FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Integrating Explicit Knowledge in the Visual Analytics Process

Model and Case Studies on Time-oriented Data

DISSERTATION

zur Erlangung des akademischen Grades

Doktor/in der technischen Wissenschaften

eingereicht von

Markus Wagner
Matrikelnummer 1326902

an der
Fakultät für Informatik der Technischen Universität Wien

1. Betreuung: Priv.-Doz. Dipl.-Ing. Dr. Wolfgang Aigner, MSc
2. Betreuung: Prof. Dr. Daniel A. Keim

Diese Dissertation haben begutachtet:

Univ.-Prof. Dr. Tobias Schreck, MSc	Univ.-Doz. Ing. MMag. Dr. Andreas Holzinger

Wien, 25.04.2017

Markus Wagner

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Integrating Explicit Knowledge in the Visual Analytics Process

Model and Case Studies on Time-oriented Data

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor/in der technischen Wissenschaften

by

Markus Wagner
Registration Number 1326902

to the Faculty of Informatics
at the Vienna University of Technology

1st Advisor: Priv.-Doz. Dipl.-Ing. Dr. Wolfgang Aigner, MSc
2nd Advisor: Prof. Dr. Daniel A. Keim

The dissertation has been reviewed by:

Univ.-Prof. Dr. Tobias Schreck, MSc	Univ.-Doz. Ing. MMag. Dr. Andreas Holzinger

Wien, 25.04.2017

Markus Wagner

Erklärung zur Verfassung der Arbeit

Markus Wagner
Mainburgstrasse 2, 3202 Hofstetten-Grünau

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

_____ _____

(Ort, Datum) (Unterschrift Verfasser)

Acknowledgements

At this point, I would like to thank all the people who have accompanied and supported me on the way of preparing my doctoral thesis:

At first, I have to give a special thank you to my supervisor Wolfgang Aigner for his helpfulness and engagement for this doctoral thesis. His manifold ideas gave me a critical and targeted access to this great research topic. Our numerous conversations on an intellectual and personal level were always an enriching and constructive exchange. I always have felt our dialogues as encouragement and motivation, for what I will always be very grateful for. I would like to thank my second advisor Daniel A. Keim for the personal and constructive feedback to my doctoral thesis. It was a great pleasure for me to meet him and his research group.

I would also like to thank all my colleagues at the Institute of Creative\Media/Technologies at St. Pölten University of Applied Sciences, especially Kerstin Blumenstein, Christina Niederer, Alexander Rind, Markus Seidl, and Matthias Zeppelzauer. They all were always a source of inspiration and have given me mental stimuli. Their honest and open criticisms as well as numerous constructive discussions were always very valuable to me. For this honesty and the collegiality, I am very grateful. Furthermore, I would like to thank Silvia Miksch and her research group at the Technische Universiät Wien for our many interesting and constructive discussions.

I also would like to give special thanks to my family, especially to my girlfriend Daniela, who always showed me her understanding and motivated me as well as she supported me to keep on track and stay focused. I also want to thank my parents, my sister and her husband who supported and motivated me and had always an open ear for me and my problems. Last but not least, I would like to say thank you to all my friends for the many evenings on which they had an open ear, for their support and their understanding. Therefore, I am very grateful.

For this reason, I would like to dedicate this work to all the people who believed in me and supported me even when I was sometimes in doubt – Thank you all!

This work was supported by the Austrian Science Fund (FWF): P25489-N23 via the KAVA-Time project.

Abstract

Visual analytics (VA) aims to combine the strengths of the human user and computers for effective data analysis. In this endeavor, the user's implicit knowledge from prior experience is an important asset that can be leveraged by both, the user and the computer to improve the analytics process. While VA environments are starting to include features to formalize, store and utilize such knowledge, the mechanisms and degree to which these environments integrate explicit knowledge varies widely. Additionally, a theoretical model and formalization of this class of VA environments is not available in the VA community yet. This doctoral thesis aims to close this gap by proposing a new theoretical high-level model conceptually grounded on the 'Simple Visualization Model' by Van Wijk supporting the visualization community. The new 'Knowledge-assisted VA Model' provides the ability to describe all components and processes to characterize knowledge-assisted VA systems. Additionally, it supports visualization experts and designers by comparing and evaluating knowledge-assisted VA systems as well by creating new solutions. To demonstrate the model's application, we use problem-driven research to study knowledge-assisted visualization systems for time-oriented data in the context of two real world problems. The first case study focuses on the domain of IT-security to support experts during behavior-based malware analysis. Therefore, we developed KAMAS, a knowledge-assisted visualization system for behavior-based malware analysis, describing its design, implementation, and evaluation. Additionally, to support clinical gait analysts during their daily work, we conducted a second case study developing KAVAGait, a knowledge-assisted VA solution for clinical gait analysis. In addition to applying the 'Knowledge-assisted VA Model' in two case studies, we also elaborate on two examples from literature. Moreover, we illustrated the utilization of the model for the comparison of different design alternatives and to evaluate existing approaches with respect to their use of knowledge. Our model provides the opportunity to inspire designers by using the model as a high-level blueprint to generate new VA environments using explicit knowledge effectively. Additionally, we observed that the VA process benefits in several ways by explicit knowledge: 1) by including it into the automated data analysis process; 2) for adapting the system's specification and 3) to faster gain new implicit knowledge about the data. Finally, we present possible future directions for future research on the integration of explicit knowledge in VA.

Kurzfassung

Visual Analytics (VA) zielt darauf ab, die Stärken des Menschen und des Computers für eine effektive Datenanalyse zu kombinieren. Hierbei ist das implizite Wissen des Menschen aus früherer Erfahrung wichtig, welches vom Menschen und vom Computer zu Verbesserung des analytischen Prozesses genutzt werden kann. Während in VA Anwendungen begonnen wird dieses Wissen eigenschaftsbasiert zu formalisieren, speichern und zu nutzen, ist jedoch der Integrationsgrad von explizitem Wissen sehr unterschiedlich und diese Art von Anwendungen wurde auch noch nie durch vorhandene VA Theoriearbeiten ausgearbeitet. Diese Dissertation zielt darauf ab, diese Lücke zu schließen, indem sie ein neues theoretisches High-Level-Modell vorschlägt, welches konzeptionell auf dem 'Simple Visualization Model' von Van Wijk basiert, um die Visualisierungscommunity zu unterstützen. Das neue 'Wissensgestützte VA Modell' bietet die Möglichkeit, alle Komponenten und Prozesse zu beschreiben, um wissensunterstützte VA Systeme zu charakterisieren. Darüber hinaus unterstützt es VisualisierungsexpertInnen und -designerInnen durch Vergleichs- und Bewertungsmöglichkeiten von wissensgestützten VA Systemen und hilft zugleich bei der Entwicklung neuer Lösungen. Um die Anwendbarkeit des neuen Modells zu demonstrieren, werden anhand eines problemorientierten Forschungsansatzes wissensgestützte Visualisierungssysteme für zeitorientierte Daten im Kontext zweier realer Probleme studiert. Die erste Fallstudie konzentriert sich auf die Domäne der IT-Sicherheit, um ExpertInnen bei der verhaltensbasierten Schadsoftwareanalyse zu unterstützen. Deshalb haben wir KAMAS, ein wissensgestütztes Visualisierungssystem für die verhaltensbasierte Schadsoftwareanalyse entwickelt und beschreiben dessen Design, Implementierung und Evaluierung. Um medizinische ExpertInnen während ihrer täglichen Arbeit zu unterstützen, führten wir eine zweite Fallstudie durch, wobei wir KAVAGait, eine wissensunterstützte VA Lösung für die klinische Ganganalyse entwickelt haben. Darüber hinaus haben wir die Verwendbarkeit des Modells für den Vergleich verschiedener Designalternativen veranschaulicht und bestehende Ansätze hinsichtlich ihrer Nutzung von Wissen bewertet. Unser Modell kann von DesignerInnen als High-Level-Blaupause verwendet werden, um neue effektive VA Anwendungen in Kombination mit explizitem Wissen zu entwickeln. Darüber hinaus haben wir festgestellt, dass der VA Prozess auf vielfältige Weise durch explizites Wissen profitiert: 1) durch Einbeziehung in den automatisierten Datenanaly-

seprozess; 2) für die Anpassung der System-Spezifikation und 3) um schneller neues implizites Wissen über die Daten zu gewinnen. Schließlich präsentieren wir zukünftige Forschungsmöglichkeiten in Bezug auf explizites Wissen. Dazu gehören unter anderem die Integration in den Interaktionsprozess, die Verflechtung mit automatisierten Datenanalysemethoden sowie die Validierung von explizitem Wissen und die Auswertung von wissensgestützten VA Systemen.

Contents

CHAPTER 1

Introduction

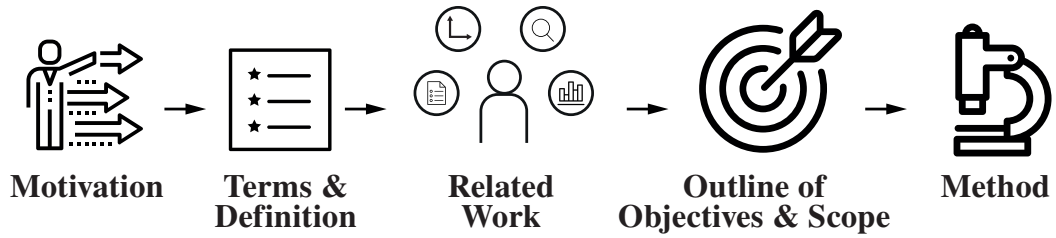


Figure 1.1: **Graphical Overview of Chapter 1** – Illustration of the main topics which are covered in this chapter.

This chapter (see Figure 1.1) describes the general motivation for this thesis (see Section 1.1) referring to: 'what is visual analytics', 'why knowledge-assisted visualization' and 'what are the benefits of stored expert knowledge'. Following this, Section 1.2 describes the major terms and definitions used in this thesis. Moreover, related work related to knowledge generation is presented in Section 1.3 including a detailed discussion. Next, the outline of objectives (see Section 1.4) including the scope and objectives of the work (see Section 1.4) as well as the general research approach and method (see Section 1.5) used to for this thesis will be described. At the end of the chapter, the dissemination (see Section 1.7) activities in relation to this thesis are presented.

1.1 Motivation

Visual analytics, "*the science of analytical reasoning facilitated by interactive visual interfaces*" (Thomas and Cook, 2005, p. 4), is a comparably young research field. It emerged from the combination of interactive data visualization with concepts from data mining, machine learning, statistics, human-computer interaction, and cognitive science (Keim et al., 2010a; Thomas and Cook, 2005). We can describe it as

> "*the method to perform tasks involving data using both computer-based analysis systems and human judgment facilitated by direct interaction with visual representations of data.*" (Rind et al., 2017a)

A major tenet of VA is that analytical reasoning is not a routine activity that can be automated completely (Munzner, 2014; Wegner, 1997). Instead it depends heavily on analysts' initiative and domain experience which they can exercise through interactive visual interfaces. Visual interfaces, especially Information Visualizations (InfoVis), are high bandwidth gateways for perception of structures, patterns, or connections hidden in the data. Interaction is "*the heart*" of InfoVis (Spence, 2006, p. 136) and allows the analytical reasoning process to be flexible and react to unexpected insights. Furthermore, visual analytics involves automated analysis methods, which perform computational activities on potentially large volumes of data and thus complement human cognition. The benefits of VA were specified by Keim et al.:

> "*Visual analytics combines automated analysis techniques with interactive visualisations for an effective understanding, reasoning and decision making on the basis of very large and complex datasets.*" (Keim et al., 2010a)

When analysts solve real world problems, they have large volumes of complex and heterogeneous data at their disposal. On the one hand, time-oriented data (see Section 1.2) is of particular importance due to its central role in many analysis contexts and tasks and on the other hand, the distinct characteristics of the dimension time make specific methods necessary. To support the analysts while performing the visual exploration and analysis of time-oriented datasets for example, a knowledge data base can be integrated into the system.

By externalization and storing of the 'implicit knowledge', it gets available as 'explicit knowledge' (see Section 1.2). In addition, to sophisticated analysis methods, implicit and tacit knowledge about the data, the domain or prior experience are often required to make sense of this data and not get overwhelmed. In this work I am going to examine 'how the visual analytics process can benefit from explicit knowledge of analysts'. This will help to develop more effective environments for gaining insights – the ability to specify, model and make use of auxiliary information about data and

domain specifics. In addition to the raw data they will help to better select, tailor, and adjust appropriate methods for visualization, interaction, and automated analysis. Potential application domains benefiting from this are healthcare, biotechnology, urban- and cyberinfrastructures, environmental science and many more.

The main goal of this thesis is to develop knowledge-assisted visualization and interaction methods (see Section 1.2) that make use of explicit knowledge to improve these methods in a context-specific manner. This reflects complicated problems which are recognized by the visual analytics (VA) community as important research challenges (Pike et al., 2009). Additionally, Purchase et al. (2008, pp. 58) described the need for models and frameworks describing visualization and "*a deeper exploration of formal, scientific models is needed for a strengthening of the field*" (Purchase et al., 2008, pp. 58) .

To achieve this goal, we compared different well known models and frameworks (see Section 2) which can be used to describe different kinds of visualization workflows and systems. It turned out that although some of these models and frameworks are including knowledge in different ways, but none of them offers the possibility to be used as a high level framework to create knowledge-assisted visual analytics solutions. To close this gap, we examined, which of these models can be used as conceptual grounding for the generalization and construction of a novel model, describing the knowledge-assisted visual analytics process. Therefore, we provide a mathematical abstraction and theoretical modeling of the visual analytics processes based on the introduction of our novel 'Knowledge-assisted VA Model'. The new model, describes all the components and processes as well as their connections for explicit knowledge integration and extraction, the integration of automated data analysis methods as well as the combination of both.

In general, the visualization and VA process can benefit from explicit knowledge of analysts (Chen et al., 2009; Wang and Ma, 2008), whereby knowledge supports the analysts with "*more efficient and effective data manipulation, management, and understanding*" (Wang and Ma, 2008). Thus, explicit knowledge helps to develop more effective environments for gaining insights – the ability to specify, model and make use of auxiliary information about data and domain specifics. In addition to the raw data, explicit knowledge will help to better select, tailor, and adjust appropriate methods for visualization, interaction, and automated analysis. Potential benefiting application domains are healthcare, biotechnology, urban- and cyberinfrastructures, environmental science and many more.

The integration of implicit knowledge into a visualization system as explicit knowledge is one of the problems which are recognized by the visual analytics community as important research challenges since 2009 (Pike et al., 2009). Leading visualization researchers have repeatedly called for the integration of knowledge with visualization. Chen (2005) lists 'prior knowledge' as one of ten unsolved InfoVis problems. In particular, two different types of prior knowledge are necessary to understand InfoVis: 1) there is the 'operational' knowledge which describes how to interact interact with the

InfoVis system; and 2) there is the 'domain' knowledge which describes how to interpret the content. Chen (2005) argues that InfoVis systems need to be adaptive for accumulated knowledge of users, especially 'domain' knowledge is needed to interpret results. In their discussion of the 'science of interaction', Pike et al. (2009) pointed out that visual analytics tools have only underdeveloped abilities to represent and reason with human knowledge. Therefore, they declare 'knowledge-based interfaces' as one of seven research challenges for the next years.

To be effective, visual analytics needs to provide 'precise' data, "*which is immediate, relevant and understandable to individual users, groups, or communities of interest*" (Kielman et al., 2009, p. 240). For example, if analysts might have hunches, which sources they believe to be trustable, which results appear plausible and which insights they deem relevant. By externalizing this knowledge and using it, analysts can avoid cognitive overload and use visualization and automated analysis methods more effectively. They can avoid reinventing the wheel, when they repeat analysis on a different dataset, a year later, or through a different technique. They can keep track of interpretations and analysis steps, communicate with co-analysts, and document results for insight provenance. Additionally, explicit knowledge provides the ability to share and combine the experts knowledge to increase the analysis quality and to learn from each other.

1.2 Terms and Definitions

In this section, the main terms and definitions which were used in this chapter will be described in detail for a better understanding.

Time-oriented Data

Visual exploration and analytical reasoning with time-oriented data are common and important for numerous application scenarios, e.g., in healthcare (Combi et al., 2010), business (Lammarsch et al., 2009), and security (Fischer et al., 2012; Saxe et al., 2012). Furthermore, time and time-oriented data have distinct characteristics that make it worthwhile to treat it as a separate data type (Aigner et al., 2011; Andrienko and Andrienko, 2005; Shneiderman, 1996). Explicit knowledge may model the relevance of data items in respect to zoom levels and recommend summarization techniques depending on task(s) and domain(s).

When dealing with time, we commonly interpret it with a calendar and its time units are essential for reasoning about time. However, these calendars have complex structures. In the Gregorian calendar the duration of a month varies between 28 and 31 days and weeks overlap with months and years. Furthermore, available data may be measured at different levels of temporal precision. Some patterns in time-oriented data

may emerge when a cyclic structure of time is assumed, for example, traffic volume by time of day, temperature by season.

In other cases, an analyst will need to balance such effects to understand long term trends. An analyst may be interested to compare developments in the data that do not cover the same portion of time. For such comparisons, they are interested in relative time to some certain events. Therefore they would align patient data by the beginning of a specific therapy, and show all events one day after the beginning (Rind et al., 2011; Wang et al., 2008).

Knowledge

As there are many competing definitions of 'knowledge' in scientific discourse, we decided to use the definition used in the community of knowledge-assisted visualization:

> "*Knowledge: Data that represents the results of a computer-simulated cognitive process, such as perception, learning, association, and reasoning, or the transcripts of some knowledge acquired by human beings.*" (Chen et al., 2009, p. 13)

This work manly focuses on the second part of this definition for knowledge. Additionally, 'knowledge' is divided into two different areas, whereby different terms are used for their description in the community.

Implicit Knowledge: On the one hand, prior literature uses the two terms 'tacit and implicit knowledge'. The term 'implicit knowledge' describes the user's own expertise on how to perform data analysis tasks and searching for insights. For example, by choosing different coloring settings or viewing positions, the user receives meaningful insights. Additionally, the user has the possibility of choosing different viewing positions analyzing the visualization results which "*can reveal more meaningful information or a more problematic scenario that requires further investigation*" (Chen et al., 2009). Wang et al. (2009) described that tacit knowledge "*is personal and specialized and can only be extracted by human*". In this work, we are concerned with both meanings and use the term 'implicit knowledge' without distinguishing between their subtle differences.

Explicit Knowledge: On the other hand, the term 'explicit knowledge' describes the users computerized representations of interests and domain knowledge (Wang et al., 2009). It "*can be processed by a computer, transmitted electronically, or stored in a database*" (Wang et al., 2009). In contrast Chen et al. (2009) used the term explicit knowledge describing "*the memory of events, facts and concepts, and the understanding of their meanings, context and associations*". This can also exist in a computerized form and be shared with others. Thus, explicit knowledge

exists in a form of data and it is independent from the users implicit knowledge. Additionally, it is different to the data and information that will be processed or visualized (Chen et al., 2009; Wang et al., 2009). In this work, knowledge which is represented in a computerized form will be referred as explicit knowledge according to the definition of Wang et al. (2009).

In this work, the focus will be to investigate how explicit knowledge can be used to support interactive visualization (knowledge-assisted visualization or VA). The specification of the users' knowledge will not be a part of this work.

Knowledge-assisted Visualization

There are numerous ways to improve visualization and interaction methods based on explicit knowledge. For example choosing variables for scatter plot axes, zooming to an area of interest instead of the viewport center, highlighting data items in a different color, or drawing reference lines in the background of a plot. Such optimizations can be applied to most aspects of the visualization and developing a general framework instead of scenario-specific solutions is a challenging task (Tominski, 2011).

Visual analytics of data is an exploratory process. If there is a given dataset, the user needs to decide, which visualization method(s) he/she wants to use for the data exploration. The objectives of knowledge-assisted visualizations include the sharing of explicit knowledge (domain knowledge) from different users. Thus, it reduces the stress on users for appropriate knowledge about complex visualization techniques (Chen and Hagen, 2010).

For example, explicit knowledge can be used to summarize and abstract a dataset. These summarizations and abstractions will form another dataset, which can be visualized through a wide range of existing visualization and interaction methods. Typically this abstraction process reduces the size of the dataset significantly. However, analysts also need to access the input dataset and switching between visualizations of both datasets should be facilitated by techniques like semantic zoom (Perlin and Fox, 1993) or brushing and linking (Becker and Cleveland, 1987). The wide ranging potential of utilizing explicit knowledge has already been demonstrated in recent research (Chen and Hagen, 2010). Despite this, most current visualization systems do not take advantage of explicit knowledge captured from domain experts.

1.3 Related Work on Knowledge Generation

The permanent growth of methods available for data visualization can be confusing for novice users and even for domain experts. Another problem is that the extensive know-how is not stored in a central place because it is separated in sub-communities (Mistelbauer et al., 2012; Nam et al., 2009). Knowledge-assisted visualizations (KAV) are a rapidly growing area which uses directly integrated expert knowledge to produce effective data visualizations. Most of the KAV systems concentrate on the integration of specific domain knowledge which can only be used for exactly these analysis tasks. Additionally it is important that the users become aware of the different methods which are needed for the data exploration and interaction but not all methods are usable or effective for the different data types to gain the expected results (Mistelbauer et al., 2012; Wang et al., 2009). Existing data visualization systems need a manual specification for each data attribute of possible visualizations. This is also significant for data which are available as linked open data and systems which represent the data as graphs with objects and weighted edges with labels (Cammarano et al., 2007). It is important to differentiate between automatic visualization systems (AVS) and automated visualization systems. Automatic visualization systems make independent decisions about the visualization activities. The automated visualization system is a programming system for the automated generation of diagrams, graphics and visualizations. In general it is necessary that the flow of an automate visualization system works like an expert would perform it (Wills and Wilkinson, 2010).

Cammarano et al. (2007) investigated the automatization of the data integration and the automatic mapping of data attributes to visual attributes. This workflow was described as the "*schema matching problem*" (Cammarano et al., 2007). During the path indexing process, each visualization attribute is matched to a set of path templates which are called schema paths, which are predicate sequences used in the next phase to find the attributes for each object. The used data model equals the Resource-Description-Framework (RDF) (RDF Working Group, 2017). Each subject-predicate-object triple of the RDF model corresponds to the edge which connects a subject with an object. Based on the provided experiments the authors showed that the needed data could be identified frequently enough that the system could be used as an exploration tool. This way it saves the user from schema-heterogeneity.

Falconer et al. (2009) treated the generation of adapted visualizations based on ontological datasets and the specification of ontological mappings. The usability of this approach was demonstrated by the use of the ontology-mapping-tool 'COGZ' in this paper, whereat ontological mappings would be translated into software transformation rules. With this transformations, the domain specific data are converted in a way to fit to a model which describes the visualization. To perform the mappings, the authors developed a rule description library based on 'Atlas Transformation Language' (ATL) (Jouault and Kurtev, 2006). With this library they converted the specific source

data into target mappings. The tests of the system showed that the system performed an automated mapping of the data, whereby the user was assisted greatly in his work.

Gilson et al. (2008) described the automated generation of visualizations from domain specific data of the web. Therefore, they described a general system pipeline which combines ontological mappings and probabilistic argumentation techniques. In the first step, they mapped a website into a domain ontology which stores the semantics of the specific subject domains (e.g. music charts). Subsequently they mapped it to one or more visual-representation-ontologies whereby each contains the semantic of a visualization technique (e.g. treemap). To guarantee the mapping between the two ontologies, they introduced a semantic-bridge-ontology which specifies the suitability of each ontology. Based on this approach, they implemented a prototype with the name 'SemViz'. For the tests of the system, they used the data of popular music websites without having prior knowledge about the pages.

Mackinlay et al. (2007) introduced in their paper the tool 'Show Me' which is an integrated set of interface commands and standard values which automatically integrate data presentations into the tool 'Tableau'. The key aspect of Tableau is 'VizQL' (visualization query language) which would be used by Show Me to generate automated presentations in a view table. One of the major aspects is the usability of the tool which has to support the flow of visual analytics. This includes the automated selection of marking techniques, commands to combine individual fields to one view and some commands to generate views from multiple fields. The APT system by Makinley (Mackinlay, 1986) forms the basis for the automated design of graphical representations of relational information. The authors implemented Bertin's semiology of graphics as algebraic operations (Bertin, 1983) and used them for the search of effective presentations for the information.

Wills and Wilkinson (2010) described the data viewer tool 'AutoVis' which reacts on content (text, relational tables, hierarchies, streams, images) and presents the containing information in an appropriate form (e.g. like an expert will do it). The design is based on the grammar of graphics (Wilkinson, 2005) and the logic is based on statistical analysis. This automatic visualization system was developed to provide a first look on the data until the modeling and analysis are finished. AutoVis was designed to protect the researchers from ignoring missing data, outliers, miscodings and other anomalies which injure the statistical adoption or the validity of the models. The design of this system contains some unique features: a spare interface, a graphics generator, a statistical analysis to protect users from false conclusions and pattern recognition.

Tominski (2011) created a new approach for event-based visualizations which contains three fundamental stages. First, the event specification is to generate event types which are interesting as a visualization for the users. This translates the user interests in an understandable representation for the computer, where they should be formulated for the user as easy as possible. The second stage specified where the interests of the users

intersects or overlaps with the data. This detection must be kept as general as possible so that it is applicable to a large number of event types. The basic task is to assess encapsulated the conditions of event types. The aim of the third step is to integrate the detected event instances in visual representations (which reflect the interests of users). The event representation has great influence on the extent to which the event-based visualization closes the Worldview Gap. The Worldview Gap is described by Amar and Stasko (2005) as "*the gap between what is being shown in a visual representation and what actually needs to be shown to intuitively draw representational conclusions*" (Amar and Stasko, 2005). Therefore, it has to be communicated: 1) what has been found; 2) highlight the events among the remaining data and 3) show why the events is interesting (Tominski, 2011). This general model allows the use of many different visualizations and the specific data-driven events focused on relational data visualizations of today.

Kadlec et al. (2010) stated that scientists are using seismic 3D data for 30 years to explore the earth crust by the interpretation of seismic data which needs a lot of expert knowledge. But it is possible to use the knowledge of experts in order to facilitate the segmentation of the geological features. To reduce the need for knowledge of seismic data and attributes, this new method uses surfaces which are growing in surfaces of known geologic characteristics. The result is a knowledge-assisted visualization and segmentation system that allows non-expert users a fast segmentation of geological features in complex data collections. The process begins with the exploration of seismic datasets using 2D slices. This 3D volume is searched interactively for possible interesting features. The elements are rendered and the user receives a feedback on the quality of the segmented features. If the system indicates a link to a non-feature, the user has the ability to repair this link. This approach transferred the expert knowledge very fast and reliable for non-expert users. This way the analysis quality of non-expert users increases similar to those of experts.

Nam et al. (2009) mentioned that the 'specific know-how' of a domain is separated in sub-communities. To overcome this problem, they started to store visualization expertises and methods in combination with possible datasets. An important aspect is to edit newly generated datasets with the existing expert knowledge from a database. Therefore, they used several levels of granularity to use the knowledge of the database correctly. Thus, they described the first step of a framework specifically in relation to the data categorization and classification by using a set of feature vectors. The usability of the framework was demonstrated by four medical datasets (knee, chest and head 2x) in a 2D application. They calculated for feature points for every dataset in a local density histogram and described them as low-level feature vectors. These were used to prepare high-level-models of the data objects. Furthermore, they intended to support a general framework for classification tasks by indexing a knowledge database for knowledge-assisted visualization systems (KAV).

Wang et al. (2009) differentiated between two types of knowledge (implicit and explicit) and defined four conversion processes between them (internalization, externalization, cooperation and combination) which were included in knowledge-assisted visualizations. They showed the applications of these four processes, their roles and utilities in real-life scenarios, using a visual analysis system for the Department of Transportation. The authors assume that the analysts can learn more through the interaction between implicit and explicit knowledge through the use of interactive visualization tools. As a further differentiation between implicit and explicit knowledge for knowledge-assisted visualization, the authors defined the following distinctions:

- "*Explicit knowledge is different from data or information.*"
- "*Tacit knowledge can only result from human cognitive processing (reasoning).*"
- "*Explicit knowledge exists in data, and is independent from the user or his tacit knowledge.*"
- "*Explicit and tacit knowledge are related and can be connected through the use of interactive visualization tools.*" (Wang et al., 2009, p. 2)

By an connection of the system to an ontological knowledge source, the visual analytics system enables the user an interactive access to the expertise of the expert. Thus, this visualization system showed that the four knowledge conversion processes are possible for the design of knowledge-assisted visualization.

Mistelbauer et al. (2012) presented a knowledge-assisted system for medical data visualization ('Smart Super Views'). This system has been tested in the medical domain and expert feedback was obtained. The Smart Super Views system contains three major steps: In the first step the information from different sources will be collected and merged. In the second step, the user decides where a region of interest (ROI) is located in the data and which visualization technique should be used. In the third step, the user interacts with the provided visualization and starts with a detailed inspection of the data. In contrast to other systems where the user himself has to select the visualization, this system will support the user in his decisions. The rule specification module of the system defines the connection between the input data and the output visualization. To model these connections, 'if-then' clauses will be used, which were specified by domain experts. Additionally, these clauses were stored in a user-readable form, in a file.

Discussion

The automation of the data integration and the automatic mapping of data attributes to visual attributes is discussed in many papers (e.g. (Cammarano et al., 2007; Falconer et al., 2009; Gilson et al., 2008; Kadlec et al., 2010; Mackinlay et al., 2007; Mistelbauer et al., 2012; Wills and Wilkinson, 2010)). The generation of adapted visualizations which are based on ontological datasets and the specification of ontological mappings are treated by Falconer et al. (2009). A similar approach was also followed by Gilson et al. (2008). They described a general system pipeline which combines ontology mapping and probabilistic reasoning techniques. The approach of Gilson et al. (2008) is described by the automated generation of visualizations of domain-specific data from the web. In contrast, Falconer et al. (2009) used the 'COGZ' tool for their approach which converts ontological mappings in software transformation rules so that it describes a model which fits the visualization. Cammarano et al. (2007) describes a similar process as "*schema matching problem*". It describes finding ways in the data model for each required visualization attribute based on visualization templates. In the end, most of the automated data mappings for visualizations aim to perform in similar ways. Gilson et al. (2008) maps the semantic data to visual-representation-ontologys, where each part contains the semantics of a visualization (e.g. treemaps). A slightly different approach by Mackinlay et al. (2007) has a set of interactive commands, defaults, automated data integration and presentations to accomplish the automated data presentation in 'Tableau'. Due to the automatic selection of markers, commands and combination of individual fields to a view, the user is able to rapidly and easily create visualizations by the use of the tool 'Show Me'. Furthermore, the tool 'AutoVis' was implemented by Wills and Wilkinson (2010) to take a first look at data which has to be visualized. For this, the system used statistical analysis for modeling the visualizations. Thus, the user should be prevented from ignoring missing data, outliers, missing codes and other anomalies. The protection (e.g., (Cammarano et al., 2007)) or the support of the users during their work (e.g., (Falconer et al., 2009; Kadlec et al., 2010; Mackinlay et al., 2007; Mistelbauer et al., 2012; Tominski, 2011)) is one of the main foci of this papers.

The 'event-based model' by Tominski (2011) permits the applicability for many different visualizations which are divided into three stages. A stepwise subdivision is also used by Gilson et al. (2008) for the required mapping instances and Mistelbauer et al. (2012) used a stepwise subdivision for the three processing steps of the 'Smart Super Views'. The three essential steps for a knowledge-assisted visualization tool according to Mistelbauer et al. (2012) are: first to collect and merge the data; second, to determine the region of interest (ROI) in the data by the user; third, the interaction of users with the generated visualization. The automated generation of visualizations respectively the assigning of the data to pre-defined visualization templates is also carried out in other approaches, which were presented in this state of the art overview, in similar ways. In

some papers it is also described that the knowledge of experts is distributed. Therefore, it is important to develop knowledge-assisted visualization systems to make the knowledge of experts available for the users (e.g., (Kadlec et al., 2010; Nam et al., 2009; Wang et al., 2009)). Commonly, the knowledge of experts is stored in files (e.g., (Mistelbauer et al., 2012)), using RDF (e.g., (Cammarano et al., 2007)) or in a knowledge database (e.g., (Nam et al., 2009)).

Based on these findings, it can be seen, that most of the discussed approaches treat the storing or the availability of explicit knowledge. Additionally, most of the currently implemented knowledge-assisted visualization systems are focused on the integration of specific domain knowledge which could only be used for precisely defined analysis tasks. Even automated generations of visualizations are described for example by Mackinlay et al. (2007). In general, it has been shown that knowledge-assisted visualization methods combined with VA are not clearly addressed in those papers. Additionally, non of the presented approaches provides a theoretical model describing the generation of knowledge-assisted VA systems. Thus it is clear that there is space for future research in the field of knowledge-assisted visualizations in combination with visual analytics. Especially in the generalization of knowledge-assisted visualization methods and the creation of a novel knowledge-assisted VA model.

1.4 Outline of Objectives

In this thesis, the overall aim is to develop knowledge-assisted visual analytics methods to gain insights effectively from time-oriented datasets (see Section 1.2). In these methods, explicit knowledge is treated as externally given, and the focus will be on how to best integrate them into the process to improve sense-making.

Knowledge-assisted visualization and interaction methods (see Section 1.2) will be developed to explore time-oriented datasets. I hypothesize that explicit knowledge (see Section 1.2) will afford more effective analytical reasoning processes (e.g., through semi-automated visualization) to prevent data interpretation errors. Finally, all developed methods need to undergo evaluation. Scenarios will be identified with target users, tasks, and datasets that act as testbeds. Designs and prototypes will be iteratively evaluated and refined. Based on these aims this work investigates the following research questions:

Main Question: How can the visual analytics process benefit from explicit knowledge of analysts / domain experts?

Sub Question 1: How can explicit knowledge be visually represented effectively in a visual analytics system?

Sub Question 2: Is it possible to generalize the interaction with knowledge-assisted visualization methods for different application scenarios?

Sub Question 3: How can analysts during the exploration of a large amount of data benefit from knowledge-assisted visual analytics methods?

The developed methods of this thesis will primarily deal with time-oriented data, but they will also be applicable for other datasets, in future work.

Scope of the Work

The goal of this thesis is to show how the visual analytics process can benefit from the use of knowledge-assisted visual analytics methods. To achieve this, 'implicit knowledge' of the domain-specific analysis experts will be stored as 'explicit knowledge' (e.g., in a database). This explicit knowledge will be used to support users during their workflow in a context-specific manner (e.g., behavior-based malware pattern analysis) to achieve their goals. Thus, that knowledge-assisted visualization methods will support the generation of more effective visual analytics environments to gain more insights and achieve better quality results compared to current methods.

In addition to the raw data, knowledge-assisted visual analytics methods will help to better select, tailor, and adjust appropriate methods for visual representation, interaction, automated analysis and prevent data interpretation errors. By externalizing the domain-specific expert knowledge and using it, analysts can avoid cognitive overload and use visualization and automated analysis methods more effectively. This way, analysts can avoid reinventing the wheel, when they repeat analysis on a different dataset, a year later, or using a different technique. Thus, they can concentrate on the important steps of interpretations and analysis, communicate with co-analysts, and document results for insight provenance.

Furthermore, the tested knowledge-assisted visualization methods will be generalized as a model for knowledge-assisted visualization (see Part I). Based on this generalizations and the results of the interviews and user studies, we will propose a new model for future knowledge-assisted visual analytics environments to support the community. The resulting model will than be applied to the two case studies (see Part II and III) described in this thesis to demonstrate the applicability of the model. Additionally, it will be demonstrated how similar knowledge-assisted visualization methods can be used for different domains based on two design studies.

1.5 General Research Approach & Method

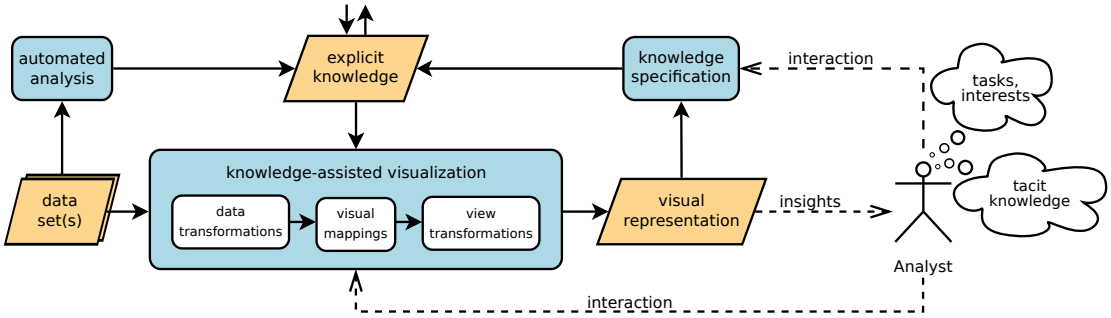


Figure 1.2: **Knowledge integration in the VA process** – This image illustrates the idea of integrating explicit knowledge for knowledge-assisted visualizations in the visual analytics process (Wagner, 2015).

In this section, the general approach on how to apply and study 'knowledge-assisted VA' methods is presented. 'Explicit knowledge' of domain experts will be used to support users during the analysis of time-oriented data.

By the use of knowledge-assisted visualizations, the available datasets will be turned into interactive and visual representations (see Figure 1.2). Thus, explicit knowledge will be used to achieve effective representations in terms of the analysis' tasks. The visualization process can be described by using the reference model of Card et al. (1999) or the data state model of Chi and Riedl (1998). Both descriptions relate to the 'internalization' of the model of Wang et al. (2009).

This thesis follows the well-known 'nested model for visualization design and validation' as proposed by Munzner (2009) (see Figure 1.3). This unified approach splits visualization design into four levels in combination with corresponding evaluation methods to evaluate the results at each level. Starting from the top, the levels of the nested model for visualization design and validation are:

Domain problem and data characterization: On this level, the goal is to understand the problem domain, the users' tasks and their goals.

Operation and data type abstraction: Within the abstraction level, domain specific vocabulary (problems and data) will be mapped to a more generic description which fits to the vocabulary of computer scientists (visualization community).

Visual encoding and interaction design: In the third level, the visual encoding of the data and the interaction methods for the data exploration will be designed.

Algorithm design: Designing of the implementation of the visual encoding and interaction methods.

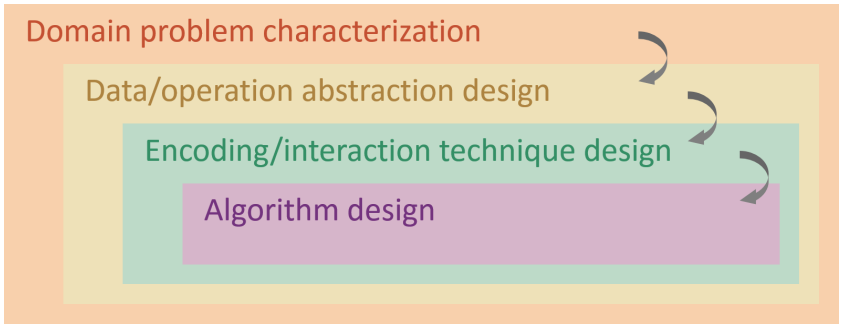


Figure 1.3: **Nested Model** – The four levels of the 'nested model for visualization design and validation' by Munzner (2009). .

Since these are nested levels, the output of the upstream level which is situated above, is the input of the downstream level which is situated below. Considering it is current practice, visual analytics was defined by Keim et al. as the "*[combination] of automated analysis techniques with interactive visualizations for an effective understanding, reasoning and decision making in the basis of very large and complex datasets*" (Keim et al., 2010a, p. 7). In general the nested model for visualization design and validation does not include automated analysis explicitly, but it can be conceptualized on the abstraction level where the data transformation takes place. This thesis will focus on knowledge-assisted visualizations for visual analytics to develop novel visual encoding and interaction methods for time-oriented data.

According to the need, that no model exists to describe the process of knowledge-assisted VA, we introduced the novel 'Knowledge-assisted VA Model' (see Part I). Therefore, we started with a literature research for established models and frameworks in the community of information visualization. In detail, we were searching for models, describing the systems architecture in relation to the needed components and processes as well as their connection. Based on this literature research, we found out that no model covers all the components and processes which are needed to describe knowledge-assisted visual analytics. But several of the found models and frameworks offers the possibility to be used as conceptual grounding. With regard to our defined requirements, which have to be fulfilled by the new model, we decided to use the well known and established 'Simple Visualization Model' by Van Wijk (2005) as conceptual grounding. To do so, we added all processes and components to describe knowledge-assisted visual analytics as well as we established a formal description for contained elements and their connections. To validate the new 'Knowledge-assisted VA Model' (see Part IV), we performed two case studies (see Part II and Part III) described in this thesis, as well as two other prototypes (Federico et al., 2015; Gove et al., 2014). Moreover, the novel model was also compared to the 'Knowledge Generation Model for VA' (Sacha et al., 2014). To describe the applicability of the model in detail, its

descriptive, evaluative and generative power (Beaudouin-Lafon, 2004) is demonstrated at the end of this doctoral thesis.

During the two performed case studies, we followed a problem-driven approach to study knowledge-assisted visualization systems for time-oriented data in the context of real world problems. At first, the research focused on the IT-security domain (see Part II). More specifically, we analyzed the needs of malware analysts in relation to their work on behavior-based malware pattern analysis (Dornhackl et al., 2014). Therefore, we designed knowledge-assisted visual analytics methods and implement a software prototype as proof of concept to test the designed methods. After this, we focused on physical therapy as a second domain. More precisely, we analyzed the needs of gait analysts dealing with clinical gait analysis data (Perry and Burnfield, 2010), collected by force plates (Winter, 2009, pp. 117). Therefore, we redesigned and extended the visual analytics methods for the interactive data exploration and extend the proof of concept software prototype to test the designed methods.

To ensure a knowledgeable research we started with a problem characterization and abstraction based on the design study methodology of Sedlmair et al. (2012b), which brings us into the first level (domain problem and data characterization) of the nested model. From there, we worked inwards along Munzner's nested model for visualization design and validation. To perform the problem characterization and abstraction, we followed a threefold qualitative research approach which consists of a systematic literature research, a focus group (Lazar et al., 2010, p. 192) and semi-structured interviews (Lazar et al., 2010, p. 184) with domain experts. Based on the results of the threefold approach, we used the 'design triangle' as proposed by Miksch and Aigner (2014) to analyze the data, the users and the tasks which fits to the second level of Munzner's model (operation and data type abstraction).

In the following steps, we started with the visualization and interaction design followed by the algorithm design and implementation based on a user centered design process (Sharp et al., 2007). Therefore, we produced sketches, followed by screen prototypes and functional prototypes (Kulyk et al., 2007, p. 50). This way, we fulfilled the third (visual encoding and interaction design) and the fourth (algorithm design) level of Munzner's nested model. During these steps, focus group members were included in the design and implementation process to get feedback about the design and the functionality of the knowledge-assisted visual analytics system. Thus it was possible to improve the design and the handling of the designed knowledge-assisted visualization methods. Additionally, user studies were performed with predefined datasets to evaluate the usability (Cooper et al., 2007, p. 70) of the new knowledge-assisted visualization methods based on the implemented visual analytics system.

After the performed user studies of the first real world problem (behavior-based malware pattern analysis) and their related knowledge-assisted visualization methods are completed, we continued to test their applicability on the second real world problem

with regard to clinical gait analysis (see Part III). Therefore, we adapted and extended the knowledge-assisted visualization methods in an appropriated way, if it was necessary, and we repeated the previously described research process in the required extent.

1.6 Conventions

The work, which is presented in this doctoral thesis was conducted in the context of the FWF-funded research project KAVA-Time. The basic designs, architectural ideas and evaluations were elaborated and carried out by myself. The resulting research and its results have been shaped on the basis of valuable discussions and inputs, mainly by my supervisors and colleagues. For this reason I decided to use the pronoun "we" instead of "I" in this thesis.

In this doctoral thesis, any formulations in male and female form are considered to be gender-neutral. All the explanations apply equally to men and women.

1.7 Dissemination

Parts of the presented results in this work have already been presented and published at scientific conferences and journals. All these results have been restructured, revised, expanded and placed in a general context in this dissertation. In the following list the corresponding publications are listed chronologically:

1. Wagner, M., Aigner, W., Rind, A., Dornhackl, H., Kadletz, K., Luh, R., and Tavolato, P. (2014). Problem characterization and abstraction for visual analytics in behavior-based malware pattern analysis. In *Int. Workshop on Visualization for Cyber Security (VizSec)*, pages 9–16. ACM

2. Wagner, M. (2015). Integrating Explicit Knowledge in the Visual Analytics Process. In *Doctoral Consortium on Computer Vision, Imaging and Computer Graphics Theory and Applications (DCVISIGRAPP)*, pages 9–18, Berlin. SCITEPRESS Digital Library

3. Wagner, M., Aigner, W., Haberson, A., and Rind, A. (2015a). Literature review in visual analytics for malware pattern analysis. In *Proc. of the Forschungsforum der österreichischen Fachhochschulen (FFH)*. FH Hagenberg

4. Wagner, M., Fischer, F., Luh, R., Haberson, A., Rind, A., Keim, D. A., and Aigner, W. (2015c). A survey of visualization systems for malware analysis. In *Eurographics Conf. on Visualization (EuroVis) State of The Art Reports*, pages 105–125. Eurographics

5. Wagner, M., Rind, A., Rottermanner, G., Niederer, C., and Aigner, W. (2016b). Knowledge-Assisted Rule Building for Malware Analysis. In *Proc. of the Forschungsforum der österreichischen Fachhochschulen (FFH)*, Vienna, Austria. FH des BFI Wien

6. Luh, R., Schramm, G., Wagner, M., and Schrittwieser, S. (2017). Sequitur-based Inference and Analysis Framework for Malicious System Behavior. In *Workshop for Formal Methods in Software Engineering (ForSE), (ICISSP)*, pages 632–643, Porto, Portugal. SCITEPRESS Digital Library

7. Wagner, M., Rind, A., Thür, N., and Aigner, W. (2017b). A knowledge-assisted visual malware analysis system: design, validation, and reflection of KAMAS. *Computers & Security*, 67:1–15

8. Wagner, M., Sacha, D., Rind, A., Fischer, F., Luh, R., Schrittwieser, S., Keim, D. A., and Aigner, W. (2017c). Visual Analytics: Foundations and Experiences in Malware Analysis. In Othmane, L. B., Jaatun, M. G., and Weippl, E., editors, *Empirical Research for Software Security: Foundations and Experience*. CRC/Taylor and Francis. In press

9. Wagner, M., Slijepčević, D., Horsak, B., Rind, A., Zeppelzauer, M., and Aigner, W. (2017d). KAVAGait – knowledge-assisted visual analytics solution for gait analysis: A design study. *IEEE Transactions on Visualization and Computer Graphics*. Under review

10. Wagner, M., Federico, P., Rind, A., Amor-Amorós, A., Aigner, W., and Miksch, S. (2017a). The simple model of knowledge-assisted visualization. *IEEE Transactions on Visualization and Computer Graphics*. Under review

Part I

Theory: Knowledge-assisted VA Model

CHAPTER 2

Model Criteria & Related Work

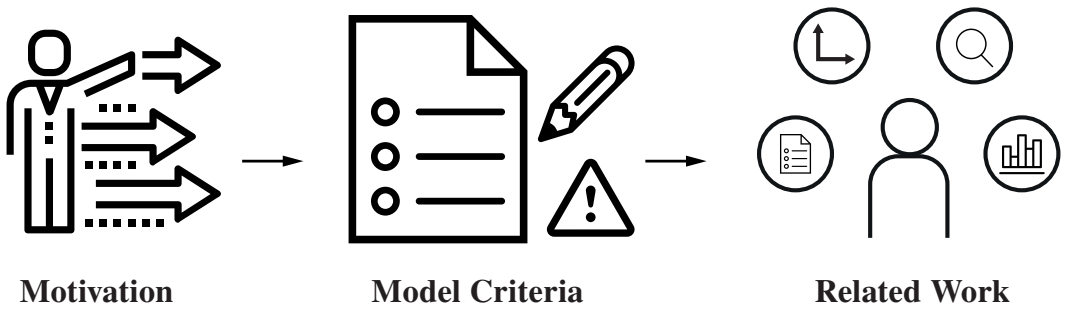


Figure 2.1: **Graphical Overview of Chapter 2** – Illustration of the main topics which are covered in this chapter to provide a general overview of the currently available models and frameworks.

As represented in Figure 2.1, this chapter starts with the motivation for the generation of a novel 'Knowledge-assisted VA Model' (see Section 2.1). Additionally, it describes with the three main criteria (see Section 2.2) to be (partially) fulfilled by the model we are searching for to describe the integration of knowledge into the visualization and VA process. Additionally, the chapter presents the related work (see Section 2.3) with regard to the found models and frameworks, which are discussed (see Section 2.4) at the end of the chapter.

2.1 Motivation

The analytical reasoning for real world decision making involves volumes of uncertain, complex, and often conflicting data that analysts need to make sense of. In addition to sophisticated analysis methods, knowledge about the data, the domain, and prior experience are required to not get overwhelmed in this endeavor. Ideally, a VA environment would leverage this knowledge to adapt itself to the specific context of domain users and the analyzed data. By explicitly taking advantage of expert knowledge in a VA system, it gives rise to more effective environments for gaining insights. I.e., making use of auxiliary information about data and domain specifics in addition to the raw data, will help to better select, tailor, and adjust appropriate methods for visual representation, interaction, and automated analysis.

To facilitate such epistemic processes, a number of visualization researchers have repeatedly called for the integration of knowledge with visualization (Chen, 2005; Thomas and Cook, 2005). In their discussion of the 'science of interaction', Pike et al. (2009) point out that VA environments have only underdeveloped abilities to represent and reason with human knowledge. Therefore, they declared 'knowledge-based interfaces' as one of seven research challenges. These calls have resulted in a number of visualization environments that include features to generate, transform, and utilize explicit knowledge. However, the mechanisms and degree to which these environments integrate explicit knowledge vary widely. Additionally, this important class of VA environments has not yet been investigated from a more systematic, conceptual perspective of VA theory. This raises the need to extend a existing model of the community or to develop a new 'Knowledge-assisted VA Model' describing the integration of explicit knowledge, its extraction and its application in the VA process.

2.2 Model Criteria

For the generation of a theoretical environment, we are searching for an existing theoretical model to describe the process of knowledge-assisted VA or a model which can be used as conceptual grounding for the needed development of a novel theoretical knowledge-assisted VA model (Purchase et al., 2008). Generally, we are searching for models and frameworks which have the main focus on a theoretical system architecture. Thereby, we are searching for a formalization that is modeling a knowledge-assisted visualization system's internal processes in combination with the system's user in a general, non-application specific manner. To do so, three high level criteria have to be fulfilled in relation to the models design and its level of detail:

Data Exploration & Analysis: In general, a VA system should contain one ore more automated data analysis methods to fulfill the "*analyze first*" part of Keim's VA

definition (Keim et al., 2010a). As VA cannot be automated completely (Wegner, 1997), the theoretical model has to describe the usage of interaction methods for data exploration and knowledge generation by the user. Therefore, the needed insights are gained by the perception of the represented visualization result at a specific time point.

Dynamics: The visual and system internal data representation can be changed by the user and/or by various system-internal processes. Therefore, on the one hand, interactions for data exploration (e.g., zooming, filtering, panning, sorting) can be performed by the user. On the other hand, automated data analysis processes (e.g., clustering, highlighting, transformations) can be applied system-internally. All these actions affect the visualization and its data representation over time, thus the temporal aspect has also to be considered in the theoretical model.

Knowledge: The integration and the generation of knowledge is also a very important aspect which has to be covered by the model. On the one hand, the model has to describe how the user gains new implicit knowledge based on the data exploration and analysis insights. On the other hand, it has to describe how the users implicit knowledge and knowledge which is gained by automated methods for example can be computerized and stored as explicit knowledge to make its system internally available for further analysis support (Chen et al., 2009; Wang and Ma, 2008).

2.3 Related Work

In this section, 14 models and frameworks (based on 17 publications) for information visualization design and implementation are presented in detail. All these models and frameworks were collected during a systematical literature review. These literature review focuses on models and frameworks describing the processes and/or components as well as their connections to theoretically describe the generation and the functionality of interactive data visualization systems. The resulting models and frameworks are sorted by their complexity (from low level to high level), their chronological order and are grouped by three types:

Visualization Pipeline Based Models: The models and frameworks included in this group are based on the 'Information visualization Design Space' by Card and Mackinlay (1997) or the 'Information Visualization Reference Model' by Card et al. (1999). These models are describing the process from the input data across the different transformation states to the visualization which is presented to the user and its interaction abilities.

Formal / Mathematical Models: These models and frameworks are describing the visualization process based on a mathematical scheme to generate a bird's eye view on the functionality (e.g., transformations, interactions) like the models by Van Wijk (2005) or Jankun-Kelly et al. (2007).

Other Model and Framework Designs: This category contains all models which were not able to be included into the former two categories (e.g., The knowledge discovery process by Han et al. (2003) or the VA Process by Keim et al. (2008)).

Visualization Pipeline Based Models

The six models and frameworks included in this group are based on the "*Information visualization Design Space*" (Card and Mackinlay, 1997) or the "*Information Visualization Reference Model*" by Card et al. (1999) describing the way from the input data across the different transformation states to the visualization presented to the user and its interaction abilities. As Extensions to this basis, authors of later publications added data states (e.g., (Chi, 2000)) or events (e.g., (Tominski, 2011)).

The Structure of the Information Visualization Design Space

Variable	D	F	D'	X	Y	Z	T	R	--	[]	CP

Figure 2.2: **InfoVis Design Space** – Illustration of the design space which is divided into three major parts, which are thick black framed (from left to right): 1) Data; 2) Automatic Processing; and 3) Controlled Processing. (Data: D := original dataset, D' := selected dataset, F := filter or recording function; Automatic Processing: (X, Y, Z) := 3D space, T := time, R := retinal properties e.g., color, size and shape, $--$:= connection properties, $[]$:= enclosure properties; Collected Processing: CP := controlled processing)

Card and Mackinlay (1997) provided an organized structure of information visualization literature and demonstrated it in relation to some examples in the paper. The result of this paper is a new framework for the design of new visualizations and augmenting existing designs. The analysis builds upon recent approaches to understand the parts of the information visualization design space.

According to Bertin (1999) visualizations have at least two different uses which should not be mixed up: The first one is for communicating information, whereby the communicator understands the transported information in advance. The second is for graphical processing, according to the use of manipulation and perception of graphical

objects to solve a problem and gain new insights. "*Graphics is the visual means of resolving logical problems*" (Bertin, 1999, p. 16).

In general, the basis of information visualization is data. For example, text can be used to compute document vectors. The major difference or distinction for data is the types of values. For this relation, there are three base types of data: 1) Nominal (these values are only equal or not equal to other values); 2) Ordered (follows a < relationship); and 3) Quantitative Values (able to perform arithmetic operations). Additionally, Card and Mackinlay (1997) also focused on the cardinality of the variables, because InfoVis allows the processing in regions of high cardinality. They also described subtypes of the variables and how they are transformed. As a short example, a dataset D which is the original dataset and a dataset D' which is a selection of the origin dataset and transformed by filters or recoding functions F. The result looks like this: $D \rightarrow F \rightarrow D'$. Basically visualizations are made of: 1) marks; 2) their graphical properties; and 3) elements requiring human controlled processing (Mackinlay, 1986). The visual processing of humans works on two different levels. First, the automatic processing which works on visual properties like color and position. It is highly parallel but limited in power. Second, there is the controlled processing which works for example on text. This one has powerful operators, but it's limited in its capacity. A basic visual presentation contains marks (e.g., points, lines, areas), a position in space (e.g., x, y for 2D, x, y ,z for 3D and especially t for time) and a set of retinal properties (e.g., color, size). They also added properties for connection and enclosure because visualizations are related to the following visual vocabulary by Card and Mackinlay (1997):

- Marks (point, line, area, surface, volume)
- Automatically processed graphical properties
  - Position (x,y,z,t)
  - Retinal encoding (color, size, shape, gray-level, orientation, texture)
  - Connections
  - Enclosure
- Collected processing graphical properties.

For a better understanding, Card and Mackinlay (1997) created a table which is divided into three major parts (see Figure 2.2) which were called: 1) Data; 3) Atomic Processing and 3) Controlled Processing. The used coding in the table is described in detail in (Card and Mackinlay, 1997). By the use of these distinctions, its possible to see the major types of visualizations. Generally, the presented analysis schema does not express all the important distinctions which could be made.

The Information Visualization Reference Model

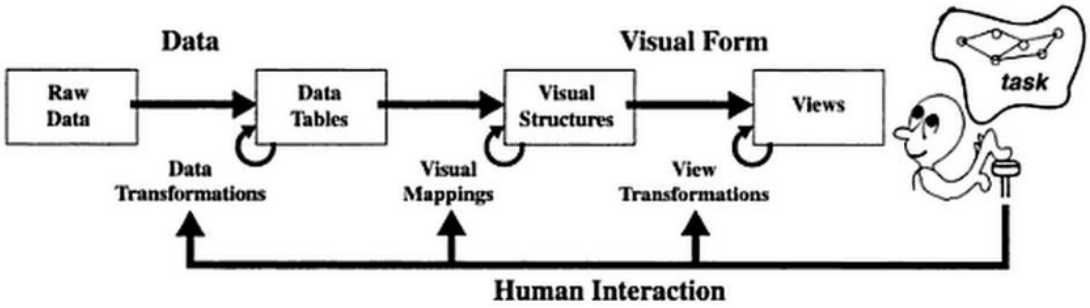


Figure 2.3: **InfoVis Reference Model** – Illustration of the InfoVis Reference Model describing the data flow from the left to the right including all transformation and mapping actions as well as the human interaction (Card et al., 1999).

Card et al. (1999) described visualizations as adjustable process from data to visual forms, whereby a reference model simplifies the discussion of information visualization systems. In the 'Information Visualization Reference Model', the raw data flows from the left to the right passing the transformation and mapping stages (which can be more than one each stage) influenced by the user/analyst to the view seen by the user.

In the first step (see Figure 2.3) the raw data is transformed into data tables whereby different types of data transformations are accomplished. Raw data are provided in many different forms, from spread sheets to text of novels. The usual strategy is, to transform these data into a more structured form (relations or a set of relations) which are easier to map to visual forms (mathematically this is a set of tuples). Additionally, three different basic types of data types are available: 1) nominal values (they are only $==$ or $!=$ to other values); 2) ordinal values (follows a $<$ relationship); and 3) quantitative values (able to perform arithmetic operations). This is done by a classification transformation which maps raw values to specified classes of values for the data table. Generally, the mapping of raw data into data tables leads to a loss or gain of information. Often, raw data contains missing or erroneous values which have to be addressed before the data can be visualized. Additionally, statistical operations can also gain additional information. Therefore, data tables often contain derived values or structures which are achieved by the use of the following four transformations:

1. "*Values* $\rightarrow$ *Derived Values*"
2. "*Structure* $\rightarrow$ *Derived Structure*"
3. "*Values* $\rightarrow$ *Derived Structure*"
4. "*Structure* $\rightarrow$ *Derived Values*" (Card et al., 1999)

An example for derived values are statistical operations like ‘mean’ and an example for derived structure is the ‘sorting of variables’.

Second, the prepared data tables are mapped by visual mappings to the visual structures, which are a spatial substrate with marks and properties to encode information. There are many possible ways to map the data tables into visual structures, but its important to preserve the data. A mapping is called expressive if all and only the data contained in the data table are represented in the visual structure. Preparing a good mapping is very difficult, because its easy for unwanted data to appear in the visual structure. This means that a good visual structure brings the data into an understandable form for the user and does not show relationships which are not contained in the data. It’s also the job of information visualization systems to setup and use visual representations of the data which follow and take advantage of the properties of human perception. The most fundamental aspect of a visual structure is space, because its perceptually dominant. Spatial positioning is such an important visual coding that the first decision has to be the spatial encoding of a dominant variable which expands the others. The next important elements are the marks which occur in space based on four elementary types: Points (0 dimensional); Lines (1 dimensional); Areas (2 dimensional); and Volumes (3 dimensional).

Third, the visual form is transformed by applying some view transformations to the view which are seen by the user. These views are “*interactively modified to turn static presentation into visualizations by establishing graphical parameters to create Views of Visual Structures*” Card et al. (1999). InfoVis’ are existing in space-time, were view transformations extract more information from the visualization than it would be possible with a statical data representation. Generally, there are three different types of view transformations available: 1) Location probes; 2) Viewport controls; and 3) Distortion. Location probes are view transformations which open up new insights onto the represented data by the use of location in the visual structure (e.g., details-on-demand, brushing & linking). Viewport controls are using affine transformations to change the point of view onto the represented data (e.g., zoom, pan, clip). Additionally, distortion transformations modify the visual structure to create focus & context views. This way, overview & detail are combined in one view (e.g., hyperbolic tree (Lamping and Rao, 1994)).

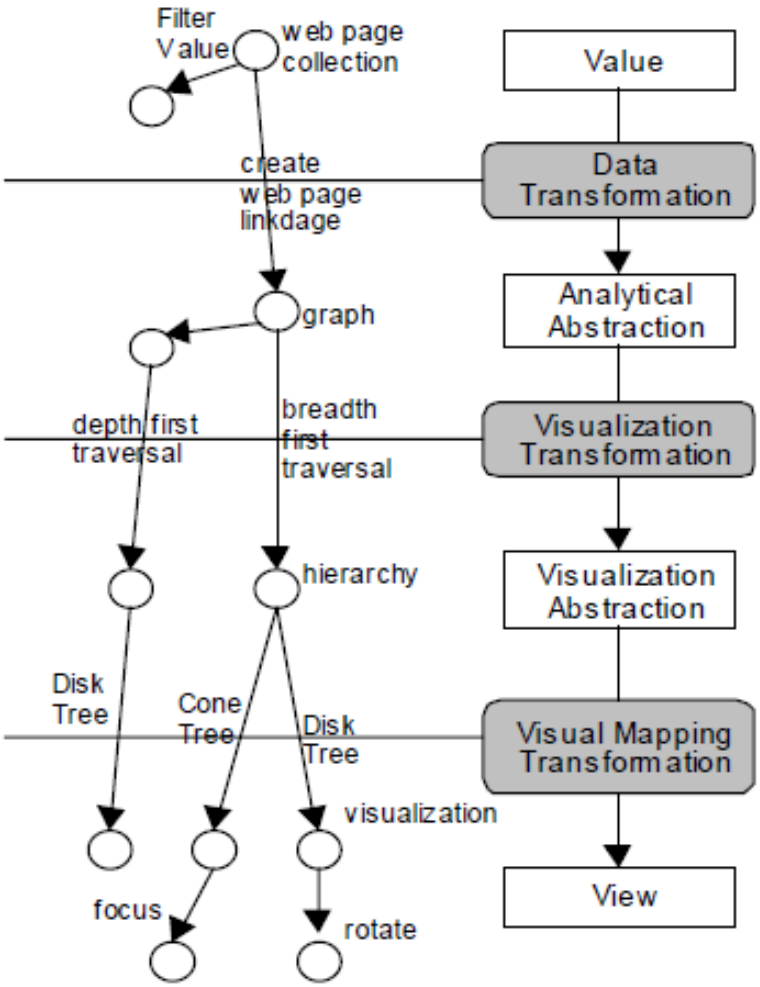


Figure 2.4: **Data State Model** – Illustration of an example of the 'Data State Model', breaking down each technique into: 1) Data Stages; 2) Data Transformation; and 3) Within Stage Operators. The 'Data Flow Model' is broken down into four data stages: 1) Value; 2) Analytical Abstraction; 3) Visualization Abstraction; and 4) View (Chi, 2002). Generally, the InfoVis Pipeline (right part of the illustration) by Chi and Riedl (1998) is based on the InfoVis Reference Model by Card et al. (1999).

Chi and Riedl (1998) described that the research in InfoVis had made great process. Therefore, many researchers have developed graphic representation semiologies to gain a better understanding of the visualization design space. In their paper, they investigated recent work on InfoVis frameworks. Grounded on their research, they developed a new

operator and user interaction based framework. The resulting 'State Model' unifies the data analysis process and the relationship between the view and the values, characterizing the 'non-interactive and interactive operators' in a visualization system.

As conceptual grounding for their framework, they used the 'Information Visualization Pipeline' by Card et al. (1999) and extended this to classify operators (see Figure 2.4 the right part). In general, they defined two categories of operators. In the first category, 'functional versus operational', on the one hand, functional operators are different for different data sets like the filtering operator. On the other hand, operational operators are similar across applications like rotation, scaling, translation and so on. The second category contains 'view versus values' operators. A value operator for example, changes the data source by processes like adding or deleting subsets of the data, filter or modifying the raw data. In contrast, a view operator changes the visualization content only like by zooming, rotation, translation or flipping. It is important to know that an operator which is closer to the view, takes more on view properties. Similar, an operator which is closer to the value pipelines end, relates more to value operation properties. Generally, the framework contains the following operators whereby their naming slightly relates to their position in the model: Data Stage Operators (DSO); Data Transformation Operators (DTO); Analytical Abstraction Stage Operators (AASO); Visualization Transformation Operators (VTO); Visualization Abstraction Stage Operators (VASO); Visual Mapping Transformation Operators (VMTO); and View Stage Operators (VSO), which were all described in the paper in detail.

From the view of the 'Visualization State Model for Operators', there are the data (values) on one end of the pipeline and the visualization(s) (view) are on the other end of the pipeline. In contrast to the 'Information Visualization Pipeline' which breaks down by more than one data source or visualization, the 'State Model' was extended with a network to allow the handling of more than one data sources and visualizations. Additionally, the developed 'State Model' uses nodes to represent a certain data state in contrast to the 'Information Visualization Pipeline'. The operators are represented by edges whereby each edge transforms the data from one state into another. In some cases, the model looks similar to a scientific visualization 'Data Flow Model', because in a flow model, the data state is not exactly represented. This 'State Model' has advantages for visualization tasks, because it makes each result in between understandable for the user, which helps for planning further operations. Moreover, Chi (2000) shows a new way to taxonomies for information visualizations by the use of the 'Data State Model', supporting researchers to understand the design space and how to apply information visualization techniques more broadly.

Chi (2002) described visualization as "*series of transformations that transcribes data values into graphically views*" (Chi, 2002). Generally, there are two models: 'The Data Flow Model' and the 'Data State Model', and both describes the relationship between view and values. The 'Data Flow Model' is well established in the scientific

visualization community and its capability and expressiveness is well understood. The main concern of this paper is the comparison of the 'Data Flow Model' and the 'Data State Model' (see Figure 2.4) and to show that the 'Data State Model' is equally expressive than the 'Data Flow Model' by modeling of the same visualizations. However, in a 'Data Flow Model', modules are created which corresponds to process nodes. Additionally, data transferring mechanisms are created to connect those modules. In contrast, a 'Data State Model' creates data stores corresponding to the data states and data processing parts which are created to connect these data states. Based on these insights, its apparent that each model has its strengths and weaknesses. The 'State Model' represents the data state better than the flow model but the representation of the visualization process is better done with the 'Data Flow Model'.

Event-Based Concepts for User Driven Visualization

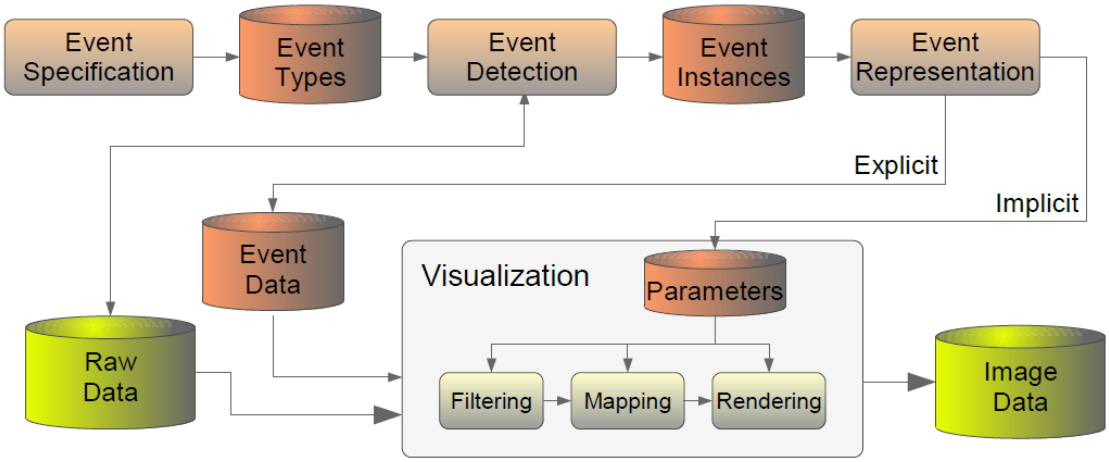


Figure 2.5: **Event-Based Visualization** – Illustration of the 'Event-Based Visualization' by Tominski (2011) consists of three stages: 1) Event Specification; 2) Event Declaration; and 3) Event Representation.

Tominski (2011) described that data visualization will become increasingly important for the exploration and the analysis of big data. Unfortunately, the needs of the users are not considered enough. Tominski (2011) moves the user into focus. Therefore, he introduced a new concept for 'Event-Based Visualization' which combines event-based methods and visualization technologies.

Former approaches were always created to solve only one problem and could not be used for others (Tominski, 2011). In this paper, he introduced a novel approach for event based visualizations which consists of three fundamental stages: 1) the user has the ability to specify their interests; 2) during the visualization, consistencies in the data are searched; 3) thereby, it is possible to automatically adjust the visualization on the

consistencies found. This way, it is possible to generate better visualizations which are adapted to the needs of the user. This new general model allows the use of many different visualizations. The general model is occupied with specific data driven events which are concentrated on relational data of today's visualizations. Additionally, he showed, how the developed methods and concepts can be used in an interactive 'Event-Based Visualization' framework, which is based on spatial and temporal data (Aigner et al., 2011; Andrienko et al., 2013) for event visualization.

Basically, the idea of 'Event-Based Visualization' consists of three stages (see Figure 2.5). In the first stage, the event specification has to generate event types which are interesting as visualization for the user. This task has to bring the user interests into a computer understandable form which has to be as easy as possible to formulate by the user. Additionally, tools have to be implemented which enables the formulation for users with different experiences. The second step is the event declaration: (1) The intersection between the user's interests and the data will be specified. This detection has to be as general as possible to be applicable for many event types. The basic task is to consider the enclosed conditions of event types. Therefore, they substituted the variables in the event formulas by concrete units of data (tuple attributes or sequences of tuple). (2) Predicates, functions and logical connections will be analyzed, thus the complete event formula returns true or false. (3) The last stage is the event representation. The goal of this stage is the integration of recognized event instances into visual representations (representing the users interests). The event representation has a major bearing on the extent to which 'Event-Based Visualization' closes the Worldview Gap. Theis gap is described by Amar and Stasko (2005) as "*the gap between what is being shown in a visual representation and what actually needs to be shown to intuitively draw representational conclusions*"(Amar and Stasko, 2005). Therefore, it has to be communicated: 1) what has been found; 2) highlight the events among the remaining data and 3) show why the events is interesting (Tominski, 2011).

Formal / Mathematical Models

These models and frameworks are describing the visualization process based on mathematical scheme to generate a bird's eye view on the functionality (e.g., transformations, interactions).

Externalizing Abstract Mathematical Models

Tweedie et al. (1996) described that their 'Interactive Visualization Artifacts' (IVAs) display data by generated mathematical models using interactively linked simple graphs. This way, the user has the ability to gain new insights by interactive data exploration. From the view of the engineering context, these insights can aid the design. For a better understanding, this paper describes two different engineering designs: 1) the influence explorer and 2) the projection matrix.

The authors described that the presented IVAs approach differs from existing models because they did not focus on the visualization of raw data. They focus on data which are recalculated or generated by mathematical models. Additionally, they excluded data which maps comfortably onto natural representations like 3D volumetric models. Instead they focus on mathematical models without an obvious representation. Therefore, they described how novel representations can be created by interactively linked simple graphs in several ways. This works for similar types of representations as well as for different representation types.

A typical task of a mathematical model is engineering design like for a light bulb whereby a model can be formulated around their parameters. Therefore, the model is a set of equations and each relating the performance of a number of parameters. In other words, for the design of a light bulb, the designer has to keep a specification in mind (e.g., lifetime, brightness). Based on a mathematical model, the design process can be immensely simplified to find the relationship between parameters and performance.

The Influence Explorer: "*Precalculation forms the backbone of the Influence Explorer. Once the data has been precalculated (as described earlier), it provides an exploration database on which to start an investigation*" (Tweedie et al., 1996). For the formulation of an external presentation of the task to be solved, the influence explorer has to allow the user to gradually build up a real similar picture of the problem which introduces the complexity in stage.

The Prospection Matrix: This matrix provides an alternative perspective of the model, whereby a set of scatter plots will be arranged in a matrix. Therefore, each scatter plot is filled with a matrix of small colored squares.

From the view of the evaluation studies, Tweedie et al. (1996) described that it is difficult to judge what a user find intuitive and how IVAs will support the user during such

practical tasks. Therefore, they performed a set of evaluations during different stages of the IVAs development. Basically, the authors stated that they learned three basically important lessons: 1) "*Maximize the directness of the interactivity*"; 2) "*Seek out the most crucial information and then represent it appropriately and simply*"; 3) "*There is a trade-off between the amount of information simplicity and accuracy*" (Tweedie et al., 1996).

The Simple Visualization Model

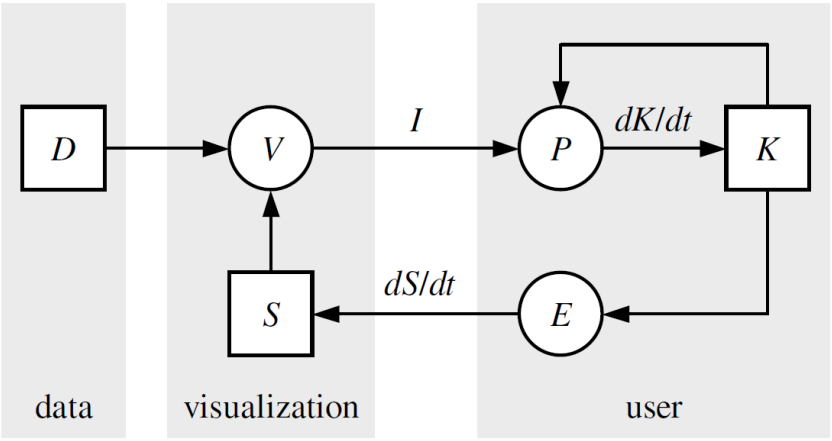


Figure 2.6: **Simple Visualization Model** – Illustration of the 'Simple Visualization Model' by Van Wijk (2005) which consists of three areas: 1) Data; 2) Visualization and 3) User. The boxes are related to containers and circles are related to processes transforming input into output.

Van Wijk (2005) stated that visualization gets mature in the relation to solved problems and new found directions. It is important to make good choices and to understand the purpose and meanings of visualizations. Thus, it would be nice to find out "*what a good visualization is*" (Van Wijk, 2005). Based on visualization systems, it is possible for analysts, engineers and the lay audience to gain new insights during the use of visualizations and during the interactive data exploration. In this way, they can detect features or patterns hidden in the data.

Research contains also some issues in relation to new visualization techniques. Many new visualization techniques are not used in real-world situations and "*many research results are nowadays considered as incrementally by reviewers*" (Van Wijk, 2005). So Van Wijk (2005) aim is the detection of overall patterns and to find a way to generalize visualization in relation to be efficient and effective.

Therefore, Van Wijk (2005) presents a generic visualization model discusses the cost and gains. The basic model is represented in Figure 2.6, whereby boxes are related

to containers and circles are related to processes transforming input into output. This model should not describe different visualization techniques, in contrast, it should be used for the description of the operation context for visualization. In general, the model is based on mathematically described operations for the transformations between the containers based on the processes. Therefore, Van Wijk (2005) defined three areas: 1) data; 2) visualization and 3) user. Each area contains different containers (boxes) and/or processes (circles). In the data area, the data container D is situated (see Figure 2.6), in the visualization area, the central visualization process V and the specification container S are included and in the user area, the processes for properties for perception and cognition P, and for interactive exploration E are included, as well as the container for the knowledge K. All the interactions between the containers and the processes are described based on formally defined transformations for a better understanding.

In relation to the knowledge contained in the user area,Van Wijk (2005) differentiated between valuable and negative knowledge. From the perspective of the valuable knowledge, the traditional aim of visualization is to gain new insights based on the data to analyze. By analyzing the data, users are able to see things they were not aware of, and based on the new gained insights, they have the ability to formulate new questions, hypotheses and data models. In the model,Van Wijk (2005) uses the term knowledge, but this contains some limitations based on a strange paradox in the visualization paradigm. In relation to the data exploration of data we don't know, we try to make pictures of interesting stuff to get new insights. But without any knowledge about the contained features and patterns, we cannot determine if we were successful or not.

The perspective of knowledge opens also another problem because visualizations can be wrong and misleading which can be introduced as negative knowledge. Therefore, Tufte (2001) introduced the lie-factor for example, which measures the difference of the visualization size in relation to the data size of an element. To explain this effect, Van Wijk (2005) gave an example were he visualizes the waves produced by ships but the data were the result of a simulation. In this context, it can be possible that the results of the simulations are wrong and as effect of this, the depending visualization shows also wrong results.

A Model and Framework for Visualization Exploration

Jankun-Kelly et al. (2007) described that a visualization technique which does not store anything is wasted and a system which does not tell the user where he/she is, where he/she have been and where he/she can go is inefficient. To overcome such limitations, these problems/issues have to be addressed before visualization systems can become effective for large scale deployments.

To do so, the authors used a three-part approach to utilize and capture the information within a visualization approach. At first, they used a formal model of the visualization approach to capture the aspects of the data exploration. The outcome of this step

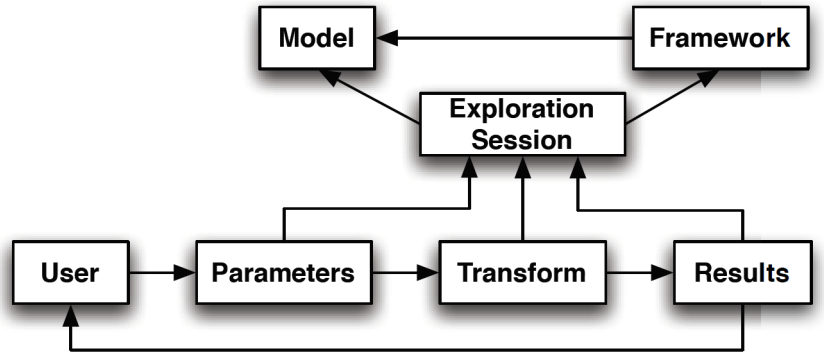


Figure 2.7: **P-Set Model** – Illustration of the 'P-Set Model' of visualization exploration by Jankun-Kelly et al. (2007). During the data exploration, the user manipulates several parameters which can be applied to a transformation generating a result. This result returns feedback to the user for further exploration.

is: "*what results were generated, how they were generated and how they were used to generate new rules*" (Jankun-Kelly et al., 2007). In the second part, the connections of a visualization session will be documented by the model for later/further analysis or dissemination to collaborators. Finally, the instances of the model will be managed by a software framework described in previous work (Jankun-Kelly et al., 2002).

Jankun-Kelly et al. (2007) noted that it is important to know that a visualization process model alone is not enough to describe the user's knowledge before and after the use of the visualization system. For the capturing of the generated knowledge and to gain insights, a meta data based model for the visualization process has also to be implemented.

The 'P-Set Model' of visualization exploration contains four major elements (see Figure 2.7): visualization transform, visualization parameter, visualization result and derivations. The visualization transforms are used to create results and the visualization parameters are generated by the performed user interactions. Additionally, the results are created by applying a set of parameter values (p-sets) to the visual transforms and the derivations describes how the new generated results are related to previous results. It is important to note that the model does not specify the visualization transform components and also does not describe the form of the parameter and result values because these elements are not the scope of this work. The model's derivation described the "*relationship between results, p-sets and other results and p-sets*" Jankun-Kelly et al. (2007). Generally, there are four derivation components: 1) time stamp; 2) parameter derivations; 3) p-set derivations; and 4) generated results. Additionally, three-part derivation calculus can be used to write derivations (Jankun-Kelly et al., 2007, p. 3).

The indication of a parameter derivation bounds the input parameter $s_i[n_i]$ from a p-set s_i which is manipulated by the user to create new output parameters p_k; n_j

represents the p-set used parameter type. By the use of a p-set derivation, the parameter value replacement will be described to change from an input p-set s_{in} to an output p-set s_{out}. And finally, the result derivation lists the result r created by the interactions of the user. To share visualizations which are captured by the p-set model, a common data format is required which is extensible to different visualization applications. These goals can be accomplished by the use of an XML structure.

Defining and Applying Knowledge Conversation Processes to VA Systems

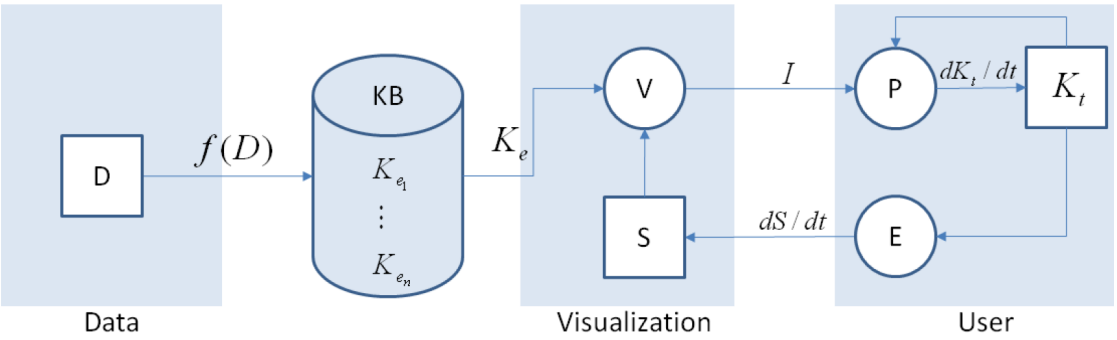


Figure 2.8: **Cognitive Conversation Process Model** – Illustration of the model by Wang et al. (2009) describing the integration of explicit knowledge into the visualization process.

Wang et al. (2009) described in their paper that the integration of knowledge into the visualization process for solving analytical tasks is a fast growing area. By integrating the experts knowledge into the visualization process, "*the experts are more capable of performing complex analytical processes*" (Wang et al., 2009). Generally, they differed between two types of knowledge: 1) tacit knowledge, which is the personal and context specific knowledge; 2) explicit knowledge, which is represented in a computerized form. Based on this definition, Wang et al. (2009) defined four 'knowledge conversation processes' for knowledge-assisted visualization (KAV) which are: 1) internalization; 2) externalization; 3) collaboration and 4) combination.

For the preparation of their model, they used as conceptual grounding the 'Simple Visualization Model' by Van Wijk (2005) and expanded it with the integration of a knowledge database (see Figure 2.8). They described that the explicit knowledge which is extracted from the data is "*represented as a visualization, which is received both perceptually and cognitively by the user via an image*" (Wang et al., 2009). Thus, they expressed tacit knowledge and explicit knowledge as a set of equations (see Equation 2.1):

$$K_e = f(D); I(t) = V(K_e, S, t); fracdK_tdt = P(I, K_t) \tag{2.1}$$

Whereby they see the explicit knowledge K_e as an extension to the data $f(D)$. By the use of the explicit knowledge K_e, the specification S, the image I at the time point t, $I(t)$, is generated by the visualization V process of the system (Wang et al., 2009). As previously mentioned, Wang et al. (2009) provided four knowledge conversation processes to the former presented knowledge definitions:

Internalization: This part is described as a "*cognitive process of acquiring skills and knowledge*" (Wang et al., 2009). In relation to KAV, this is the process where the analysts are supported by explicit knowledge while understanding and gaining insights of the visualization and transforming the explicit knowledge into tacit knowledge (Wang et al., 2009).

Externalization: Defines the process where the analyst transforms his tacit knowledge into explicit knowledge by gaining insights, concepts and more (Wang et al., 2009).

Collaboration: This relates to the process of learning from others and contains the sharing of knowledge as well as the learning and the building of the concurrence by the use of computers (Wang et al., 2009).

Combination: In general, explicit knowledge exists in many different forms (e.g., books, research papers, social networks), the combination of these different bodies is mentioned as important by the authors (Wang et al., 2009).

Additionally, the authors also presented the implementation of a knowledge-assisted VA bridge management system consisting of two major components: 1) a visualization interface for interactive data exploration and 2) a knowledge management structure for domain concept and knowledge management.

Other Model and Framework Designs

This category contains all models which were not able to be included into the former two categories.

A Visualization Model of Interactive Knowledge Discovery Systems

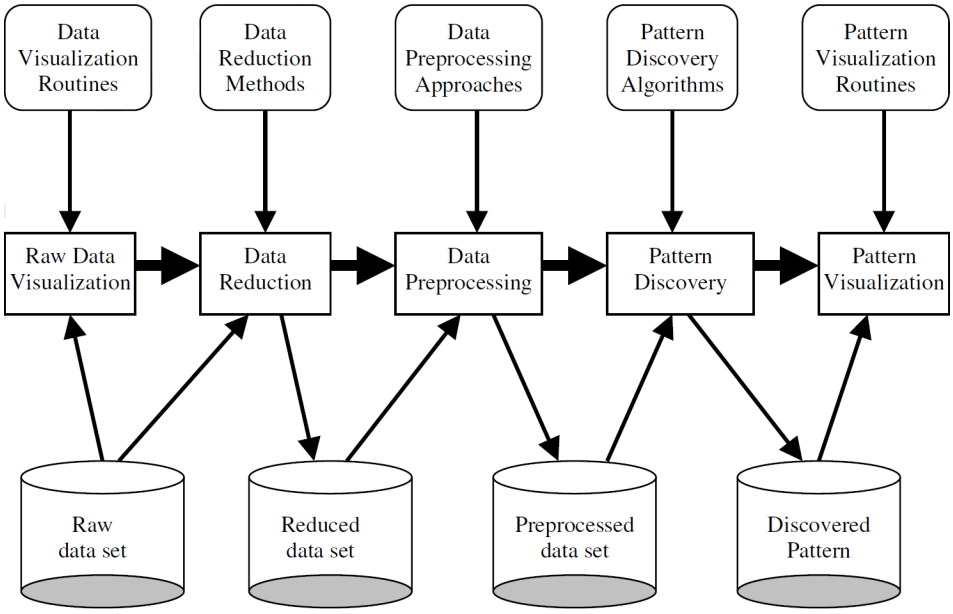


Figure 2.9: **RuleViz Model** – Illustration of the 'RuleViz Model' and its five component: 1) raw data visualization; 2) data reduction; 3) data preprocessing; 4) pattern discovery 5) and pattern visualization (Han et al., 2003).

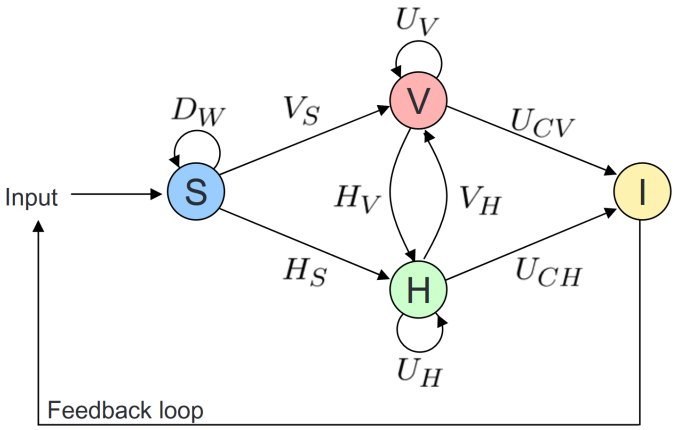
Han et al. (2003) introduced in their paper a visualization model called 'RuleVis' consisting of five components for knowledge discovery and data mining which are: 1) data preparation and visualization; 2) interactive data reduction; 3) data preprocessing; 4) pattern discovery; and 5) pattern visualization. Based on this, they discuss three implementation paradigms: the image-based paradigm, the algorithm-embedded paradigm, and the interaction-driven paradigm. Mostly, algorithm-based approaches are used in the areas of artificial intelligence, information retrieval, databases, statistics and more. In contrast, visualization-based approaches are used in the areas of graphics, scientific and InfoVis, and for visualization techniques to summarize or extract complex visualization results rather than mathematical, logical, or textual results.

KDD is a process of searching for knowledge which is represented as relationships and patterns of large data sets. This also includes the employment of this knowledge to solve problems or interpret phenomena. The KDD process is generally interactive

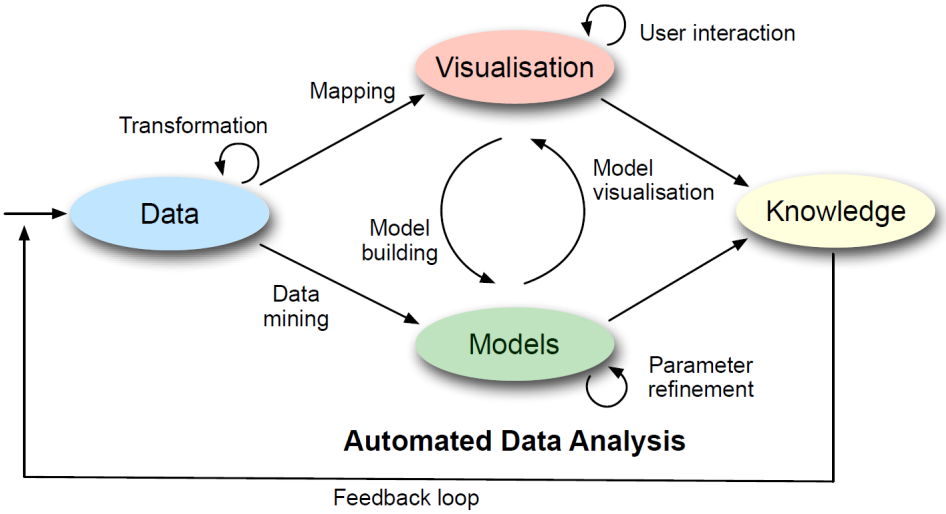
and intuitive including many steps of decisions which are to be made by the user(s). Existing approaches can be characterized as algorithm-based or visualization-based. The algorithm-based approach includes: rule induction, concept learning, association mining, decision tree construction, neural networks and many more. In contrast the visualization-based approach specifies the relationship between data items or variables by means of visualization metaphors (Han et al., 2003). In general, the basic tasks of knowledge discovery are: data classification; data clustering; regression and prediction; association and correlation; sequence discovery; summarization and generalization; characterization and discrimination; temporal or time series analysis; and path traversal patterns extraction. It's important to note that the KDD process involves many different steps to fulfill all these tasks. Additionally, there is no exact definition available how many steps and which steps has to be included in the KDD process. A common process has to accept raw data input, select relevant data items, reduce, process, enrich the data set, transform the data, find patterns, interpret and discover the results (see also Figure 1 in (Han et al., 2003).

The implemented 'RuleViz' model contains five components, including raw data preparation and visualization, data reduction, data preprocessing, pattern discovery, and pattern visualization (all these steps are visualized in Figure 2.9. The introduced 'AViz' framework is implemented along the image-based paradigm. It plots the raw data into an image to present it to the user and it provides interaction tools to help the user selecting interesting data areas from the data set to reduce the data. Additionally, it contains a pattern visualization component to help the user to understand and interpret the patterns discovered. Therefore, this framework consists of five major parts: 1) data preparation and visualization $\rightarrow$ to specify the raw data file and its attributes with specific format, numeric attributes X and Y and the quantitative attribute Z which constitutes the consequences of the associated rules to be discovered; 2) interactive data reduction $\rightarrow$ business data sets contain millions of data items. For the discovery of associations between attributes of the huge data set, the raw data must be read from disk or as input from other systems, which is stored in memory; 3) discretization of numerical attributes with visualization $\rightarrow$ after the performed data reductions, the new generated dataset has to be re-drown in the whole available window; 4) discovering associated rules $\rightarrow$ for each distinct value of Z, the authors attempt to find an optimized region on the $X \times Y$ plan, construct an associated rule; 5) visualizing associated rules $\rightarrow$ finally the discovered rules will be visualized by a very simple visualization schema because every associated rule corresponds to an optimal region on the plane (Han et al., 2003).

Han et al. (2003) also presented two example experiments based on the 'AViz' framework. These experiments demonstrated that the framework is useful for the users of the KDD process. Therefore, they asked ten people to evaluate the understanding of the system, whereby they only focused the evaluation on the KDD process and its results.



(a) VA process by Keim et al. (2008).



(b) VA process by Keim et al. (2010a)

Figure 2.10: **Visual Analytics Process** – Illustration of the 'Visual Analytics Process' defined by Keim et al. (2008) in Figure 2.10a and Keim et al. (2010a) in Figure 2.10b characterizing the interaction between the four major elements of the process in combination with a feedback loop. The elements contained in Figure 2.10a are: S := source, D_W := data pre-processing, V_S := functions visualizing the data, V := visualization, U_V := user interaction on visualization, U_{CV} := insights from visualization, I := insight, U_{CH} := insights from hypothesis, H := hypothesis, U_H := user interaction on hypothesis, H_S := functions generating hypothesis from data, H_V := hypothesis from visualization, V_H := functions visualizing hypothesis.

Keim et al. (2008) and Keim et al. (2010a) provided schematic overviews of the VA process. The structure of this process "*combines automatic visual analytics methods with tight coupling through human interaction in order to gain knowledge from data*" (Keim et al., 2010a). Based on the two models presented in Figures 2.10a and 2.10b the process contains in general four major elements with different labels but the same meaning: 1) data or data set := S; 2) visualization := V; 3) model or hypothesis := H and 4) knowledge or insights := I, connected by different functions.

Mostly, data sources need to be integrated before applying the visualization V or automated analysis methods H. To do so, as pre-processing steps, transformations of the data D_W are needed to make them usable for further exploration. After this, the transformed data are used to generate the models or hypothesis H by data mining methods (hypothesis generation H_S) and to prepare the visualization V by applying the visual mapping V_S. The visualization V and the models H are connected by functions for model building (hypothesis from vis. H_V) and model visualization (vis. from hypothesis V_H). Generally, user interactions can be performed or are integrated in the visualization like zooming or selecting (U_V), and in the models area for parameter refinement or hypothesis generation (U_H). Based on this iterative and interactive process, the analyst gains new insights I or knowledge based on the results of the visualization V and the performed automated methods H. This interactive process is given by the feedback loop whereby the analyst refines the input data S to validate different hypothesis/models (Keim et al., 2010a, 2008).

Domain Knowledge in the Visual Analytics Process

Lammarsch et al. (2011) described that the combination of automated analysis methods with interactive visualizations is "*a necessary step*". Thus, most of the currently available models are human-focused which makes it challenging to use such models as template for the design and development of VA systems.

To overcome this problem, Lammarsch et al. (2011) developed a process description grounded on a combination of Keim et al. (2008) and Bertini and Lalanne (2009). The resulting process based model (see Figure 2.11) offers the user the ability to interact with all elements included in the gray area as well as all connections leading inside and outside (Lammarsch et al., 2011). The input data are real world values which are collected by one or more prior processes (a dataset). In addition to Keim et al. (2008), domain knowledge is included into the system, consisting of hypotheses and models from prior processes. The interactive visual interface is the center of the model and combines the humans' hypothesis and the models, based on automated analysis to gain new insights. Additionally, the interactive visual interface gains the ability to transfer data from the hypothesis to the automated analysis methods by user interaction. Lammarsch et al. (2011) stated that a result in the models, which cannot be validated is called a hypothesis. Generally, a hypothesis is only correct for the current state of the data, which are

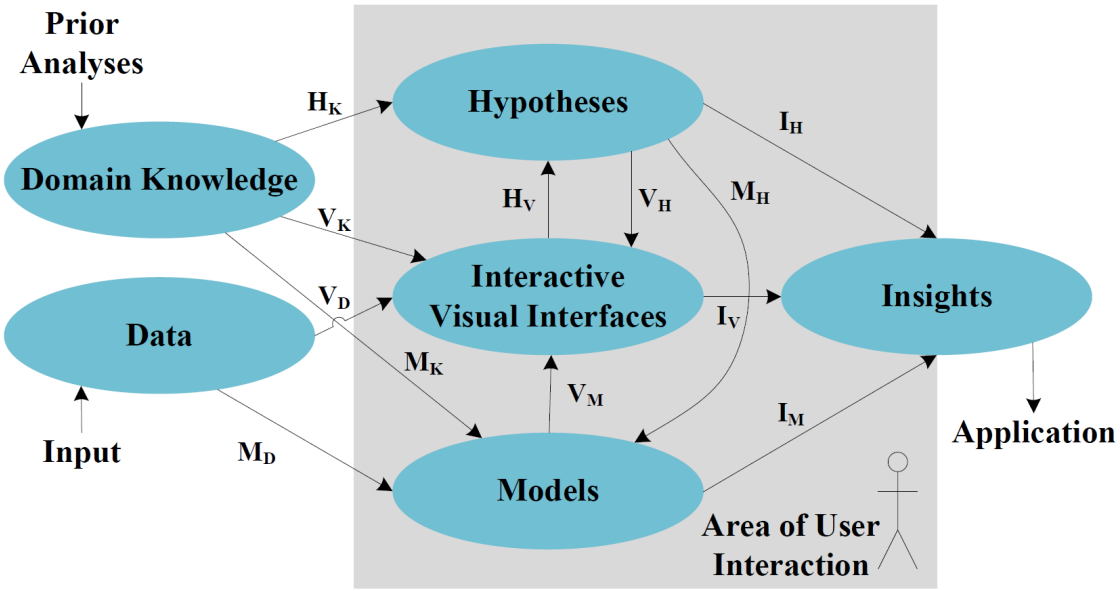


Figure 2.11: **Domain Knowledge in the VA Process** – Illustration of the 'VA Process' by Lammarsch et al. (2011) using domain knowledge for user support. (H_K := hypotheses from domain knowledge, V_K := directly visualize domain knowledge, M_K := models from domain knowledge, V_D := data visualization, M_D := generate models from data, H_V := hypotheses based on visualization, V_M := visualize models, V_H := visualize hypotheses, M_H := validate hypotheses for models, I_H := insights from hypotheses, I_M := insights from models, and I_V := insights from visualizations)

represented in the system (same for automated analysis methods). The benefit of VA is the ability to validate hypothesis by using interactive data exploration methods.

Based on this novel model (see Figure 2.11), Lammarsch et al. (2011) gains the ability to handle time-oriented data in relation to the structure of time (e.g., year, quarter, month) and to the VA process model. The application of the new model was demonstrated on a prior developed tool in detail.

A Reference Model for Adaptive Visualization Systems

Nazemi et al. (2011) stated that one key issue of information visualization and adaptive user interfaces is the information overload. Both areas have well performing algorithms, methods and applications but a real merge of these has not been placed yet. In this paper, the authors presented a reference model for adaptive visualization systems which allows the adaption of the visualization types as well as the adaption of the visualization parameters. Additionally, they derived a framework for adaptive visualizations of semantic data.

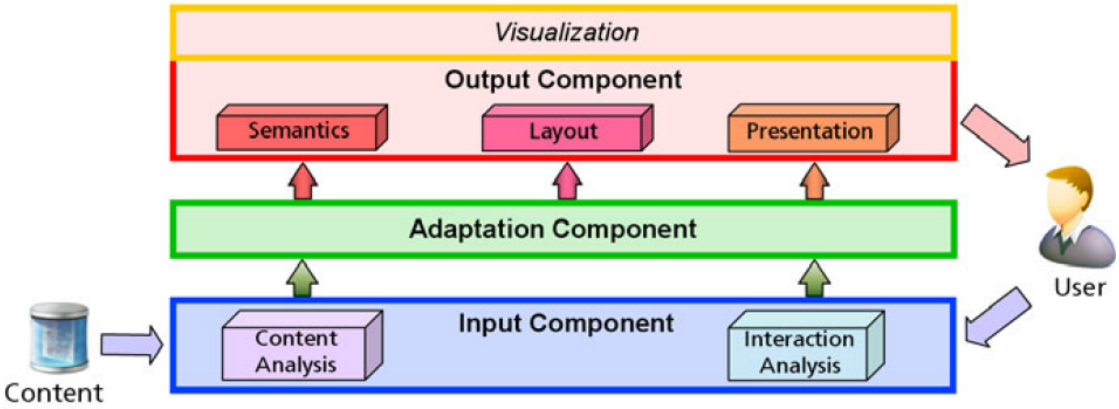


Figure 2.12: **Model for Adaptive Visualization Systems** – Illustration of the 'Model for Adaptive Visualization Systems' representing the content on the left and the user on the right. Additionally, in the center, the three basic components are represented which are: 1) input; 2) adaption and 3) output (Nazemi et al., 2011).

From the view if InfoVis, Card et al. (1999) presented a general reference model which contains different phases that are needed for the mapping and the transforming of raw data into a visual representation. According to this reference model, different extensions are available today (Keim et al., 2008). Aaltonen and Lehikoinen (2005) redefined the data table and visual structure by an extension with a visual adaption of the original reference model by Card et al. (1999). To visualize information on different ways, the InfoVis community provides a large set of frameworks (e.g., Prefuse by Heer et al. (2005)) based on the reference model by Card et al. (1999)). There is only one framework called 'e-Vis' by Tominski (2011) which contains the aspects of adaption.

In general, The reference 'Model for Adaptive Visualization Systems' (MAVS) (see Figure 2.12) consists of three basic components which are: 1) input; 2) adaption and 3) output. Additionally, this model can be configured by parameters which are defined by certain impact and influence factors like knowledge or the users behavior or the underlying data structure or amount. The input component receives two different impact factors for the visualization adaption and parametrization which are the user interaction and the data. The interaction analysis appears as part of interaction events which are captured with different contextual information about the interaction type, the layout method and the content. Therefore, an interaction event has the form of a triple `<type, layout, data>`. From the view of the content analysis, semantically annotated data can contain geographic or time dependent information, to identify visualizations which are able to represent the prepared data in an appropriate way. Therefore, the data will be analyzed to extract relevant attributes of them.

The central element of the presented model in this paper is the adaption component. This component has interfaces to all components and modules in the model to control

them. It collects and combines information from the input component and transforms it for the output module. The whole process is based on a (static) visualization capability model and a (dynamic) user preferences model. The adaption of the output is performed in two steps. First, in relation to the visualization capability model, the user's preferences in relation to the visualization, content and activities are selected. Second, these visualizations are ordered to the preferences of the user and the parameter for each output component module. The output component contains the last three elements of the model: 1) semantic; 2) layout and 3) presentation. Generally, visualizations can be described around the following three questions: What will be displayed; where is it displayed and how is it displayed? The semantics module defines which data will be displayed depending on information of the data, its structure and their amount. By the use of the layout model, it will be defined where and how the data will be visualized, depending on user preferences, different graphical metaphors and different layout algorithms. At the end, the presentation module specifies how the data will be visually represented to the user by setting texture, color, size or shape. Additionally, the visualizations are separated into three different groups. Group one are general visualized components (GVCs) which are abstracted visualizations from the user interface. The second group is called semantics visualization components (SVCs) which visualizes the structure of semantic data and provides the possibility to interact. The third and last group contains the content visualization components (CVCs) which represents content which is referenced by the semantics (e.g., pictures or HTML views).

Overall, this model allows the goal-oriented adaption of specific parts of the system with appropriate level of detail in combination with the modular structure of the system's conceptual architecture.

Knowledge Generation Model for Visual Analytics

Sacha et al. (2014) described that VA has great success by helping domain experts during the exploration of large and complex data sets. This is possible by the combination of "*effective delegation of perceptive skills, cognitive reasoning and domain knowledge*" by the human and "*computing and data storage capability on the machine side*", which are effectively intertwined by VA (Sacha et al., 2014).

The knowledge generation model for VA by Sacha et al. (2014) is generally grounded on the VA process by Keim et al. (2008) and Keim et al. (2010a) and is split into two parts: 1) the computer system, which includes the visualization, the model and the data and 2) the human side which is gaining insights during the data analysis by a knowledge generation process. Sacha et al. (2014) stated that a clear separation between the computer and the human is not possible because the computer misses the creativity of the human and the human misses the efficiency to deal with a vast amount of data. VA combines the benefits of the computer and the human for data exploration and gaining new insights.

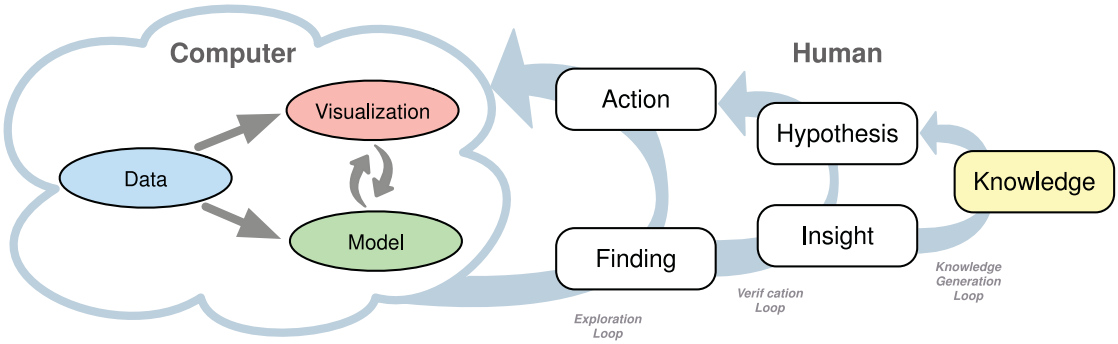


Figure 2.13: **Knowledge Generation Model** – Illustration of the 'Knowledge Generation Model' for VA combining human and machine concepts. On the left side there are the VA system components and mappings, and on the right side there are the human concepts embedded into a three loop framework (Sacha et al., 2014).

The knowledge generation process of the analyst is represented on the right side of Figure 2.13 and consists of three loops: 1) exploration; 2) verification and 3) knowledge generation. The process of knowledge generation is very complex including numerous reasoning activities, hypothesis generation and testing, combined with direct system interactions (Sacha et al., 2014).

Exploration Loop: This loop describes the direct interactions with the VA system and may effect on all the components of the system (such as the data selections, preprocessings, visual mappings, navigation, and adaptions of the underlying computations). Each interaction causes an observation, which has to be made by the analyst in order to spot a finding (Sacha et al., 2014).

Verification Loop: The analyst has to understand and interpret new patterns by the use of domain knowledge gaining new insights in the verification phase. These insights are visual findings by the humans that contribute to validate, refine, or reject a given hypothesis whereby the analyst builds a mental model during this phase (Sacha et al., 2014).

Knowledge Generation Loop: The whole verification process is affected by the analysts knowledge of the domain, his/her expertise or experience. Thus, hypothesis and assumptions about the data are gained by the analysts knowledge and by a feedback loop, this knowledge can be integrated into the system (Sacha et al., 2014).

Additionally, Sacha et al. (2014) demonstrated the applicability of their 'knowledge generation model for VA' on several systems.

The Human-Computer System

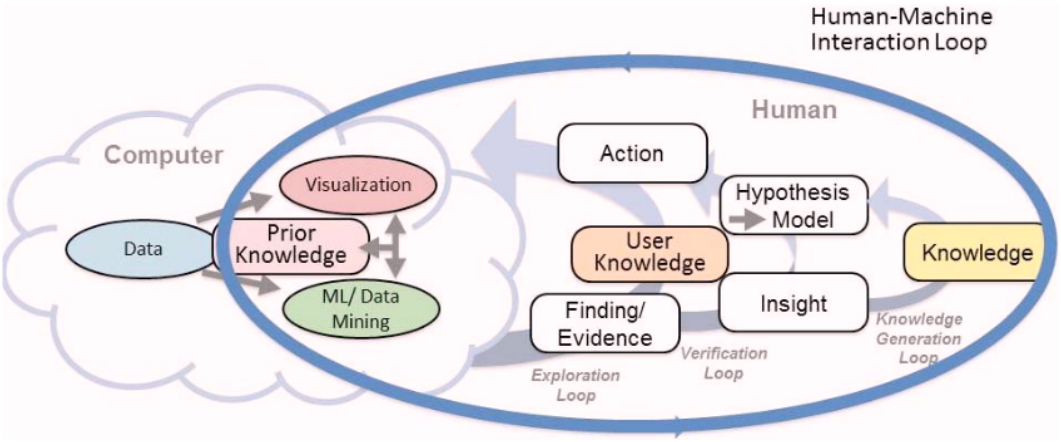


Figure 2.14: **The Human Computer System** – Illustration of the 'Human Computer System' extending the Knowledge Generation Model by Sacha et al. (2014) with 'Human-Machine Interaction Loop' describing the communication. Additionally the model contains prior knowledge on the computers side and user knowledge on the users side which are both driven by the user (Ribarsky and Fisher, 2016).

Ribarsky and Fisher (2016) contributed in their paper a new 'Human-Computer Model' which describes the connection between the human and the computer working together (see Figure 2.14). Generally, the new model is conceptually grounded on the 'Knowledge Generation Model for VA' by Sacha et al. (2014). The authors described that models which are focusing on the humans side (e.g., cognition, reasoning, decision-making), mostly did not provide much detail on the computers side. Based on the 'Human-Machine Interaction Loop', they demonstrated the connection and overlap between the human and the computer side of the model. On the computer side of the model, Ribarsky and Fisher (2016) added machine learning components to the data mining process including automated or semi-automated data analysis. Based on the 'Human-Machine Interaction Loop', the analyst gets the ability to communicate with the computer by interaction and the computer communicates to the human by visualization. The analytical discourse was also extended by the integration of prior knowledge (generated by external knowledge) and user knowledge (implicit knowledge), which are both driven by the user to establish a knowledge base. Additionally, Ribarsky and Fisher (2016) described that prior knowledge can also include knowledge which was produced by collaboration with others. It is important to note, that all the elements which are placed on the human side in Figure 2.14 or the elements described in the 'Human-Machine Interaction Loop' are human-centered but computer supported. Additionally, the authors stated that annotations in relation to the reasoning steps should be possible.

2.4 Discussion

To provide a better overview of the different available models and frameworks to describe the process of interactive data visualization (information visualization or VA), we divided the collected papers into three categories: 1) 'Visualization Pipeline Based Models'; 2) 'Formal / Mathematical Models' and 3) 'Other Model and Framework Designs'. These three used categories are described in detail at the beginning of Section 2.3. As former mentioned, we are searching for models and frameworks which can be used to describe knowledge-assisted visualization or VA systems from the view of the systems architecture and needed processes, as well as the connection with the user (e.g., visualization, perception, insights, knowledge, interaction).

Visualization Pipeline Based Models

The papers of the first category are based on the **Visualization Pipeline** introduced by Card and Mackinlay (1997) and in a second publication by Card et al. (1999). In general, these models are describing the way of the input data across the different transformation states to the visualization presented to the user and its interaction abilities. Chi and Riedl (1998) started to extend these models in parallel where they developed a new framework based on operators and user interaction. The resulting state model unifies the data analysis process and the relationship between the view and the values. To do so, they defined two categories of operators. The first category depends on functional versus operational operators. Thereby, functional operators have to be adapted for different data sets like filtering operators, and operational operators are similar across applications (e.g., rotation, scaling, translation). The second category contains view versus values operators. A value operator changes the data source like adding or deleting subsets for example. In contrast, a view operator changes the visualization content (e.g., zooming, rotation, translation, flipping). This schema was extended by Chi (2000, 2002) who showed the similarities between the Data Flow and the Data State model. The main concern of this paper is the comparison of the Data Flow and the Data State Model and to show that the Data State Model is equally expressive than the Data Flow Model by the modeling of the same visualizations. The main difference between these two models is that the Data State Model is working with operators and is mostly used for information visualization, whereby in contrast the Data Flow Model is mostly used for the description of scientific visualizations (Chi, 2002). The Event-Based Model by Tominski (2011) can be seen as an effective extension to the frameworks of Chi and Riedl (1998) and Chi (2000). Tominski (2011) used events to describe the visualization process in more detail, whereby he divided the workflow into three fundamental stages: 1) the user has the ability to specify their interests; 2) during the visualization, consistencies in the data will be searched; 3) thereby, it is possible to automatically adjust the visualization on the consistencies found. This way, it is possible to generate better

visualizations which are adapted to the needs of the user. As shown in Table 2.1, non of the included models and frameworks in this category supports the extraction of implicit knowledge to be included as computerized explicit knowledge as well as automated data analysis methods.

Formal / Mathematical Models

The second category deals with **Formal and Mathematical Models** and frameworks to describe the visualization process generating a bird's eye view on the functionality (e.g., transformations, interactions). In general, this group consists of four models (Jankun-Kelly et al., 2007; Tweedie et al., 1996; Van Wijk, 2005; Wang et al., 2009). Tweedie et al. (1996) described that their interactive visualization artefacts (IVAs) is displaying data by generated mathematical models, using interactively linked simple graphs. Thus, the user gains new insights by interactive data exploration, whereby these insights can aid the systems design. The model is based on a set of equations and each relating the performance of a number of parameters which can be used for light bulb design for example. Additionally, the simple visualization model introduced by Van Wijk (2005) is also based on several mathematical equations. In general, the model consists of 3 areas: 1) the data; 2) the visualization and 3) the user. The visualization part of this model can be seen as the central process whereby the data will be transformed into an image based on the specification (which is generated by the user's knowledge during the data exploration). It is very important to note that this model explicitly describes the generation of knowledge for the user. The model by Van Wijk (2005) was used as conceptual bases by Wang et al. (2009) who integrated a knowledge base into the system to support the user during the data exploration. Therefore, they defined four knowledge conversation processes: 1) internalization; 2) externalization; 3) collaboration and 4) combination. Jankun-Kelly et al. (2007) stated that visualization systems which do not store anything, do not tell the user where he/she is and where he/she has been are inefficient. To overcome these limitations, such issues have to be eliminated. Therefore, they created the p-set model containing four major elements: 1) vis transform; 2) vis parameter; 3) vis result and 4) derivation. This way, Jankun-Kelly et al. (2007) established a usable model for many different visualization applications whereby a common data format (XML structure) is needed to use the framework for reaching the goals. Since the extraction of the users implicit knowledge is a criteria for a model to describe knowledge-assisted VA, the models by Van Wijk (2005) and Wang et al. (2009) are considering the implicit knowledge of the user and Wang et al. (2009) especially also focuses on the integration of explicit knowledge to support the analysis workflow. However, no model of framework focuses on the extraction of implicit knowledge to store it as explicit knowledge during the analysis.

Other Models and Frameworks

The third group deals with **Other Model and Framework** designs. This category contains all models (seven models out of eight papers) which were not able to be included into the former categories (Han et al., 2003; Keim et al., 2010a, 2008; Lammarsch et al., 2011; Nazemi et al., 2011; Ribarsky and Fisher, 2016; Sacha et al., 2014) in terms of their specific setting. KDD (Han et al., 2003) is a process of searching for knowledge which is represented as relationships and patterns of large data sets which includes the employment of knowledge solving problems or interpret phenomena. In general, the KDD process is interactive and intuitive including many steps of decisions, which will be made by the user(s), Whereby existing approaches can be characterized as algorithm-based or visualization-based. Generally, the KDD process does not describe how to extract the users implicit knowledge to store it system internally. The VA process described by Keim et al. (2010a, 2008) combines automated analysis methods with human interaction to gain knowledge or insights from the data based on the generated models or hypothesis supported by a feedback loop for interaction. Conceptually grounded on the VA process by Keim et al. (2008), Lammarsch et al. (2011) developed an extension by including domain knowledge. Additionally, the new model offers the user the ability to interact with all elements included in the gray area as well as all connections leading inside and outside. The interactive visual interface is the center of the model and combines the humans' hypothesis and the models, based on automated analysis to gain new insights. Additionally, Sacha et al. (2014) developed an extension to the VA process by Keim et al. (2010a). Thereby, they divided the grounded VA process into two areas: 1) computer, which includes the data, the visualization and the model and 2) human, which is focusing on the knowledge. For gaining insights or knowledge, Sacha et al. (2014) defined three loops on the human side: 1) the exploration loop; 2) the verification loop and 3) the knowledge generation loop. Based on these three loops, it is possible to describe the levels of detail in relation to insight gaining supported by the VA system. Ribarsky and Fisher (2016) extended the 'Knowledge-Generation Model for VA' by Sacha et al. (2014) with a 'Human-Machine Interaction Loop' describing the systems interactions and reasoning steps. They extended the model on the computer side with prior knowledge combined with automated data analysis methods and on the human side with the integration of the users knowledge (implicit knowledge) into the exploration loop. Both types of knowledge, which are integrated in the system, are driven by the user or by collaboration to establish a knowledge base. In contrast, the Reference Model for Adaptive Visualization Systems (MAVS) by Nazemi et al. (2011) consists of three basic components, which are: 1) Input; 2) Adaption and 3) Output. Additionally, this model can be configured by parameters which are defined by certain impact and influence factors like knowledge or the user's behavior or the underlying data structure or amount. The input component receives two different impact factors for the visualization adaption and parametrization. The interaction analysis is a part of

interaction events which are captured with different contextual information about the interaction type, the layout method and the content. In this category, all out of one model (Nazemi et al., 2011) are including automated data analysis methods supporting the analysis workflow. Additionally, four models and frameworks also including the users knowledge and its extraction in different levels of detail (Han et al., 2003; Lammarsch et al., 2011; Ribarsky and Fisher, 2016; Sacha et al., 2014), but no model concerns the dynamic aspect.

	(Card and Mackinlay, 1997)	(Card et al., 1999)	(Chi and Riedl, 1998)	(Chi, 2000)	(Chi, 2002)	(Tominski, 2011)	(Tweedie et al., 1996)	(Van Wijk, 2005)	(Jankun-Kelly et al., 2007)	(Wang et al., 2009)	(Han et al., 2003)	(Keim et al., 2008)	(Keim et al., 2010a)	(Lammarsch et al., 2011)	(Nazemi et al., 2011)	(Sacha et al., 2014)	(Ribarsky and Fisher, 2016)
Data Exploration	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Automated Analysis	-	-	-	-	-	-	-	-	-	-	✓	✓	✓	✓	-	✓	✓
Dynamics	✓	✓	✓	✓	✓	✓	-	✓	-	✓	✓	-	-	-	-	-	-
Knowledge	-	-	-	-	-	-	-	-	-	✓	-	-	-	✓	-	✓	✓

Table 2.1: **Requirements** – An overview of the included models and frameworks categorized along the model criteria (see Section 2.2) which has to be fulfilled by a new knowledge-assisted VA model.

Summary

Summarizing the concerned aspects of the included models based on the model criteria (see Table 2.1), all models and frameworks are containing interactive data exploration. It is interesting to see that models which are including the dynamic aspect (focusing on time), do not include the integration of automated data analysis methods. From the view of knowledge, only four out of 14 models (17 papers) are meeting this aspect. However, as the integration of knowledge into a model or framework can be fulfilled in different degrees of accuracy, Wang et al. (2009) and Lammarsch et al. (2009) are containing a knowledge base using explicit knowledge for analysis support.

In general, the models and frameworks by Chi and Riedl (1998) and Chi (2000) are providing the ability to be extended with a knowledge generation part which has to be built in parallel to the visualization pipeline. In contrast, the 'Event-Based Model' by Tominski (2011) can be extended by integrating the knowledge directly into the model which raises some needs of adapting of the given connections for the included

elements. Tweedie et al. (1996) and Jankun-Kelly et al. (2007) are focusing on the visualization and the needed transformations for data representation. For the integration of the knowledge generation process into these models, a completely new part has to be developed. From the view of knowledge integration, MAVS (Nazemi et al., 2011) offers a good ability describing knowledge externalization in a machine readable form. Additionally, the models by Keim et al. (2010a, 2008), Lammarsch et al. (2011) and Sacha et al. (2014) also offer the ability to be used as conceptual grounding for the integration of explicit knowledge, but it would be very complex to be implemented on this level of detail. Such an integration was described by Ribarsky and Fisher (2016) grounded on Sacha et al. (2014) adding prior knowledge into the computer side and users knowledge into the user side of the model to create a user-driven knowledge base. They integrated a 'Human-Machine Interaction Loop' describing the interactions and displaying the reasoning steps. Since this model is very human centered, it does not clearly support the development of knowledge-assisted visualization systems with regard to the needed components, processes and their connections but it describes very well the different operations performed on the users side. Van Wijk (2005) and Wang et al. (2009) are the only approach which are describing the knowledge generation of the systems user. Additionally, Wang et al. (2009) also demonstrated the expandability of the model by Van Wijk (2005) with a knowledge base. However, based on Wang et al. (2009) it is possible to describe visualization (VIS) and knowledge-assisted visualization (KAV) system's but it did not support visual analytics (VA) or KAVA systems. Additionally, the model offers no possibility to extract the implicit knowledge of the analyst and store it as an explicit knowledge system internally. Thus, this is a good starting point for the integration of explicit knowledge, the combination of the user's knowledge and the knowledge generated by automated analysis methods translated in a machine readable context.

CHAPTER 3

The Knowledge-assisted VA Model

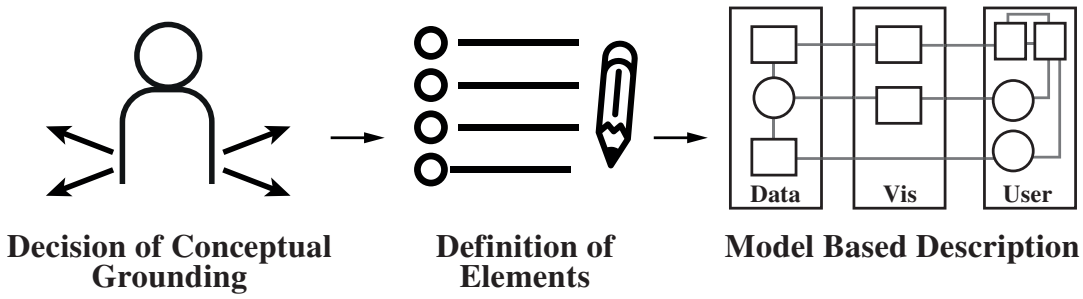


Figure 3.1: **Graphical Overview of Chapter 3** – Illustration of the topics which are covered in this chapter, providing the description of the conceptual grounding, the definition of the model elements and of the model.

As illustrated in Figure 3, this chapter presents the elements of the new 'Knowledge-assisted VA Model' which are described in Section 3.2, followed by a textual and formal description (see Section 3.3). At the end of the chapter, a discussion of the new model and the different describable system types is included.

3.1 Decision of the Conceptual Grounding

As discussed in Section 2.4, the currently available models do not cover all the needs to describe the process of knowledge-assisted VA. Thus, we decided to develop a new model describing knowledge-assisted VA conceptually grounded on the 'Simple Visualization Model' by Van Wijk (2005). This model was also used as basis by Wang et al. (2009) who included a knowledge base as extension but no possibility to extract the users implicit knowledge as well as the integration of automated data analysis methods. Thus, based on the extended model by Wang et al. (2009) it is possible to describe knowledge-assisted visualization systems without the iterative extraction of implicit knowledge during the runtime. Generally, the model by Van Wijk (2005) is structured in tree areas: 1) Data; 2) Visualization and 3) User, whereby the required model-oriented bird's eye view on the system is ensured to include the connection to the user. Furthermore, the ability to integrate a knowledge database was demonstrated in the extension by Wang et al. (2009). Since the model by Van Wijk (2005) contains also a formal description based on functions, derivations and integrations, the aspect of dynamics is covered.

3.2 Definition of the Model Elements

This section presents a short introduction of the different elements used to describe the novel 'Knowledge-assisted VA Model' in combination with the definition of its elements and formal symbols (see Figure 3.2):

A **:= Automated Analysis:** Components used for automated data analysis based on different algorithms that can be used depending on the analysis problem.

D **:= Data:** Is used as the general term describing the two different types of data (D^r and D^a) which are included in the system.

D^r **:= Raw Data:** Specifies the raw input data of the system which are used as input for different automated analysis methods for example.

D^a **:= Pre-analyzed Data:** Refers to the output which is generated by one ore more automated analysis methods (A).

E **:= Exploration:** This is based on the user's implicit knowledge K^i to adjust the visualization V by the implicit specification S^i.

I **:= Image:** Is the visual representation generated by the visualization V which is perceived P by the user.

K^i **:= Implicit Knowledge:** Contains the users personal knowledge about the data and the insights gained during the perception P of the presented images I (see Section 1.2).

K^e **:= Explicit Knowledge:** The computerized knowledge stored system internally, generated by the extraction X of the users implicit knowledge and automated analysis methods A (see Section 1.2).

K^x **:= Extracted Knowledge:** Specifies the extracted and computerized version of the users implicit knowledge K^i.

K^a **:= Automated Analysis Knowledge:** Specifies the computerized knowledge which is generated by the use of one or more automated analysis methods A.

P **:= Perception:** The process how the user gains new insights to generate implicit knowledge K^i.

S **:= Specification:** The combination of the specification S^i based on the users exploration E by using implicit knowledge and the specification S^e based on the explicit knowledge K^e stored system internally.

S^i **:= Specification by** K^i**:** The specification part which is based on the exploration E of the users implicit knowledge K^i.

S^e **:= Specification by** K^e**:** The specification part which is based on the explicit knowledge K^e stored system internally.

t **:= Time:** Because data analysis is a interactive process, many components (e.g., K^i, K^e, S^i, S^e) are changing over time.

V **:= Visualization:** The process generating an image I from the data based on the specification which is affected by the users input and the system itself.

X **:= Externalization:** The process how the implicit knowledge K^i is computerized to be stored system internally as extracted knowledge K^x.

3.3 Description of the Model

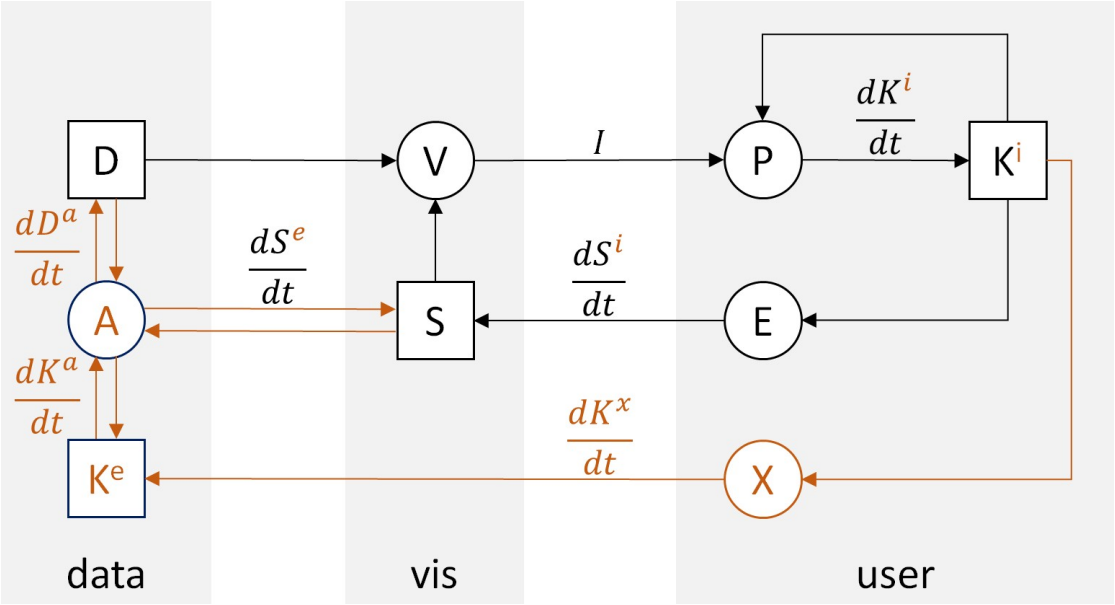


Figure 3.2: **The Knowledge-assisted VA Model** – Illustration of the new designed 'Knowledge-assisted VA Model' based on the 'Simple Visualization Model' by Van Wijk (2005). The orange elements are describing the new added elements for generating, storing and integrating explicit knowledge into the VA process.

In the 'Simple Model of Visualization' (Van Wijk, 2005) the input data D are seen as static, thus, they cannot change over time. From a general perspective, a visualization system gets raw data D^r as input data that can be transformed or restructured into a pre-analyzed dataset D^a by automated analysis methods A if needed. For example, if the input data are temperature data measured every minute, the analysis A step calculates the mean value for each hour, day, and month to remove a seasonal component of the cycle length. Therefore, the analysis step uses the explicit knowledge K^e (see Equation 3.1) which is generated by a combination of the extracted implicit knowledge K^i of the user (K^x) and the knowledge generated by automated analysis methods A defined as (K^a).

$$\frac{dD^a}{dt} = A(D^r, K^e, t) \tag{3.1}$$

whereby the generation of the pre-analyzed dataset D^a follows an integration over time t (see Equation 3.3), assuming that D^a_0 is the initial pre-analyzed dataset containing the same data like D^r so that the initial dataset can be marked as D (see Equation 3.2)

before the first analysis is carried out (or especially if no analysis is performed). Additionally, a new D^a is created by the combination of the current D^a_n and the new calculated D^a_{n+1} (see Equation 3.4) (i.e., a cascade of automated analysis steps):

$$D = D^r = D^a_0 \tag{3.2}$$

$$D^a_{n+1} = \int_0^t A(D^r, K^e, t)dt \tag{3.3}$$

$$D^a(t) = D^a_n + D^a_{n+1} \tag{3.4}$$

As described by Van Wijk (2005), in the model, the visualization can be seen as the central process. The dataset D^a will be transformed into a time depending image $I(t)$ based on the specification S (see Equation 3.5):

$$I(t) = V(D^a, S, t) \tag{3.5}$$

This image I will be perceived by the user's perception P which results as an increase of the users implicit knowledge K^i (see Equation 3.6):

$$\frac{dK^i}{dt} = P(I, K^i, t) \tag{3.6}$$

The current implicit knowledge K^i of the user follows an integration over the time t, assuming that K^i_0 is the initial implicit knowledge at the time point t_0 (see Equation3.7):

$$K^i(t) = K^i_0 + \int_0^t P(I, K^i, t)dt \tag{3.7}$$

A further important aspect is the exploration E described as $E(K^i)$. The user decided to adapt the specification S^i (implicit part) of the visualization V based on the users current implicit knowledge K^i. This happens through further exploration E (see Equation 3.10):

$$\frac{dS^i}{dt} = E(K^i, t) \tag{3.8}$$

whereby the current implicit specification S^i follows an integration over time t, judging from S^i_0 as initial specification for the implicit knowledge K^i (see Equation 3.11):.

$$S^i(t) = S^i_0 + \int_0^t E(K^i, t)dt \tag{3.9}$$

Based on the definition of knowledge K by Chen et al. (2009, p. 13), we differ between knowledge which is generated by the extraction of the users implicit knowledge K^x and the knowledge which is generated by automated analysis methods K^a.

The combination of these two knowledge parts (K^x and K^a) will be referred as explicit knowledge K^e in this work. At this point it is important to note that automated analysis methods A which are integrated in a system, do not necessarily need to generate knowledge (K^a) that can be stored.

To retain (parts of) the users implicit knowledge K^i for further analysis support, it can be extracted X (extraction) and stored as extracted knowledge K^x in a computerized form (see Equation 3.10) whereby the knowledge extraction was also covered by Wang et al. (2009) in a similar way:

$$\frac{dK^x}{dt} = X(K^i, t) \tag{3.10}$$

The extracted knowledge K^x also follows an integration over time t, assuming that K_0^x is the initial extracted knowledge which will increase by further extraction of the users implicit knowledge K^i (see Equation 3.11):

$$K^x(t) = K_0^x + \int_0^t X(K^i, t)dt \tag{3.11}$$

Additionally, to retain (parts of) the knowledge generation by automated computerized analysis methods operating on dataset D^a which is based on the specification S, can be stored as analysis knowledge K^a in a computerized form (see Equation 3.12):

$$\frac{dK^a}{dt} = A(D^a, S, t) \tag{3.12}$$

Thus, the analysis knowledge K^a also follows an integration over the time t, assuming that K_0^a is the initial automated analysis knowledge which can increase by further automated analysis of the dataset D^a , based on the specification S (see Equation 3.13):

$$K^a(t) = K_0^a + \int_0^t A(D^a, S, t)dt \tag{3.13}$$

As former mentioned, the explicit knowledge K^e can be seen as the sum of the extracted knowledge K^x (generated from the implicit knowledge K^i) and the automated analysis knowledge K^a (generated by automated analysis methods A) (see Equation 3.14):

$$\frac{dK^e}{dt} = \frac{dK^x}{dt} + \frac{dK^a}{dt} \tag{3.14}$$

whereby the explicit knowledge K^e (composed from the user's extracted knowledge K^x and the automated analysis knowledge K^a) follows an integration over time t assuming $K_0^e = K_0^x + K_0^a$ as initial explicit knowledge K^e (see Equations 3.15, 3.16 and

3.17) whereby K_0^e can also contain knowledge which was integrated during the system development:

$$K^e(t) = K_0^x + \int_0^t X(K^i, t)dt + K_0^a + \int_0^t A(D^a, S, t)dt \tag{3.15}$$

$$K_0^e = K_0^x + K_0^a \tag{3.16}$$

$$K^e(t) = K_0^e + \int_0^t (X(K^i, t) + A(D^a, S, t))dt \tag{3.17}$$

In order to achieve a knowledge support, the explicit knowledge K^e (stored computerized knowledge) is used for exploration and analysis support of the dataset D^a. this also described by the 'Visual Analytics Mantra': "*Analyze first, show the important, zoom, filter and analyze further, details on demand*" by Keim et al. (2010a). Thereby, the explicit specification component S^e is produced (see Equation 3.18):

$$\frac{dS^e}{dt} = A(D^a, S^e, t) \tag{3.18}$$

Wherein the current explicit specification S^e follows an integration over time t, when starting from S_0^e as initial specification for share explicit knowledge K^e (see Equation 3.19):

$$S^e(t) = S_0^e + \int_0^t A(D^a, K^e, t)dt \tag{3.19}$$

In summary, the specification S can be seen as the sum of the implicit specification S^i (depending on the implicit knowledge K^i) and the explicit specification S^e (depending on the explicit knowledge K^e) (see Equation 3.20):

$$\frac{dS}{dt} = \frac{dS^i}{dt} + \frac{dS^e}{dt} \tag{3.20}$$

whereby the specification S (composed from the implicit S^i and explicit S^e specification) follows an integration over time t assuming $S_0 = S_0^i + S_0^e$ as initial specification for the combination of the implicit K^i and explicit K^e knowledge (see Equations 3.21, 3.22 and 3.23):

$$S(t) = S_0^i + \int_0^t E(K^i, t)dt + S_0^e + \int_0^t A(D^a, K^e, t)dt \tag{3.21}$$

$$S_0 = S_0^i + S_0^e \tag{3.22}$$

$$S(t) = S_0 + \int_0^t (E(K^i, t) + A(D^a, K^e, t))dt \tag{3.23}$$

Seen from an general perspective and extending the description by Van Wijk (2005), visualization and the extraction of knowledge K (composed from implicit K^i and explicit K^e knowledge ($K = K^i + K^e$) from the data D are objective processes in relation that the results do not depend on the person performing the analysis. Additionally, the analysis has to be repeatable by others and has to provide the same results under the same conditions (Van Wijk, 2005). However, visualization is not a well-defined process (always the same result relating to the same data). That means that the implicit knowledge K^i does not change only based on D, it is also related to the specification S (e.g., given by hardware, parameter, algorithms and explicit knowledge K^e), the perception P of the user and his/her implicit prior knowledge K_0^i (see Equation 3.24):

$$\frac{dK}{dt} = P(V(D, E(K^i, t) + A(D, K^e, t), t)K^i, t) \tag{3.24}$$

3.4 Discussion

We used the 'Simple Visualization Model' by Van Wijk (2005) as conceptual grounding to generate the 'Knowledge-assisted VA Model'. Therefore, we included an additional path to the 'Simple Visualization Model' (see Figure 3.2) which describes the externalization X of the user's knowledge in a machine readable structure. Additionally, we also included a part describing the knowledge generation by automated pre-analysis A (described as "*Analyze first*" by Keim et al. (2010a)) in combination with the externalization of the users implicit knowledge K^i. It is important to note that the 'analyze first' criterion is only possible if one can apply automated analysis methods A to the dataset D^r to prepare a dataset D^a. This implies that knowledge-assisted visual analytics requires a share of explicit knowledge K^e to be able to support and extend this preliminary analysis methods A. Thus, the explicit knowledge K^e can also be used without corresponding data D (e.g., a knowledge corresponding experiments) because also the explicit knowledge K^e alone can provide insights or helps to gain insights on corresponding datasets. On the contrary, it is important to note that it is not possible to fulfill the 'analyze first' step without automated analysis methods A which can be extended with 'integrate explicit knowledge' K^e to fulfill all the needs for a knowledge-assisted VA system.

$$\textbf{System Types} := \begin{cases} |K^i| \geq 0,\ |K^e| = 0,\ A = 0,\ V > 0 \ \Rightarrow\ \textbf{VIS} \\ |K^i| \geq 0,\ |K^e| > 0,\ A = 0,\ V > 0 \ \Rightarrow\ \textbf{KAV} \\ |K^i| \geq 0,\ |K^e| = 0,\ A > 0,\ V > 0 \ \Rightarrow\ \textbf{VA} \\ |K^i| \geq 0,\ |K^e| > 0,\ A > 0,\ V > 0 \ \Rightarrow\ \textbf{KAVA} \\ |K^i| > 0,\ |K^e| = 0,\ A > 0,\ V = 0 \ \Rightarrow\ \textbf{A} \\ |K^i| > 0,\ |K^e| > 0,\ A > 0,\ V = 0 \ \Rightarrow\ \textbf{KAA} \end{cases} \tag{3.25}$$

Keim et al. (2010b) declared that VA can be characterized by the use of two problem classes: "*(1) Analytical Problems and (2) General Application Areas of IT*" (Keim et al., 2010b). To solve the problems in the named classes, they pointed out to "*three methodological classes: a) Automatic Analysis, b) Visualization, and c) Visual Analytics*" (Keim et al., 2010b). Based on the article by Keim et al. (2010b) in combination with the novel 'Knowledge-assisted VA Model', it is now possible to distinguish between four different visualization system types and two systems without visualization described in Equation 3.25.

The first time a visualization is used without explicit knowledge K^e and automated analysis methods A, the 'Visual Information Seeking Mantra': "*Overview first, zoom and filter, then details-on-demand*" (Shneiderman, 1996) comes in use. Such 'Visualization' (VIS) systems can be described with the model by Van Wijk (2005) and Wang et al. (2009) for example. If, the user integrates step by step his/her implicit knowledge K^i and/or knowledge generated by the integration of automated methods A in a machine-readable way, explicit knowledge K^e is generated and integrated in the system as support for exploration and insight gaining. Based on this, the related systems can be defined as 'Knowledge-assisted Visualization' (KAV) which can be described by the model of Wang et al. (2009) or 'Knowledge-assisted Visual Analytics' (KAVA) depending on the integration of automated analysis A or not. If the system supports preliminary data analysis by automated analysis methods A without the integration or storing of explicit knowledge K^e, further analysis will follow the 'Visual Analytics Seeking Mantra' by Keim et al. (2010a) and is can be described as 'Visual Analytics' (VA) system.

Generally, automatic analysis methods can also benefit from the use of prior knowledge which plays a fundamental role in the knowledge discovery process (KDD) (Fayyad et al., 1996). Knowledge-based systems enable the integration of explicit knowledge (also called as background knowledge in this context (Hand, 1998)) into the reasoning process to model exceptional rules, preventing the system to reason over abnormal conditions (Perner, 2006). Novel knowledge-assisted data analysis and interpretation approaches using computer-readable explicit knowledge have obvious advantages in contrast to systems that do not use this (Zupan et al., 2006). Assuming that there are systems available without containing a visualization V, the model also allows to describe

'Analysis' (A) systems and 'Knowledge-assisted Analysis' (KAA) systems. These systems can be seen as subtype of VA and KAVA systems but without including a visual interface for data representation.

An additional interesting aspect is that Van Wijk (2005) expects that the data D did not change over time t, it seems that he considers this appears as a static entity throughout exploration / visualization. Thus, during the data exploration, no new datasets can be added to the system. Based on this assumption, the model could now be expanded by the integration of dynamic datasets or data sources $D(t)$ (e.g., different types of (time-oriented) streaming data) in the future.

Next Steps

To validate the new 'Knowledge-assisted VA Model', two case studies were performed solving real world problems of two different application fields. The first case study (KAMAS see Part II) takes place in the field of IT-security. More precisely, we developed a 'Knowledge-assisted Malware Analysis System' to support the analysts during behavior-based malware analysis. The second design study (KAVAGait see Part III) is related to the field of clinical gait analysis supporting the clinicians during clinical decision making. Generally, both prototypes are receiving time-oriented data as input, but they are different in their structure depending on the included data. Based on these two design studies, in Part IV the applicability of the new 'Knowledge-assisted VA Model' is demonstrated. Additionally, its descriptive, evaluative and generative power (Beaudouin-Lafon, 2004) are discussed.

Part II

Case Study 1: Behavior-based Malware Analysis in IT-Security (KAMAS)

CHAPTER 4

Motivation & Related Work

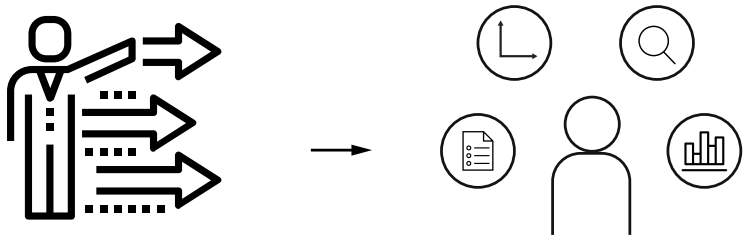


Figure 4.1: **Graphical Overview of Chapter 4** – Illustration of the main topics which are covered in this chapter in relation to the first research domain: behavior-based malware analysis.

This chapter starts with the general motivation (see Section 4.1) for the first design study in relation to behavior-based malware analysis and the need for knowledge-assisted visualization systems in this area (see Figure 4.1). Additionally, this chapter presents the related work (see Section 4.2) according to behavior-based malware analysis systems.

4.1 Motivation

Malware (malicious software) is undoubtedly one of today's greatest threats to the Confidentiality/Integrity/Availability (CIA) triangle of information security (Stoneburner et al., 2002). It has become a common tool in digital theft, corporate and national espionage, spam distribution and attacks on infrastructure availability. Thus, monitoring of vulnerable systems will become increasingly important. This applies to networks, individual computers, as well as mobile devices (e.g. (Gelenbe et al., 2013; Trinius et al., 2009; Yee et al., 2012)). Malicious software, or malware, can be defined as "*any software that does something that causes harm to a user, computer, or network*" (Sikorski and Honig, 2012). Examples include viruses, trojan horses, backdoors, worms, rootkits, scareware, or spyware. Malware analysis, in turn, is defined as "*the art of dissecting malware to understand how it works, how to identify it, and how to defeat or eliminate it*" (Sikorski and Honig, 2012). The number of malicious programs, is growing at a tremendous rate. The sheer number of newly discovered malware variants poses a significant challenge to the security community. Only, in the third quarter of 2014, 20 million new samples were discovered (PandaLab, 2014) which amounts to more than 150,000 pieces of malicious software that need to be triaged every day. What some argue to be a manageable annoyance for personal computer users have the potential to cause severe damage in high-availability environments or safety critical infrastructures.

To be effective for such analysis, accurate detection mechanisms are needed (Dornhackl et al., 2014). Malicious behavior in software is identified through static or dynamic analysis (Egele et al., 2012, p. 7). On the one hand, when statically analyzing a possibly malicious software sample, the binary file is usually disassembled and dissected function by function. Dynamic analysis, on the other hand, focuses on the sample's behavior (Egele et al., 2012, pp. 8): The file is executed on a test system and its activity is observed and recorded. Both approaches yield patterns or rules that are later used for detection and classification of malicious software. This leads to two different methods to detect malicious programs which are the signature-based approach and the behavior-based approach.

Signature-based Malware Recognition

Current malware detection/classification commonly uses a signature-based approach: Known malware is described by its syntactic characteristics – mostly bit strings or simple patterns (e.g., defined by regular expressions). This approach is used for the most common antivirus systems but the signature-based detection has several shortcomings (Christodorescu et al., 2008, p. 5): Firstly, obfuscation techniques commonly utilize polymorphic or metamorphic mutation to generate an ever-growing number of malware variants that are different in appearance but functionally identical. Secondly,

signature-based techniques can only detect malware that has already been identified and analyzed; new species or hitherto unknown variants are generally overlooked.

Behavior-based Malware Recognition

To overcome the limitations of signature-based approaches, behavior-based malware detection can be used (Egele et al., 2012). Here, a sample's activity is analyzed during execution using dynamic analysis techniques. Afterwards, a previously defined set of rules is applied to the generated report in order to decide whether the sample's behavior is malicious or not. Behavioral analysis of suspicious code samples is, despite the disadvantage in performance, a promising approach to detecting and pre-classifying malware: A specific piece of malware is not characterized by its syntactic appearance, but rather by its dynamic behavior – in whatever disguise it might appear.

This use case focuses on the visualization of the pattern extraction and recognition process used in a research project on formal definition of malware behavior (Dornhackl et al., 2014).

The Need for Knowledge-assisted Visualization

Because of the fact that manual analysis by domain experts is very cumbersome, automated data analysis methods are needed. In order to automate this process as much as possible, patterns of particular call sequences need to be specified and categorized as being potentially harmful or harmless. These patterns can then semi-automatically be detected and analyzed in context. On the other hand, this process cannot be automated completely as domain experts need to be in the loop to identify, correct, and disambiguate intermediate results. This combination of large amounts of data, complex data analysis needs, and the combination of automated data analysis with analytical reasoning by domain experts lends itself very well to the notion of visual analytics (Keim et al., 2010a; Thomas and Cook, 2005).

A major tenet of visual analytics states that analytical reasoning is not a routine activity that can be automated completely Wegner (1997). Instead it depends heavily on analysts' initiative and domain experience. Furthermore, visual analytics involves automated analysis methods which computationally process large volumes of data and thus complement human cognition. In addition to challenging analysis methods, 'implicit knowledge' (Chen et al., 2009) or 'tacit knowledge' (Wang et al., 2009) about the data, the domain experience or prior experience are often required to make sense of the data and not become overwhelmed. By externalizing some of the domain experts' implicit knowledge, it can be made available as explicit knowledge and stored in a knowledge database (Chen et al., 2009) to support the analysts and to share their knowledge with others.

4.2 Related Work

Currently, problem-oriented research is underrepresented in visualization literature (Lam et al., 2012; McKenna et al., 2016, 2015; Pirker and Nusser, 2016; Sedlmair et al., 2012b), even though these are essential for design and implementation of suitable visual analytics solutions. Therefore, we will first present some notable examples of problem characterization papers in other domains and then focus on visualization work of malware analysis.

Design Studies

Sedlmair et al. (2008) studied the daily routines of automotive engineers and their tool support using interviews and task observation. After analysis along different collaboration settings, their main contributions are system requirements for multiple display environments. The MizBee design study (Meyer et al., 2009), set in comparative genomics, starts with the characterization of questions asked in this problem domain as the first of four contributions. For this, they conducted interviews with two expert biologists, who work in the area, followed by a taxonomic analysis of the visual encoding of the data. Guided by the knowledge of the problem characterization and analysis they designed the multiscale system called MizBee. Tory and Staub-French (2008) conducted a field study in the domain of building design to investigate the current use of visualization in meetings. The design requirements for RelEx by Sedlmair et al. (2012a) were based on detailed characterization of data, tasks, and existing tools, which were obtained using literature research, contextual observation, semi-structured interviews, and a focus group. This way, RelEx allows automotive engineers to specify and optimize traffic patterns in such networks. In contrast, Pretorius and Van Wijk (2009) point out that it is important for visualization designers to ask themselves: "*What does the user want to see?*" and "*What do the data want to be?*" as well as how these two points mutually enhance one another.

In the problem domain of cyber security, Fink et al. (2009) developed a set of visualization design principles with a focus on high-resolution displays and presented prototypes according to these design principles. Additionally, they presented prototypes according to these design principles. Goodall et al. (2004) conducted contextual interviews to gain a better understanding of the intrusion detection workflow and proposed a three-phased model in which tasks could be decoupled by necessary know-how to provide more flexibility for organizations in training new analysts. However, none of these user-centered studies tackled behavior-based malware pattern analysis.

Malware Analysis and Visualization

Previous research explored the area of malware analysis from different points of view. Lee et al. (2011) made a good case for visualization in malware analysis, which they propose is needed to recognize and extract unseen malware patterns. For the classification of the malware the authors used a technique which extracted the properties of malicious software which were provided by anti virus companies. Dornhackl et al. (2014) introduced a workflow for malware pattern extraction. They log the system calls of malicious software which is executed on a host and analyze them. In this way, it is possible to extract and define malicious system call patterns. The survey by Shiravi et al. (2012) described and categorized 38 different network security visualization systems, which they divided into five different classes of use-cases. The definition of the classes was based on the behavior of the malicious activities, which could be detected with these tools. Some of the presented approaches are also supporting methods for interactive data exploration. In contrast to this survey, we are looking at the behavior-based malware analysis on system and API call level. Likewise, a part of Conti's book (2007) is dedicated to malware analysis.

However, all of the mentioned approaches are related to visualization of network traffic and not of malware execution traces. Software visualization (Diehl, 2007) tackles some related data sources in the form of static and dynamic software but has completely different analysis goals. There is, however, general literature on 'automated techniques' for malware detection and analysis as well as surveys for areas related to malware analysis: Siddiqui et al. (2008) provide a compact overview of 19 malware detection approaches using data mining on file features. Additionally, they categorize these 19 malware detection approaches based on the included file features, the analysis type and the detection strategy. Complementarily, Egele et al. (2012) survey 18 approaches for dynamic analysis of malware samples and compare them alongside emulation/instrumentation technologies, the granularity of recorded malware behavior, and obfuscation techniques. Furthermore, some of their systems support clustering or automatic report generation. Bazrafshan et al. (2013) survey 22 approaches for heuristic malware detection and categorize them by the data source used. Idika and Mathur (2007) survey malware detection approaches based on anomalies, specifications, or signatures. In 2015, we presented a survey on visualization systems for malware analysis (see Chapter 5 and (Wagner et al., 2015c)). Thereby, we described 25 malware visualization tools and categorized their main analysis focus based on the 'Malware Visualization Taxonomy'.

The landscape of mobile malware was surveyed by Felt et al. (2011), who summarized the characteristics of 46 malware samples for iOS, Android, and Symbian operating systems. Additionally, they discussed the effectiveness of preventive measures against such mobile malware. Finally, the topic of port scanning was surveyed by Bou-Harb et al. (2014) and Bhuyan et al. (2011). Approaches for malware detection tech-

niques in general are covered in the surveys of Egele et al. (2012) and Bazrafshan et al. (2013). This surveys is characterizing three different detection approaches. First, there is the signature-based technique which was used by most of the anti virus programs but it only works for known malware. Each file has its own signature and so the detection of various known malicious software is more efficient and faster than all the other approaches. Second, the behavior-based approach that analyses the execution behavior of malware and third, the heuristic-based approach that aims to combine the advantages of the signature based and the behavior based method.

Summary

Based on the work which has been performed in visualization for malware detection, it can be recognized that so far, no design study has been published in the domain of malware analysis from a visual analytics perspective. To close this gap and provide a basis for further designers of visual analytics systems in this domain, we chose to investigate it using methods from human-computer interaction (Lazar et al., 2010; Sharp et al., 2007). We follow a threefold approach consisting of a systematic literature survey, a focus group, and semi-structured interviews with domain experts as described next.

CHAPTER 5

Survey on Visualization Systems for Malware Analysis

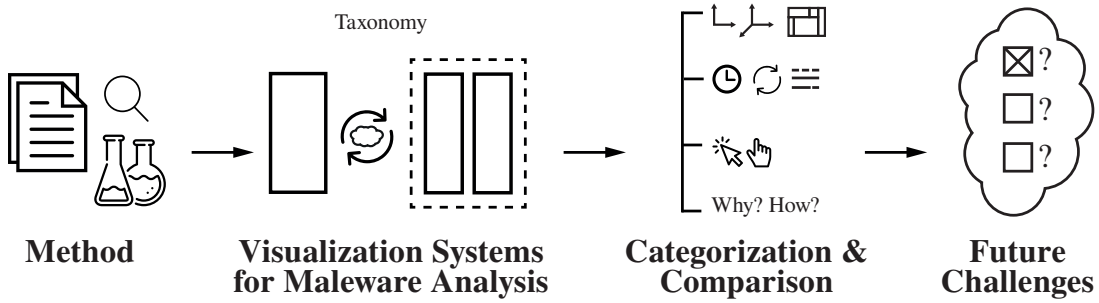


Figure 5.1: **Graphical Overview of Chapter 5** – Illustration of the main topics which are covered in this chapter in relation to the presented survey.

This chapter presents a detailed survey on visualization systems for malware analysis (see Figure 5.1). All the surveyed systems are categorized along the new introduced 'Malware Visualization Taxonomy' (see Figure 5.7) dividing such systems into three categories ('Individual Malware Analysis', 'Malware Comparison' and 'Malware Summarization') (see Section 5.2). Additionally, all the surveyed systems are categorized and compared (see Section 5.3) in relation to the used visualization techniques (Keim, 2002), mapping and representation space (Aigner et al., 2011), temporal aspects (Aigner et al., 2011), interactivity, and problems/actions ("*Why?*") (Munzner, 2014). At the end of the survey, we present a discussion and future challenges (see Section 5.4) as well as a summary (see Section 5.5) to complete the survey.

5.1 Method

To get a comprehensive overview of visualization methods supporting malicious software analysis systems in the field of IT security, we used a number of 'digital libraries' (IEEE Xplore, ACM digital library, Google Scholar, and Academic Research Microsoft) (see Figure 5.2). A skeleton of common search terms was used in all of them. To improve our search results we individually refined the different keywords and keyword combinations for each of the used search engines in order to achieve maximum topical coverage. This was necessary since each search engine has its own strengths and weaknesses (e.g., on IEEE Xplore it is possible to structure your own advanced search by selecting different search parameters). All the used search terms and combinations are included in the appendix (see Appendix A). Based on the keywords and combinations used, we found about 200 publications.

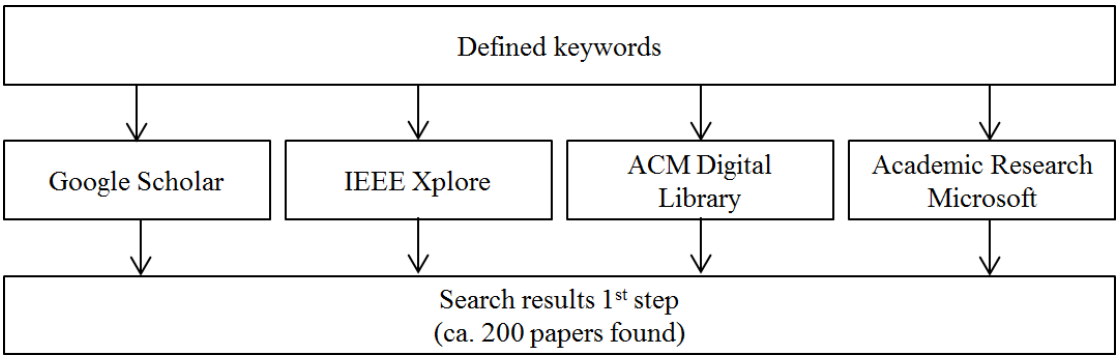


Figure 5.2: **Literature Search Step 1** – Illustration of the first step ending up with ca. 200 papers found.

In a second step (see Figure 5.3), we identified the 'authors' of the most relevant papers and refined our search to include other publications by these researchers. For the research with IEEE Xplore, we used the advanced search options, so it was possible to select the right parameter for each author and keyword (e.g., the parameters: Document Title, Authors, Abstract, Full Text and Metadata). Additionally, we visited the homepages of the identified authors to look for additional material related to the research topics. Based on the employed search strategies it was possible to identify more than 220 different scientific papers and articles in the respective area.

In order to sort out inappropriate papers (see Figure 5.4), we perused all the abstracts and the conclusions for relevant information. Through this process, we verified whether the identified papers really fit the main topic of malware analysis systems that make use of visualization methods. Thus, it was possible to reduce the findings to 42 papers. The categorization process and the elimination of inappropriate papers were performed in each search step of the research process.

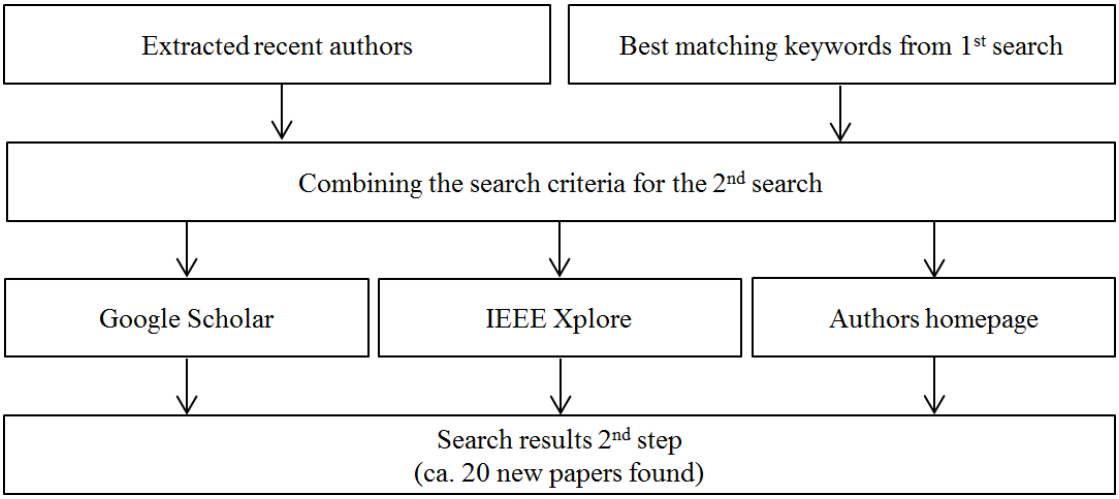


Figure 5.3: **Literature Search Step 2** – Illustration of the second step ending up with ca. 20 additional papers found.

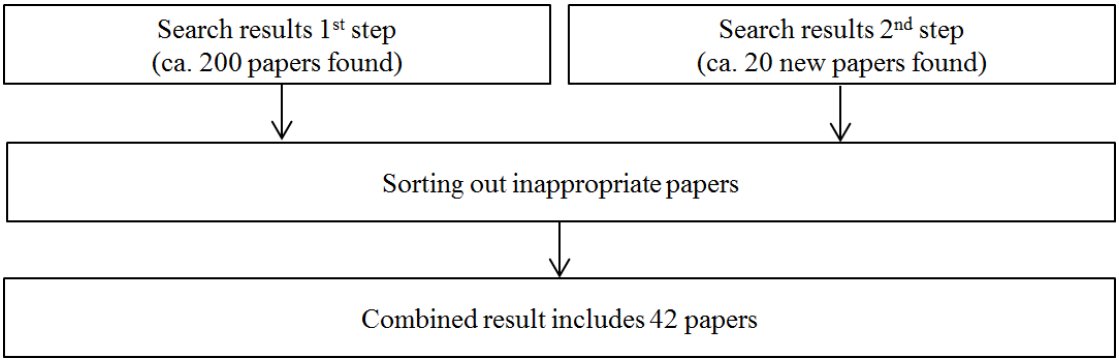


Figure 5.4: **Literature Search 1st Paper Reduction** – Illustration of the first process of paper reduction based on the first and the second search results.

In addition to the results of the search engines, we wanted to make sure to include all papers published at 'VizSec (Visualization for Cyber Security)' (see Figure 5.5) which is the premier venue for discussing malware visualization systems as it brings together security and visualization experts. To explore VizSec publications, we utilized our publicly-available search interface for VizSec papers (http://vizsec.dbvis.de/) and skimmed through the entirety of publications. In the end, we identified 3 additional papers directly related to malware (most had already been found earlier). Finally, we investigated all the references of the current paper collection to check whether there are any papers still undiscovered.

During a specific review phase (see Figure 5.6), eventually identified 25 papers out of the 45 papers matching our specific topic of malware visualization systems. Some

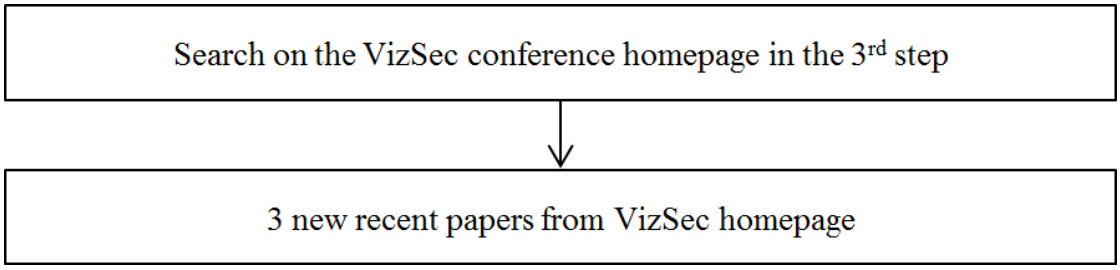


Figure 5.5: **Literature Search Step 3** – Illustration of the third paper search performed on the 'VizSec' home page.

papers present incremental work which leads to the fact that Quist and Liebrock (2009) is similar to Quist and Liebrock (2011), because it is an extension journal paper of the same system. Similarly, Han et al. (2014a) is related to Han et al. (2013), and Shaid and Maarof (2014) to Shaid, S.Z.M. and Maarof, M.A. (2014). However, we still decided to include all versions in the survey in order to present an extensive overview of all academic publications that are in the scope of this work.

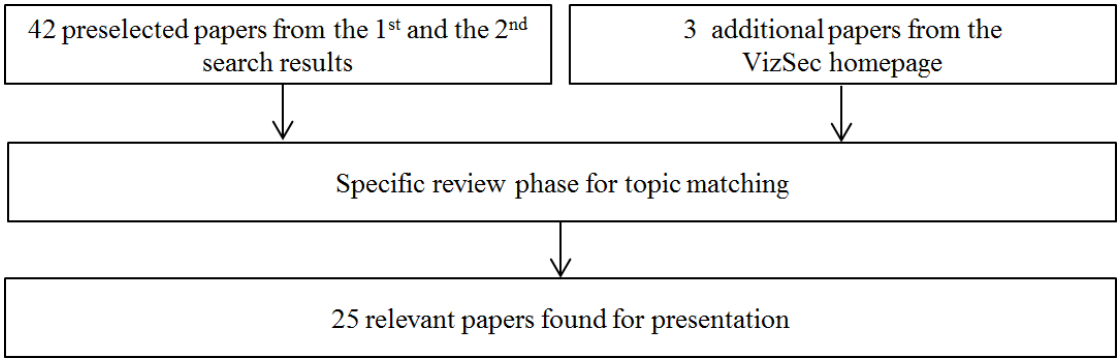


Figure 5.6: **Literature Search 2nd Paper Reduction** – Illustration of the second paper reduction process, focusing on papers matching our specific topic of malware visualization systems .

To classify and categorize the identified papers, we built an interactive web application to gather responses and collect reviews of all the members of our team. The web application directly connects to a shared Zotero collection using the Zotero API (Roy Rosenzweig Center for History and New Media, 2015). We decided on an extensive list of features and criteria to categorize and review the visualization systems. Two researchers extensively reviewed all the papers. The results were directly entered into our web application which stored them in a database and eventually synchronized them to the Zotero collection in form of tags. Afterwards, all criteria where no consensus was reached were discussed to agree on a common approach for their description.

The public part of the web application is available at http://malware.dbvis.de/ provided by our collaborators ('University of Konstanz') for the related publication (Wagner et al., 2015c). All tables in this survey can be interactively explored using the mentioned web application.

5.2 Visualization Systems for Malware Analysis

Based on our literature research, we identified various general trends and objectives prevalent in malware visualization systems. Using visualization obviously helps to understand malware behavior, which is helpful for forensics and 'malware detection'. Additionally, visual analysis can help to support the 'malware classification' process. Malware detection does mostly refer to the automatic identification of malware (e.g., anti-virus software for end users). However, in more complex scenarios, targeted attacks, or for unknown malware, manual analysis by malware experts is unavoidable. Such analysis helps to identify suspicious behavior, to eventually create rules and signatures, which can then be used to improve automated malware detection. Malware classification focuses on the aspect to assign an unknown malware sample to a known group of malware types.

Ultimately, we split the tools into four main categories representing the main goals of visualization support in malware visualization systems: (1) individual malware analysis, (2) comparison of malware images, (3) comparison of malware characteristics, and (4) malware summarization. Table 5.1 shows the categorization of the reviewed systems based on the aforementioned groups.

	(Yoo, 2004)	(Panas, 2008)	(Conti et al., 2008)	(Quist and Liebrock, 2009)	(Tirinius et al., 2009)	(Nataraj et al., 2011a)	(Grégio and Santos, 2011)	(Quist and Liebrock, 2011)	(Yee et al., 2012)	(Grégio et al., 2012)	(Zhuo and Nadjin, 2012)	(Saxe et al., 2012)	(Anderson et al., 2012)	(Paturi et al., 2013)	(Han et al., 2013)	(Wu and Yap, 2013)	(Kancherla and Mukkamala, 2013)	(Donahue et al., 2013)	(Shaid, S.Z.M. and Maarof, M.A., 2014)	(Han et al., 2014b)	(Han et al., 2014a)	(Shaid and Maarof, 2014)	(Gove et al., 2014)	(Wüchner et al., 2014)	(Long et al., 2014)
Individual Malware Analysis	-	-	✓	✓	✓	-	✓	✓	✓	-	✓	-	-	-	-	-	-	✓	-	-	-	-	-	✓	-
Malware Comparison ► Feature-Based Approach	-	-	-	-	-	-	-	-	-	✓	-	✓	-	-	-	-	-	-	-	-	-	-	✓	-	✓
Malware Comparison ► Image-Based Approach	-	✓	-	-	-	✓	-	-	-	-	-	-	-	-	-	✓	✓	-	✓	✓	-	✓	-	-	-
Malware Summarization	✓	-	-	-	-	-	-	-	-	-	-	-	✓	✓	✓	-	-	-	-	-	✓	-	-	-	-

Table 5.1: **Visualization Category** – An overview of our taxonomy and categories of visualization systems for malware analysis.

In general, there are two different main goals of malware visualization systems. On the one hand, there are systems for malware forensics, which are used to understand the individual behavior of a malicious malware sample and on the other hand, there are

malware classification tools, which are used to identify the common behavior of malware samples. Based on these main groups, we differentiate between three underlying main categories. We developed the 'Malware Visualization Taxonomy' (see Figure 5.7 and also Wagner et al. (2015c)) which represents the three categories:

Individual Malware Analysis: These systems support the individual analysis of primarily single malware samples to gain new insights of its individual behavior related to malware forensics.

Malware Comparison: This category fits to visualization tools that are primarily used for the comparison of multiple malware samples for the identification of common behavior (e.g., the malware family) to support malware classification. In general, we have identified two different subcategories:

- These tools that use a 'Feature-Based Approach' explore and compare different malware samples based on extracted features. Those tools use various data visualization techniques to compare characteristics with each other.
- Tools based on an 'Image-Based Approach' generate visual images based on binary data or the behavior logs of the malicious software. Eventually, those visual fingerprints are compared by using computer vision techniques.

Malware Summarization: Systems of this category summarize the behaviors of n different malware samples to identify similarities and to gain new insights of their common behavior.

As sketched in Figure 5.7, eventually, one or several malware analysis tools can be combined to generate rules and signatures for malware samples or malware families based on the generated insights. Additionally, the increasing use of visual analytics methods will enhance the forensics and classification methods for malware detection.

Discussion: From the taxonomy as seen in Figure 5.7, it becomes obvious that 9 tools focus on individual malware analysis, 11 on malware comparison, and 5 on malware summarization to provide visual summaries of large amounts of malware samples and their characteristics. Additionally, it is interesting to see that only 4 tools for malware comparison are using primarily the feature-based approach, while 7 focus on image-based approaches.

Based on the various publication years (see Figure 5.8), it becomes apparent that using malware characteristics (based on features extracted through static and dynamic malware analysis) is becoming more common since 2013 and that fewer systems focus

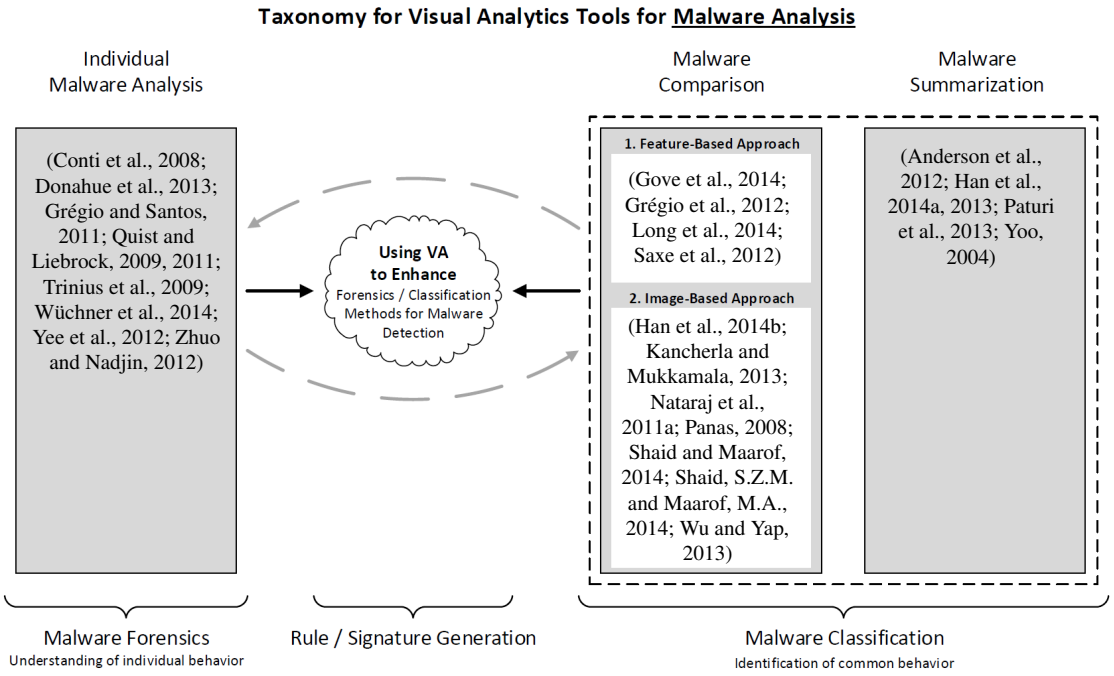


Figure 5.7: **Malware Visualization Taxonomy** – Categorization of malware visualization systems into three categories, namely (1) Individual Malware Analysis, (2) Malware Comparison, and (3) Malware Summarization. All systems have the ultimate goal to generate rules and signatures for fully-automated malware detection systems. While the first category tackles the problem of understanding the behavior of an individual malware sample for forensics. The latter two focus on the identification of common behavior for malware classification.

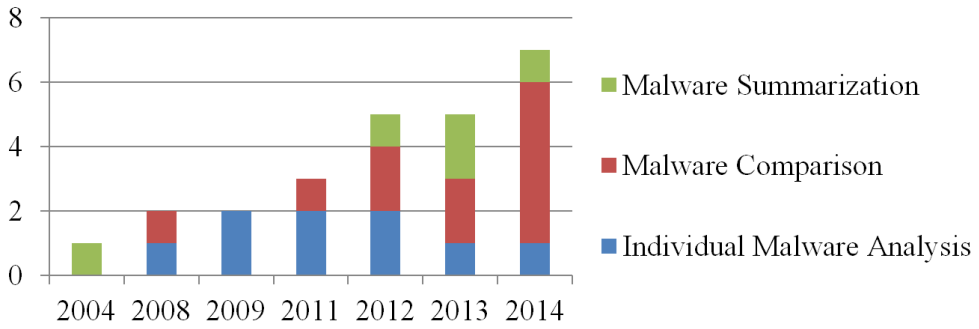


Figure 5.8: **System Types over Time** – Illustration of the three categories of malware visualization systems and their focus over the years.

on individual malware analysis (malware forensics). Most of the research for individual

malware analysis was performed between 2004 and 2012. In the past 10 years, visualization seems to be used more often to generate image-like representations of malware samples which are then used for visual comparisons.

Next, we are describing the three identified types of malware visualization systems (Individual Malware Analysis, Malware Comparison and Malware Summarization) in detail in combination with the presentation of further research directions for each category.

Visualization for Individual Malware Analysis

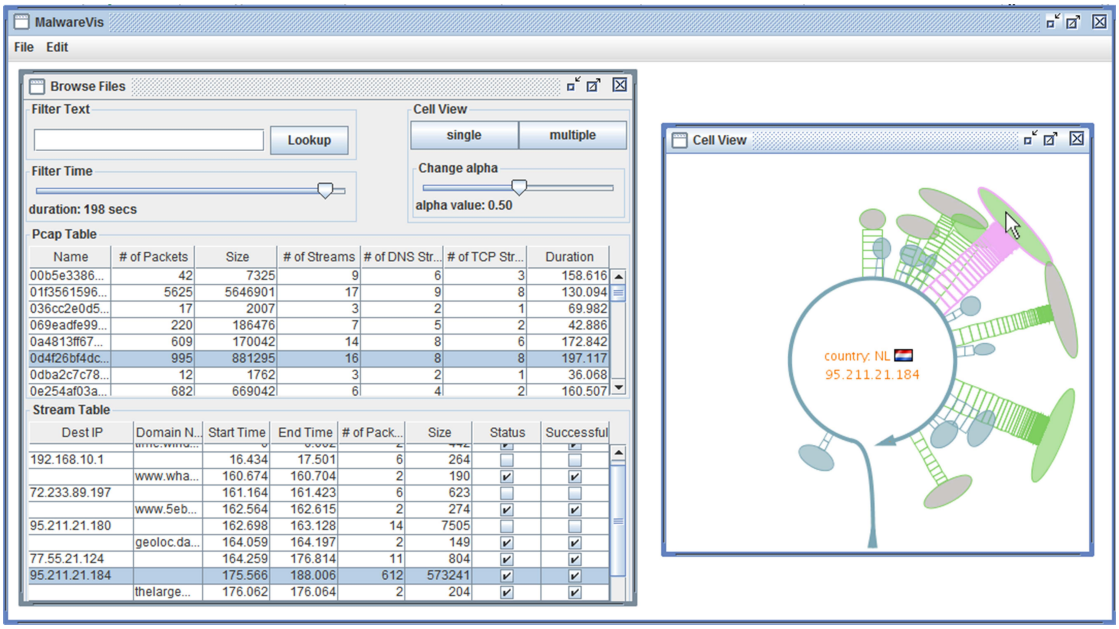


Figure 5.9: **Individual Malware Analysis** – This interactive system visualizes network activity of an individual malware sample (Zhuo and Nadjin, 2012).

The first group contains visualization systems geared towards the extensive analysis of 'individual' malware samples (Conti et al., 2008; Donahue et al., 2013; Grégio and Santos, 2011; Quist and Liebrock, 2009, 2011; Trinius et al., 2009; Wüchner et al., 2014; Yee et al., 2012; Zhuo and Nadjin, 2012).Zhuo and Nadjin (2012), for example, focus on only one specific type of malware behavior – the network activity of a malware sample – which is then visualized by a glyph-like chart as can be seen in Figure 5.9. This specific feature can be explored in detail which is not possible in other, less specialized visualization tools.

Other tools consider various features at the same time, but still focus on the individual analysis of single malware samples. Trinius et al. (2009) use treemaps and so-called

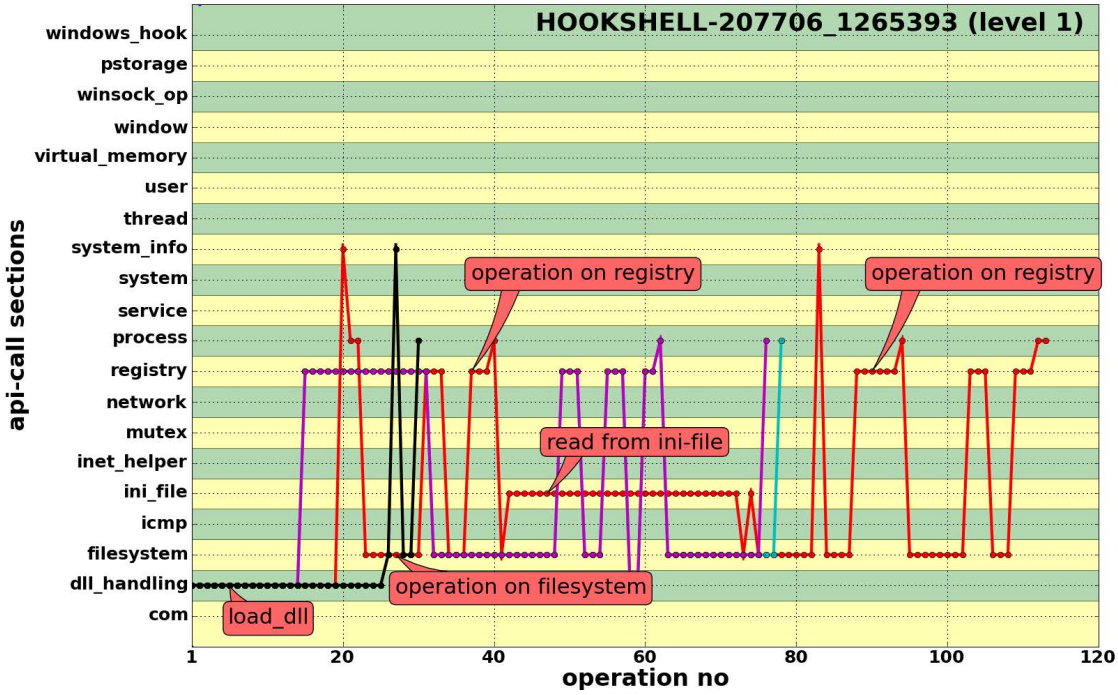


Figure 5.10: **Individual Malware Analysis** – Visual representation of system calls issued over time by an individual malware sample (Trinius et al., 2009).

thread graphs, as seen in Figure 5.10, to visually analyze system calls executed by the selected malware. While basic comparison is also possible with most of the tools in this category (e.g., using multiple instances of the same tool), they do not specifically support bulk analysis. This lack of functionality becomes apparent when the analyst attempts to compare hundreds of malware samples at the same time.

Future Research Directions: The visual analysis of individual malware samples leads the analyst to a better understanding of the specific behavior and can help to judge if an unknown sample is indeed malicious or not. However, current work could be improved with respect to malware detection. Because many of those tools do not include classification methods to compare the observed behavior to the behavior of known malware types. In the future we expect more visual analytics tools to combine individual malware analysis with automated methods and to incorporate methods to directly relate and compare findings with the behavior of known or previously analyzed samples. Automatic highlighting of important or possibly malicious aspects, would help the analyst to quickly focus on most suspicious behavior, first to reduce the time which is needed for manual analysis.

Visualization Support for Malware Comparison

While the individual analysis is needed to get a deep understanding of a malware samples, the comparison with already known malware samples is crucial for malware classification. On the one hand, this step helps to reduce the number of samples that need time-consuming manual analysis. On the other hand, comparison with other samples can help to identify groups or malware families. All the systems which are represented in this category use visualizations to enhance the comparison of n with m malware samples for the identification of their common behavior (e.g., to identify related samples, find the correct malware family). Technically, we distinguish between feature-based and image-based approaches.

Feature-Based Approach

Figure 5.11: **Comparison of Malware Characteristics** – Identifying similar malware samples to a focus on a sample by comparing them along different sets of characteristics (e.g., capabilities) (Gove et al., 2014).

Feature-based approaches (Gove et al., 2014; Grégio et al., 2012; Long et al., 2014; Saxe et al., 2012) use visual analytics techniques to let the user filter, search, compare, and explore a wide range of properties extracted during analysis. These systems provide means to compare malware samples based on their similarities of features.

Individual exploration of these features is also possible, but is much more limited, compared to the previous category. While some of the tools of the previous category were specifically designed to do an in-depth analysis of network activity or to fully explore the temporal sequence of system calls. Feature-based malware comparison tools try to focus on a broad set of different features and characteristics, and try to make them all accessible to the analysts. This leads to more abstract representations, higher aggregation levels, and eventually less details for individual features (e.g., ignoring the temporal aspects of network connectivity).

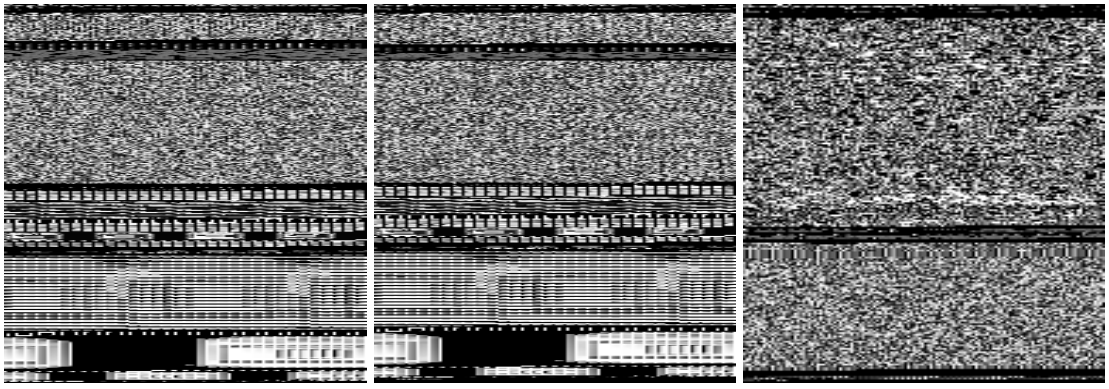
Figure 5.11 shows a screen shot of a visual analytics system by Gove et al. (2014) used to interactively explore and compare large sets of characteristics or attributes of samples in malware corpora. The approaches advantage is that the analyst can directly compare various features. This helps to understand in which features malware binaries are related and in which they are not. However, on the other hand it is harder to get a quick visual overview of occurring patterns.

Future Research Directions: The comparison of characteristics helps to visually enhance the malware classification process in various ways. Tools in this category also focus on the question of which features can be extracted and used for comparison. Comparing such malware characteristics helps to identify related samples based on similarity metrics and to identify the common behavior of the explored samples for classification. Especially, the possibility to compare many different features at once and the possibility to apply standard methods from the field of data analysis (e.g., MDS, PCA, clustering) opens a promising research direction. Using visual interfaces to guide the analyst in the selection of features seems to be a good way to better support malware classification. Such visual analytics interfaces would eventually help to define better classifiers to improve malware classification models.

Image-Based Approach

Image-based approaches (Han et al., 2014b; Kancherla and Mukkamala, 2013; Nataraj et al., 2011a; Panas, 2008; Shaid and Maarof, 2014; Shaid, S.Z.M. and Maarof, M.A., 2014; Wu and Yap, 2013) have in common that they use visual mappings to render an image for each malware sample.

For example, the analyst might need to correlate a given suspicious file to a cluster of malware variants in order to associate the file to a specific malware family. Similar images can be visually clustered using either a manual or an automatic approach based on algorithms from the areas of computer vision and image processing. Some systems vi-



(a) FakeRean.D (b) FakeRean.E (c) Mebroot

Figure 5.12: **Comparison of Malware Images** – Visualizing malware executables as grayscale images is a common technique to visually identify similarities with low computation costs.

sualize the binary data and directly map the (raw) byte-code representation or respective entropy values to an image (e.g., (Han et al., 2014b; Nataraj et al., 2011a)). We applied this technique to variants of the 'FakeRean' malware as seen in Figure 5.12a. We use this to detect similar images representing related malware samples (Figure 5.12b). These particular malware samples can be visually distinguished from Figure 5.12c, which represents a 'Mebroot' malware sample, sharing no visual patterns with the other malware family.

Nataraj et al. (2011b) extract various texture features from such images, to eventually use them for classification. The advantage of this technique is, that it can be applied to any file and can be computed efficiently, which is important for large malware corpora. While classification accuracy is quite comparable for many malware variants, the approach is limited because it does not make use of any dynamic analysis and only relies on the actual bytes found in the binaries. Another problem is, that the visual impression is strongly dominated by possible images embedded in the resource section of an executable, which could be avoided by malware authors to create less obvious visual patterns.

To overcome this drawback, the approach was extended to visualize disassembled CPU instructions or API calls (e.g., (Panas, 2008; Shaid and Maarof, 2014; Shaid, S.Z.M. and Maarof, M.A., 2014)) in a similar way, however, resulting in higher computation costs.

Future Research Directions: One possible future research direction could be the implementation of interaction methods to segment a region of interest or to characterize these texture patterns. Automated image comparison would help analysts to visually

identify common code portions or specific instruction blocks within a sample. This information could be used to directly highlight relevant sections in the image. Additionally, the integration and combination of image- and feature-based methods could be promising. Image-based methods using static analysis together with a probability score can be used as efficient first step in a classification pipeline. Afterwards, the more expensive feature-based methods together with dynamic analysis would only be applied to those samples, which share less distinctive image representations, eventually leading to a more scalable classification process.

Visualization Support for Malware Summarization

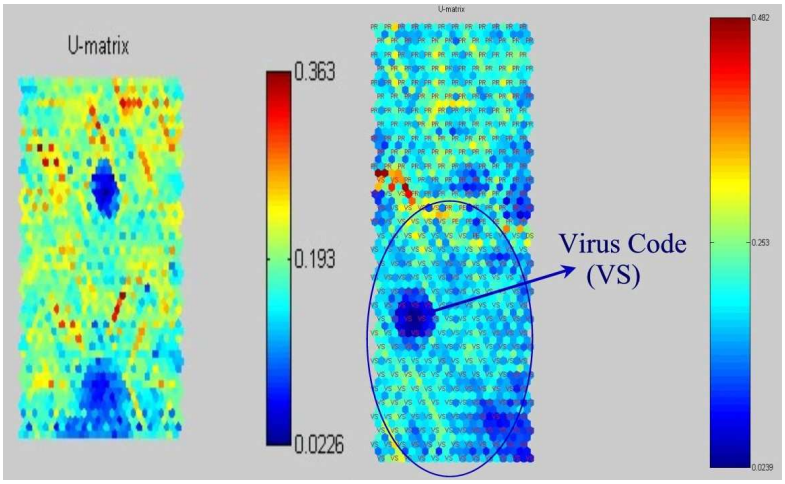


Figure 5.13: **Visualization Support for Malware Summarization** – A self-organized map is calculated and visually represented by the system to summarize many malware variants to extract common regions. With this technique it is possible to create a topologically ordered data mapping in terms of a discretized main surface or curve (Yoo, 2004).

While this category is more diverse, the associated tools all provide primarily some kind of summarization capability for a large number of malware samples within the visualization (Anderson et al., 2012; Han et al., 2014a, 2013; Paturi et al., 2013; Yoo, 2004). Some identify a visual mask that is common for all selected samples (e.g. (Yoo, 2004)) as seen in Figure 5.13. Others summarize and extract a single combined representative out of many malware variants (e.g., (Han et al., 2014a, 2013)). Finally, others use visual representations to show hierarchical clusters (Paturi et al., 2013) or use heatmaps to visually represent kernels used for a support vector machine classifier to summarize and eventually classify malware samples (Anderson et al., 2012).

Future Research Directions: The combination of different types of base data and data provider analysis modes are frequently stated as future work in this category. This will result in larger amounts and more heterogeneous data as input for visualization systems. Another direction into larger amounts of data can be the comparison of malware families as a whole based on their summarization. Finally, the integration of malware summarization with malware comparison and malware forensics using semantic zoom which is for example a promising direction. One direction could be to find out virus pattern similarities. This should not only focus on the identification of same virus families but also to the generation of virus masks based on the infected file structures. A second interesting research direction is the investigation of a unified data gathering system which combines the data of different data sources from static and dynamic analysis tools. With such a system it will be possible to compare different instruction sequences and to create algorithms for automatic malware classification and similarity analysis. Based on similarity measures it could be possible to detect mobile malware different malware variants and/or similar structures.

5.3 Categorization & Comparison

To provide a systematic overview of the findings from our literature research, we decided to consistently categorize all tools by the type of provided data, used visualization techniques (Keim, 2002), mapping and representation space (Aigner et al., 2011), temporal aspects (Aigner et al., 2011), interactivity, and problems/actions ("*Why?*") (Munzner, 2014). Thus, all the used categorizations are based on well-established taxonomies used in the visualization community and are described in detail in this section.

Visualization Techniques

For the categorization of the different visualization techniques we used the "*Information Visualization and Data Mining*" taxonomy by Keim (2002). More precisely, we focused on the part discussing visualization techniques. Based on this taxonomy it is possible to divide the used techniques into 5 generalized categories which will be described in the followings:

Standard 2D/3D Displays includes visualization techniques like 'x-y (x-y-z) plots' (e.g., scatter plots), 'bar charts' and 'line graphs' for example (Keim, 2002).

Geometrically-transformed Displays are aiming to interesting transformations of multidimensional datasets. This group includes 'scatter plot matritzes' (e.g., (Andrews, 1972)) and techniques which can be summarized as 'projection pursuit' (Huber, 1985) which are techniques from exploratory statistics. Additionally, this group includes geometric projection techniques like 'prosection views', 'parallel

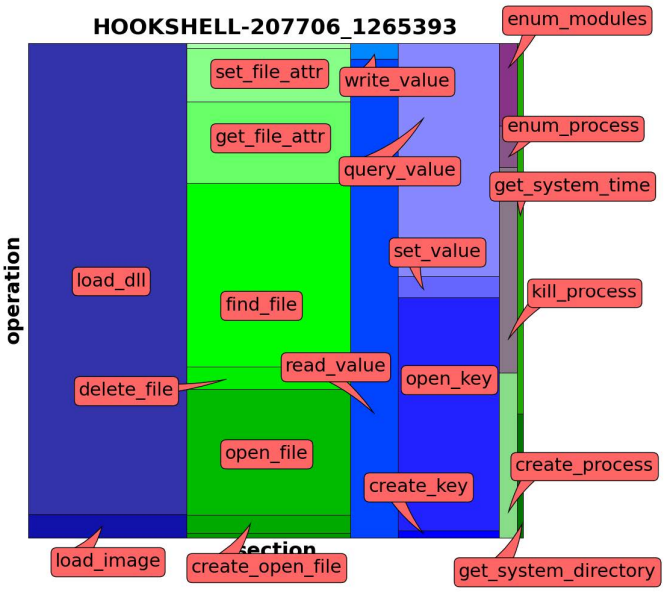


Figure 5.14: **Malware Treemap** – Visualization providing an overview of the most frequently used system call operations for an individual malware sample. It is evident that this sample uses calls from 6 out of 20 sections with operations from the 'dll handling' and 'file system'sections occuring most often (Trinius et al., 2009).

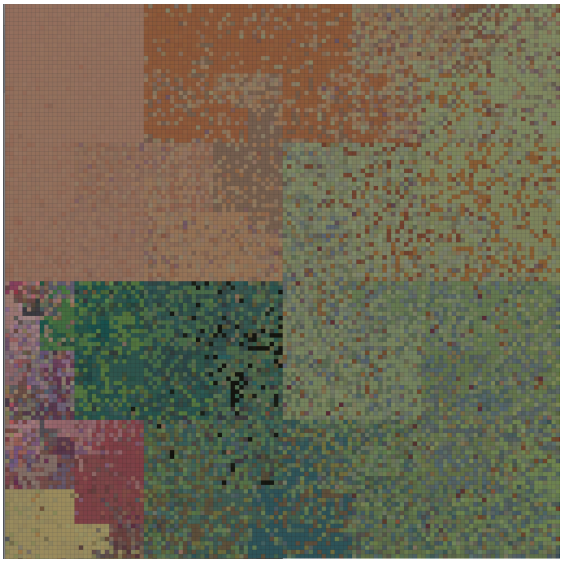


Figure 5.15: **Dense Pixel Displays** – Each pixel shape represents a malware sample. Similar malware samples are arranged next to each other and assigned similar color values to visualize commonalities (Saxe et al., 2012).

coordinates' as described by Keim (2002) and 'stardinates' (Lanzenberger et al., 2005). By the use of the parallel coordinates technique, the k-dimensions will be mapped onto the two display dimensions using k equidistant display axis which are parallel on one of the display axis.

Iconic Displays are mapping the attributes of a multidimensional data items onto the features of an icon for the representation. These icons can be defined in many different ways. It is possible to use 'small faces' (e.g., chernoff faces (Chernoff, 1973)), 'needle icons', 'star icons', 'sticky figure icons' (Pickett and Grinstein, 1998), 'color icons' and 'tile bars' for example. To generate the visualization, all the attributes of the data item will be mapped/transformed to the features of the used icon. By the use of sticky figure icons, 2 dimensions will be mapped to the 2D position of a glyph and all the other data values will be mapped to the angles and/or the limb length of the icon for example (Keim, 2002). Additionally, by the use of small faces, 2 dimension will also be mapped to the position and the other data attributes will be mapped to the face shape and elements.

Dense Pixel Display has the basic idea to map each dimension pixel to a colored pixel. These pixels will be grouped into adjacent areas belonging to the dimension. Based on the use of one pixel per data value, this visualization technique allows the largest amount of visualized data belonging to the available display dimensions. One of the main questions is to find an efficient way for the arrangement of the pixel to find hotspots in the data. Examples for a good arrangement of the data are the circle segmentation technique (Ankerst et al., 1996), the recursive pattern technique (Keim et al., 1995), the 'Peano-Hilbert curve' and the 'Morton curve' as shown by Keim (2000). Additionally, 'matrix' visualizations will be situated in this area.

Stacked Displays are a traditional approach to represent hierarchical data in a partitioned way. In the case of multidimensional or high dimensional data, it is necessary to select the right/needed data dimension for the space partitioning. Examples for stacked displays are 'dimensional stacking' (LeBlanc et al., 1990), 'hierarchical stacking', 'treemaps' or 'neighborhood treemaps' (Duarte et al., 2014) which are also called 'Nmaps'. The basic idea of this visualization technique is to integrate one coordinate system into another coordinate system. In the first step the outer coordinate system will be divided into rectangular cells for the representation of the selected top-level categories/data. In the next step the inner coordinate system of each generated rectangle will be divided into rectangles to integrate the contained data from the top-level area and so on.

Discussion: Our findings are summarized in Table 5.2. It is interesting that stacked displays and iconic displays are not commonly used in this domain. More research in

	(Yoo, 2004)	(Panas, 2008)	(Conti et al., 2008)	(Quist and Liebrock, 2009)	(Trinius et al., 2009)	(Nataraj et al., 2011a)	(Grégio and Santos, 2011)	(Quist and Liebrock, 2011)	(Yee et al., 2012)	(Grégio et al., 2012)	(Zhuo and Nadjin, 2012)	(Saxe et al., 2012)	(Anderson et al., 2012)	(Paturi et al., 2013)	(Han et al., 2013)	(Wu and Yap, 2013)	(Kancherla and Mukkamala, 2013)	(Donahue et al., 2013)	(Shaid, S.Z.M. and Maarof, M.A., 2014)	(Han et al., 2014b)	(Han et al., 2014a)	(Shaid and Maarof, 2014)	(Gove et al., 2014)	(Wüchner et al., 2014)	(Long et al., 2014)
Standard 2D Display	-	-	-	✓	✓	-	✓	✓	-	✓	-	-	-	-	-	✓	-	✓	✓	✓	-	✓	✓	✓	✓
Standard 3D Display	-	✓	-	✓	-	-	-	✓	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Geometrically-transformed Display	✓	✓	-	✓	-	-	✓	✓	✓	-	✓	-	-	✓	-	-	-	-	-	-	-	-	-	✓	✓
Iconic Display	-	-	-	-	-	-	-	-	-	✓	✓	✓	-	-	-	-	-	-	-	-	-	-	✓	-	-
Dense Pixel Display	✓	-	✓	-	-	✓	-	-	-	-	-	✓	✓	-	✓	-	✓	✓	✓	✓	✓	✓	✓	-	-
Stacked Display	-	-	-	-	✓	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	✓	-	-

Table 5.2: **Visualization Techniques** – A general overview of the most frequently used types of visualization techniques based on the taxonomy of Keim Keim (2002).

appropriate glyph design seems to be promising because of the compactness of such visualization techniques. Most analysis support tools use standard 2D displays. Trinius et al. (2009) use treemap representations to analyze system call operations for individual malware samples, as seen in Figure 5.14. Interestingly, a total number of 13 tools use visualization techniques which fall into the category of dense pixel displays. The reason for this is that a wide range of tools depict malware samples as image-like representations (Figure 5.12) which can already be interpreted as dense pixel displays. Dense pixel displays are also used to convey similarities between malware samples as can be seen in Figure 5.15 (Saxe et al., 2012).

Mapping to Time & Representation Space Dimension

While a large variety of visual representations are possible for analysis of malware data, we can categorize these by two fundamental dichotomies: the dimensionality of the representation space and whether physical time is used to convey data Aigner et al. (2011).

Mapping to Time

Time-oriented data have to pass the mapping step of the visualization pipeline like any other data variables that have to be visualized. Thereby the data will be made visual comprehensible likewise by mapping to some geometry and visual attributes (e.g., shapes and colors) in the presentation space. To visualize the dynamics of time, the data could be mapped to dynamics of a visual representation for example. In practically there are two different types of mapping time for the visual representation. On

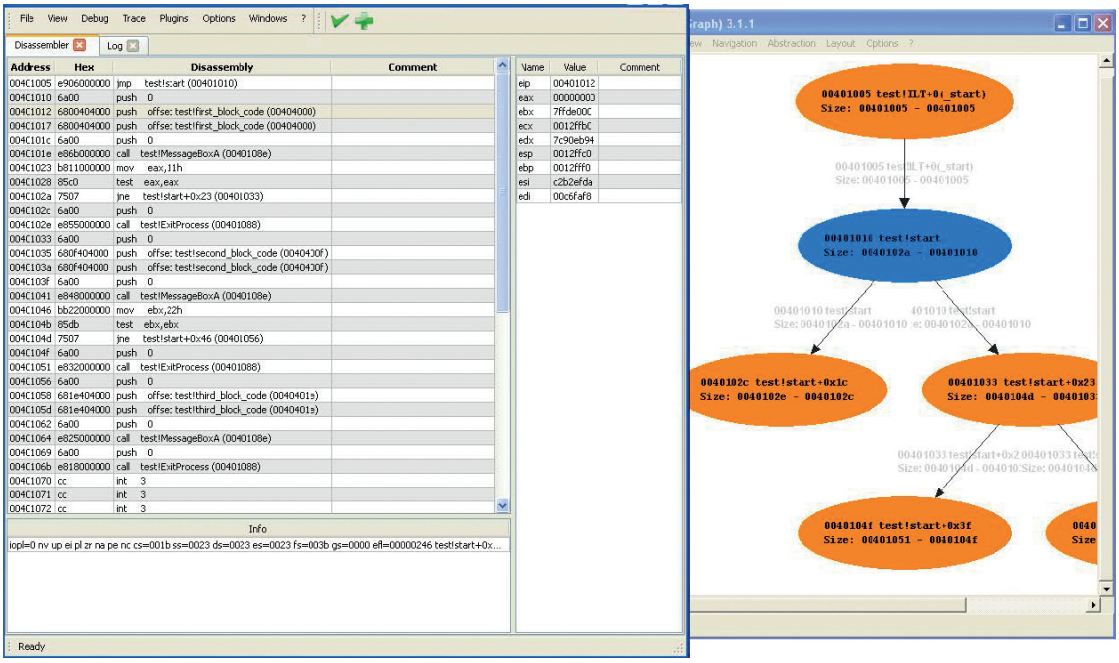


Figure 5.16: **Dynamic Mapping** – A visual debugger for malware analyis which is using node-link diagrams with replay capabilities to show the execution flow of a malware sample over time (Yee et al., 2012).

the one hand, there is the 'static' mapping, which means that the time and the data are mapped to a coherent space which does not change over time. 'Dynamic' mapping on the other hand, maps the time as real time elements in the visual representation. With this mapping type, it is possible to change the visualization over time like slide shows or animations (Aigner et al., 2011).

Static: There exist many ways for the mapping of time to visual variables. Visual representations which uses the static mapping of time, includes a time axis. Thereby, most of the time the x-axis is used as the time axis, and the time depending variables will be situated along the y-axis. If more than one axis would be used for the representation of the time, the visualization would become more and more complex. To show the different granularities of the time (e.g., years, quarters, weeks, days, and so on), the time axis will be subdivided for the illustration in a hierarchical way. Additionally, one further aspect is that the time is mapped to a coherent space (Aigner et al., 2011).

Dynamic: If the visualization of the data requires a lot of space like, visualization of multivariate data, graph structure visualization etc., it is very difficult to embed an own time-axis into the display space. Therefore, as alternative, the physical

time could be used to encode time, whereby many visualization frames will be rendered based on the time steps in the data. Theoretically, it is possible to implement a dynamic visualization based on real time which means a "*time-to-time mapping between time and frames*" (Aigner et al., 2011). However, in practice, this is not very often possible because the time steps are too big for example. Therefore, an interpolation will be made to calculate intermediate time points between the given time steps from the data.

Representation Space

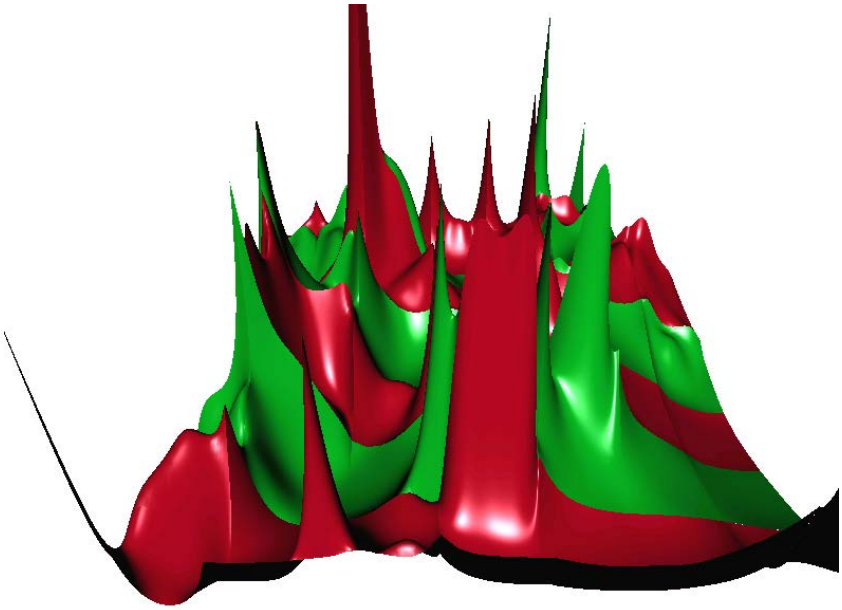


Figure 5.17: **3D Visualization** – Panas proposes 3D visualization to generate a specific visual signature that helps to identify anomalies within a malware family. For the data visualization, Panas mapped the number of control transfer instructions (x-axis), the number of data transfer instructions (y-axis) and the number of instructions (z-axis) (Panas, 2008).

In general, the representation space for a visualization can be either '2D' (two-dimensional) or '3D' (three-dimensional). In the community there exist different opinions about which one of the representation approaches is better. One group argues that '2D' visualizations are better for the data analysis because the third dimension makes it more difficult. The other group argues that two dimensions are not enough for the visualization of complex data sets, because with the third dimension it is possible to use one more natural dimension for the data representation (Aigner et al., 2011). However,

the main question is not which visual representation dimensionality should be used, but rather what are the analytical goals to be achieved with the data to be visualized.

2D Visualizations addresses to the two dimensions (x-axis and y-axis) of a computer display, whereby all visual elements are described with respect to these coordinates. Useable graphical elements in the '2D' space are: dots, lines, circles and arcs for example. The time axis will be mapped to one of the two axis of the display in the case of a '2D' visualization, but this is not always necessary (Aigner et al., 2011).

3D visualizations additionally uses the third dimension (the z-axis) for the representation of a geometry, whereby the visualization gets more complex by the use of volumetric structures. The perception of a human is naturally turned into a '3D' world. Based on this fact, '3D' representations are potentially communicating such structures on a better way than '2D' representations for humans. A computer display only contains two dimensions, so projections are required to generate the z-axis for '3D' rendering (Aigner et al., 2011).

	(Yoo, 2004)	(Panas, 2008)	(Conti et al., 2008)	(Quist and Liebrock, 2009)	(Trinius et al., 2009)	(Nataraj et al., 2011a)	(Grégio and Santos, 2011)	(Quist and Liebrock, 2011)	(Yee et al., 2012)	(Grégio et al., 2012)	(Zhuo and Nadjin, 2012)	(Saxe et al., 2012)	(Anderson et al., 2012)	(Paturi et al., 2013)	(Han et al., 2013)	(Wu and Yap, 2013)	(Kancherla and Mukkamala, 2013)	(Donahue et al., 2013)	(Shaid, S.Z.M. and Maarof, M.A., 2014)	(Han et al., 2014b)	(Han et al., 2014a)	(Shaid and Maarof, 2014)	(Gove et al., 2014)	(Wichner et al., 2014)	(Long et al., 2014)
Mapping ► Static	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Mapping ► Dynamic	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-
Dimensionality ► 2D	✓	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Dimensionality ► 3D	-	✓	-	✓	-	-	-	✓	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 5.3: **Representation Space & Mapping** – An overview of used representation spaces in visualization systems. Almost all tools focus on static mappings.

Discussion: Table 5.3 summarizes the findings with respect to representation space and mapping to time. Obviously, most of the tools focus on static mapping. The reason for this might be that many of the tools use base data that does not consider chronological order. Another reason might be that a dynamic representation makes it harder for the analyst to focus on specific characteristics gleaned through static analysis. Therefore, most systems provide a static mapping. Only the tool by Yee et al. (2012) uses a more

dynamic mapping. They provide a visual debugger with node-link diagrams which can be used to replay the control flow of a malware sample (Figure 5.16). On the representation space side, 4 out of 25 tools map the data to a 3D representation space in addition to a 2D visualization. The remainder utilizes 2D representation only. Only one tool (Panas, 2008) (see Figure 5.17) solely uses 3D representation of the analysis data. Additionally, only two tools ((Yee et al., 2012) and (Wüchner et al., 2014)) provide a static and dynamic mapping to time. All the remaining tools only provide static mapping. In contrast to static mapping, physical time (dynamic mapping) can be used to encode data. Therefore, several frames will be rendered for the time steps in the data so that a 1:1 mapping could be implemented between time steps and frames. However, in practice this is not always realizable.

Temporal Aspects

Time-oriented data plays an important role in malware analysis: For example, the 'execution order of system or API calls' is relevant to identify certain behavior patterns (such as the creation and subsequent deletion of files). The time passed between two specific calls could also be of importance. Time can be modeled in different ways depending on analysis goals. For our categorization, we use a selection from time-oriented data aspects introduced by Aigner et al. (2011).

First there is to make a clear distinction between the physical time and the time in information visualization systems. In information visualization systems, the main goal is to provide a suitable reflection of the time to support the analysis task as much as possible and not to perfectly imitate the physical time. There was performed extensive research in different fields of computer science to formulate the notion of time (e.g., simulation, data mining, artificial intelligence and many more). Aigner et al. "*presented the overall aspects of modeling time, and not a particular model*" (Aigner et al., 2011). Therefore, they described a number of design aspects which are important to modeling time aspects.

Scale

Based on the 'scale' aspect, they will take a look at the time by the elements of the given model. Therefore, they differ between 3 types which are the 'ordinal', the 'discrete' and the 'continuous' time scale.

Ordinal: The 'ordinal' time domain represents only relations or relevant time aspects (e.g., before, after). For example, *Thread x was set to sleep before API call y was called* and *Thread x woke up after some seconds of sleep* could be modelled by using an 'ordinal' time scale. It is important to note that only relative statements can be given and it is not possible to differ in the given example if thread x woke

up before or after the API call y. This time primitive might be adequate if only qualitative temporal relationships are the area of interest (Aigner et al., 2011).

Discrete: Discrete time domains are also able to represent distances. Thereby, the modelling of quantitative time values will be enabled by the mapping of the time values to a set of integers. Aigner et al. (2011) described that "*discrete time domains are based on a smallest possible unit (e.g., seconds or milliseconds as in UNIX time) and they are the most commonly used time model*" in information visualization systems. For example, thread x woke up in the same second as the API call y was executed, but it is not possible to exactly order the events.

Continous: With this type of time model, a mapping to real numbers is possible. For explanation, between any two time points, there is existing another point in time (Aigner et al., 2011). Based on the example of the thread and the API call, now it is possible to say exactly which event comes first and how much time was left in between.

Arrangement

In the next design aspect, Aigner et al. (2011) took a look at the 'arrangement' of the time domain. Therefor they decided to divide the 'arrangement' into 2 types which are called the 'linear' and the 'cyclic' form.

Linear: The 'linear' form corresponds to our perception of time, from the past to the future (like a timeline) whereby each element has a predecessor and a successor. The 'linear' arrangement is very common in all types of provided data (e.g., monthly averages and many more) (Aigner et al., 2011).

Cyclic: If the data is composed of a set of recurrent time values, they are talking about 'cyclic' arrangement (e.g., the 4 seasons of the year). For example, a time value A is preceded and succeeded by any other time value B (Aigner et al., 2011).

Granularity and Calendars

To describe the complexity of time, Aigner et al. (2011) described 3 different abstraction levels of granularity ('none', 'single' and 'multiple'). Based on the level of granularity, time values can be mapped to larger or smaller conceptual units.

None: If time values are not mapped (divided) by any kind of granularity (e.g., years, quarters, months and so on), then the system will have 'none' granularity. This means for example that the time values are only ordered or divided by abstract ticks (Aigner et al., 2011).

Single: A ‘single’ granularity describes the mapping of the time values to only one type of granularity (e.g., years, quarters, months and so on) (Aigner et al., 2011). An example of this is when the granularity of the time values is only given in milliseconds.

Multiple: With the mapping to ‘multiple’ granularities, it is possible to divide the time values into years, quarters, months and so on, which means that the structure refers to a calendar. In particular, this means that the time values are mapped to human-meaningful time values which include mappings between pairs of granularity. These pairs can be represented as a graph. In the common Gregorian calendar, dates are expressed as <day, month, year> triple, were each of the fields cycle as time passes (Aigner et al., 2011).

Time primitives

To relate the data to the time, Aigner et al. (2011) presented a set of basic elements which are called ‘time primitives’. These ‘time primitives’ are divided into 3 elements which are called ‘instant’, ‘interval’ and ‘span’ which are divided into 2 groups (absolute and relative primitives). ‘Instant’ and ‘interval’ are related to absolute primitives. They are located on a fixed position on the time domain. In contrast the ‘span’ is a relative primitive because it has no absolute position in time. Additionally, the definition of time primitives is possible at all levels of granularity.

Instant: A single point in time is called an ‘instant’ like January 16, 2015. Additionally, depending on the scope, if an ‘interval’-based or a point-based time model is used, an ‘instant’ can also contain a duration (Aigner et al., 2011). Examples of an ‘instant’ are the beginning of a malware analysis or the discovery of a new malware.

Interval: An ‘interval’ is a part of a time domain which will be represented based on two ‘instants’ which marks the beginning and the end of the ‘interval’. Additionally, an ‘interval’ could also be modelled by the combination of a ‘instant’ and a positive ‘span’ (duration). In this case the ‘instant’ defines the start or the end point (Aigner et al., 2011).

Span: The ‘span’ is an unanchored primitive which represents a directed duration of time (e.g., 4 hours) in terms of a number of granularities in a given granularity. For example a ‘span’ of 4 days is a count of 4 granularities of the type hour. A ‘span’ could be positive or negative. A positive ‘span’ denotes the forward motion of time and a negative ‘span’ denotes the backwards motion of time. In the case of irregular granularities like years, the precisely length of the span is not known

because a ‘span’ of 2 years could include 365 or 366 days depending on the particular time context (Aigner et al., 2011). In this case, the exact length could only be termined exactly, if it is related to an anchored time domain.

	(Yoo, 2004)	(Panas, 2008)	(Conti et al., 2008)	(Quist and Liebrock, 2009)	(Trinius et al., 2009)	(Nataraj et al., 2011a)	(Grégio and Santos, 2011)	(Quist and Liebrock, 2011)	(Yee et al., 2012)	(Grégio et al., 2012)	(Zhuo and Nadjin, 2012)	(Saxe et al., 2012)	(Anderson et al., 2012)	(Paturi et al., 2013)	(Han et al., 2013)	(Wu and Yap, 2013)	(Kancherla and Mukkamala, 2013)	(Donahue et al., 2013)	(Shaid, S.Z.M. and Maarof, M.A., 2014)	(Han et al., 2014b)	(Han et al., 2014a)	(Shaid and Maarof, 2014)	(Gove et al., 2014)	(Wüchner et al., 2014)	(Long et al., 2014)
Scale ► Ordinal	-	-	-	✓	✓	-	✓	✓	✓	✓	✓	-	-	-	✓	-	-	-	✓	-	✓	✓	-	-	-
Scale ► Discrete	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-
Scale ► Continuous	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Arrangement ► Linear	-	-	-	✓	✓	-	✓	✓	✓	✓	✓	-	-	-	✓	-	-	-	✓	-	✓	✓	-	✓	-
Arrangement ► Cyclic	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Granularity ► None	-	-	-	✓	✓	-	✓	✓	✓	✓	-	-	-	-	✓	-	-	-	✓	-	✓	✓	-	-	-
Granularity ► Single	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Granularity ► Multiple	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-
Time primitives ► Instant	-	-	-	✓	✓	-	✓	✓	✓	✓	-	-	-	-	✓	-	-	-	✓	-	✓	✓	-	✓	-
Time primitives ► Interval	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	✓	-
Time primitives ► Span	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 5.4: **Temporal Aspects** – An overview of used time primitives. It is interesting to see that only 12 of the reviewed systems focus on temporal aspects, while the others do not specifically focus or do not convey temporal aspects of the malware behavior in the visual representations.

Discussion

Interestingly, only 12 out of 25 presented tools use temporal aspects. All these 12 tools have a ‘linear’ arrangement, 11 use an ‘ordinal’ time scale (see Table 5.4) and only the tool by Wüchner et al. (2014) uses a ‘discrete’ time scale. Only 2 tools use an ‘interval’ based time primitive, whereby the tool by Zhuo and Nadjin (2012) uses a ‘single’ granularity (as seen in Figure 5.9) and the tool by Wüchner et al. (2014) uses ‘multiple’ granularities. The remaining 10 visualization systems use ‘instants’ as time primitives and do not feature any granularity.

Interactivity

For the categorization if a presented system supports interaction, we decided to check, if it is possible to use functionalities like zooming, filtering, panning, details on demand or

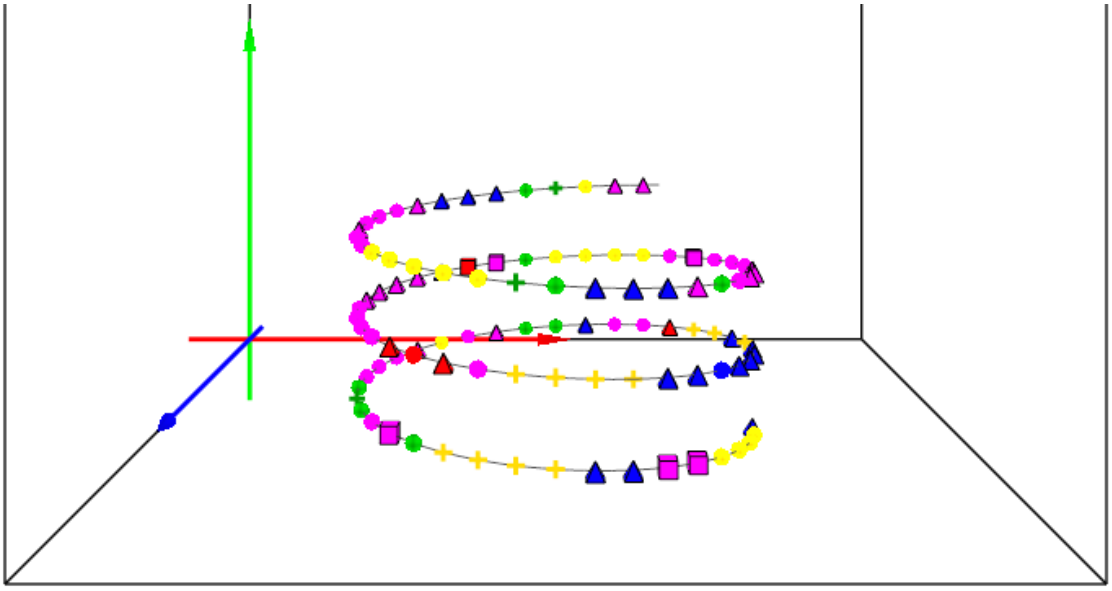


Figure 5.18: **Interaction** – This tool represents an ordered sequence of the malicious actions using an iconic representation in a spiral form. For the data exploration it is possible to zoom in and out, rotate, tilt, select different behavior slices, view the textual logs and compare it with other available behavioral data (Grégio et al., 2012).

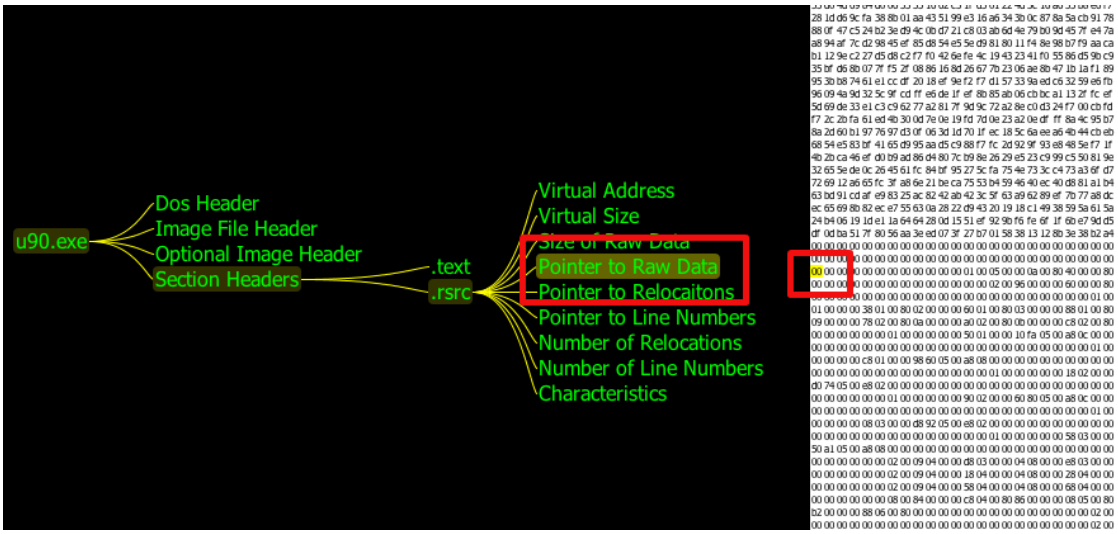


Figure 5.19: **Interaction** – Linking the hex editor on the right to an interactive tree representation enhances navigation and understanding of malware header data (Donahue et al., 2013).

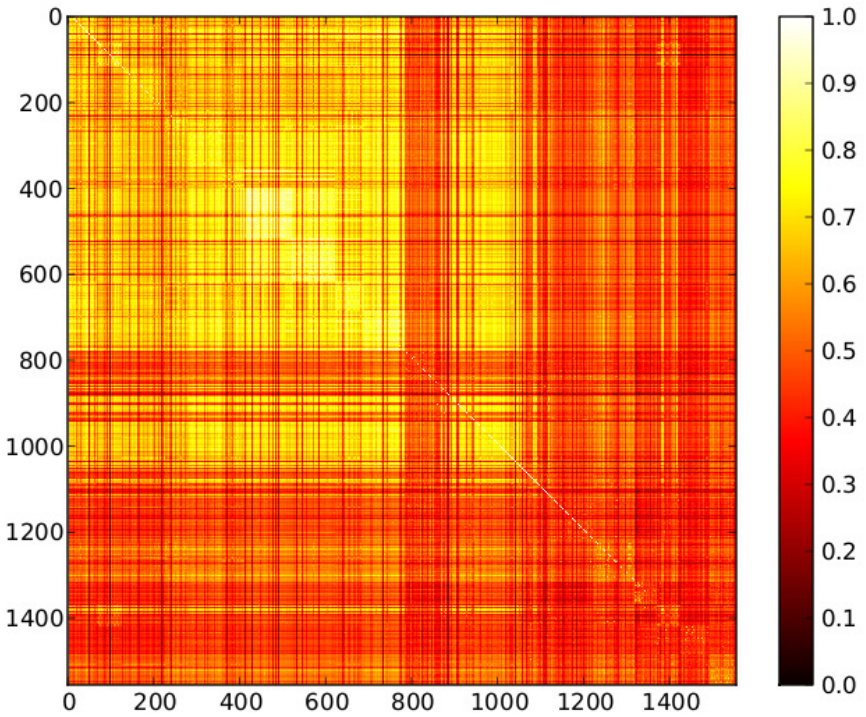


Figure 5.20: **No Interaction** – Example for a non-interactive dense pixel visualization showing similarity between 780 malware samples (top left) and 776 benign samples (bottom right) (Anderson et al., 2012).

linking and brushing for example (e.g., (Keim et al., 2008; Shneiderman, 1996; Thomas and Cook, 2005)). Additionally, we tried to find out if it is possible to switch dynamically between different visual data representations and more. The main problem for this categorization was that many of the papers did not describe exactly which of the former described features they support. Most of the time, the tools were only described as interactive, although the pictures in the papers showed that there are some of the listed interactions used. However, they were rarely described explicitly, so we decided to limit the categorization only to the possibility for interaction.

Discussion: Based on this simple categorization we found that 13 out of 25 tools support interaction (Conti et al., 2008; Donahue et al., 2013; Gove et al., 2014; Grégio et al., 2012; Grégio and Santos, 2011; Long et al., 2014; Quist and Liebrock, 2009, 2011; Saxe et al., 2012; Wu and Yap, 2013; Wüchner et al., 2014; Yee et al., 2012; Zhuo and Nadjin, 2012). An example representation for an interactive analysis tool can be seen in

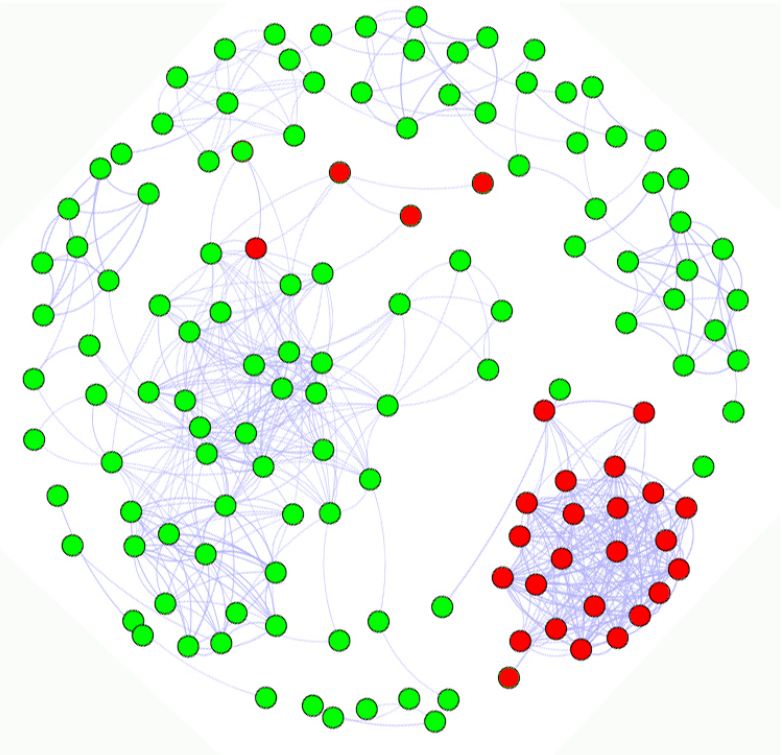


Figure 5.21: **No Interaction** – Malware samples are arranged in a node-link diagram with edge weights based on how many antivirus systems label them in the same category. The red nodes belong to a known malware family.The placement of the nodes were calculated automatically and the user could not interact with them (Grégio and Santos, 2011).

Figures 5.18 and 5.19. A non-interactive solution is depicted in Figures 5.20 and 5.21.

Problems/Actions ("Why?")

Brehmer and Munzner (2013) and Munzner (2014) propose a multi-level typology to describe abstract visualization tasks. The abstraction of domain specific vocabulary helps to identify similarities between them but it is hard to abstract them in a comparable way. If this step is made in an appropriate way, you can see that many people from different domains may aim the same goals. A problem can appear, if the translation is not made in an appropriate way so that the established vocabulary is not understandable for the people. The analysis framework includes a "*small set of carefully chosen words to describe 'why' people are using vis*" (Munzner, 2014). In this framework they describe

	(Yoo, 2004)	(Panas, 2008)	(Conti et al., 2008)	(Quist and Liebrock, 2009)	(Trinius et al., 2009)	(Nataraj et al., 2011a)	(Grégio and Santos, 2011)	(Quist and Liebrock, 2011)	(Yee et al., 2012)	(Grégio et al., 2012)	(Zhuo and Nadjin, 2012)	(Saxe et al., 2012)	(Anderson et al., 2012)	(Paturi et al., 2013)	(Han et al., 2013)	(Wu and Yap, 2013)	(Kancherla and Mukkamala, 2013)	(Donahue et al., 2013)	(Shaid, S.Z.M. and Maarof, M.A., 2014)	(Han et al., 2014b)	(Han et al., 2014a)	(Shaid and Maarof, 2014)	(Gove et al., 2014)	(Wüchner et al., 2014)	(Long et al., 2014)
Interaction	-	-	✓	✓	-	-	✓	✓	✓	✓	✓	✓	-	-	-	✓	-	✓	-	-	-	-	✓	✓	✓
No Interaction	✓	✓	-	-	✓	✓	-	-	-	-	-	-	✓	✓	✓	-	✓	-	✓	✓	✓	✓	-	-	-

Table 5.5: **Interactivity** – An overview of the level of interactivity in the visualizations used by the tools.

performed actions and targets which will be reached. Therefore, Munzner defined three levels of actions to specify the goals of the users. The top-level area tells how to consume existing data of a visualization and how to produce additional data. The mid-level describes the used search methods for known and unknown targets and locations. In the bottom-level, Munzner describes the choices of different query actions which are defined as identify, compare and search. Additionally, this framework translates all the needed domain-specific terminologies into a generic terminology. We make use of this typology to describe why the malware analysis tasks are performed by using visualization. In the following, we report how we applied those tasks to the context of malware visualization (Munzner, 2014).

Analyze

In this area, Munzner (2014) differs between two possible goals of the users. Some users want to consume existing information from the visualization and some users want to produce new information (gain new insights) from the visualization. Additionally, it is also possible to do both at the same time by one user.

Consume: This subcategory is divided into tree types of actions which were called ‘discover’, ‘present’ and ‘enjoy’.

- With the **discover** action, the user will use the visualization for the generation and verification of hypotheses using visual exploration, to gain new knowledge about the presented data which was not known before (Munzner, 2014). For example, related to malware analysis, the user tries to find new (not) malicious pattern in system and API call combinations to gain more insight of different executable files.

- **Present** aims to the goal of incisive information communication or storytelling based on the visualized data. Additionally, the presentation will guide the user in the context of decision making, planning and many more (Munzner, 2014).
- **Enjoy** refers to casual usage or pure enjoyment of visualization without specific goals or needs at hand. For example, a visualization will be designed along the needs of a user group and another group of people will test this visualization based on the interest of the functionalities for data analysis (Munzner, 2014).

Produce: In this case, the user's intend is to generate new material or output which will be used as input for a further instance or for some visualization related tasks later on. Additionally, this subcategory is also divided into 3 types of actions which were called 'annotate', 'record' and 'derive' (Munzner, 2014).

- Based on the **annotation** goal, the user of the visualization has the option to textually or graphically annotate the visualization. This annotations have to be typically made by hand by the user (e.g., tagging the points of a scatter plot) (Munzner, 2014). For example, Gove et al. (2014) described in their paper that they have noticed many different printable strings which were different spellings of same strings. Based on this insights, they annotated the malware samples to help the analysis process finding alternative versions of these strings.
- With the **record** goal it is possible to capture or save visualization elements as persistent elements. These artifacts include screen shots, parameter settings, logs, annotations and many more (e.g., everything what makes sense to store). In contrast to the 'annotate' action, the 'record' action saves the visualization as a persistent element. Thus, to save the annotations which has been made by a user, it is necessary to 'record' them (Munzner, 2014).
- Based on the **derive** goal is possible to produce new data elements which are based on existing data elements. Thus, it is possible to derive new attributes from existing information or to transform a data type into another (Munzner, 2014). In relation to malware analysis this functionality could be realized if you have a matrix with found system and API calls on both axes (x, y). In this matrix view you can see which call will be executed after the other.

Search

All the actions which were presented in the top-level 'analyze' area requires 'search' activities for the elements of interest which will be described in this mid-level area.

Munzner divided this area into four categories whereby the identifier and the location of the target elements are known or not (Munzner, 2014). In this survey, we used the 'malware sample' as the 'target' while the 'malware characteristics' was used as the 'location'.

Lookup: If the user knows what he/she is looking for and were it is, than it is simple a 'lookup' action because the target and the location are known (Munzner, 2014). Applied to malware visualization, the analyst loads a specific malware sample to analyze a specific set of characteristics. Such a lookup action would help to analyze a specific case to eventually understand the malware's behavior.

Locate: In the case that the user knows what he/she is searching for, but he/she does not know where he/she has to search, the search action will be called 'locate'. Thus, the location is unknown and the target is known (Munzner, 2014). Applied to malware visualization, the analyst loads a specific malware sample to analyze. However, it is unclear that, which are the interesting behavior features, so the target of interest is known, but the location and characteristics need to be located.

Browse: If the user does not know the exact identity of the target he/she is looking for but he/she knows the location of it, the used search action is called 'browse' (Munzner, 2014). Applied to malware visualization, the analyst is interested in many different malware samples, the precise target is unknown, however, he/she knows what characteristics to look for. This could be identified as a browse action.

Explore: If the user does not know what he/she is looking for and he/she also does not know where he/she has to search, the processed search action will be called 'explore'. In this case the target and the location of it are both unknown (Munzner, 2014). Applied to malware visualization, the analyst is interested in many different malware samples, the precise target is unknown. It is also unclear yet, which characteristics are relevant, so the precise location to look at is also unknown. This resembles an explore action. The analyst explores the data using the visualization to search for relevant targets (malware samples) and for interesting characteristics (locations).

Query

If there has been found one target or a set of targets for a search, a bottom-level goal is to query these targets. Munzner (2014) named three different types of queries, which are 'identify', 'compare' and 'summarize' whereby 'identify' "*refers to one target*", 'compare' "*refers to multiple targets*" and 'summarize' "*refers to the full set of possible targets*" (Munzner, 2014).

Identify: As described before, the 'identify' query refers to a single target. If the search result is a known target especially found by 'lookup' or 'locate', the 'identify' query returns the characteristics of the target. In contrast, if the targets of a search activity like 'browse' or 'explore' matches to special characteristics of a target, the 'identifie' query returned value will be the reference of the found target or targets (Munzner, 2014).

Compare: A 'comparison' query activity takes more sophisticated "*idioms*" than an 'identify' query activity, to support the user (Munzner, 2014). For the comparison of different targets, the inspection of a single in detail is not adequate. If we break this down to system and API call comparison, this means that the system has not only to compare the call-type. Additionally, it also has to compare the call parameters and if it is needed the return values of the system and API calls too.

Summarize: The 'summerize' query scope is to work with all possible targets. This is similar to the overview task which is well-known in the InfoVis and VA community. By the use of the 'summarize' query, there should be provided "*a comprehensive view of everything*" or "*a summary display of everything*" (Munzner, 2014).

	(Yoo, 2004)	(Panas, 2008)	(Conti et al., 2008)	(Quist and Liebrock, 2009)	(Trinius et al., 2009)	(Nataraj et al., 2011a)	(Grégio and Santos, 2011)	(Quist and Liebrock, 2011)	(Yee et al., 2012)	(Grégio et al., 2012)	(Zhuo and Nadjin, 2012)	(Saxe et al., 2012)	(Anderson et al., 2012)	(Paturi et al., 2013)	(Han et al., 2013)	(Wu and Yap, 2013)	(Kancherla and Mukkamala, 2013)	(Donahue et al., 2013)	(Shaid, S.Z.M. and Maarof, M.A., 2014)	(Han et al., 2014b)	(Han et al., 2014a)	(Shaid and Maarof, 2014)	(Gove et al., 2014)	(Wüchner et al., 2014)	(Long et al., 2014)
Analyze ► Consume ► Discover	-	-	✓	✓	✓	-	✓	✓	✓	✓	✓	✓	-	✓	-	✓	-	✓	✓	✓	-	✓	✓	✓	✓
Analyze ► Consume ► Present	✓	-	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	-	-	-	✓	✓	-	✓	✓	-	✓	✓	✓	✓
Analyze ► Consume ► Enjoy	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Analyze ► Produce ► Annotate	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	✓	✓	-
Analyze ► Produce ► Record	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Analyze ► Produce ► Derive	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Search ► Lookup	-	-	-	✓	-	-	✓	✓	✓	-	✓	✓	-	-	-	-	-	✓	-	-	-	-	✓	-	-
Search ► Browse	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	✓	-	✓
Search ► Locate	-	-	✓	✓	✓	-	✓	✓	✓	-	✓	✓	-	-	-	-	-	✓	-	-	-	-	✓	-	-
Search ► Explore	-	-	-	-	-	-	✓	-	-	-	✓	✓	-	-	-	✓	-	-	-	-	-	-	✓	✓	-
Query ► Identify	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	✓	✓	-
Query ► Compare	✓	✓	-	✓	✓	✓	✓	✓	-	✓	✓	✓	-	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	-	✓
Query ► Summarize	✓	-	-	-	-	-	-	-	-	-	-	✓	✓	✓	✓	-	-	-	-	-	✓	-	-	-	-

Table 5.6: **Problems/Actions ("Why?")** – The analysis based on the main actions supported by the visualization systems helps to identify gaps and unexplored research areas.

Discussion: Table 5.6 offers an overview of the main focus of the visualization systems in terms of supported actions. Such action analysis also helps to define general categories. Systems that feature the 'summarize' action are especially suited for the corresponding category discussed in Section 5.2. Identifying a specific malware sample is not a commonly used action used in the reviewed systems. Most systems instead focus on comparing given malware samples to a larger set, e.g., to assign them to certain malware families.

5.4 Discussion & Future Challenges

Having surveyed and systematically compared the state of the art in visualization systems for malware analysis we can extract a number of findings and propose challenges for future work. These results provide particular guidance for visual analytics professionals working in the domain but also benefit both communities – i.e. information visualization and IT security.

Bridge Between Categories

In Section 5.2 we identified three categories of malware visualization systems tackling different sub-problems of 'malware forensics' and 'classification' at the levels of individual malware samples, comparison of malware samples, and common features summarized from malware families. It is surprising that these categories cleanly partition the state-of-the-art in malware visualization. Furthermore, the prevalence of systems using malware samples either individually (9) or in comparison (11) is evident in comparison to systems working with the summaries of malware families (5). These systems for malware summarization are in sharp contrast to the domain literature's emphasis on the increasing number of malicious software, malware families and variants in the wild (e.g., (Bazrafshan et al., 2013; Dornhackl et al., 2014; Lee et al., 2011)). Since there is a common goal of generating rules or signatures, it can be assumed the potential target users of all three visualization system categories overlap. Thus, future malware visualization systems should investigate comprehensive designs: for example to switch perspective between summarization and comparison or to semantically zoom into individual analysis mode. Likewise the integration of common features of malware families can be integrated into individual malware forensics to make it more expressive.

Integrate Different Data Sources

Malware analysis is based on a wide range of base data collected by different data providers (Wagner et al., 2015c) under different analysis modes. As malware gets more sophisticated in detecting and avoiding analysis, there is an increasing need to combine

different data providers – for example to combine static and dynamic analysis. This involves not only supporting different data formats but also handling the resulting heterogeneous data in a suitable way, for example through multiple coordinated views.

Problem Characterization and Abstraction for Tailored Visualization

Many systems use visualization only on a very simple level and rely on standard displays. However, these visual representation methods are limited in their visual scalability. Yet there is potential to create novel or adapted representation methods to cater the special needs of malware analysis. Problem-driven visualization research thrives from interdisciplinary collaboration with domain experts but needs to start from a solid problem characterization and abstraction as base for design and evaluation (Miksch and Aigner, 2014; Munzner, 2014; Sedlmair et al., 2012b). Such research on the requirements for malware visualization can constitute an independent contribution to research (see Chapter 6 and e.g., (Wagner et al., 2014)).

Involve Expert Knowledge Through Interaction

For keeping up with the large number and dynamic evolution of malware families, malware analysts need to continuously adapt the settings of their visualization systems. Interactivity is a key strength of visualization systems which allows domain experts to take other points of view including immediate feedback (Keim et al., 2010a; Shneiderman, 1996; Thomas and Cook, 2005). However, most malware analysis systems surveyed here, are very limited in this regard – only 13 of 25 systems reported any kind of interaction. Even if these deficits in interaction are a recurrent theme in visualization research, malware analysis in particular can profit from more extensive interaction and annotation features as it is a very knowledge-intensive job. It should even be considered to provide knowledge-oriented interactions allowing to externalize knowledge that can subsequently be used in the analysis process to improve analysts' performance (Sacha et al., 2014).

Intertwine Analytical Methods with Visualization

Currently most systems build their visual metaphors directly on the output of the data providers and only few systems such as Saxe et al. (2012) use additional analytical methods to classify or cluster the data. Following the visual analytics agenda (Keim et al., 2010a; Thomas and Cook, 2005), analytical methods must be considered alongside visual representation methods for scalable and problem-tailored visualization solutions. Furthermore, analytical methods should not be treated as a black box but should allow adaption by experts through interaction (Mühlbacher et al., 2014). Future systems will

be required to handle the influx of malware mutations and to maintain full compatibility to new behavioral tendencies and trends. Therefore, new systems will be needed which open up the possibility for automated behavior analysis and common behavior identification to support the experts during their work.

5.5 Summary

In this survey we presented a systematic review of visualization systems for malware analysis. Each analysis system was then assigned to one of the 3 main categories, depending on their general approach to processing and visualization, which were defined in the 'Malware Visualization Taxonomy' as presented in Figure 5.7. We also categorized these systems by their input files and formats, the visualization techniques utilized, the representation space and the mapping to time, certain temporal aspects, their interactive capabilities, and the different types of available user actions. Many of the surveyed systems gather analysis data internally and base their analysis on the sample's binary code. Others use external data providers to retrieve specific properties. In terms of visualization techniques we discovered that stacked displays and iconic displays are not commonly used in the malware domain; most tools utilize static 2D displays to support the analyst. Dynamic or 3D representations are rare – only 4 of the explored systems are able to map data to a 3D representation space. With regard to the used representation space and the mapping to time, we found out that most of the systems use a static mapping. On the temporal side we determined that only 12 out of 25 analysis systems consider time at all. All time-aware systems use a linear arrangement. 11 out of these 12 tools use an ordinal timescale and one uses a discrete one. Most of the tools use the 'instant' time primitive. Only one tool uses the 'interval' primitive and one other tool uses 'instant' and 'interval' primitives. In relation to the granularities, only 2 tools out of 12 are using these primitives, whereby one tool uses a single granularity and one tool uses multiple granularities. Surprisingly, only 13 of the surveyed systems support interaction while the remainder relies solely on non-interactive representations. Of the available user actions, 'discover', 'present' and 'compare' operations are the most common. It is interesting to see that the identification of specific malware samples is usually not a priority. All the results of this survey are publicly available for interactive exploration on our supplementary website found at http://malware.dbvis.de/.

Furthermore, we defined various future challenges and perspectives in Section 5.4 to further improve visual analytics for malware analysis to eventually help to enhance cyber security.

CHAPTER 6

Problem Characterization & Abstraction

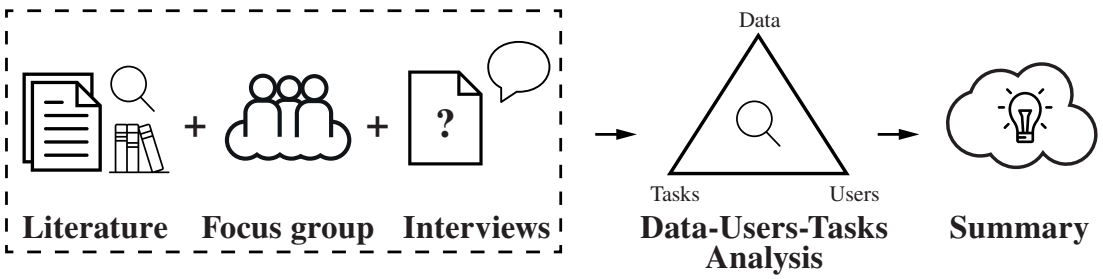


Figure 6.1: **Graphical Overview of Chapter 6** – Illustration of the main topics which are covered in this chapter in relation to the problem characterization and abstraction of the first research domain: behavior-based malware analysis.

This chapter generally follows the paradigm of problem-oriented research, i.e., working with real users to solve their tasks (Sedlmair et al., 2012b). To be able to support the domain experts during their work we performed a problem characterization and abstraction in the first step, which is also the first contribution of a design study (Sedlmair et al., 2012b). Thereby, we analyze and characterize the data to be dealt with as well as the user requirements and the features needed for a malware detection system which are supported by visual analytics methods. In particular, we follow the two top levels of the 'nested model for visualization design and validation' as proposed by Munzner (2009)

defined as *domain problem and data characterization* and *operation and data type abstraction*, for our problem characterization and abstraction (see Figure 6.1). These two steps are crucial in reaching a common language and understanding between domain experts and visualization researchers, and the requirements gathered provide the basis against which design proposals can be judged (Sedlmair et al., 2012b).

To ensure a knowledgeable characterization and abstraction of malware pattern analysis along the design triangle of data, users, and tasks (Miksch and Aigner, 2014), we followed a threefold research approach. This research approach consists of a systematic literature research, a focus group (Lazar et al., 2010, pp. 192) and semi-structured interviews with domain experts (Lazar et al., 2010, pp. 184), which are presented in this Chapter in detail.

The methods applied in our research are related to cognitive task analysis (CTA) (Wei and Salvendy, 2004). Our threefold approach includes the classification families 1 ('observations and interviews') and 2 ('process tracing') of Wei and Salvendy's classification of CTA methods (Wei and Salvendy, 2004). For the 'observations' and 'process tracings' we used example views during the interviews. The specifics of the conducted methods will be outlined below.

6.1 Literature Research

To get a good overview of existing visualization approaches with regard to the interests of behavior-based malware analysis, we used different search engines. Thereby, we looked for publications which are related to the specific needs of our collaborators in in the area of IT-security.

Design & Procedure

In the first step, we used different keyword combinations (e.g. malware, malicious software, visual analytics, visualization, time-oriented data, security, etc.). In the second step, we searched for the authors of the currently best matching papers and combined them with the currently best matching keywords of the previous research. Based on this search strategy, it was possible to find 26 different scientific publications on malware analysis for IT-security on local hosts. In order to refine our results, we investigated all the abstracts and conclusions and removed less appropriate papers. Furthermore, all the papers had to be from well known conferences and publishers (e.g. conferences: 'VizSec, VAST, ...'; publishers: 'ACM, IEEE, Springer'). This led to a number of six highly relevant papers and five tools that could be identified.

Apparatus & Materials

As search engines for our our literature review, we used 'ACM digital library', 'IEEE Xplore', 'Google Scholar', 'Academic Research Microsoft'. Additionally, all used keywords and keyword combinations were documented in a text document.

Results

We examined five existing approaches from six papers from the perspective of information visualization based on Shneiderman's 'Visual Information Seeking Mantra' (Shneiderman, 1996) as well as the visualization techniques used.

Yao et al. (2005) describe the design of an interactive framework for automated trust negotiation (ATN). With ATN, it is possible for the user to show credentials, policies and to analyze the relations of negotiated components. These sessions used a 'Trust Target Graph' (TTG) which is built up from the two negotiated ATN sessions. For the visualization of the ATN session, a node-link diagram (Heer et al., 2010) was used. To draw the TTG and the ATN sessions, the authors used the Grappa (Barghouti et al., 1997) system and the Java port of GraphViz (Ellson et al., 2004).

Willems et al. (2007) describe 'CWSandbox' and a combination of behavior-based malware detection with dynamic malware detection, API hooking, and DLL injection. Additionally, Trinius et al. (2009) created a parameterized abstraction of detailed behaviors of malicious software based on CWSandbox. For the visualization of the malware they used two different visualization methods. On the one hand, they used treemaps (Shneiderman, 1992) and on the other hand they used thread graphs (Xia et al., 2008) for the representation of the malicious software.

Shabtai et al. (2006) report that visualization of raw data did not make sense for the experts. To improve the results for the visualization, they implemented an intelligent visualization interface called 'VISITORS' which was an extension to the 'KNAVE-II' tool applied earlier which is used mostly in the medical domain. The tools contained the following features: temporal data abstraction, knowledge-based interpretation, summary of data, queries, visualization, and exploration of a large amount of time-oriented data. Of course, the system also supported a temporal abstraction of the available data. Additionally, the system includes a signal-concept visualization over time (using divided/stacked bar charts (Cleveland and McGill, 1984) and index charts (Heer et al., 2010)), a visualization for multiple concepts' association over time, using a sort of parallel coordinates (Inselberg and Dimsdale, 1991), and indented lists.

Yee et al. (2012) worked on reverse engineering of a binary executable by transforming a stream of bytes from a program into a sequence of machine instructions. The static and the dynamic debugger system interacted with a graph (Heer et al., 2010; Herman et al., 2000) visualization system called 'Mini-Graph'. With this system, they visualize the analysis data of the targeted executable file. Additionally, the system contained a

tabular text visualization area which represents different data (e.g. address, hex, comments). This way, the program flow of the targeted system could be reconstructed and it was easier to detect fragments of malicious instructions.

The 'VERA' approach introduced by Quist and Liebrock (2009) uses dynamic analysis to visualize the flow of a program. The visualization system uses a 2D representation of the data, which was transformed into a 3D space. They claim that a 2D representation is very useful for a quick initial analysis, but the 3D view provides a convincing view of the data by offering better introspection and zooming features for the observed code. For the generation of the graph layout, they used the Open Graph Drawing Framework (OGDF) (Chimani et al., 2007). With VERA, the authors provided a navigable tool to explore the data (panning, zooming and filtering). Based on a user study, they showed that the system is very helpful for them and for (non)experienced system users.

Summary

All the presented tools operated on the local host and use 2D visualizations to present the data. Only VERA uses 2D and 3D visualizations for the data representation. ATN, CWSandbox, and Mini-Graph visualize malware data using node-link diagrams. Furthermore, the CWSandbox tool also uses treemaps. Only VISITORS uses bar charts for the visualization and combines this with index charts and parallel coordinates. It can be seen that most utilized visualization techniques are node-link diagrams or graphs, closely followed by bar charts. The major disadvantage of all the presented tools was that no tool supported all the benefits of information visualization based on the 'Visual Information Seeking Mantra' by Shneiderman (1996). Particularly, interactivity is rather restricted in the mentioned approaches. CWSandbox does not support any kind of interaction for data exploration. All the other tools provide basic interaction features, but only VERA and VISITORS are more elaborated. VERA supports interaction, zooming, filtering, and panning functionalities. The VISITORS tool supports zooming, filtering and details on demand functions. Based on this insights, we concluded that information visualization methods are less used for malware detection systems operating on the local hosts. This provides novel research opportunities in the area of information visualization and visual analytics applied to the area of malware analysis. Thus, it became apparent that information visualization and visual analytics methods are less common in current available approaches for malware analysis.

6.2 Focus Group

The main intent of these performed focus group meetings was to match the domain-specific vocabulary and to establish a mutual understanding. In addition to that, we tried to find out:

- Which datasets are useful and interesting to visualize?
- Which data structures will be processed?
- What are possible visualization scenarios?
- How could expert knowledge be extracted or included?

Furthermore, it created a bird's-eye view to establish a better understanding of the different analysis steps of behavior-based malware detection (see Figure 6.2).

Participants

The focus group consisted of 7 people: 4 IT-security experts who are working in a research project on malware recognition via formal description of the malware behavior (MalwareDef project team) and 3 visual analytics experts working on knowledge-assisted visual analytics methods for time-oriented data (KAVA-Time project team).

Design & Procedure

The iterations of the focus group meetings were established on a monthly basis with a duration of approximately 3 hours (4 sessions for this project phase).

Apparatus & Materials

The results of each meeting were documented by written notes in combination with drawings. These documentation was used during all focus group meetings to be extended or adjusted to build up a solid base for the common understanding.

Results

In the first meeting, the visual analytics experts and the IT-security experts established a common understanding of their respective fields of work and their objectives to create a good basis for the collaboration. In the subsequent meetings, the technical vocabulary has been developed and clarified. Furthermore, we iteratively worked on discussing the four questions as former formulated.

Data Gathering: The data to be visualized will be generated by extracting relevant behavior patterns, whereby this process is split into several stages (see Figure 6.2). Initially, the sample under scrutiny is executed inside an isolated, partially virtualized laboratory environment. Tools such as APImon (Rohitab, 2016) and Procmon (Microsoft, 2016) monitor all activities and generate a report (i.e. trace) of system and API calls sequentially invoked by the sample. These traces are then clustered using Malheur, an automated behavior analysis and classification tool developed by Rieck (2016). In the next step, all traces within a cluster are concatenated and processed using the Sequitur algorithm (Nevill-Manning and Witten, 1997). Originally developed for file compression, Sequitur automatically replaces frequent patterns with short symbols, effectively generating a context-free grammar in the process, referred to as cluster-grammar (Luh et al., 2017). A human analyst must then assess this grammar and extract rules describing potentially relevant behavior.

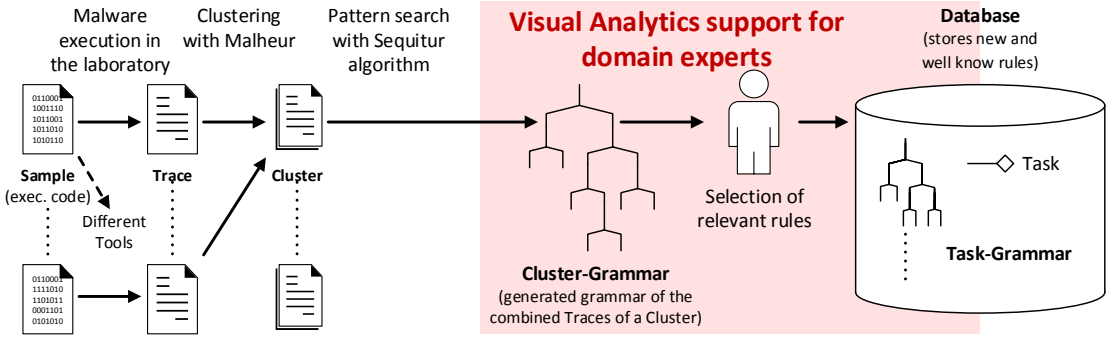


Figure 6.2: **Bahavior-based Malware Detection Approach** – Different stages of a behavior-based malware detection approach as identified during the focus group sessions with IT-security experts. The red area shows the part to be supported with visual analytics methods.

Visualization: Here is where visualization comes in, whereby one possible visualization scenario consists of the Sequitur (Nevill-Manning and Witten, 1997) file. The selection of relevant rules (call sequence) relies heavily on the accessibility and readability of the extracted information. Thus, it should be possible to select one rule to see, if it is a known pattern or not. Once a relevant rule has been identified, it should be possible to store it as expert knowledge in the system (i.e., in a database) to make it available for automated analysis and sharing with other analysts.

Cluster Grammar Structure: Table 6.1 shows an excerpt of a grammar example which was produced by the Sequitur algorithm: Column 1 (“Rule”) contains the production rules of the grammar. In the example (see Table 6.1) in line one, 27 is a

Rule	Count	Sequence
27 → 3311 3329	8	`RegOpenKeyW RtlEnterCritical...`
28 → 40 41	5	`RegQueryValueExW RtlInitUnic...`
29 → `RtlNtSt...`	76	`RtlNtStatusToDosError`
...	...	...

Table 6.1: **Cluster Grammar Example** – Example of a Sequitur cluster grammar file.

non-terminal symbol which is replaced by a sequence of the non-terminals 3311 followed by 3329. which contains all the system and API calls of the 3^{rd} column. The right arrow symbol → is the separator between the number on the left side which is comparable with the parent node of the derivation and the numbers on the right side which are the child nodes of the derivation. The numbers on the right side (e.g., 3311 and 3329) usually have a size of two non-terminal and nominal elements but sometimes it goes up to more than 10. If the rule on the left side is the last child node of the derivation tree, it points to a single terminal symbol on the right side which represents a single API call. Column 2 ('Count') gives the number of times the rule is applied within the derivation of all the traces of one cluster stringed together. The values of column 3 ('Sequence') show the result of applying all production rules recursively down to the leaves and contain all system and API calls of the rule separated by spaces (the terminal string). Thus, all values which are represented in column 2 and 3 can be derived from the values of column 1.

6.3 Interviews

Based on the results of the focus group, we developed interview guidelines for semi-structured interview sessions.

Participants

We selected a group of 10 IT-security experts to participate in the interview sessions. All the interviewed persons are working in the field of malware detection or in a related field of IT-security at three different Austrian companies. The interviewed group includes two female and eight male participants (see Table 6.2 for further details) with a degree in computer science. One person has a doctor degree (equivalent to PhD), eight persons have a master's or a diploma engineer degree and one person has a bachelor degree. Four of the interview partners were also members of the focus group from which we obtained the data four our visual analytics project.

Person	Organization	Age	Knowledge	Gender	Education
P1	R1	50-59	expert	m	PhD
P2	R1	30-39	expert	m	MSc
P3	R1	20-29	expert	m	MSc
P4	R1	20-29	expert	m	MSc
P5	C1	20-29	expert	m	MSc
P6	C1	20-29	expert	m	MSc
P7	C2	20-29	expert	f	MSc
P8	R2	30-39	basic	f	MSc
P9	R2	30-39	basic	m	BSc
P10	R2	30-39	basic	m	MSc

Table 6.2: **Interviewees** – Data of the interviewed persons. All the persons of R1 are related to the focus group (R := research group, C := company).

Design & Procedure

Each interview was scheduled for approximately one hour and was documented by means of audio recording and notes. The results of the interview sessions were combined and evaluated for the presentation of the results. The interviews were divided into two parts: brainstorming and example view exploration. The first part was structured along the following main questions related to behavior-based malware detection:

- What is the workflow of analysis?
- Which tools are used?
- How is data collected and analyzed?
- Which data records are interesting?
- What will be done with the discovered patterns?
- Are the data records currently graphically evaluated?
- Are there any visualization tools available for this field?

After the brainstorming part of the semi-structured interview each person was exposed to 6 visualization techniques (Arc Diagram (Wattenberg, 2002), Multiple View (Gotz et al., 2012), OutFlow (Wongsuphasawat and Gotz, 2011), Wordtree (Wattenberg and Viegas, 2008), Parallel Tag Cloud (Collins et al., 2009), Pixel-Oriented Visualization (Keim,

2000)) shown in Figure 6.3. We selected these visualization techniques after our preliminary problem understanding from the focus group sessions for being potentially applicable in the application domain and covering a broad range of different options for graphical representations. With these views our goal was to evaluate which visualization techniques could be useful for the visualizations and which of them are familiar to the analysts. Thus we aimed to get answers for the following questions:

- Is this or a similar visualization method already known?
- What kind of information can you identify using these visualizations?
- Is this visualization method potentially useful for your work?

Apparatus & Material

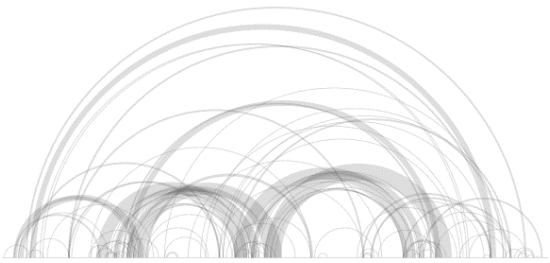
As described earlier, the interviews were divided into two parts. For the brainstorming and the example view exploration, we used an audio recorder and paper to make some notes. As extension for the second part (the example view exploration), we printed the six example views on paper and presented them in the same order to each person.

The complete interview guidelines used (in German) are included in Appendix B and the example views, shown to the participants during the interview sessions, can be seen in Figure 6.3.

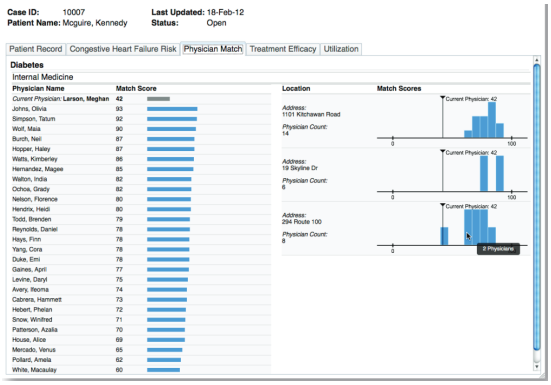
Brainstorming Results

This part presents the results of the first interview part which was structured around 7 questions.

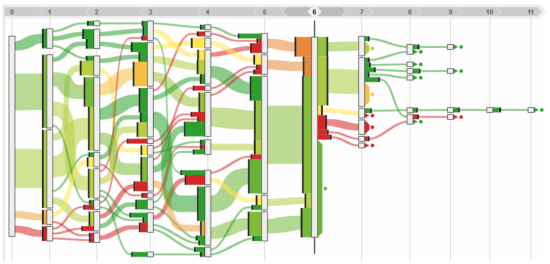
What is the workflow of analysis? As a first result it became apparent that the workflow depends on whether you are working as an anti-virus manufacturer or as a malware analyst. From the view of a malware analyst, the best choice of action is to perform a combination of static and dynamic analysis. The basic approach is to execute malware samples in an isolated area on virtual hosts and/or on native hosts for analysis and pattern finding (testing on native host takes more time because reinstalling of the machine after the test is more time consuming). C2 stated: "*The more malware samples you have, the easier it is to find unknown malware patterns.*" With regard to used operating systems, the participants reported that most of the time, malware samples are executed on all currently used Windows operating systems (Windows XP - Windows 8.1). This is done on both system architectures (x86 and x64) to determine the sample's target system. All the activities of the malware samples are logged by a wide range of tools. Additionally, the research group R1 gave a detailed overview of the single steps which will be performed in their analysis workflow:



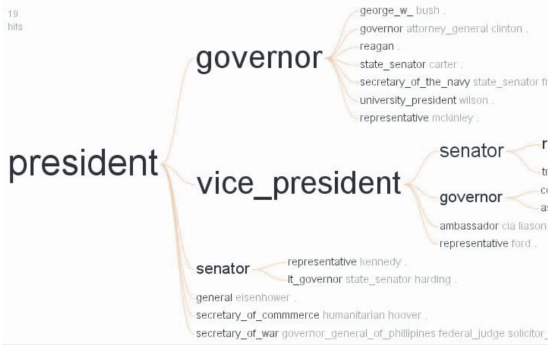
(a) Arc Diagram (Wattenberg, 2002)



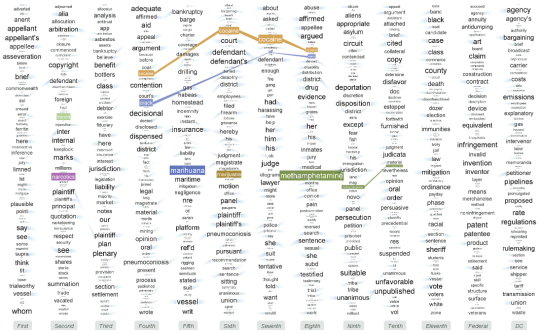
(b) Multiple View (Gotz et al., 2012)



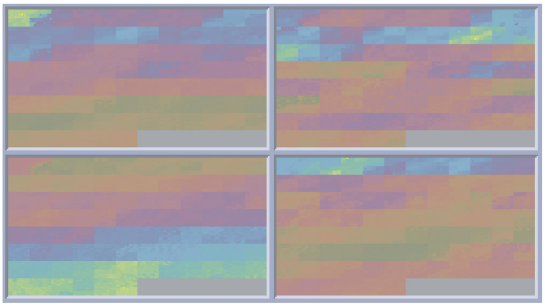
(c) OutFlow (Wongsuphasawat and Gotz, 2011)



(d) Wordtree (Wattenberg and Viegas, 2008)



(e) Parallel Tag Cloud (Collins et al., 2009)



(f) Pixel-Oriented Visualization (Keim, 2000)

Figure 6.3: **Example Views** – Shows the six example views used during the second part of the interview sessions.

1. After the logged execution of the malware samples (reports), the generated traces will be combined to some clusters by the use of the Malheur software tool. The number of clusters which will be generated depends on the number of the tested malware samples and the included system and API calls.

2. Next, the generated clusters will be compressed by the use of a lossless compression algorithm which is called 'Sequitur' (Nevill-Manning and Witten, 1997).

3. In the next step, they were searching for patterns in the automatically prepared data. For that, the analysts will use different text editors and occasionally Shell commands. In any case they have to analyze the data by hand.

4. The found pattern contained in the former generated cluster will be rated by hand and stored in a database (they also have to check if a pattern currently exists). The pattern matching between new findings and the existing patterns in the database will be automated in a future work step of the research group.

5. In the last step they perform a pattern based 'context free grammar' construction by the use of the tool Ox (Bischoff, 1992). Based on this context free grammar, the experts are searching for malicious patterns.

Which tools are used? For report generation, the research group R1 primarily utilized APImon and Procmon. T3sim (proprietary software of IKARUS Security Software (IKARUS, 2017)), Joe Sandbox (Joe Security LLC, 2017) and FireEye (FireEye, Inc., 2017) were occasionally used to complement specific analyses. Furthermore, they use Malheur to cluster reports generated by the other tools. P7 used APImon, Procmon, Cuckoo Sandbox (Cuckoo Foundation, 2017), IDAPro (Hex-Rays SA., 2017), FireEye and Joe Sandbox. P7 stated: "*IDAPro is the Swiss Army Knife of a malware analyst.*" The members of C1 use IDAPro, Anubis (formerly TTAnalyze) (Bayer et al., 2006) and some different sandbox tools that were not specifically named. R2 told us that they also used some different sandbox tools as well as IDAPro and Procmon.

How is data collected and analyzed? Our interview partners explained that after the application of one of the tools, the generated files have to be evaluated by hand. This is a very labor intensive task because each file contains several thousand lines of system and API call combinations. Additionally, the analysis of the used system and API calls is a big part. Thus, not only the execution of malware samples is important, but also the examination of the final state of the machine on which a malware sample was executed. C1 mentioned: "*Anubis provides information about the general maliciousness of a sample. The output of IDAPro, however, has to be analyzed by hand.*"

Which data records are interesting? In general there are many examples of interesting data records. The participants gave quite a number of different examples of interesting data records. Specifically, it was named that when you are examining the static part of the data, you are able to see signatures, hashes, and strings (parts of the program code which could be identified as a collection of ASCII symbols). It has also been mentioned, that the network communication of a program is very interesting: information like where does the program connect to?, upload?, download? can be answered by exploring this data. A further important finding is that it is possible that malware changes

its activities at runtime. This means that malware could contain encrypted code which will be decrypted at runtime. In addition to the program activities, our interview partners mentioned that it is very interesting to see whether a program registers itself with one of the autostart lists of the operating system. All the described activities are harmless by themselves but might become malicious in certain situations and combinations.

What will be done with the discovered patterns? R1 reported that all the patterns he is going to find will be compared with the currently stored patterns in the database. Furthermore the semantic of a found pattern needs to be manually associated to a number of predefined categories. All the other persons explained that they do not store found patterns, but only report them if there is enough time. According to our interview partners, there is no tool that allows the storing of found patterns for future evaluation activities. Finally an interesting insight was that all interviewed experts had their very own approaches toward pattern recognition that made it difficult to consolidate them to a more generalized view.

Are the data records currently graphically and visually evaluated? R1 told us that there are some visualization tools available but they did not fit the needs of their research. One of the tools they described was the Procdot (CERT, 2017) visualization tool generating a call graph to visualize the program calls of the analyzed malware sample. Additionally, C1 and C2 used Anubis which colored the results green or red if they are malicious or not.

Example Views Results

This part presents the results of the second part of the interview structured around six former described example views (see Figure 6.3).

Arc Diagram: The feedback of the interviewees implies that this visualization technique is quite conceivable for pattern recognition and tamper detection for system and API calls. Similarly, it has been mentioned to be well suited for grammar and database visualization. By means of the arc thickness and diameter of the circle, the frequency and the intensity of the connection could be shown. In addition, color differentiation is also very important and helpful to distinguish the malware and the system and API call types. Furthermore, it was stated that this visualization technique could be highly suitable for the visualization of temporal processing but also that it looks a bit unstructured. The number of calls could then be visualized by the thickness of the arcs. (C1) Of course, this technique is also quite conceivable for the visualization of different modules (For example: one sample or multiple samples). One possible problem identified could be the scalability of the visualization technique with the amounts of data to be dealt with in malware analysis.

Multiple Views: The example we have used consisted of an overview of the data using bar charts and text on the left and a detailed view of a selected data entry on the right. This visualization method displays the details of the selected data on the right side because some tools of other domains are using this approach. Thus, this method had a very high brand recognition by the participants. A visual indicator could be a traffic light coloring of the bar charts on the left side, to support user exploration activities. The interviewed experts suggested that this method could be well suited to represent behavior scores on the left and the sample's inherent API and system calls, including frequency of occurrence on the right. Additionally this visualization method was said to be potentially well-suited for the visualization of the Sequitur results and for comparing several different samples on system and API call level (e.g. on the left side there could be the sum of the same system and API calls and on the right side, in the detail view, one could compare them).

OutFlow: Outflow was found to be applicable to visualize various system and API calls which yield the same results. Malignant combinations could be highlighted using color and after each intermediate step an assessment of the malignancy of the samples in percent could be specified. For example there are several different combinations of system and API calls to add a program to a system's auto start list/area/directory. As an extension to recognize loops, the interview partners suggested back links. Furthermore, OutFlow was identified for opening up the possibility to recognize unnecessary or more importantly, obfuscated code trying to mask the true intent of the sample. This method could also be used to visualize different execution threads and their dependencies (e.g. file-handles).

Wordtree: When discussing Wordtree, our interview subjects suggested that using a color differentiation of various malware families would be very helpful. It was mentioned that the use of different font sizes for the representation of the frequency of concurrency is not as important. A uniquely occurring system or API call combination is sufficient to wreak considerable damage. Furthermore, this technique was said to be potentially helpful for a stepwise categorization by visualizing an understandable hierarchy of subdivisions in order to specify the focus of the executed sample (e.g., network focused, calculation focused). Additionally, it seems to be well-suited for the visualization of system and API call sequences and possibly for the database structure, too. A good expansion option would be to lead the individual strands back together to locate patterns which lead to the same result. This creates the possibility to discover patterns with the same intentions which were created with different system and API calls.

Parallel Tag Cloud: Considering this method it was mentioned that it would be useful for the side-by-side comparison of various samples. It would be interesting to correlate system and API calls to specific malware families ore to search for calls

which are used by different malware families. However, it would be important to put the focus on the connections between the nodes rather than on the text size. R2 also put forward a word of caution: R2 stated: "*It seems as if only the most common elements of the data to be compared are displayed - this could be misleading.*" In general, this visualization method has been described as useful but R2 (P7, P9) would rather use this method for reporting.

Pixel-Oriented Visualization: On the one hand, many of the interviewed experts mentioned that this technique would be well-suited to show data from different samples for comparison or it can be used for the comparisons of reports over time (e.g. the occurrence of different types of malware over a time period). Additionally this method could be used for the visualization of disk partitions or encrypted data in the samples. On the other hand, the technique was also critically viewed, as for example by R1: "*This visualization technique is ill-suited for the group's purpose. It rather seems to be handy for certain statistical evaluations.*"

Possible view combinations: At the end of the interviews, the respondents suggested different combinations of the presented visualization methods. These statements were then compared to select the preferred techniques. Most of the interviewees indicated that a combination of multiple views, Arc Diagram, and Wordtree would be preferred, followed by OutFlow and pixel-oriented visualization. In addition P3, P4 and P6 suggested to rotate the Arc Diagram by $90°$. The Parallel Tag Cloud, in turn, has been described as the least useful solution.

6.4 Data–Users–Tasks Analysis

Above we have characterized the domain problem of behavior-based malware analysis using literature research, focus group meetings, and semi-structured interviews. In addition to these results, we followed the approach of Miksch and Aigner (2014) to structure the domain problem and data characterization (which is the first stage of the well-known *nested model for visualization design and validation* by Munzner (2009)) along the Data-Users-Tasks triangle (see Figure 6.4). This high-level framework is structured around three questions:

- "*What kinds of data are the users working with? (data)*"
- "*Who are the users of the VA solution(s)? (users)*"
- "*What are the (general) tasks of the users? (tasks)*" (Miksch and Aigner, 2014)

Based on the answers to these questions, designers of VA methods will be supported to find or design appropriate visual representations of the data along with appropriate analysis and interaction methods to support the users in their work.

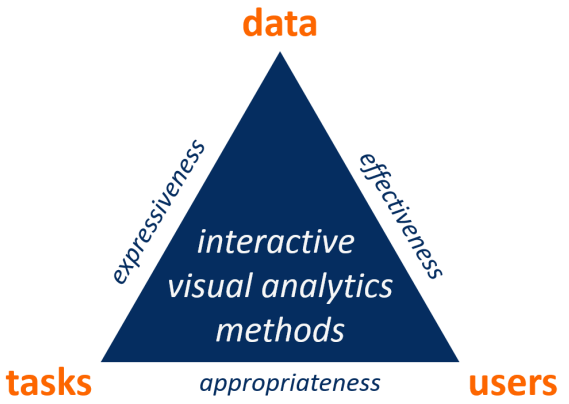


Figure 6.4: **Design Triangle** – Data-Users-Tasks Design Triangle (Miksch and Aigner, 2014).

Data

In dynamic analysis malware analysts work with collections of traces, which are sequences of relevant system or API 'calls'. In addition, call parameters and return values of the calls can be exposed. However, they do not examine these traces directly because of the large data volume. Our collaborators uses the Sequitur algorithm (Nevill-Manning and Witten, 1997) to generate context-free grammars from the clusters of traces, which they refer to as 'cluster grammars' (Figure 6.2).

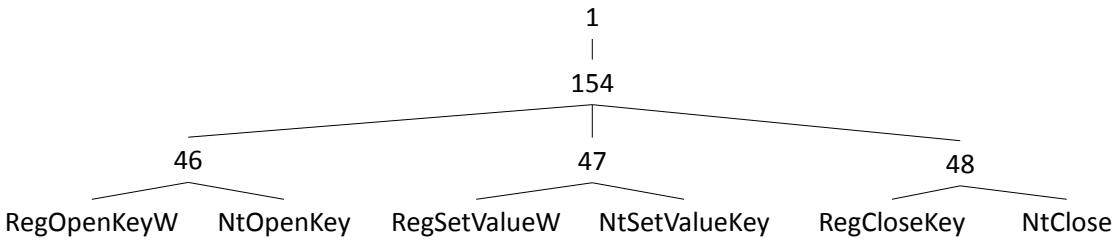


Figure 6.5: **Cluster Grammar Structure** – Illustrative example of the cluster grammar as directed acyclic graph.

Each grammar describes the derivation of a terminal string (all traces of one cluster stringed together). Each node of the parse tree has a rule of the grammar associated with it and derives a sub-sequence of terminal symbols. Table 6.1 shows examples of such 'rules' along with the number of occurrences of this rule in the parse tree and the terminal sub-sequence derived from this rule. Additionally, the distribution of rules over the traces in a cluster is available. For example a cluster of 20 malware samples might yield a total trace of 20,000 calls and a cluster grammar of 1,000 rules. These data are currently stored in a database together with metadata for system and API calls and a

taxonomy of malware behaviors. Furthermore the database contains call sequences described by rules that have already been assigned to a certain malicious behavior, referred to as *task grammar*.

The system and API calls have a value on a 'nominal' scale with a cardinality greater than 100. (Sub-)sequences of calls are 'time-oriented data' on an 'ordinal time scale' with 'instants' as time primitives (Aigner et al., 2011, p. 66). The parse tree for each cluster is a graph with rules associated to nodes (except terminal nodes), the edges representing the expansion of the rule (Figure 6.5). The parse tree of a cluster can be modeled as a 'simple directed acyclic' graph (Kerren et al., 2014, p. 2) (i.e. a tree). The intermediate nodes are non-terminals of the grammar represented by a number and the end nodes are terminal symbols of the grammar represented by system calls (see Figure 6.5). Of main interest is the number of reoccurrences of a non-terminal and their locations. The cluster grammar can be modeled as a 'network' with an underlying directed acyclic graph with rules as nodes and their composition as edges (see Figure 6.5). The underlying graph is 'simple, directed, acyclic', and usually not planar (Kerren et al., 2014, p. 2). Node's attributes are primarily the call sequences and quantitative data such as secondary occurrence counts or distribution over traces. Generally, there are no edge attributes.

Users

Malware analysis is performed by domain experts, 'malware analysts'. These users have a strong computing background – typically a university degree in computer science or IT security. They command background knowledge about system and API calls, malware vectors, and a particular intuition how harmless calls can combine to malicious behavior. The users are comfortable with combining a range of different tools such as command line, text editor, and ad-hoc developed software but have no dedicated experience with Visual Analytics solutions. Yet, they are willing to familiarize themselves with a new tool because they need to perform malware analysis often and for extended periods. However, malware analysis is a specialist activity, so there will be relatively few users.

Tasks

The primary task of malware analysts is to 'select relevant rules from the cluster grammar, categorize them by a malicious behavior task, and store them with the task grammar' (i.e. database). Secondary tasks include the manual adaptation and fine-tuning of rules found in the cluster grammar, comparing rules from the cluster grammar to rules already existing in the task grammar, and creating new rules manually either directly from traces or from their background knowledge/literature. The finding of relevant rules, is an ill-defined problem and depends on many factors. In particular such factors are the occurrence count, the distribution over traces, and the background knowledge

of the involved system and API calls. Overall, malware analysis is ‘pattern discovery’ (Laxman and Sastry, 2006), i.e. discovering relevant call sequences in traces. The primary task can be abstracted (Brehmer and Munzner, 2013) as ‘producing’ rules for the task grammar. For this, users must first ‘identify’ rules by ‘exploring’ the cluster grammars, ‘browsing’ by particular occurrence counts, or ‘locating’ with ‘special focus’ on system or API calls.

6.5 Summary

Based on the performed literature research, focus group meetings, and semi-structured interviews we formulated a problem characterization and abstraction. The interviewees enumerated many tools for the different work steps depending on the focus of their work/research. Additionally, they analyze the collected data usually manually because the currently available tools do not cover all the needs of the interviewees. By means of the six presented example views it was possible to identify preferred visual representation combinations with the interviewees. Thus, by this combinations the data analysis can be greatly facilitated (e.g. multiple view + arc diagram + word tree).

Summarizing by means of ‘data–users–tasks’, we can abstract the parse tree of a cluster grammar as a simple directed acyclic graph with nominal data attributed to the nodes. It consists of individual rules which in turn are composed of sequences of system and API calls which have a cardinality greater than 100. The users of the future system will be malware analysts (domain experts). Additionally, the main tasks are to select different rules, categorize them by their task and store them in the database as well as manual adaption and/or tuning of found rules.

Unlike the existing work in IT-security, this problem characterization and abstraction focused on malware pattern analysis and constitutes a solid base for future work. It allows visual analytics designers to create and judge design proposals for future solutions and it helps to identify similarities to other domains and their visual analytics solutions. Finally, it also aids domain experts to reflect about their own work. While designing such visual analytics solutions domain experts should of course stay involved in a user-centered design process (Sharp et al., 2007). We intend to pursue this path in collaboration with the members of the focus group.

Next Steps

In the next steps, we start with the interface design and implementation, which is based on user centered design (Sharp et al., 2007) process. In order to do so, we produced sketches in the first step followed by screen prototypes and finally the functional prototype. Additionally, we include all focus group members during the design and implementation process to get feedback about the design and the functionality of the system. Thus, it is possible to improve the design and the handling of the tool. During the implementation of the functional prototype, we perform formative evaluations. This way, it is very easy to eliminate design and functional problems very quick. Finally, we perform a user study with predefined datasets to evaluate the prototype.

CHAPTER 7

Visualization Design & Prototypical Implementation

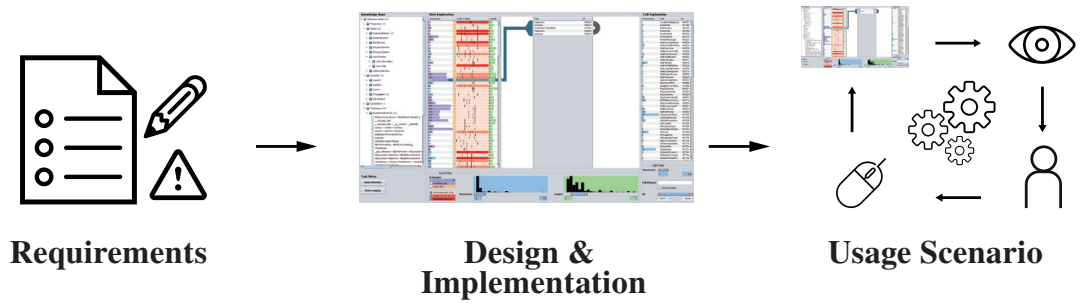


Figure 7.1: **Graphical Overview of Chapter 7** – Illustration of the main topics which are covered in this chapter with regard to the prototypical implementation of KAMAS, the prototype of the first research domain: behavior-based malware analysis.

In this chapter, we present the design and prototypical implementation of our knowledge-assisted malware analysis system (KAMAS) according to the findings described in Chapter 6. To achieve the best possible results, we worked in accordance with the *nested model* by Munzner (2009). Specifically, we focused on the third ('visual encoding and interaction design') and fourth ('algorithm design') level of Munzner's model (Munzner, 2009) and describing all steps in detail (see Figure 7.1).

7.1 Requirements

Reflecting the insights gained in Chapter 6 we could abstract the parse tree (the data structure which specifies the result of pattern search) of a cluster-grammar as a simple directed acyclic graph with nominal data attributed to the nodes. These nodes consist of individual rules which are composed of sequences of system and API calls. These have a cardinality typically greater than 100. The users are malware analysts (domain experts) whose main tasks are to select different rules, categorize them by their task and store them in the database as well as manual adaption and/or tuning of found rules (see Chapter 6).

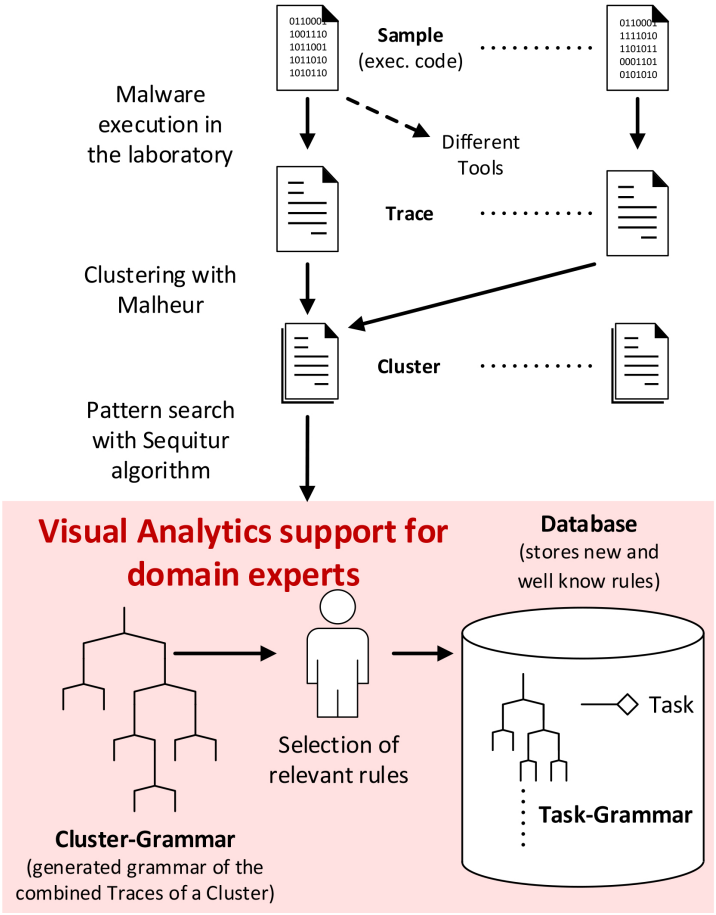


Figure 7.2: **Behavior-based Malware Analysis Process** – Behavior-based malware analysis process as conducted by our collaborators.

Based on these insights, we can define four key requirements (R) which have to be fulfilled by the KAMAS prototype:

R1 Data: 'Handling of complex data structures in behavior-based malware analysis.' To ensure the effective analysis of a cluster-grammar, a large and complex data structure needs to be modeled, stored, and analyzed. This includes detailed information about the contained rules in the form of a directed acyclic graph with nominal attributes.

R2 Visual Representation: 'Appropriate visual representations for IT-security experts.' For the main parts, analysts investigate the collected data manually because the available tools do not cover all of their needs. In a preliminary study, we found out that malware analysts preferred visualization concepts containing multiple views and arc-diagrams or word trees.

R3 Workflow: 'Workflow-specific interaction techniques.' The analysis workflow for behavior-based malware analysis contains several preprocessing and analysis steps (see Figure 7.2). In relation to that, it is important to provide familiar interaction techniques to the experts, supporting them during the analysis in finding new malicious patterns and gaining new insights of the data.

R4 Expert Knowledge: 'Externalization of expert knowledge to reuse and share.' When analysts solve real world problems, they have large volumes of complex and heterogeneous data on their disposal. By externalizing and storing of the experts' implicit knowledge (Chen et al., 2009), it gets available system-internally as computerized knowledge to support further analysis or other analysts.

We designed the visualization and interaction techniques of KAMAS, focusing on the defined requirements, followed by the algorithm design and implementation based on a user-centered design process (Sharp et al., 2007). In this way, we addressed the third and fourth level of Munzner's nested model (Munzner, 2009). During these steps, we collaborated with a group of three IT-security experts to get formative feedback for the design and implementation process of KAMAS. Two of them were malware analysts who were also involved in the previous focus group for problem characterization and abstraction (see Chapter 6). All domain experts had more than 5 years experience in behavior-based malware analysis and experience with several tools. For formative evaluation of usability (Nielsen, 2010), user studies were performed with predefined datasets provided by our IT-security collaborators.

7.2 Terms & Definitions

In this section we define the recently used terms in relation to the malware analysis process as shown in Figure 7.2 (Chapter 6 (Dornhackl et al., 2014; Wagner et al., 2014)):

Sample: An executable program that is assumed to be malicious.

System and API call: The way a computer program requests a service (a function) from the operating systems kernel to be executed.

Trace: The behavioral execution logs of a sample in the correct execution order, e.g., a sequence of system and API calls.

Cluster: The concatenation of the traces (log-files) of n related samples in one file.

Cluster-grammar: The context-free grammar extracted by the Sequitur algorithm of the cluster file (Luh et al., 2017).

Rule: A grammar element, i.e. a sequence of system and API calls that occur in the generated cluster-grammar.

Task-grammar: An identified relevant rule, stored in the KDB assigned to a task of the malicious behavior schema (Dornhackl et al., 2014). The task-grammar is the foundation of the automatically generated rules, which are used to detect malicious behavior in newly loaded clusters.

KDB: The 'Knowledge Database' storing the extracted implicit knowledge of the IT-security experts as explicit knowledge which is used for the automated analysis and visualization.

7.3 Design & Implementation

To support malware analysis experts during their work, we set up a design study project to find a visualization solution that followed a user-centered design process (Sharp et al., 2007). Therefore, we involved a group of three domain experts in malware analysis and IT-security to keep the design in line with the analyzed needs of our prior work. We iteratively produced sketches, screen prototypes, and functional prototypes (Kulyk et al., 2007). Based on these three prototype stages, we gathered feedback about the design's usability and how well it supports their analysis needs.

The design study resulted in the KAMAS prototype (see Figure 7.3), which is implemented in Java. In relation to the 'Malware Visualization Taxonomy' (see Chapter 5) we can categorize KAMAS as a 'Malware Forensics' tool with a regard to the analysis of malicious software execution traces. Additionally, we can also categorize KAMAS

Figure 7.3: **Interface of KAMAS** – User interface of the KAMAS prototype with its three main areas. 1) The (1.a) tree structure of the knowledge database (KDB) and the (1.b) test menu used for logging during the user study. 2) The rule explorer including the (2.a) rule overview table with the graphical summaries, the (2.b) connection line to the (2.c) rule detail table representing the included calls and the (2.d) arc-diagram for pattern recognition support. Additionally, on the bottom there are different filtering options (2.e–h). 3) The call explorer interface including (3.a) the table of calls available in the loaded file and different filtering options (3.b–d).

as a 'Malware Classification' tool for malware comparison in relation to the automated analysis, of the included call sequences contained in a loaded file and based on the explicit knowledge stored in the KDB. Next, we elaborate on central design decisions.

Input Data

In general, the input data are sequences of system and API calls which are logged during the execution of malware samples in protected laboratory environments (Wagner et al., 2015c) by the use of behavior-based malware analysis as described in Chapter 6 and Wagner et al. (2014). In some preprocessing steps, the data of several analyzed malware samples (e.g., from the same malware family) are clustered and transformed into a context-free grammar in order to reduce the immense number of calls, simplify the analysis and reduce storage costs. The outcome of these preprocessing steps are loaded into KAMAS.

In our specific case, the following preprocessing steps are adopted from our collaborators' malware analysis process (see Figure 7.2): First, the sample under scrutiny

is executed inside isolated, partially virtualized laboratory environment using a number of different tools (e.g., APImon (Rohitab, 2016), Procmon (Microsoft, 2016)). By monitoring all activities, these systems generate a report (i.e. trace) of system and API calls sequentially invoked by the sample. In the second step, the traces are clustered with Malheur, an automated behavior analysis and classification tool developed by. Rieck (2016). In the third step, all traces within a cluster are concatenated and processed using the Sequitur algorithm (Nevill-Manning and Witten, 1997). In relation to this step, the system and API call parameter are neglected (e.g., memory addresses, user names) which simplifies the creation of a context-free grammar of the combined samples in the cluster and generates a bird's-eye view. If some parameters are needed, they can be considered by attaching them behind the call using a separator (e.g., `CreateProcess#winword.exe`). It is important to note that the alphabet of the context-free grammar will increase by considering the call parameters. Originally developed for file compression, Sequitur automatically replaces frequent patterns with short symbols, effectively generating a context-free grammar in the process, which is referred to as cluster-grammar and forms the input data format for the KAMAS prototype. The data structure of the cluster-grammar is a simple directed acyclic graph. Leafs are called 'terminals' and represent distinct system calls. Other nodes are called 'non-terminals' and are represented by numbers. Additionally, a sequence of system calls (terminals) composed from the cluster-grammar is called 'rule' and represents a part of the sample's actual execution order.

While our collaborators only analyzed malware samples on Windows operating systems, it will also be possible to analyze malware for other operating systems based on data providers for these operating systems. After the transformation into the input data structure as described above, they can be loaded into KAMAS.

Internal Data Handling Concept

To increase the performance of our prototype, we decided to use a data-oriented design (Fabian, 2013) (e.g., used in game development and real time rendering) to organize and perform transformations on the loaded data. For example, we implemented map and set data structures as translation tables and for the input data, we mapped the strings to integers. Based on this mapping, we built an internal data structure to increase performance and to decrease memory usage. In combination with these structures, we implemented an action pipeline (similar to a render pipeline) in which it is possible to link filter options in order to realize high performance dynamic query environments and to perform all operations on the data in real time.

To test the robustness and performance of KAMAS, we loaded and worked with several different cluster-grammar files containing between ten and 42 traces of malware samples in the cluster-grammar at sizes between 61 and 7,278 rules. The included malware samples were collected by our collaborators from the IT-security department

in 2014 to test their behavior-based malware analysis and clustering system. Overall, they collected a sample set with 800 different malware samples of different families (worms, trojans and bots). Additionally, with these datasets, the system handles more than 8,000 different windows system and API calls.

Visualization Concept

When loading an input file, all of the included system and API calls are presented in the 'Call Explorer' area (see Figure 7.3.3).

Call Explorer: The call table (see Figure 7.3.3.a) in general contains three columns. From left to right, it provides at first detailed information on the call occurrence, which shows the appearance in the loaded file, thus helping the analyst to identify interesting calls depending on their occurrence frequency (e.g., if the cluster was generated with 16 samples, calls that occur 16 times or a multiple of 16 times are more interesting than others). The occurrence frequencies are quantitative data presented as bar charts, which serve as a quick visual decision support, and numbers for a precise comparison (graphical tables). The second column visualizes the name of the calls as a string which is a nominal value and the third column displays the unique ID of the call as an ordinal value.

Rule Explorer: The included rules of the loaded input file are visualized in the 'Rule Exploration' area (see Figure 7.3.2) found in the rule overview table (see Figure 7.3.2.a). From left to right, 'Occurrence' tells the analyst how often the visualized rule occurs in the loaded file visualized by bar charts and numbers. These values are related to the occurrence histogram by color (see Figure 7.3.2.g) in the 'Rule Filter' area, which serves as an overview of the rule distribution depending on their occurrence. The second column '==', provides information if the rule is equally distributed in all the traces of the loaded cluster or not. More precisely, if the represented rule occurs equally often in all traces combined in the cluster-grammar the related column will be colored orange, otherwise it is colored gray. In the third column ('Calls in Rule') the graphical summary (see Figure 7.4) visualizes the included calls.

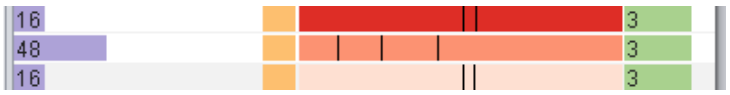


Figure 7.4: **Graphical Summary** – Detail of the graphical summary in the rule overview table. This is a space-efficient rule representation with lines marking distinct calls contained in the rule, regardless of the occurrence order and number.

Graphical Summary in the Rule Explorer: Due to the fact that a rule consists of 1 to n calls, it is difficult for the analysts to read them completely, to remember them in their correct order and to compare them with other rules. To simplify the unfavorable textual representation and to make the structures of the rules more comparable, we developed a graphical summary for the representation of the contained calls. In the graphical summary, each pixel of the horizontal position is related to a specific system call ID and has a fixed position to allow their comparison. Each call ID related pixel signifies whether a call of a certain type is present in the rule (colored black) or not. Thus, fingerprints of rules that have many calls in common, look visually similar. If a call occurs more often in a rule, only one line is displayed for it. For example, if a rule contains four calls (length of four) and less than four lines are displayed in the graphical summary (e.g., three or two), one or more calls occur more often than once. Based on the graphical summary, the analyst has the opportunity to compare the included rules and to recognize similar rules faster in relation to the included calls. Furthermore, the graphical summary which is used for the visualization of the calls contained in one rule, is based on the 'Gestalt principle of similarity' (Johnson, 2014) in order to support the analyst in quickly recognizing related call combinations. That is to say, rules that contain similar sets of calls have similar line patterns. This allows analysts to quickly assess similarity without the need to consider exactly which calls are part of the involved rules. Additionally, a tool tip is provided for a quick look at the included calls in the rule. The three different color intensities from dark to light red of the graphical summary informs the analyst whether a rule in the KDB is 'fully known' (already stored), 'partially known' (partially stored) or 'not known' (unknown) whereas these rules are defined by the analyst or other users. A 'fully known' rule which is highlighted dark red, is included as-is in the KDB. A 'partially known' rule (middle red) describes a rule which contains at least one more call at the beginning or at the end of a 'fully known' rule, and is not stored in this form in the KDB. The highlighting of the rules is performed internally by comparison of the rules contained in the input file and the rules stored in the KDB. In the last column, the analyst receives information about the length of the visualized rule (number of included calls). If the analyst compares the number of lines from the graphical summary with the related rule length, the analyst can recognize quickly whether it matches or not as described above. Like the occurrence column, this column is also related to a histogram (see Figure 7.3.2.h) in the filter area showing the distribution of the lengths of the rules.

Rule Detail Table in the Rule Explorer: Additionally, in the 'Rule Exploration' area, the rule detail table (see Figure 7.3.2.c) visualizes the included calls and the unique ID of a selected rule in its actual execution order. To support the ana-

lysts during their work on pattern search, we included a real time pattern search. For the visualization of patterns which are included in the represented execution order, arc-diagrams (Wattenberg, 2002) are used (see Figure 7.3.2.d). In this way, the analyst receives a visual representation of recurrence patterns up to the five largest patterns in a rule.

Connection Line in the Rule Explorer: To connect the rule overview table and the rule detail table, we integrated a table connection line (see Figure 7.3.2.b). This line connects a selected rule from the overview table to the detailed execution order of the integrated calls in the detail table. Thus, the user is informed of where the selected rule is located during scrolling in the view. By a click on the connection line, the selected rule comes back into focus.

Visual Interface Design Concept

IT-security experts are well skilled programmers and programming IDEs usually have a similar structure and workflow. Based on the findings of our prior research described in Chapter 6, we decided to imbue the design of KAMAS with programming IDE interfaces such as Eclipse or NetBeans. Based on this interface structure we could establish a familiar workflow concept on multiple views for the IT-security experts. In relation to these well-known interface structures, we situated the tree structure of the KDB (see Figure 7.3.1) to the left side of the interface like the project structure. Additionally, we positioned the 'Rule Explorer' in the center of the screen (see Figure 7.3.2) like the development area. This is the most used element of the prototype – the main screen, used to explore and analyze the calls which are included in a rule. Moreover, we positioned the 'Call Exploration' area to the right of the interface (see Figure 7.3.3), like the functions overview area in commonly used programming IDEs. In this area, the user has the ability to explore all of the calls which are included in the rules in the middle and to obtain detailed information about them. In addition to the design decision in relation to a programming IDE, we used Gestalt principles (Johnson, 2014) to improve interface clarity. Each exploration area (Rule Explorer and Call Explorer) contains its own filtering area below the data visualization (based on the Gestalt principles of proximity and similarity).

Interaction Concept

For a better understanding of its functionality, we describe KAMAS according to five steps based on the visual information seeking mantra by Shneiderman (1996): overview first, rearrange and filter, details-on-demand, then extract and analyze further.

Overview: When the malware analyst loads an input file, the tables for rule overview (see Figure 7.3.2.a) and call exploration (see Figure 7.3.3.a) are filled with the

complete data of all extracted rules and all calls occurring in these rules. Furthermore, the histograms (see Figures 7.3.2.g and 7.3.2.h) give an impression of the distribution in rule occurrence and length of the currently loaded cluster grammar file.

Rearrange: The analyst can now rearrange both the display of rules and calls by sorting the tables by any column.

Filter: The next step is to reduce the number of rules under consideration. For this purpose, the interface offers a selection of several filtering options. The analyst can start by filtering the calls (see Figures 7.3.3.b–c) or selecting calls directly from the call exploration table. Furthermore, the analyst can filter rules by their number of occurrence, their length, whether their occurrence is equally distributed across samples, and whether they match or partially match rules from the KDB (see Figures 7.3.2.e–h). The rules displayed in the center table are updated immediately, where the graphical summary gives an impression of the included calls.

Details-on-Demand: If a rule catches the analyst's interest, it can be selected from the rule overview table. This action opens the rule in the rule detail table (see Figure 7.3.2.c), where the analyst can read the included calls in their sequential order. The arc-diagram provides information about repeated subsequences within a rule (see Figure 7.3.2.d). All the contained subsequences are analyzed in real time. In order to not confuse the analyst, only the five largest subsequences are presented in the visualization.

Extract: Once the analyst has found a possibly malicious rule, it can be added to the KDB by dragging it from the rule overview table into the tree structure of the KDB (see Figure 7.3.1.a). Alternatively, the analyst can select some calls of a rule in the rule detail table and add them to the KDB by drag & drop. This updates the background color of the rule overview table, which allows further analysis under consideration of the added rule.

Filter Possibilities

The implemented filtering options are organized in two separated filter action pipelines where the input data depends on the loaded analysis file. The first pipeline connects all filters for the call exploration area by using an 'and' operator. The result of the first filter pipeline affects the call exploration (see Figure 7.3.3) and the rule exploration (see Figure 7.3.2) area. The output of the call exploration action pipeline is the basic input of the rule exploration action pipeline. In this pipeline, all included filters work similar to the call exploration action pipeline and control only the rule exploration area of the interface (see Figure 7.5). The integration of action pipelines for data filtering has

the benefit of being easy to change, and quickly include or exclude new or not needed filtering options into KAMAS.

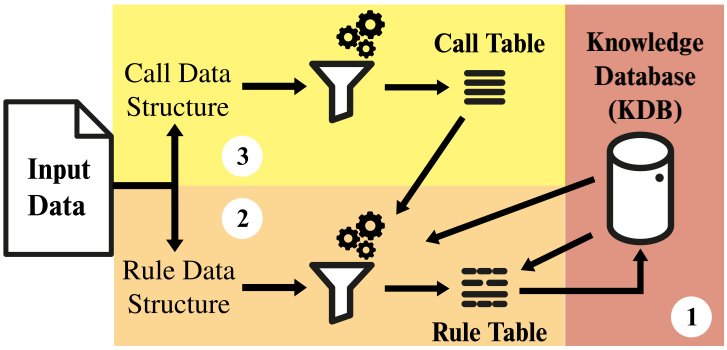


Figure 7.5: **Data Flow Diagram** – Data flow diagram of the KAMAS prototype including the two filter action pipelines. 1) The KDB which affects the rule table depending on the stored knowledge. 2) The rule table is representing the included rules of the loaded input data. Its filter pipeline gets affected by the output of the KDB and the call table filters. 3) The call table visualizes the included calls. Its filtering pipeline affects the call visualization and biased the rule table filter pipeline.

Call Filters: Below the call table, three filter options are arranged. All these filters determine the contents of the call table and subsequently the rule exploration. The 'Occurrence' filter (see Figure 7.3.3.b) is implemented as range slider to select the region of interest. Below is the 'Call (Regex)' filter (see Figure 7.3.3.c) with a separated option for 'Case Sensitive' search. This filter allows the entry of a regular expression or plain strings which are internally converted into a regular expression for a sub-string search. The third filter option operates on the unique 'ID's' (see Figure 7.3.3.d) of the calls. If the user selects one or more calls in the call table (see Figure 7.3.3.a), a fourth filter level will be established. All of these filters are connected in a dynamic query system (see Figure 7.5.2).

Rule Filters: From left to right, we included the following filtering options according to the suggestions of the collaborating domain experts: The 'multiples only' option provides the opportunity to show only rules that occur with a multiple number of times of the included samples in the loaded cluster (e.g., 16 samples included $\rightarrow$ only rules that occur 16, 32, ... times will be shown). The option 'equal only' eliminates all rules which are not equally distributed in the included samples (see Figure 7.3.2.e). Below the knowledge database filter options are arranged (see Figure 7.3.2.f) to activate or deactivate the visualization of rules based on their knowledge state. Additionally, the rule exploration filtering unit includes two range sliders (Ahlberg and Shneiderman, 1994) combined with histograms (see

Figure 7.3.2.g-h) for the visualization of the data distribution and for the selection of a region of interest. Figure 7.3.2.g serves the filtering option based on the 'Occurrence' of the rules and the related histogram shows the distribution of the rules in relation to the loaded file. The user can also select a region of interest depending on the 'Length' (see Figure 7.3.2.h) by range slider interaction, similar to the 'Occurrence' workflow.

Externalized Knowledge Integration

To support the malware analysts during their work, we integrated a KDB related to the malware behavior schema by Dornhackl et al. (2014), which is included on the left side of the interface (see Figure 7.3.1) as an indented list (tree structure). At the end of each folder description, the number of the contained rules in the integrated subfolders is shown. To add new rules to the KDB, the system provides two possibilities: On the one hand, the user can add a full rule by drag and drop from the rule overview table (see Figure 7.3.2.a), and on the other hand, the user has the ability to select a set of calls from the rule detail table (see Figure 7.3.2.c) and add them by drag and drop to the KDB. When adding a new rule, the tree automatically unfolds the hovered folder. All of the rules which are integrated in the KDB will be checked against the loaded input data automatically. By this way, the system distinguishes between three types of rules (see Figure 7.3.2.a and 7.3.2.f): not known (light red), partially known (middle red) and fully known rules (dark red). Additionally, the KAMAS prototype provides the ability to activate or deactivate (parts of) the KDB or to select the types of rules which should be shown in the rule overview table.

Knowledge Generation Loop

Figure 7.6 provides an overview of the system's knowledge generation loop, starting at the dark gray inner loop. In general, the system's KDB stores all known rules, which were generated by former cluster file analysis sessions.

If the analyst loads a new cluster file, (1) the included rules will be checked automatically against the activated KDB parts. Depending on the prior automated analysis, (2) the system provides a visual representation of loaded rules in relation to its knowledge state. Henceforth, the analyst can carry out the data exploration and analysis (the analyst is part of the knowledge generation loop). During the cluster analysis, (3) the analyst has the ability to extend the KDB with new rules found during the data exploration process. By adding new rules to the KDB, the system automatically refreshes the rules highlighting depending on the new knowledge state (4), which brings us into the outer (light gray) loop. Here the analyst is part of the continuously recurring loop (5), for data exploration (6) and knowledge generation (7).

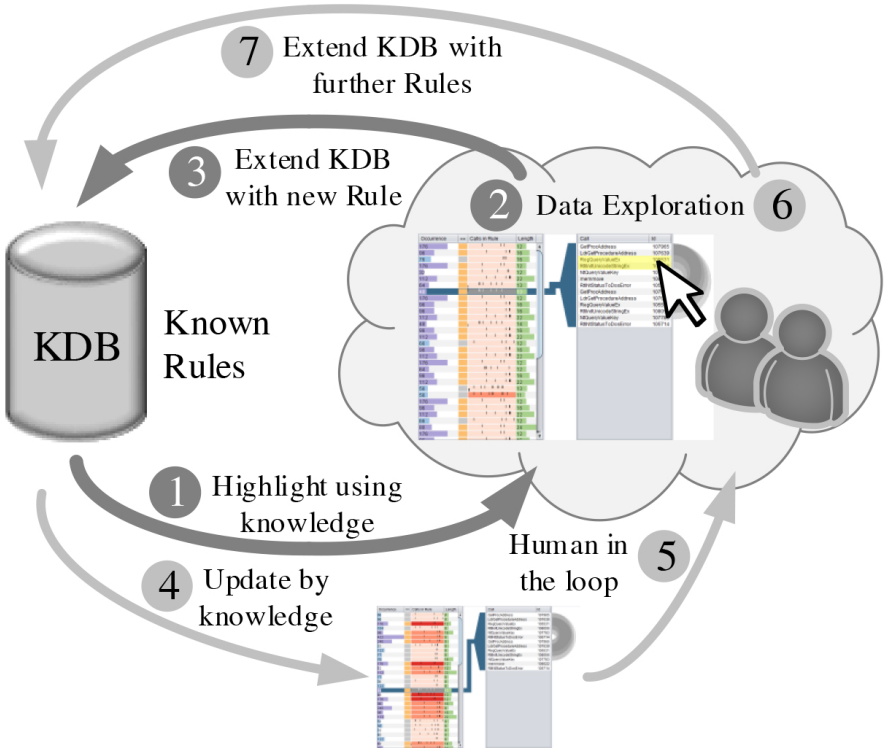


Figure 7.6: **KAMAS Knowledge Loop** – Overview of the system's knowledge generation loop, beginning with the dark gray inner loop after loading a new file and changing to the light gray outer loop for interactive data exploration and knowledge generation cycle. The analyst is a major part in both loops.

Test Menu

To log the user's activities during the user study, we included a test menu (see Figure 7.3.1.b) in our prototype. For this, we integrated two buttons, whereby the first one provides a 'reset' option for all filters and selections in the interface (like a new prototype start) and the second one 'starts' and 'stops' the logging of the user actions. In this way it was possible to generate a log file with a unique ID for each analysis task, using the 'EvalBench software library' (Aigner et al., 2013).

7.4 Usage Example of KAMAS

When malware analysts want to gain insights into their collected execution traces, they usually have to analyze the data manually, for example, using shell scripts with only text editors for visual support (see Chapter 6 and (Wagner et al., 2014)). With KAMAS, analysts can visually analyze software execution traces to find malicious subsequences (patterns). These patterns can be added to a KDB for automated data preprocessing, support during analysis, or the training of novice analysts.

If an analysts loads new malware data, KAMAS provides a general overview of all included system and API calls and all preprocessed rules which are sequences of calls (see Figure 7.3). Furthermore, in the 'Rule Explorer' (see Figure 7.3.2.a) the analyst can see a graphical summary with different background colors depending on the knowledge state of the KDB. If a rule is fully known in the KDB, the rule's background is dark red, a light red background tells the analyst that a KDB rule is included in the represented rule but surrounded by at least one other call. The light red background tells the analyst that this rule did not match with the KDB, which may mean that it is not hostile or was not yet recognized. Based on the different filtering options provided, the analyst has the possibility to search for rules of interest iteratively.

Rule Selection & Investigation

If the analyst selects a rule in the 'Rule Explorer', the selected rule becomes highlighted (see Figure 7.3.2.a). At the same time, on the right side, a detail table is generated which represents the individual calls in chronological sequential order (see Figure 7.3.2.c). Additionally, a vertical arc-diagram (see Figure 7.3.2.d) represents sub-sequence repetitions found, which supports the analyst in finding the smallest sequence without sub-patterns.

Searching for Specific Calls

If the analyst searches for a specific call, the calls name can be entered into the name filter in the 'Call Filter' area (see Figure 7.3.3.c). Here, the analyst has the ability to enter regular expressions for the search. Thus, it is also possible to add only a part of the call's name to search for a specific group. Additionally, it is also possible to select one or more calls of interest in the 'Call Exploration' area (see Figure 7.3.3.a). All of these abilities affect the 'Rule Exploration' area so that each represented rule has to contain at least one of the selected (filtered) calls.

Storing new Patterns

If the analyst wants to store a new rule in the KDB, the system provides two options. The first option is that the analyst selects the full rule in the overview table and add this rule by drag & drop into the KDB (see Figure 7.3.1.a). The second option relates to the detail table. The analyst selects only the calls of interest of a found rule and adds them as a new rule by drag & drop to the KDB. No matter which option the analyst selects, the tree structure in the KDB automatically unfolds the hovered categories so that the analyst can add the selection in the right category or sub category. It is important to note that currently only users (e.g., analysts) have the ability to add new rules to the KDB, it is not thought that new rules can be imported directly into the KDB by an external administrator for example.

CHAPTER 8

Validation Strategies & Results

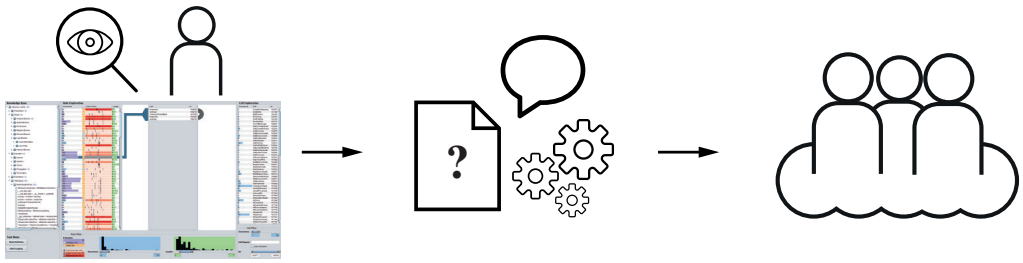


Figure 8.1: **Graphical Overview of Chapter 8** – Illustration of the main topics which are covered in this chapter in relation to the validation of KAMAS, the prototype of the first research domain: behavior-based malware analysis.

As described by Sedlmair et al. (2012b), the validation of a visualization design is the second contribution of a design study. To validate the KAMAS prototype and to provide evidence for its effectiveness, we followed a threefold research approach consisting of moderated expert reviews, user studies and focus group meetings (Lazar et al., 2010) (see Figure 8.1). All of the insights were documented in detail to ensure reproducibility (Smuc et al., 2015) and used to improve the research prototype. All materials used, such as task descriptions and interview guidelines, are included in Appendix B as well as prototypes including different input data are available on http://phaidra.fhstp.ac.at/o:1264.

8.1 Expert Review

In the first step, we conducted moderated expert reviews to eliminate usability issues in the basic functionality and appearance of the interface.

Participants

To validate the visual interface design, we invited three usability experts to contribute in this step (see Table 8.1). Each of them has between two and 20 years of experience in this field. Two of them are between 20 and 29 years of age, hold a master's degree and advanced knowledge in usability. One of them is between 40 and 49 years old, holds a PhD degree and expert knowledge in usability engineering from industry and research projects.

Person	Age	Knowledge	Gender	Education
E1	40-49	4	m	PhD
E2	20-29	3	f	MSc
E3	20-29	3	m	MSc

Table 8.1: **Usability Experts** – Overview of usability experts who participated in the expert reviews of the KAMAS prototype. (Knowledge: 1 := basic, 2 := skilled, 3 := advanced, 4 := expert).

Design and Procedure

Each usability expert received a short introduction to the basic features and the workflow of the system. Next, each expert was led through each feature individually and was asked to critique potential usability issues.

Apparatus and Materials

As evaluation material, we generated a fully functional build of KAMAS and used the same version for each expert review. The review sessions were performed on a 15″ notebook with a full HD screen resolution and an external mouse for navigation. Each expert review was documented on paper by the facilitator.

Results

The basic color scheme of the prototype was found to be easily recognizable. Only the coloring of the 'multiples only' option was pointed out as being not well differentiated from the other elements (see Figure 7.3). Basically, the dynamic query features (Ahlberg et al., 1992) were described as being very useful. If the user left the focus of a filtering input box, the interface had to apply automatically the entered parameter. Additionally, in the filtering option 'call', the experts suggested that it would be helpful to update the search results after each input. Overall, all of the usability experts provided positive feedback on the design structure of the prototype. All of the experts' suggestions were included for a redesign and revision of the prototype in order to prevent the users from having basic interface issues.

8.2 User Study

A user study with six IT-security experts was performed in October 2015. It lasted one hour on average and encompassed five analysis tasks, the system usability scale questionnaire (SUS) (Brooke, 1996), and a semi-structured interview. The goals (G) and non-goals (NG) of the user study were defined as:

G1: Testing the the functionality of the research prototype.

G2: Testing the visualization techniques for comprehensibility in relation to the domain.

G3: Testing the utility of knowledge storage and representation in the system.

NG1: Comparison of the research prototype with another analysis system.

NG2: Conducting performance tests.

Participants

We invited six IT-security experts (see Table 8.2) to participate in the user study. Two participants were also members of the focus group for the user-centered design process and gave feedback on sketches and early prototypes (see Section 7.3). All subjects are working in the field of behavior-based malware detection and analysis or in a closely related field of IT-security. All of them are involved in malware analysis projects in cooperation with different industry partners.

Person	Organization	Age	Knowledge	Education
P1	R	30-39	4	PhD
P2	R	30-39	4	MSc
P3	R	30-39	4	MSc
P4	R	30-39	4	PhD
P5	R	30-39	3	MSc
P6	F	60-69	4	PhD

Table 8.2: **Study Participants** – Data of the user study participants (R := research group, F := faculty; knowledge: 1 := basic, 2 := skilled, 3 := advanced, 4 := expert).

Design and Procedure

At the beginning of the user study, each participant was asked about a general impression of the user interface and the functions which could be recognized. This step took approximately five minutes. Subsequently, each participant had to solve five guided **analysis tasks**. The first three included a step-wise introduction to the system and the last two were combined analysis tasks to determine the understanding of the prototype's workflow (this step was also required for the subsequent SUS questionnaire). Each analysis task was read aloud to the participant at the beginning of the task, and for reference, each task was handed over to the participant in a printed form. For the analysis tasks, the participants spent approximately 30 minutes. After the analysis task session, each participant had to fill out a **SUS questionnaire** in less than five minutes. The SUS is a standardized, technology-independent questionnaire to evaluate the usability of a system (Brooke, 1996). During this test, the participants were asked ten questions on a five-level Likert scale from strongly agree to strongly disagree: 1) I think that I would like to use this system frequently; 2) I found the system unnecessarily complex; 3) I thought the system was easy to use; 4) I think that I would need the support of a technical person to be able to use this system; 5) I found the various functions in this system were well integrated; 6) I thought there was too much inconsistency in this system; 7) I would imagine that most people would learn to use this system very quickly; 8) I found the system very cumbersome to use; 9) I felt very confident using the system; 10) I needed to learn a lot of things before I could get going with this system. Finally, we performed **semi-structured interview** sessions with an average duration of 20 minutes. For this, we used an interview guideline consisting of ten major questions addressing general system usability, filtering, using the KDB, and individual visual metaphors used in KAMAS.

Apparatus and Materials

We used a silent and clean room without any distractions to perform the three parts of the user study for all participants under the same conditions. The five **analysis tasks** were performed on a 15″ notebook with full HD screen resolution and an external mouse. As dataset for the analysis tasks, we used a set of 16 malware samples (from the same family) executed and logged on a Windows operating system and transformed into KAMAS's input data format containing 794 rules as described in Section 7.3. To achieve the best outcome, we asked the participants to apply thinking aloud (Nielsen, 2010). We recorded the screen and the participant using the notebook's internal webcam. In parallel, the prototype logged user interactions using EvalBench (Aigner et al., 2013) (see Figure 7.3.1.b) and the facilitator took notes in our pre-defined test guideline. The **SUS questionnaire** and **semi-structured interview** were conducted on paper in the participants' native language (German). For the detailed questions, we used small images in the interview guidelines to support the participants in recalling the respective parts of the prototype.

Results on Analysis Tasks

During the five analysis tasks, all participants described their solutions to the tasks at hand by thinking aloud. They expressed problems as well as benefits of the systems during their work. A major problem pointed out by most participants was that in many cases they had to read tool tips about different filter options but the text disappeared to quickly [P1, P2, P3, P5]. Additionally, all participants told us that they had problems in understanding the arc-diagrams. Only P3 stated: "*It is an indication that something is included. I understood and applied it.*" In contrast, all but one participant noted that they quickly understood the handling of the KDB and its features in the prototype [P1, P2, P3, P5, P6].

Results on System Usability Scale (SUS)

By applying the SUS, it was possible to obtain a good indication concerning the handling of our KAMAS prototype. The results show a SUS value of 75.83 points out of 100, which can be interpreted as good without significant usability issues according to the SUS description by Bangor et al. (2009). They described the SUS questionnaires' result from different perspectives: From the perspective of the acceptability range, a value between 70 and 100 points is labeled as 'acceptable' and from the adjective rating perspective, the KAMAS SUS result lies in a range between good (73 points) and excellent (85 points). In general, all participants were able to recognize the interaction design and interface changes in relation to the work steps they fulfilled. Based on the SUS description and the average evaluation of the system by the participants, the result

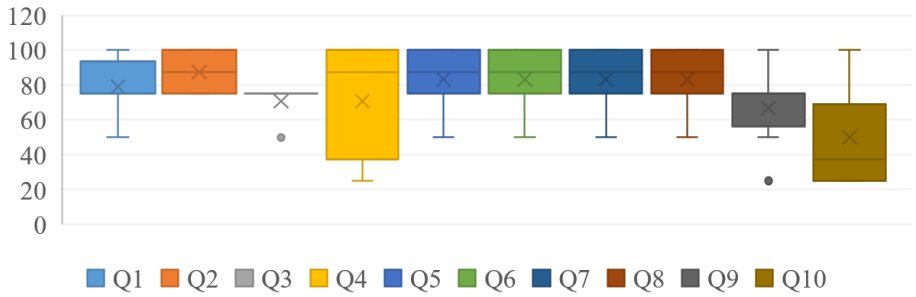


Figure 8.2: **Results of the SUS** – Illustration of the SUS questionnaires' results divided into the ten contained questions (Q1 to Q10).

of the usability assessment was very positive. Sauro (2011) compared 500 SUS scores and identified the average score as 68 points. Additionally, he showed that only 36% of the compared tests reached an SUS score higher than 74, and only 10% of the systems reached an SUS score greater than 80, which shows that our system receives a grade of 'B' at this implementation state (see Figure 8.2).

During the test, the participants again addressed the tool tip complication and the missing brief introduction of the included visualization techniques. These comments were also apparent concerning SUS question ten: "*I needed to learn a lot of things before I could get going with this system*" (Brooke, 1996), with a score of 50%. Additionally, the participants' opinions were also diverse in questions four and nine. Some of them argued that they will need a brief introduction of the included visualization techniques to better understand the workflow of the system. Other argued that they felt very confident using the system. In contrast, the SUS questions concerning the complexity, functionality integration, consistency, learnability and usability (number two and five to eight) clearly reached more than 80%. Thus, focus areas for further improvements were identified.

Results on Interviews

Here, we present the results of the performed interviews structured along the ten main questions asked to the participants. All participant statements quoted in this section have been translated from German into English by the authors.

Are the filter options understandable? In general, the filter options were classified as very useful by all participants (P1 – P6), e.g., as P6 stated, "*Yes, nothing more to say about it.*" P4 and P5 also added that all filters were well labeled so that it was possible to recognize their mode of action immediately. In the 'call filter by name' part, it was confirmed by the participants that the regular expression support was a great advantage. Additionally, P1 was very glad that the system automatically performs a full text search

if there are no regular expression elements added. P1 and P3 added that the range sliders were very helpful and reflected the selected area very well. Likewise, it was mentioned that the arrangement is intuitive.

Were the different visualization techniques helpful? P1 and P3 – P6 indicated that the various visual elements contributed significantly to understanding the system. P1 and P6 stated that the graphical summary (see Figure 7.4) was very useful. Although participant P6 added that a brief description would have been helpful. P3 mentioned that the interesting rules were immediately found by adhering to the graphical representations (e.g., coloring, bars in the table columns, arc-diagrams for pattern recognition) and they provided a good overview of how often they occur. P1 and P3 – P6 noted that the colored background of the individual categories and elements aided significantly in finding relationships and helped in understanding the system. P1 and P6 indicated that the arc-diagram was not really understandable. Only P3 immediately recognized that it is used to visualize recurring patterns in the data. P3 added that the histograms were very helpful to see the loaded file's data distribution.

Did you understand the use of the knowledge database? The handling and the meaning of the KDB was assessed as very useful by all subjects (P1 – P6). However, P5 indicated that a short description of the functional advantage would have been helpful in understanding it more quickly. Likewise, it was described as very easy to insert new rules in the database. The tree structure was very well received by the participants. P3 stated that the drag & drop worked very well for the insertion of new rules. Interestingly, P2 stated that "*The knowledge database can also be interpreted as a kind of pre-filter for data use.*" This relates to the automated analysis by using the KDB to highlight the knowledge state of included rules in the loaded input file. Thereby, the analyst gets the ability to in- or exclude these rules depending on the explicit knowledge.

Were the different ways of knowledge representation helpful in finding the right choices? In general, all participants (P1 – P6) stated that the KDB is ([P4] "*probably*") helpful for the decision making. P1 and P5 described it as well suited for checking the rules in the loaded file. That answers frequently asked analysis tasks such as: "*Is the sample doing what the database says?*" [P1]; "*What is the type of the considered sample?*" [P1, P5]. P3 and P5 mentioned that the database would also be used as a search option to instantly find or investigate rules with predefined content. In this regard, P5 added that the KDB was very helpful for quickly searching and filtering. "*If the knowledge database is filled, it makes sense to focus on this*" [P1]. P2 indicated that it depends on the task.

How was the expert knowledge represented in the system? P1, P3 and P4 related to the KDB and their tree structure were representing the explicit knowledge stored in the system. Additionally, P2, P5 and P6 referred to the color gradation in the graphical summary representing the knowledge state of a rule in the KDB. It was very well understood by the participants, that the KDB could be filled by other experts or by themselves.

Thus they had the chance to explore existing externalized knowledge, learn from it and add their own. "*Any expert has added this into the database marked as important or prominent*", stated P3. P2, P5 and P6 referred to the red three-step color map which is used for the background in the graphical summary. By this way, the user recognizes immediately if a rule matches with the database (not known, partially known, or fully known). The red three-step color map was understood by all participants and described as clear and logical. P1 added, "*If a malware changes over time, it can be found by means of expert knowledge.*"

Were the included bar charts in the table cells helpful? The bar charts in the table cells (see Figure 7.4) were generally described as very helpful and useful, because one immediately gets an overview of the distribution and thus can estimate the frequencies at a glance. Additionally, P1, P2, and P3 added that the sorting of the data can be recognized immediately, thus min and max values could be found very quickly. P1 – P3, P5 and P6 mentioned that the bar charts were very easy to read, helped recognizing frequencies, and in comparing the occurrence, the length, and the distribution. P4 and P5 added that this presentation is much better for comparison than numerical representations because they can be estimated. In case it is necessary to get the exact value of a cell entry, all participants stated that they would use the exact number.

Were the histograms displaying rules helpful? P1, P3, P4, and P6 stated that the histogram information on the distribution of the length and occurrence of the rules in relation to the loaded file was important additional information. P1 additionally mentioned that it would be helpful to gray-out the areas of the histogram which are not selected in relation to the underlying range slider. "*A solution by means of shading would also be feasible and would require less space*", stated P4. P5 indicated that such a representation is not applied in other known tools.

Was the connection between the two tables helpful? The connecting line was described by all participants as very or absolutely useful. P3 and P6 called the connection optically very supportive because it helped the eyes' during the transition to the second table. Additionally, P1 and P6 valuated the connection line as helpful for scrolling because you can always find the associated rule again. "*After recently trying it out, the connection line was very helpful*", stated P5.

Were the arc-diagrams next to the detail table helpful? The arc-diagram provoked mixed opinions among the participants. All of the participants stated that interesting elements in the resolved rule were made visible by the arcs. However, the precise usage and meaning, was recognized only by P3. "*This is a part of a pattern in a rule which is again discoverable on the first glance. This might be confusing, but at least you always see if patterns are contained*", stated P3. In general, four out of six participants (P1, P2, P4, and P5) considered this visualization method to be confusing and would likely to reject it. They demanded a different form of presentation for pattern highlighting. Other critical comments were: "*The overlap of same colors does not make sense*" [P4].

"*With a description certainly useful*" stated P2 and P5. "*I did not understand it without [visualization] background knowledge*" [P5].

What is your overall impression of the software? In general, all participants (P1 – P6) described the prototype as very clear after having a brief explanation. The overall layout was described as clear, and the graphical summary was also found to be very helpful. "*Simple and very good. Definitely useful for the application*" stated P6. Likewise, it was stressed that the prototype includes many functions but it is not overloaded. P2 commented the system as useful for exploratory tasks in malware analysis. P5 added that a longterm test would be interesting. Participant P4 suggested to change the arrangement of the KDB and the call exploration (see Figure 7.3) in a way that a workflow results from left to right.

Combined Results

The systems included filtering options were described as very useful by all participants. Additionally, their arrangement was noted as intuitive. The various visualization elements included in the system contributed significantly to the understanding. Thus, the graphical summary was mentioned as very useful and the colored background aided significantly in finding relationships. In general, the participants told us during the analysis tasks that they had problems understanding the arc-diagrams. In contrast all participants out of one had understood the KDB very quickly. Likewise, it was easy for them to insert new rules by using the drag & drop option which relates to the automated analysis by using the KDB to highlight the knowledge state of included rules in the loaded input file. Additionally, the KDB was described by all participants as very helpful for decision making. Regarding to the representation of the expert knowledge, on the one hand, the participants related to the tree structure of the KDB, and on the other hand, they related to the color gradation in the graphical summary. Based on the color gradation, it was easy to find out if a rule is included in the KDB or not. Additionally, the participants recognized that the KDB can be filled by other experts or by themselves. Thus they had the chance to explore existing externalized knowledge, learn from it and add their own.

The histogram information on the distribution of the length and occurrence of the rules in relation to the loaded file was important additional information for the participants. Additionally, the simple looking connecting line was described by all participants as very or absolutely useful, because it helped the eyes' transition to the second table. Over all, the participants described the prototype as very clear and understandable after having a brief explanation. Based on the combined insights from the five analysis tasks, the SUS and the interviews, we compiled and rated a list of found issues inspired by Nielsen's severity ratings (Nielsen, 2010). This list includes 'feature requests' (FR), 'severities' (SE) and the 'effort' for their implementation (see Table 8.3). All of these adjustments were included before the following focus group meetings.

Description	FR	SE	Effort
KDB: Include number of underlying elements	2	-	3
KDB: Gray out inactive elements	2	-	2
KDB: Automatically unfold by hover	3	-	3
Filter: Adding the number of elements in relation to the knowledge state	2	-	1
Filter: Consistency of interface labels	-	2	1
Connection line: Selected rule has to come back into focus by click	2	-	2
Arc-diagram: Reduce number of arcs	-	2	1
Tables: Change 'Rule' to 'Calls in Rule'	-	2	1
Tables: Change '=' to '==' (for equal distribution)	-	2	1
...			

Table 8.3: **Severity Rating** – List of identified feature requests and severities (FR: 1 := nice to have, 2 := good feature, 3 := enhances usability; SE: 1 := minor, 2 := big, 3 := disaster; Effort: 1 := min, 2 := average, 3 := max).

8.3 Industry Focus Group

After the most important adjustments (see Table 8.3) were carried out in the prototype, we conducted two focus group meetings to get feedback on the changes and gather additional suggestions for improvement. Members of two different professional companies in the IT-security domain participated in these sessions. Each focus group contained three IT-security experts from industry.

Design and Procedure

The focus group meetings were designed as guided meetings – we showed the prototype to the group and explained the work steps for analysis tasks. Each group member was asked to express feedback and provide suggestions at any time.

Apparatus and Material

As material, we used a fully functional build of the KAMAS prototype including all improvements based on prior evaluation results. The guided focus group meetings were performed on a 15″ notebook with HD screen resolution and an external mouse. Both focus group meetings were held in early November 2015, with a duration of approximately one hour and documented on paper by the experiment facilitator.

Results

All focus group members reported that the prototype is well designed and very interesting to use because it gives each user the ability to benefit from the expert knowledge of others. One group member noted that it is hard to build a mental connection between the histogram and the related table columns. He recommended changing the histogram's background to the color of the related bars in the columns. This suggestion was immediately included into KAMAS. One of the major inputs of the industry focus groups was that not only malignant rules, but also benign rules are important for good analysis result. Additionally, one company offered to carry out a long-term field study and to expand the prototype for their requirements.

CHAPTER 9

Reflection & Discussion

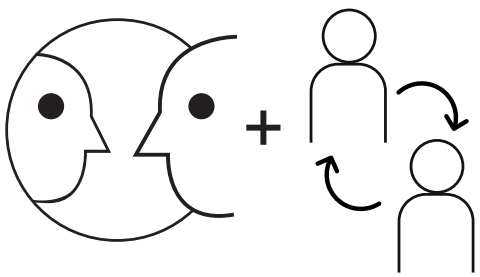


Figure 9.1: **Graphical Overview of Chapter 9** – Illustration of the reflection and discussion topic which is covered in this chapter in relation to the validation of KAMAS, the prototype of the first research domain: behavior-based malware analysis.

Following the design study methodology (Sedlmair et al., 2012b), reflection is the third contribution to a design study (retrospective analysis and lessons learned) for the improvement of current guidelines (see Figure 9.1). When we broke down the reflection on our requirements from Section 7.1, the different validation steps confirmed that our interactive visualization prototype fulfills the requirements (R1 – R4) of malware analysts.

Figure 9.2: **Suggested Color Map** – Current three-step color map (left) versus a suggested five-step color map for knowledge representation (right). For color-blind users we decided to select to a diverging red-blue scale (red := malicious, blue := benign).

R1 Data: The cluster-grammar provided by IT-security experts is a complex data structure consisting of the derivation rules, detailed derivation information and occurrences. Using this information as basis, we designed three analysis tables for representing the data: 1) a call overview table, visualizing all calls included in the cluster-grammar combined with their total occurrence; 2) the rule overview table to represent the calls included in a rule by a graphical summary – to provide the comparability of rules in combination with the total occurrence of the rule, it's length and derivation information; 3) a rule detail table, showing the included calls of a rule in their sequential order. To gain better insights into the rules' distribution in relation to their occurrence and length, KAMAS provides two different histograms, each combined with a range slider which can be used as filtering option.

R2 Visual Representation: In general, the decision for an interface similar to programming IDEs was well received by the participants. It was easy for them to understand the handling and to work with it. Additionally, the participants and the focus group members appreciated the prototype's wide range of coordinated features, which they regarded as useful for data exploration while not being overloaded. A particularly interesting outcome of the tests from a visualization design perspective is, that the arc-diagrams did not provide the benefits we expected (e.g., easily locate and compare patterns in the call sequences). One participant realized that something interesting was in the data, but he could not pinpoint the meaning. Yet, the simple connection line between the rule overview table and the rule detail table, which originally was considered 'nice to have' by designers, turned out to be a much appreciated and valuable feature. Thus, the connection line supports the analysts in finding the connection between these two representations.

R3 Workflow: All included filter methods were very well received. Additionally, the dynamic query concept and its fast response was understood very well by the participants. In general, they described the relationships of the filters as intuitive and the usage of the KDB by drag & drop actions as easy to use. By subsequent focus group meetings, further improvements were integrated, such as the colored background of the filters, used to emphasize visualization elements and connections. One of the major inputs of the industry focus group was that not only malignant rules, but also benign rules are important for a good analysis. Therefore, partic-

ipants suggested to change the three-step red color scale to a five-step scale with a neutral color in the center (see Figure 9.2). Based on the insights we gained from the tests, we found that the visualization of the expert knowledge and also the handling of the KDB was easy to understand and to use for the participants.

R4 Expert Knowledge: As previously mentioned, the KDB's tree structure was well received by the participants and focus groups members. Another improvement, added as a result of the user study, was the addition of brackets with numbers of included rules in brackets at the end of each node. Additionally, we added a counter for each type of represented knowledge in the interface (see Figure 7.3.2.f). The industry focus group members noted that the newly added numbers were helpful for getting a better overview of the loaded data. Reflecting on the insights we gained in the performed tests, we found that the analysts appreciated and could benefit from the externalized expert knowledge by sharing and activating or deactivating KDB elements during the analysis process.

Categorization of KAMAS

If we categorize KAMAS along the 'Malware Visualization Taxonomy' as described in Chapter 5 and (Wagner et al., 2015c), we can see that KAMAS can be categorized on the one hand as a 'Malware Forensics' tool which regards to the analysis of malicious software execution traces. On the other hand, KAMAS can also be categorized as a 'Malware Classification' tool for malware comparison in relation to the automated analysis. This automated analysis works on the included call sequences contained in a loaded file and is based on the explicit knowledge stored in the KDB.

Relation to Knowledge-assisted VA Model

Since KAMAS was described in detail in Chapter 7, now we describe its functionalities using our new 'Knowledge-assisted VA Model' (cmp. Chapter 3) as shown in Figure 9.3. KAMAS supports malware analysts (IT-security experts) in behavior-based malware analysis in order to learn about previously unknown samples of malware or malware families. Therefore, they need to identify and categorize suspicious patterns from large collections of execution traces, represented in the form of preprocessed rules in their sequential order. These rules are generated by the use of 'Sequitur' (Luh et al., 2017) (see Chapter 6), which can be seen as an external automated data analysis method $\textcircled{A}$. A 'Knowledge Database' (KDB) storing explicit knowledge $\boxed{K^e}$ in the form of rules is integrated into KAMAS to ease the analysis process and to share it with colleagues. Based on this, automated data analysis methods $\textcircled{A}$ are comparing the rules included in the loaded cluster file of malware samples $\boxed{D}$ based on the specification $\boxed{S}$ with the stored explicit knowledge (see Figure 9.3). Thereby, the specification gets

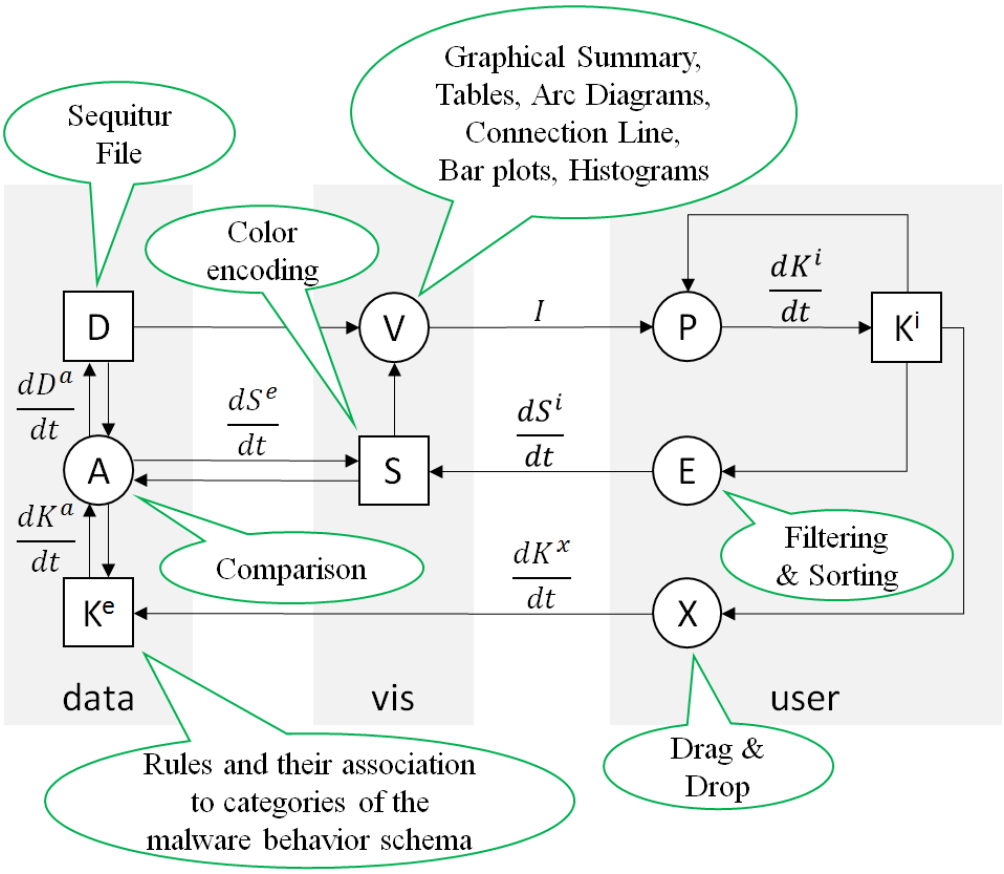


Figure 9.3: **Instantiation of Knowledge-assisted VA Model for KAMAS** – Illustrating specific elements in relation to the components and processes of the'Knowledge-assisted VA Model'.

adapted to highlight known rules { $\boxed{D}$, $\boxed{K^e}$, $\boxed{S}$ } $\rightarrow$ $\textcircled{A}$ $\rightarrow$ $\boxed{S}$. Additionally, the explicit knowledge can be turned on and off partially or completely by interaction: $\textcircled{E}$ $\rightarrow$ $\boxed{S}$. If the analyst loads a cluster file of malware samples into the system, the contained rules are visualized based on the systems specification { $\boxed{D}$, $\boxed{S}$ } $\rightarrow$ $\textcircled{V}$. If there is no specification prepared in the first visualization cycle (e.g., zooming, filtering, sorting), all read-in data are visualized and compared to the KDB. The image is generated by the visualization process and perceived by the analyst to gain new implicit knowledge $\textcircled{V}$ $\xrightarrow{I}$ $\textcircled{P}$ $\rightarrow$ $\boxed{K^i}$, which also influences the user's perception $\boxed{K^i}$ $\rightarrow$ $\textcircled{P}$. Depending on the gained implicit knowledge, the analyst has now the ability to interactively explore the visualized rules by system provided methods (e.g., zooming, filtering, sorting), which are affecting the specification $\boxed{K^i}$ $\rightarrow$ $\textcircled{E}$ $\rightarrow$ $\boxed{S}$. During this interactive process, the analyst gains new implicit knowledge based on the adjusted visualization.

For the integration of new knowledge into the KDB, the analyst can add whole rules or the analyst can add a selection of interesting calls (both by drag & drop), extracting his/her implicit knowledge $\boxed{K^i} \rightarrow \textcircled{X} \rightarrow \boxed{K^e}$. Moreover, KAMAS provides the ability to directly visualize the stored explicit knowledge in the KDB transforming it into data $\boxed{K^i} \rightarrow \textcircled{A} \rightarrow \boxed{D} \rightarrow \textcircled{V}$.

Contributions of KAMAS

In relation to the design study presented in this part, the main contributions are:

- We presented a detailed literature review with regard to visualization systems for malware analysis in combination introducing the 'Malware Visualization Taxonomy' for system categorization. In general, the taxonomy divides the categorization of malware visualization systems into three categories: 1) Individual Malware Analysis; 2) Malware Comparison; and 3) Malware Summarization.

- We provided a detailed problem characterization and abstraction to establish a common understanding between domain experts and visualization researchers. Thereby, we are describing the data to be visualized, the future system users and their tasks to be solved in detail.

- We presented the concept and the implementation of KAMAS as systematically designed, developed and evaluated instantiation of an knowledge-assisted VA solution for the handling of a vast amounts of data in behavior-based malware analysis. The calls have a nominal value and can be described as time-oriented data on an ordinal time scale with instances as time primitives.

- We show that applying knowledge-assisted VA methods allows domain experts to externalize their implicit knowledge and profit from this explicit knowledge during their analysis workflow.

- For the visualization of the explicit knowledge we provided a three-step color map for knowledge highlighting in combination with a graphical summary helping the analyst while comparing different rules. Additionally, for the exploration of the explicit knowledge, we provided an intended list.

- To provide evidence for the effectiveness of the developed methods, we provide a rigorous and reproducible validation of the introduced techniques with malware analysis experts.

Lessons Learned

During this design study, we learned that explicit knowledge opens the possibility to close the gap between different categories of malware analysis systems which are described in the 'Malware Visualization Taxonomy' in Chapter 5 and (Wagner et al., 2015c). Thus, it combines features for Malware Forensics and Malware Classification. In contrast to other malware analysis systems which build their visual metaphors directly on the output of the data providers, KAMAS uses an input grammar generated by a combination of Malheur (Rieck, 2016) and Sequitur (Nevill-Manning and Witten, 1997) for cluster and data classification. Therefore, we use analytical and visual representation methods to provide a scalable and problem-tailored visualization solution following the visual analytics agenda (Keim et al., 2010a; Thomas and Cook, 2005). For keeping up with the large number and dynamic evolution of malware families, malware analysts need to continuously adapt the settings of their visualization systems, whereby interactivity is a key strength of visualization systems. Malware analysis in particular profits from extensive interaction and annotation features as it is a very knowledge-intensive job. By providing knowledge-oriented interactions, externalized knowledge can subsequently be used in the analysis process to improve analysts' performance.

Transferability

The knowledge generation loop can be generalized for other domains taking into account domain-specific data structures and patterns of interest. On a general level, the workflow for knowledge generation and extraction is mostly similar and always includes the system user as an integral part of the loop (Endert et al., 2014). Focusing on n stepped colored highlighting and easy to understand summarization techniques it is faster and more effective to find similarities in the data.

Next Steps

In the next step, we will extend our research to a different problem domain in order to generalize our results. More precisely, we will focus on clinical gait analysis in health care. Therefore, we have to adapt and extend the knowledge-assisted visualization methods and the interface as necessary in relation to the users' needs, which we will find out starting again with a problem characterization and abstraction (Sedlmair et al., 2012b) to specify the data, the users and the tasks (Miksch and Aigner, 2014).

Part III

Case Study 2: Clinical Gait Analysis in Health Care (KAVAGait)

CHAPTER 10

Motivation & Related Work

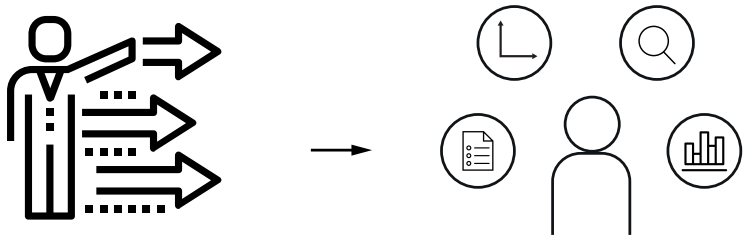


Figure 10.1: **Graphical Overview of Chapter 10** – Illustration of the topics which are covered in this chapter in relation to the second research domain: clinical gait analysis.

This chapter starts with the general motivation (see Section 10.1) for the second design study in relation to clinical gait analysis and the need for knowledge-assisted visualization systems in this area. Additionally, this chapter presents the related work (see Section 10.2) according to VA in movement time series and multivariate time series analysis (see Figure 10.1).

10.1 Motivation

According to the 2014 Disability Status Report of the US (Erickson et al., 2016), 5.5% of working age adults (ages 21 to 64), amounting to more than 10 million nationwide, suffer from an ambulatory disability. Walking and stair-climbing are essential motor functions that are prerequisites for participation in activities of daily living. Disruptions to these motor skills hold severe health and socio-economic implications if left unattended. Therefore, gait rehabilitation is a crucial issue for clinicians.

Gait analysis tools allow clinicians to describe and analyze a patient's gait performance to make objective data based decisions. The systems commonly used for capturing gait data range from simple video cameras and force-distribution sensing walkways to highly sophisticated motion capture systems (Nigg and Herzog, 2007; Winter, 2005). The latter is often referred to as the gold standard in clinical gait analysis, as this method assesses the gait's underlying kinematic and kinetic components (Cappozzo et al., 2005).

However, motion capture systems' widespread use is limited due to it's substantial monetary and infrastructural costs, prolonged time commitment for data collection, and it's requirement for specialized technicians. Thus clinics with a large daily influx of patients must rely on more practical and affordable methods. Force plates and cost-effective two-dimensional gait analysis tools are popular alternatives to determine external forces applied to the ground (ground reaction force, GRF) during gait (Kirtley, 2006, pp. 83–96) as well as the associated kinematic variables (e.g. joint angles). These assessments generate vast amounts of multivariate, time-oriented data, which need to be interpreted by a clinician in a short period of time.

Automated data analysis methods bear the potential to support the clinician during this challenging process. Still, it is a difficult task to interpret the obtained data as several parameters are inter-linked and requires considerable domain expertise. The combination of vast amounts of inter-linked clinical data derived from clinical examinations, the need for sophisticated data analysis methods, and clinical decision making requiring the judgment and expertise of clinicians strongly lends itself to the notion of VA (VA) (Keim et al., 2010a; Thomas and Cook, 2005).

Because the fact that manual analysis by domain experts is very cumbersome, automated data analysis methods are needed. In order to automate this process as much as possible, spatio-time parameter (STP) (e.g., (Tahir and Manap, 2012)) ranges for particular 'gait interferences' in comparison to the 'normal gait' are needed to be specified and categorized. This way, STP from a patient can then semi-automatically be analyzed and matched in this context.

On the other hand, this process cannot be automated completely as domain experts need to be in the loop to identify, correct, and disambiguate intermediate results. This combination of large amounts of data, complex data analysis needs, and the combination of automated data analysis with analytical reasoning by domain experts is suited to the

notion of VA (Keim et al., 2010a; Thomas and Cook, 2005).

VA may support the clinician with powerful interactive visualization techniques that are integrated in combination with semi-automated data analysis methods. Consequently this may support the clinician in interpreting complex data and drawing appropriate clinical conclusions. The clinicians' 'implicit knowledge' from prior experience is essential in the analysis process, but it is not shared with other experts or integrated in the VA system. Thus, it is logical to externalize some of the domain experts' 'implicit knowledge' and make it available as 'explicit knowledge' in the VA process (Chen et al., 2009; Wang et al., 2009). As such, it can be used to augment the visual display of data and to support (semi-) automated data analysis (knowledge-assisted VA methods). Additionally, joint learning between clinicians is enabled and the collection of expert knowledge across several clinicians allows the constructing of a comprehensive clinical knowledge database (Chen et al., 2009) (refereed to as 'explicit knowledge store' later in this article for the proposed prototype).

This work follows the paradigm of 'problem-oriented research' (Sedlmair et al., 2012b), i.e., working with real users (clinicians), thus aims at solving the aforementioned problem by means of VA.

In detail, a comprehensive prototype was developed which is intended to support the clinician in interpreting gait data during everyday clinical practice. The methods proposed in this work aim at externalizing implicit knowledge of clinicians into a knowledge database that makes these data available as explicit knowledge to other clinicians. To be able to support domain experts in their work on clinical gait analysis, it is imperative to conduct a design study (Sedlmair et al., 2012b) including a 'problem characterization and abstraction', a 'validation of the visualization design' and a 'reflection'. Specifically, we followed the 'nested model for visualization design and validation' as proposed by Munzner (2009). This model is a unified approach which structures visualization design into four levels by combining them with appropriate validation methods. This consequently reduces threats to validity at each level.

10.2 Related Work

From a data perspective, gait measurements are multivariate time series. To visualize and analyze such data, a variety of different VA approaches have been introduced in prior work.

VA for Movement Time Series

Andrienko et al. (2013) give a broad overview how VA can be used to visualize locomotion, which they refer to as 'Visual Analytics of Movement'. In their work they give recommendations how such data can be represented in the context of VA and how

these data may be for example resampled. However, they mostly focused on geo spatial datasets in relation to time. In the field of sport science, two VA systems (Janetzko et al., 2014; Perin et al., 2013) support soccer analysts in analyzing position-based soccer data at various levels of detail. Janetzko et al. (2014) additionally enrich the analysis with manually annotated events such as fouls and suggest further candidate events based on classification. An effective full automated method for human motion sequences segmentation for character animation purposes was introduced by Vögele et al. (2014). They described the fast detection of repetitions in discovered activity segments as a decisive problem of motion processing pipelines. For testing these method, they used different motion capture databases and visualized the results with stacked bar charts for comparison with other techniques.

In the context of medicine, sports and animation, domain experts can use the 'MotionExplorer' system (Bernard et al., 2013) for the exploration of large motion capture data collections. These data represent a special type of multivariate time series. Following an iterative design approach, Bernard et al. (2013) demonstrated the functionality of the 'MotionExplorer' through case studies with five domain experts. A similar approach was pursued by Purwantiningsih et al. (2016), who collected data on patients' quality of movement using serious games and different motion sensing devices. To make these multivariate time-series data accessible to clinicians, their VA solution allows hierarchical clustering and navigation in time. The exploration of equine motion and performance is an important field for veterinary medicine. In detail, gait analysis in this context is essential in diagnostics and in the emerging field of research of long-term effects of athletic performance. The VA system 'FuryExplorer' (Wilhelm et al., 2015) allows the experts to interactively explore captured multivariate time-oriented data. This system resulted in a more efficient analytical workflow for evaluation of horse motion data by the domain experts.

VA for Multivariate Time Series

The analysis of time-oriented data is an important problem for many other domains beyond movement data. In a systematic review Aigner et al. (2011) surveyed more then 100 visualization approaches and systems for time-oriented data. They give recommendation of what should be visualized, why it should be visualized and how visualization can be designed. For the purpose of visual exploration, interaction techniques and analytical methods are required as well, for which they present recommendations. Many approaches for visualizing multivariate time series are based on a form of small multiples (Tufte, 2001) where the many charts – one for each univariate time series – are juxtaposed on a common time axis. Space-efficient visualization techniques like horizon graphs (Heer et al., 2009), braided graphs (Javed et al., 2010), and qualizon graphs (Federico et al., 2014) have been designed and experimentally evaluated for such purposes. The 'LiveRAC' sytem by McLachlan et al. (2008) for IT systems management visu-

alizes time series for hundreds of parameters in a reorderable matrix of charts. The system allows for the reordering and side-by-side comparison with different levels of detail. The 'PieceStack' system Wu et al. (2016) provides an interactive environment to split and hierarchically aggregate time series based on stacked graphs Havre et al. (2002).

A different approach to tackle multivariate data applies dimensionality reduction to project multivariate measurements to 2D space, where they can be displayed as trajectory (such as a connected scatter plot) (Haroz et al., 2016). The 'TimeSeriesPaths' system (Bernard et al., 2012) applied this approach as a visual data aggregation metaphor to group similar data elements. Based on their VA approach, they provided a kind of hatching based visualization (a kind of barchart using transparency to visualize the frequency) for inner class comparison. Schreck et al. (2009) showed trajectories in small multiples and applied self-organizing maps to spatially cluster the trajectories. 'Gnaeus' by Federico et al. (2015) provides knowledge-assisted visualizations based on guidelines for electronic health records of two medical scenarios based on multivariate time series data.

Summary

The presented work is focusing on multivariate time series data to solve problems in different domains. However, none of the identified approaches provide the ability to extract and store implicit knowledge of experts in the form of explicit knowledge in a database to share with other experts, or to use automated analysis. This is a desirable feature for a VA tool, especially in clinical gait analysis, as it would support clinicians in decision making when analyzing a patient's gait and would support joint learning between different clinical experts. We only found two tools using expert knowledge to support the analysis workflow. Gnaeus (Federico et al., 2015) is designed for the analysis of electronic health care records supported by guidelines for knowledge-assistance. Additionally, the soccer data explorer by Janetzko et al. (2014) uses a 'feedback loop' to tag unknown events which are used as training data (knowledge) for the systems automated classifier, but it does not support direct interactive knowledge exploration and comparison.

CHAPTER 11

Problem Characterization & Abstraction

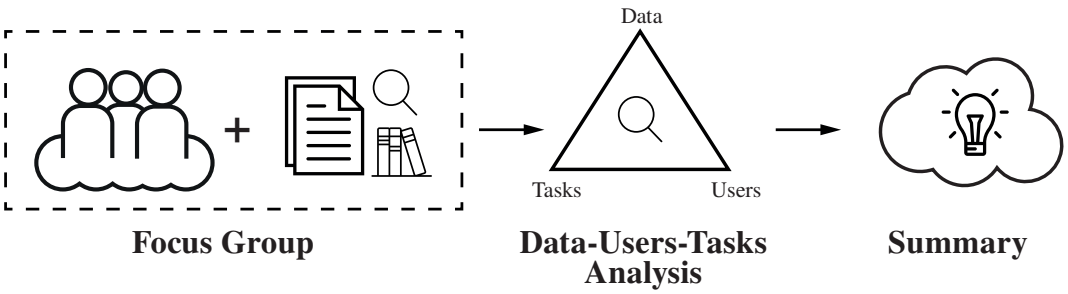


Figure 11.1: **Graphical Overview of Chapter 11** – Illustration of the topics which are covered in this chapter in relation to the second research domain: clinical gait analysis.

One primary goal in clinical decision making during gait rehabilitation is to assess whether a recorded gait measurement displays normal gait behavior or if not, which specific gait patterns (abnormalities) are present. To ensure knowledgeable results for the domain of clinical gait analysis and rehabilitation, along the triangle of data, users and tasks (Miksch and Aigner, 2014), we followed a user-centered design process (Sharp et al., 2007). Information was gathered primarily from focus group meetings (Lazar et al., 2010, pp. 192) and set in context with domain-specific literature. Based on this, we addressed the first (*domain problem and data characterization*) and second level (*operation and data type abstraction*) of the nested model by Munzner (2009). Figure 11.1 shows a graphical overview of this chapter.

11.1 Focus Group

The primary aim of the focus group meetings was to match the domain-specific vocabulary between the computer scientists and clinical experts. Additionally, these meetings were used to establish a mutual understanding of the following questions:

- What is the general workflow in the setting of a clinical gait laboratory?
- How does the clinician interact within this setting?
- Which data result in this setting and how are they structured?
- How can implicit expert knowledge be extracted and/or included?

Furthermore, we created a graphical overview for a better understanding of the different steps performed during clinical gait analysis (see Figure 11.2).

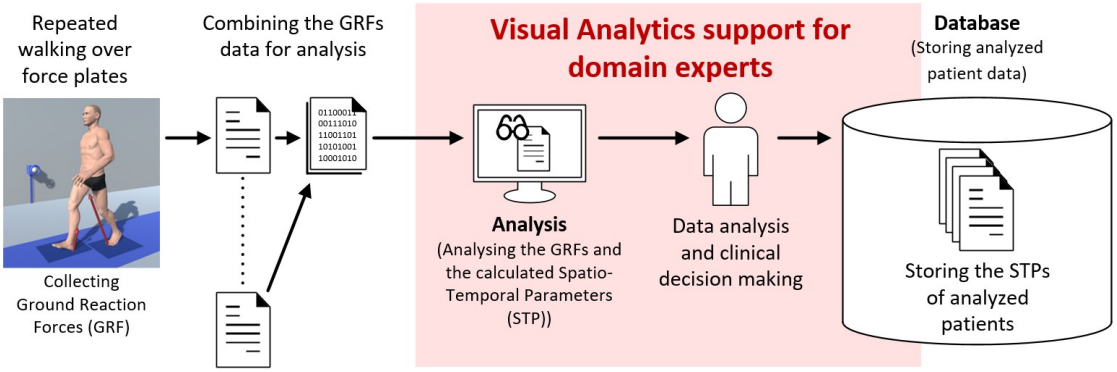


Figure 11.2: **Clinical Gait Analysis Approach** – Shows an overview of a typical gait analysis process. A patient is walking repeatedly across a walkway in the clinical gait laboratory with integrated force plates. The collected GRF data will be combined in a dataset for further analysis. Based in these data, different STPs are calculated for clinical decision making. All data of analyzed patients are stored in a database.

Participants

Seven participants comprised the focus group: two clinical gait analysis experts and two image processing experts who are working in research projects for (automated) gait pattern classification and three visual analytics experts working on knowledge-assisted VA methods for time-oriented data.

Design & Procedure

The focus group members shared a co-working space so that frequent discussions were possible and questions could be resolved quickly. Additionally, focus group meetings were held to discuss detailed questions with all members. Generally, the discussions and meetings were performed over a 13-months time frame.

Apparatus & Materials

The results of the frequent discussions and meetings were regularly documented by notes on paper, which resulted in an extensive basis for a common mutual understanding. These notes were subsequently transformed into the manuscript at hand. Additional materials were shared between focus group members using a cloud service.

Results

A sufficient amount of patient gait data is necessary to develop visualization and pattern recognition applications for clinical practice. While there have been attempts to provide such gait analysis databases (Tirosh et al., 2010), the amount of public available data is too limited. Most gait data are still located directly at the clinics.

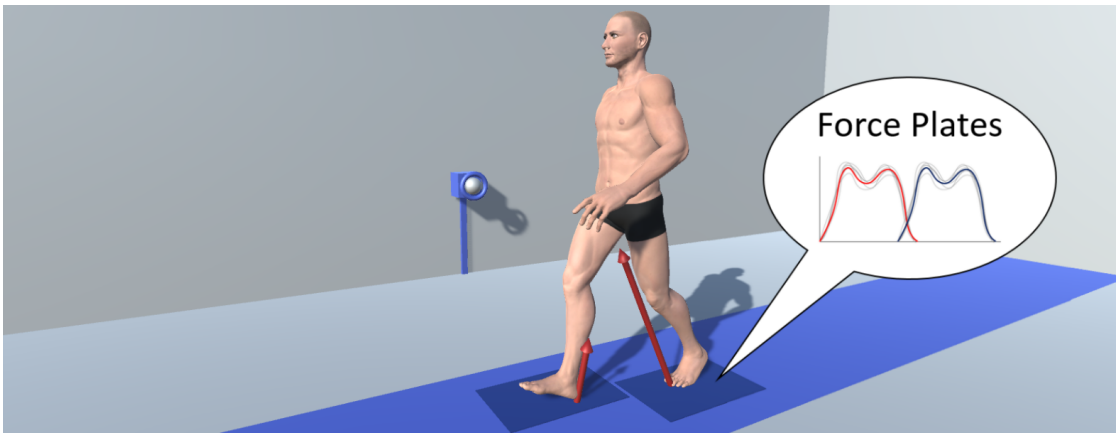


Figure 11.3: **Simple Gait Analysis Arrangement** – A simple clinical gait analysis arrangement. Two force plates are integrated into the walkway with a time-synchronized lateral video camera. The arrows show the ground reaction force measured by each force plate.

Clinical Partner: The Workers' Compensation Board (in German: Allgemeine Unfallversicherungsanstalt, AUVA[1]) is the social insurance for occupational risks for

[1] www.auva.at, accessed December 02, 2016.

more than 3.3 million employees and 1.4 million pupils and students in Austria and runs several rehabilitation centers. These centers typically use force plates to determine GRFs to assess patient gait disorders and to evaluate patient progress during physical therapy treatment. This allows for a high patient turnover and simplicity of gait measurements. The prototype described in this manuscript was developed along the needs of the AUVA's clinical gait laboratories and practice. The data incorporated in the prototype were derived retrospectively from the AUVA's database.

Gait Analysis Protocol: Typically gait analysis involves the patient walking repeatedly across an approximately 10 meter walkway in the clinical gait laboratory (see Figure 11.3). In the center of the walkway, one or more force plates with a dimension of approximately 600 x 400 mm, are integrated in the ground. During a typical gait analysis session, the patient is asked to walk across the walkway until he/she feels comfortable with the laboratory setup. Then a set of ten footsteps which is starting with the left or the right foot is produced. Each step has to be a respectively clean strikes at the center of the plate. Additionally, corresponding videos with regard to the steps are recorded.

Force Plates – Principle of Operation: There are various mechanical and electrical devices available to quantify the effect of force. Force plates in clinical settings typically adopt strain gauges or piezo-electric quartz crystals to convert force into electric signals (Nigg and Herzog, 2007, pp. 311–324). Force plates currently on the market are constructed as rectangular plates where force transducers for each axis of direction are mounted in each corner.

GRF Data Recording: The raw signals from these transducers are amplified, analog-digital converted and sampled with typically 1000 Hz to 2000 Hz. Resulting data are then used to determine the total vertical, anterior-posterior and medio-lateral force components in Newtons (Nigg and Herzog, 2007, pp. 325–326). This results in time-oriented data for all three spatial dimensions.

GRF Data Processing and Reduction: A recording of all three components during healthy gait is shown in Figure 11.4. The vertical component typically comprises a typically M-shaped waveform. This is the result of transferring weight from one leg to the other, shock absorption and active push-off for propulsion. The anterior-posterior shear forces is a breaking force during the first half of stance and propulsive during the second half (active push-off). Medio-lateral forces are often directed medially in response to the lateral motion of the body. To make absolute GRF data comparable between patients, the GRF values are normalized to body mass and expressed in percent body mass (%BM). As step times vary within and between patients, data are time-normalized to 100% stance time. These data

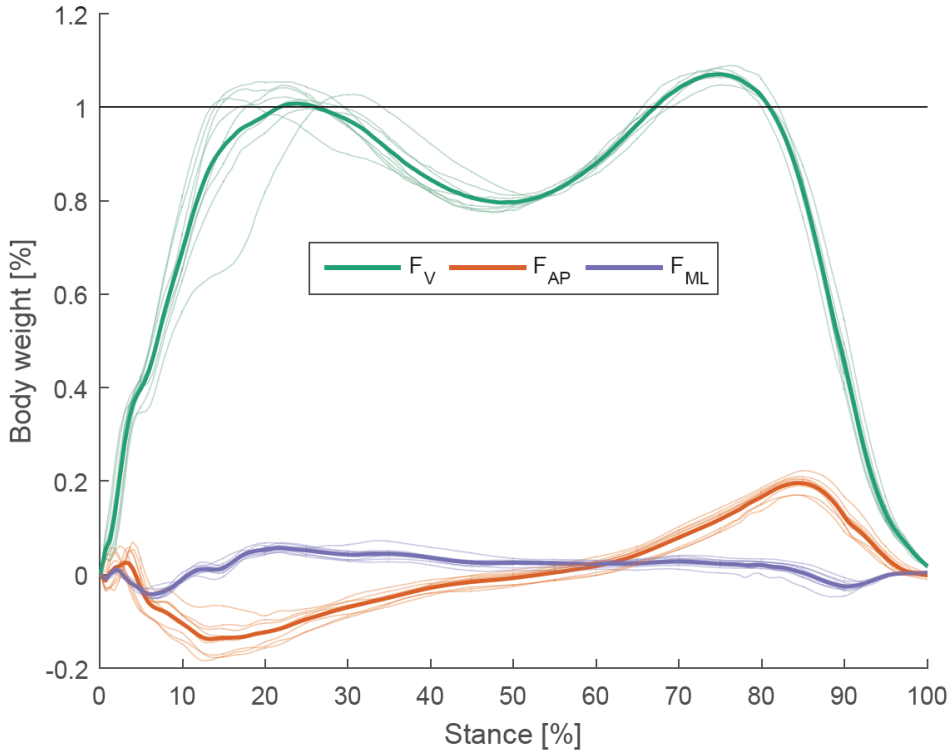


Figure 11.4: **Typical GRF Recording** – Typical recordings (consistency graphs) of GRF data during a clinical gait analysis session. A total of 10 steps are recorded, time-normalized and amplitude normalized to body mass.

are then visualized by plotting so called consistency graphs, where all ten trials are plotted in one graph to inspect variability across the steps recorded (see Figure 11.4). Then, for visual inspection simplicity, a mean representative curve from all trials and corresponding standard deviation bands are calculated and plotted. From these data several discrete biomechanical parameters can be derived, such as local peaks and valleys, and loading rates with their corresponding time points.

Spatio-Temporal Parameters of Gait: Although GRF is a very sensitive measure of gait pathology, its specificity is low since GRF comprises the motion and acceleration of whole body dynamics (Kirtley, 2006, pp. 95). Thus, additional measurements are necessary to describe the gait pattern of an individual in detail. In clinical gait analysis, the repeated movement of steps are referred to as a gait cycle, which starts with initial contact of one leg with the ground to the next ground contact of the same leg. Within this concept one can assess spatial and temporal parameters, referred to as spatio-temporal parameters (STPs) of gait (Baker, 2013). Spatial parameters may include the length of a step or a stride (two con-

secutive steps). Temporal parameters comprise the time duration of a single step, a stride, the swing phase (when the foot does not have contact to the ground), the time duration for the single support phase (when only one foot has contact to the ground), and double support (when both feet have contact). Additionally, the cadence (steps per minute), number of gait cycles per specified time, and walking speed are used to express the temporal aspect of gait.

Clinical Decision Making: The information captured during clinical examination of a patient typically sums up to a large amount of demographic data, medical-history data and gait data. The clinicians distinguish between norm data describing healthy gaits and specific gait patterns describing different categories of abnormal gait behaviors. The vast amount of data need to be interpreted by a clinician in a short period of time. Furthermore, it is difficult to interpret the obtained data as the derived parameters are inter-linked with each other. Thus, automated and sophisticated data analysis and visualization methods may bear the potential to support the clinician during this challenging process (see Figure 11.2).

Summary

All of these biomechanical data produce a large amount of inter-linked parameters and time series. These have to be interpreted by the clinician in a short period of time. Thus, several years of clinical experience as well as a good biomechanical understanding is necessary. Depending on the fact that the investigated objects are human, the diagnosis can not be fully automated. However, there is the desire of the analysts to support them in their work or to provide the analysis tools with interactive visualizations in order to improve and simplify the analysis.

11.2 Data–Users–Tasks Analysis

Above we have characterized the domain problem of clinical gait analysis using focus group meetings. In addition to these results, we followed the approach of Miksch and Aigner (2014) to structure the 'domain problem and data characterization' (which defines the first stage of the *nested model* by Munzner (2009)) along the Data-Users-Tasks triangle. This high-level framework is structured around three questions:

- "*What kinds of data are the users working with? (data)*"
- "*Who are the users of the VA solution(s)? (users)*"
- "*What are the (general) tasks of the users? (tasks)*" (Miksch and Aigner, 2014)

Based on the answers to these questions, designers of VA methods will be supported to find or design appropriate visual representations of the data along with appropriate analysis and interaction methods to support the users in their work.

Data

The data to be analyzed were collected during clinical gait analysis via force plates. The collected time series data are discrete quantitative values with an ordinal timescale and a single granularity (ms) (Aigner et al., 2011). Generally, each force plate provides three separate but synchronized time series, one for each direction in 3D space. For example, if the gait analysis setting contains two force plates, one for each foot (see Figure 11.3), the system provides six synchronized time series for analysis. Additionally, eight STPs for each foot (16 in combination) can be calculated based on the time series data.

Users

Clinical gait analysis is performed by domain experts, *physicians, physical therapists, biomedical engineers or movement scientists*. These users have a strong background in gait and movement analysis – typically holding a university degree. They command background knowledge about anatomy, bio-mechanics, gait analysis and a particular intuition on pathological gait functions which are derivations to the normal gait (walk). The users are comfortable with combining a range of different data representations such as spreadsheets, diagrams, box plots, and Matlab plots (e.g., line plots) that are mostly developed for a special hardware setting for their institution. Thus, depending on the software solutions they are working with, they have no dedicated experience with VA solutions. Yet, they are willing to familiarize themselves with a new tool because they need to perform gait analysis often and for extended periods. However, gait analysis is a specialist skill which requires experience, therefore a relatively small pool of users will employ the system.

Tasks

The primary task of a clinician in gait rehabilitation is to *assess gait performance, to analyze and interpret the acquired data and to use these information for clinical decision making*. Secondary tasks involve the identification of specific gait patterns (abnormalities) and to compare the observed data to already existing patient data sets (e.g. in the clinic's database). To support these tasks, expert knowledge might be stored in some sort of database, so that this information will be available for other clinicians. Over time such approaches could gather a vast amount of expert knowledge, which could guide experienced or young clinicians during the process of clinical decision making and interpretation of patient data.

11.3 Summary

Based on the performed focus group meetings combined with the state of the art literature we formulated a problem characterization and abstraction. Typically gait analysis involves the patient walking repeatedly across a walkway a laboratory containing one or more force plates which are integrated the ground. The raw signals from these transducers are amplified, analog-digital converted and sampled with typically 1000 Hz to 2000 Hz. The resulting in time-oriented data are then used to determine the total vertical, anterior-posterior and medio-lateral force components in Newton. To make these GRF data comparable between patients, their values are normalized to body mass and expressed in percent body mass (%BM) and are visualized by plotting consistency graphs, containing all trials. From these data several discrete biomechanical parameters can be derived (e.g., STPs) which produce a large amount of inter-linked parameters. Spatial parameters may include the length of a step or a stride. Temporal parameters comprise the time duration of a single step, a stride, the swing phase, the time duration for the single support phase, and double support. Additionally, the cadence, number of gait cycles per specified time, and walking speed are used to express the temporal aspect of gait. For the data analysis, clinicians are using various representations such as spread sheets, diagrams, box plots, and Matlab plots (e.g., line plots) that are mostly developed for a special hardware setting. Depending on the used software, they have no dedicated experience with VA solutions. The primary task of a clinician in gait rehabilitation is to *assess gait performance, to analyze and interpret the acquired data and to use these information for clinical decision making*. Secondary tasks involve the identification of specific gait patterns (abnormalities) and to compare the observed data to already existing patient data sets (e.g. in the clinic's database).

Next Steps

In the next steps, we start with the interface design and implementation based on the user centered design (Sharp et al., 2007) process. Therefore, we produced sketches in the first step followed by screen prototypes and at last the functional prototype. Additionally, we included all focus group members during the design and implementation process to get feedback about the design and the functionality of the system. Thus, it was possible to improve the design and the handling of the new tool. During the implementation of the functional prototype, we performed formative evaluations. This way, it was easy to eliminate design and functional problems quickly. At last, we performed a user study with predefined datasets and a case study with a national gait analysis expert to evaluate the prototype.

CHAPTER 12

Visualization Design & Prototypical Implementation

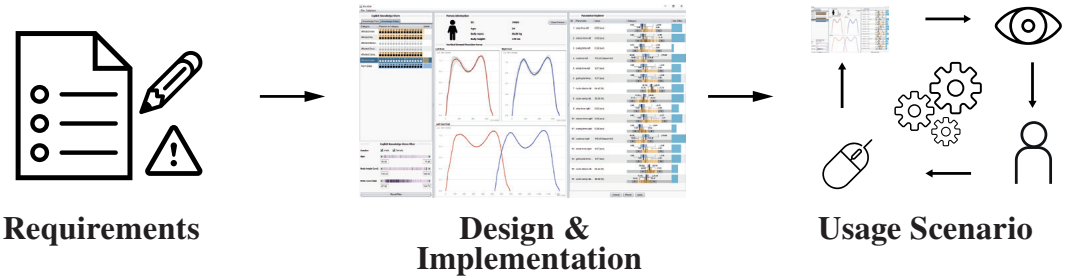


Figure 12.1: **Graphical Overview of Chapter 12** – Illustration of the main topics which are covered in this chapter with regard to the prototypical implementation of KAVAGait, the prototype of the second research domain: clinical gait analysis.

In this chapter, we present the design and prototypical implementation of our knowledge-assisted gait analysis system (KAVAGait) according to the findings described in Chapter 11. To achieve the best possible results, we worked in accordance with the *nested model* by Munzner (2009). Specifically, we focused on the third ('visual encoding and interaction design') and fourth ('algorithm design') level of Munzner's model (Munzner, 2009) and describing all steps in detail (see Figure 12.1).

12.1 Requirements

Based on the insights gained during the Data-Users-Tasks analysis in Section 11.2, we defined four key requirements (R) which have to be fulfilled by the KAVAGait prototype:

R1 Data: *Handling of complex data structures in clinical gait analysis.* To ensure the effective exploration and analysis of the GRFs and calculated STPs, a large and complex data structure needs to be modeled, stored and analyzed.

R2 Visual Representation: *Visual representations appropriate for gait analysis experts.* Clinicians use different types of diagrams, such as box and line plots, to conduct their analyses.

R3 Workflow: *Workflow-specific interaction techniques.* In relation to the clinical gait analysis workflow, it is important to provide familiar interaction techniques and metaphors to the clinicians which are needed for the identification of specific gait pattern and the observed data comparison to already existing data sets of patients for clinical decision making.

R4 Expert Knowledge: *Externalization of expert knowledge to reuse and share.* When analysts solve real world problems, they have a vast amount of data at their disposal to be analyzed and interpreted. By storing the clinicians' implicit knowledge, it can be made internally available in the system and usable to support the analysis process.

These four requirements form the basic pillars of KAVAGait, which have to be fulfilled during the design and implementation phase. While designing the visualization and interaction techniques, we focused on the defined requirements, followed by a user-centered design process (Sharp et al., 2007) for the algorithm design and implementation (see Section 12.3). During these steps, we collaborated with our focus group members (see Section 11.1) to get formative feedback.

12.2 Terms & Definitions

In this section we define the recently used terms in relation to the clinical gait analysis process as shown in Figure 11.2) (see Chapter 11:

GRF: The 'Ground Reaction Force' contains the data points collected by the patients walk(s) over the force plates in a gait laboratory. These GRFs are describing the forces for all three dimensions: the vertical forces, the anterior-posterior forces and the medio-lateral forces (Nigg and Herzog, 2007).

STP: 'Spatio-Temporal Parameters' are calculated based on the GRFs to describe and compare the gait of a patients. In KAVAGait the time duration of a single step, a stride, the swing phase and the cadence are calculated for comparison ((Baker, 2013; Kirtley, 2006; Tahir and Manap, 2012)):

Step time: Is measured in sec from the initial contact from one side (e.g., left foot) to the initial contact of the other side (e.g., right foot)

Cadence: Represents the count for the steps per minute.

Stride time: Describes two consecutive steps of the patients walk (120 / cadence).

Stance time: Measures the time past between the initial contact to to toe off of one foot in sec.

Swing time: Measures the time past between the toe off to the initial contact of one foot in sec.

Gait Cycle time: Represents the combination of the swing time and the stance time in sec for one foot.

Cycle Stance Ratio: Describes the stance time proportion in $\%$ of the gait cycle time.

Cycle Swing Ratio: Describes the swing time proportion in $\%$ of the gait cycle time.

EKS: The 'Explicit Knowledge Store' storing the extracted implicit knowledge of the clinicians as explicit knowledge which is used as support for automated gait analysis and visualization.

12.3 Design & Implementation

To keep the design in line with the needs and requirements defined earlier (see Section 12.1), we continued our user-centered design process (Sharp et al., 2007) by involving three domain experts in clinical gait analysis. We iteratively produced sketches, screen prototypes, and functional prototypes (Kulyk et al., 2007). Thus, we could gather and apply feedback about the design's usability and how well it supports the needs of the clinicians. This way, we addressed the third (*visual encoding and interaction design*) and fourth level (*algorithm design*) of the nested model (Munzner, 2009).

The design study resulted in the KAVAGait prototype (see Figure 12.2, 12.3 and 12.4), which is implemented in Java. Next, we elaborate on central design decisions.

Figure 12.2: **Interface of KAVAGait** – User interface of the KAVAGait prototype with its three main areas for gait analysis. 1) The (1.a) table structure of the 'Explicit Knowledge Store' (EKS) and the (1.b) used for filtering the included data in the EKS. 2) The patient explorer including the (2.a) 'Person Information', the (2.b) visualization of the vertical GRF (F_v) for each foot on an separated scale and the 2.c) visualization of the combined F_v from both feet. 3) Shows the 'Parameter Explorer' visualizing the 16 calculated spatio-temporal parameters (STPs) of the loaded person in relation to the 'Norm Data Category' and a second 'Selected Category'.

Input Data

The primary input data for the KAVAGait system are clinical gait analysis data, i.e. the vertical component of GRF (F_v) of both feet collected by the use of two synchronized force plates in the form of synchronized time series data combined in one import file. From these time series, we calculated eight STPs for each foot (16 discrete numbers at all) (e.g., (Kirtley, 2006; Tahir and Manap, 2012)).

These parameters can be stored in the 'Explicit Knowledge Store' (EKS) by the expert and used as explicit knowledge. This knowledge is used for automated analysis represented as graphical summary and matching (see Figure 12.2.1.a). Additional patient data on gender, age, body mass, and body height are available.

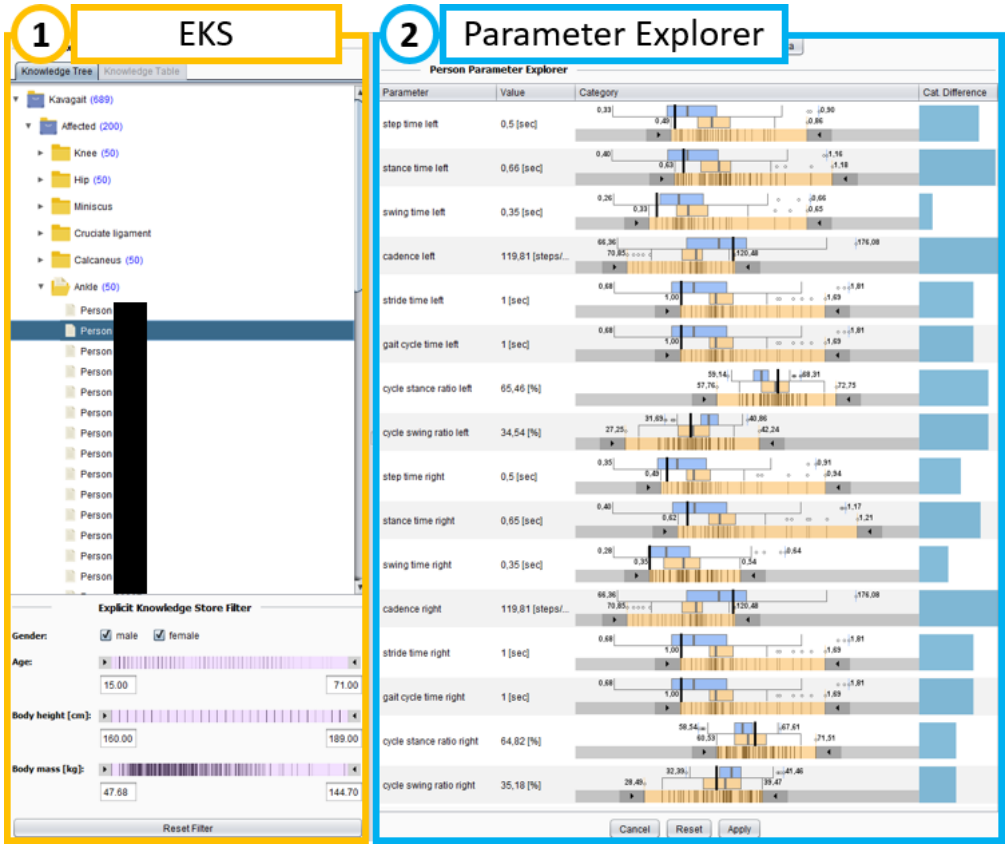


Figure 12.3: **Interface of KAVAGait** – User interface for EKS exploration and adjustment in relation to stored single 'Patients'. 1) The tree structure of the EKS while selecting a single 'Patient' for comparison and adjustment 2) with other patients in relation to the 'Norm Data Category' and the 'Category' including the patient (the black bar is used to hide the patient IDs for anonymization).

Internal Data Handling Concept

To increase the performance of our prototype, we decided to use a data-oriented design (Fabian, 2013) (e.g., used in game development and real time rendering) to organize and perform transformations on the loaded data. For example, we implemented map and set data structures as translation tables and for the input data, we mapped the strings to integers. Based on this mapping, we built an internal data structure to increase performance and to decrease memory usage. In combination with these structures, we implemented an action pipeline (similar to a render pipeline) in which it is possible to link filter options in order to realize high performance dynamic query environments and to perform all operations on the data in real time.

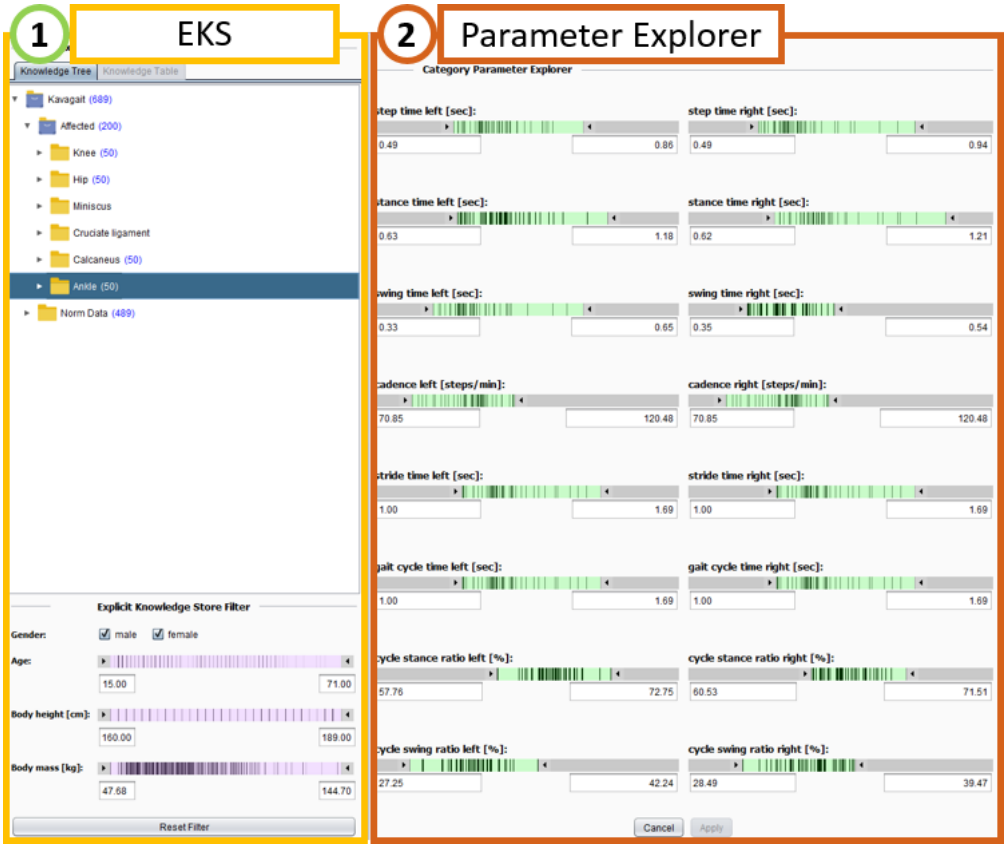


Figure 12.4: **Interface of KAVAGait** – User interface for EKS exploration and adjustment in relation to 'Categories' containing several 'Patients'. 1) The tree structure of the EKS while selecting a 'Category' for comparison 2) based on the hatching representing the time-distance parameters of the patients included in the category.

Explicit Knowledge Store (EKS)

To support gait analysts during their work, we designed the EKS related to STPs for different categories of gait patterns. Therefore the EKS is organized as categories of specific abnormal gait patterns and norm data of healthy gaits. For each category, the EKS contains the previously assigned patients and the values of their STPs. These categories are used for intercategory comparisons (between the 'Norm Data Category' and a 'Selected Category') provided by 'Interactive Twin-Box-Plots' (ITBP's) (see Figures 12.2.3 and 12.3.2) to support the analyst during clinical decision making. Additionaly, clinicians can refine the value range $[min, max]$ for each STP and category manually.

Visual Interface Design Concept

To best support physical therapists and gait analysis experts, we created an interface structure which allows working from left to right to fulfill the analysis tasks. This interface structure establishes an easy workflow concept based on multiple views for gait analysis experts. In relation to this interface structures, we situated the table structure of the EKS (see Figure 12.2.1) as well as the tree structure of the EKS (see Figure 12.3.1 and 12.4.1) to the left side of the interface to select individuals or categories of interest for exploration that always includes the related filtering options. In general, KAVAGait provides three different views, one for the exploration of newly loaded patient data (see Figure 12.2) and two for the exploration and adjustment of the EKS (see Figure 12.3).

Interaction Concept

For a better understanding of its functionality, we describe KAVAGait according to five steps based on the 'visual information seeking mantra' by Shneiderman (1996): overview first, rearrange and filter, details-on-demand, then extract and analyze further.

Overview: When the clinician loads an input file, the patient information and the F_v data from the performed analysis are displayed in the center of the prototype (see Figure 12.2.2). Additionally, the automatically calculated matching of the patient to the stored EKS categories will be represented on the left side (see Figure 12.2.1.a).

Rearrange: The clinician has the ability to rearrange each display, represented as a table, by sorting the columns (see Figures 12.2.1 and 12.2.3).

Filter: To reduce the number of patients used for the calculation of the automated category matching, the interface offers a selection of several filtering options. Thereby, the clinician can filter the EKS data by 'Gender', 'Age', 'Body height' and 'Body mass' (e.g., see Figure 12.2.1.b). The matching results displayed in the 'Knowledge Table' are updated immediately, and the graphical summary of the'Parameters in Category' (described in detail in the following 'Visualization Concept' part) gives an impression of the 16 matched value ranges of the calculated STPs (see Figure 12.2.1.a).

Details-on-Demand: If a matching result catches the clinician's interest, it can be selected from the knowledge table. This action opens a detailed visualization of the matching calculation underlying parameters in a separated table – the 'Parameter Explorer' (see Figure 12.2.3). In this table, the clinician can compare the calculated parameters of the loaded patient to the 'Norm Data Category' and to the 'Selected Category' based on ITBPs. Additionally, the clinician still gets

the information how different the categories are used for comparison ('Category Difference').

Extract: Once the clinician has found possible analysis result for the patients gait, all calculated parameters can be added to the selected category of the EKS by pressing the 'Apply' button Figure 12.2.3). Alternatively, the clinician can select some parameters of the patient in the 'Parameter Explorer' table to add only them to the selected EKS category by using the 'Apply' button. From this moment, these data are immediately integrated into the automated analysis.

Visualization Concept

In general, the KAVAGait system provides three different views to analyze new loaded patient gait data (see Figure 12.2) and for explicit knowledge generation, exploration and adjustment (see Figures 12.3 and 12.4).

New Patient Data Exploration: When loading new 'Patient' data, the person's information (see Figure 12.2.2.a) containing the 'ID', 'Age', 'Body mass', 'Body height' and 'Gender', and the measurements of the 'Vertical Ground Reaction Forces' F_v (see Figures 12.2.2.b and c) are visualized in the center view of KAVAGait (see Figure 12.2.2). These F_v data are represented for each foot (see Figure 12.2.2.b), whereby the light gray lines representing a single step and the red (left foot) or blue (right foot) line representing the mean F_v data from the single steps. Additionally, a combined representation of the F_v in a combined coordinate system is available (see Figure 12.2.2.c) for further analysis and comparison. To make the F_v (see Equation 12.1)

$$F_v := m \cdot g + m \cdot a_v \tag{12.1}$$

of a new loaded patient comparable with others, the system uses the vertical body acceleration a_v in combination with the gravity g (see Equation 12.2).

$$g + a_v = F_v/m \tag{12.2}$$

Thus, a value < 1.0 describes a negative 'slope' and a value > 1.0 describes a positive 'slope' (Kirtley, 2006, p. 85). The 16 STPs are used for comparing the newly loaded patient to 'Categories' (pathologies) of specific gait patterns (gait abnormalities) or norm data (describing healthy gait). These 16 calculated STPs are the input for the visualizations in the 'Knowledge Table' (see Figure 12.2.1.a) in the 'Params in Category' column.

Depending on the 16 STPs, the so called 'Graphical Summary' (see Figure 12.5) tells the clinician if a patient parameter x is in range $[min, max]$ with a filled black square or if it is out of range ($x < min \vee x > max$) with a outlined black square based

Figure 12.5: **Graphical Summary** – Illustration of the 'Graphical Summary' representing the calculated from the input data 16 STPs. If a patient parameter x is in range $[min, max]$ its represented by a filled black square, if it a parameter is out of range ($x < min \vee x > max$) it is represented by a outlined black square based on the explicit knowledge stored in the EKS. If the EKS did not contain data for a category (empty category), the graphical summary represents a gray square.

on the explicit knowledge stored in the EKS. If the EKS did not contain data for a category (empty category), the graphical summary represents a gray square. Thus, these three states are providing a first impression of the patient's STPs. Additionally, the third column ('Match') represents how newly loaded patients matches to the stored categories in the EKS (see Equation 12.3) supporting the clinicians during clinical decision making. In each category, Equation 12.3 has to be used where c defines the matching criteria results for the category, i iterates over all 16 STPs, σ_i is the standard deviation and μ_i is the mean for the specific STP group in the category. Additionally, x_i defines the specific STP of the newly loaded patient.

$$c = \sum_{i=1}^{n} \frac{\sigma_i}{|\mu_i - x_i|^2} \tag{12.3}$$

By using the included filtering options, explicit knowledge can be represented and modified by changing the ranges of the individual STPs for different gait abnormalities and normal gait (see Figure 12.2.1.b). When selecting a category of interest, the loaded 'Patient' can be compared to other patients in the 'Parameter Explorer' view (see Figure 12.2.3). These table contains five columns. The first column represents the STP 'ID' to create a connection to the number in the graphical summary represented in the former described 'Knowledge Table'. In the second column, the 'Parameter' name is represented and column three contains the calculated STP 'Value' for the loaded patient. The fourth column provides the Interactive Twin-Box-Plot (ITBP) (see Figure 12.6) for intercategory comparison in relation to (1) the 'Norm Data category' represented as blue box plot, (2) the 'Selected Category' of a specific gait patterns represented as orange box plot in combination with a 'Hatching Range-Slider' (HRS), which is conceptually based on the 'scented widgets' by Willett et al. (2007) and (3) the STP value of the currently loaded patient. Additionally, based on the HRS, the clinician has the ability to quickly visually adjust typical value ranges (the explicit knowledge) of the 'Selected Category'.

The last column represents the difference d between the 'Norm Data Category' and the 'Selected Category' which are visualized in the ITBP based on the Fisher discrimi-

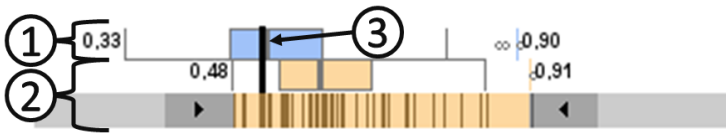


Figure 12.6: **Interactive Twin Box Plot** – Illustration of the 'Interactive Twin Box Plot' (ITBP) for intercategory comparison. 1) represents the 'Norm Data Category' as blue box plot, 2) represents the 'Selected Category' of a specific gait pattern as orange box plot in combination with a 'Hatching Range-Slider' and 3) represents the actual STP of the currently loaded patient for comparison.

nant function (see Equation 12.4) (Fisher, 1936). Based on this function, the difference between the two compared classes is calculated (the larger the difference the greater the value d). Hereby, μ_1 specifies the mean value and σ_1^2 the variance of the first category parameters and vice versa μ_2 specifies the mean value and σ_2^2 the variance of the second category parameters. It is important to note, the bigger the bar, the bigger the difference. After a clinician has finished exploring the newly loaded patient data, he/she can add them to the currently selected knowledge table category in the EKS by using the 'Apply' button. Likewise, the clinician has the possibility to undo various changes in the EKS at any time by using the 'Reset' button.

$$d = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \tag{12.4}$$

EKS Exploration and Adjustment: As mentioned before, KAVAGait contains two additional views for the exploration and adjustment of the explicit knowledge stored in the EKS. On the one hand, (see Figure 12.3.1) the clinician has the ability to select a single 'Patient' in the EKS for comparison (see Figure 12.3.2) with other patients by using ITBPs which show the relation to the 'Norm Data Category' and a 'Selected Category' of abnormalities. This visualization works the same way as formerly described for the exploration of newly loaded patient data (see Figure 12.2.3). On the other hand, (see Figure 12.4.1) the clinician can select a category visualizing each STP value range set by HRS (see Figure 12.4.2) included in the 'Selected Category'. Here, the clinician has the ability to change (overwrite) the automatically estimated range by moving the HRS for each parameter. Thereby, the HRS' color will change to dark orange and by applying the changes, the category receives an orange triangle in the tree structure to remind the clinician that changes were applied manually.

In general, each HRS included in the visual interface (see Figures 12.2, 12.3 and 12.4) can be used for filtering or adjustment of the EKS included 'Categories'.

Filter Possibilities

The implemented filtering options are organized as a filter action pipeline providing a dynamic query environment. These pipeline affects and restrict the explicit knowledge stored in the EKS (e.g., see Figure 12.2.1.b). More precisely, it deactivates stored patient data which did not match with the filter criteria combination. Thus, the range (min and max value) for each of the 16 STP changes in relation to these restrictions which effects the matching calculation (see Figure 12.2.1.a). The integration of an action pipeline for data filtering has the benefit of being easy to change, and quickly include or exclude new or not needed filtering options into KAVAGait. From top to bottom, we included the following filtering options according to the suggestions of the collaborating domain experts: The 'Gender' option provides the opportunity to include or exclude male or female data by check boxes. Below, the 'Age', 'Body height' and 'Body mass' are arranges as three HRS to set a min and a max value for the data which have to be included in the matching calculations.

Externalized Knowledge Integration

To support gait clinicians during their work, we designed the EKS related to different specific gait patterns (abnormalities) and norm data (healthy gaits). The EKS is included on the left side of the interface in two different forms depending on the task to be supported.

First, when the clinician explores newly loaded patient data, the EKS is presented in a table format (the 'Knowledge Table') (see Figure 12.2.1.a). All of the 'Categories' which are integrated in the EKS will be checked against the loaded input data automatically. Based on the explicit knowledge, the system distinguishes between the three states (in range, out of range, or no data) of the graphical summary for each of the 16 STPs in the 'Parameters in Category' column. Additionally, the system calculates how newly loaded patients match to the stored 'Categories' in the EKS. To add new knowledge to the EKS, the system provides two possibilities in this setting: On the one hand, the clinician can add the full patient dataset, representing each parameter as ITBP, by using the 'Apply' button in the 'Parameter Explorer' (see Figure 12.2.3) to the 'Selected Category' in the 'Knowledge Table'. On the other hand, the user has the ability to select a set of parameters of interest from the 'Parameter Explorer' table and add them using the 'Apply' button in the 'Parameter Explorer' to the 'Selected Category'.

Secondly, when the clinician explores the explicit knowledge, and adjusts it for single patient data or a category, the EKS is presented as an indented list (the 'Knowledge Tree') (see Figure 12.3.1). On the one hand, (see Figure 12.3.1) the clinician has the ability to select a single 'Patient' from the EKS for comparison (see Figure 12.3.2) with other patients by using the ITBP in relation to the 'Norm Data Category' and the 'Selected Category' including the selected 'Patient'. On the other hand, (see Figure 12.4.1)

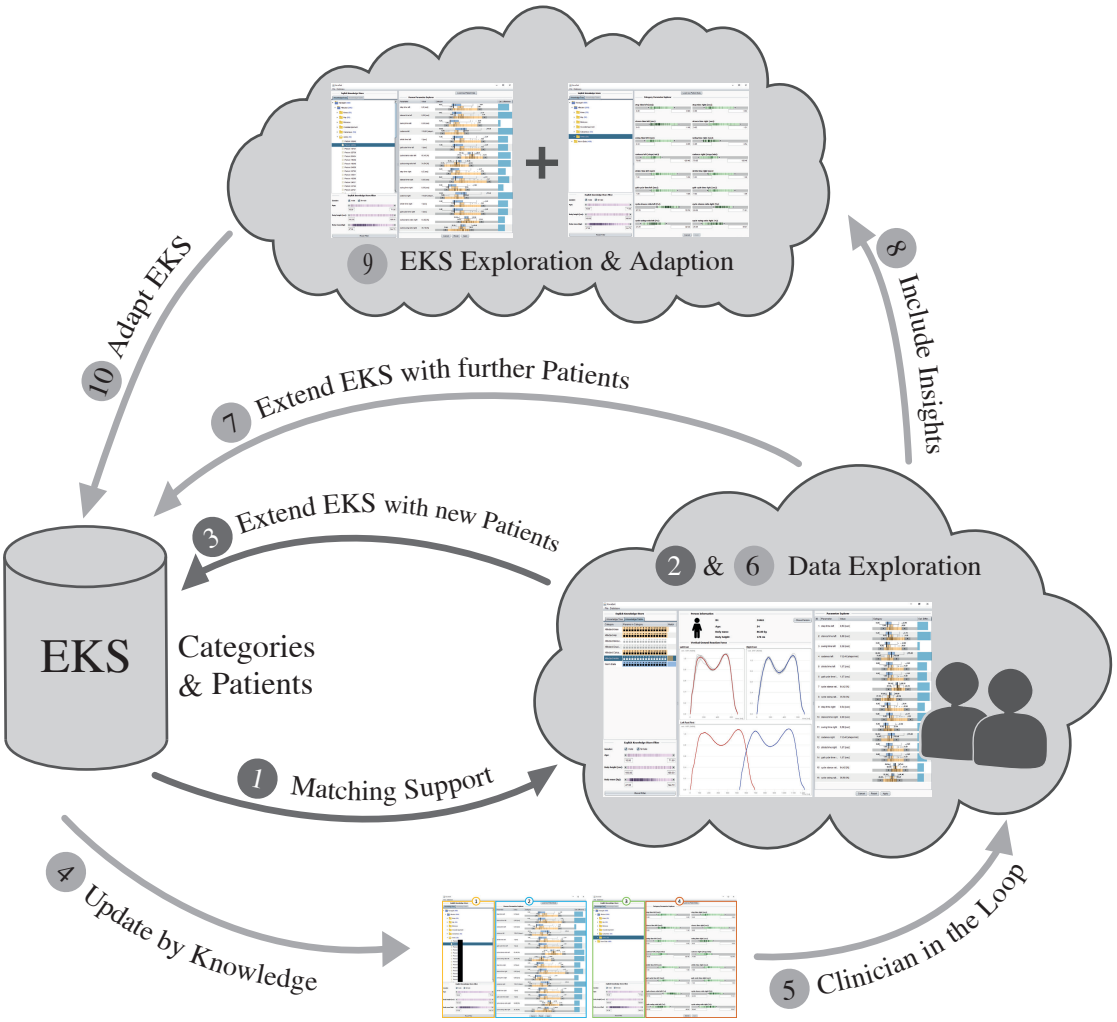


Figure 12.7: **KAVAGait Knowledge Loop** – Overview of the system's knowledge generation loop, beginning with the dark gray inner loop after loading a new patient and changing to the light gray outer loops for interactive data exploration and knowledge generation. The clinician is a major part in both loops.

the clinician can select a category visualized by HRS (see Figure 12.4.2) for each STP of the patients included in the 'Selected Category' of the EKS. Generally, at the end of each 'Category', the number of contained 'Patients' is shown in blue brackets.

Knowledge Generation Loop

Figure 12.7 provides an overview of the system's knowledge generation loop, starting at the dark gray inner loop. In general, the system's EKS stores all 'Patient' data in

several ‘Categories’ depending on patients’ pathologies (gait abnormalities) which were generated by former gait analysis sessions. If the clinician loads a new patient file, the calculated STPs will be checked automatically against the EKS (1) to calculate the category matching. Depending on the automated matching calculations, the system provides a visual representation of the results (2). From this point, the clinician can carry out the patient data exploration and analysis (the clinician is part of the knowledge generation loop). (3) During the patient data analysis, the clinician has the ability to include them into a ‘Category’ in the EKS. By adding a new ‘Patient’ to the EKS or setting filters, the system automatically refreshes the matching calculations depending on the new knowledge state (4), which brings the user into the outer (light gray) loop. Here the clinician is part of the continuously recurring loop (5), for data exploration (6) and knowledge generation (7). Additionally, the clinician has the ability to continuously include new insights (i.e. gait categorizations, value range limits) (8) depending on ‘Patient’ and ‘Category’ exploration and adjustment (9) to adapt the EKS for further automated analysis (10).

12.4 Usage Example of KAVAGait

If a clinician loads new gait analysis data of a patient, KAVAGait provides a general overview of the patients personal data and F_v from the left and the right foot (see Figure 12.2.2). In the ‘Knowledge Table’ the analyst can see a graphical summary of the 16 calculated STPs of the patient in relation to the different categories stored in the EKS (see Figure 12.2.1.a). Additionally, the matching of the patient to the different categories of the EKS will be calculated automatically and represented to the clinician as bar chart (the size of the bar relates to the match).

Gait Pattern Selection & Investigation

By selecting a category of interest in the ‘Knowledge Table’, the included data will be represented to the analyst in the ‘Parameter Explorer’ on the right side as ITBPs (see Figure 12.2.3). This way, the analyst has the ability to perform intercategory comparisons of the ‘Selected Category’ to the ‘Norm Data Category’ in combination with the actual loaded patient data for clinical decision making. Based on the different filtering options provided, the analyst has the possibility to compare the new loaded patient to a patient group of interest (see Figure 12.2.1.b).

Storing new Patterns

If the analyst wants to store new analyzed patient data to a gait pattern category in the EKS, the system provides two options. The first option is that the analyst adds the

full patient data set by pressing the ‘Apply’ button to the currently selected category of the ‘Knowledge Table’ (see Figure 12.2.3). The second option relates to a specific selection of patient parameters. The analyst selects only the parameters of interest of the currently loaded patient and adds them as a new element by pressing the ‘Apply’ button to the EKS.

EKS Exploration

To explore the explicit knowledge in the EKS, KAVAGait provides two different views to the analyst. On the one hand, the analyst has the ability to select a patient of interest in the ‘Knowledge Table’ (see Figure 12.3.1) which ends up in an ITBP representation similar to the exploration of new loaded patient data (see Figure 12.3.2). As difference to the patient exploration, this visualization does not show the F_v data of a currently stored patient. On the other hand, the clinician has the ability to select a category for exploration (see Figure 12.3.3). Here the category data are represented as HRS to the analyst (see Figure 12.3.4). Each of these two visualizations provides the ability to explore and to adjust the currently stored data in the EKS. This way, the analyst can enable or disable patient data areas which are not helpful for automated analysis based on his/her implicit knowledge. By doing so, the implicit knowledge of the analyst will be also stored as explicit knowledge in the EKS.

CHAPTER 13

Validation Strategies & Results

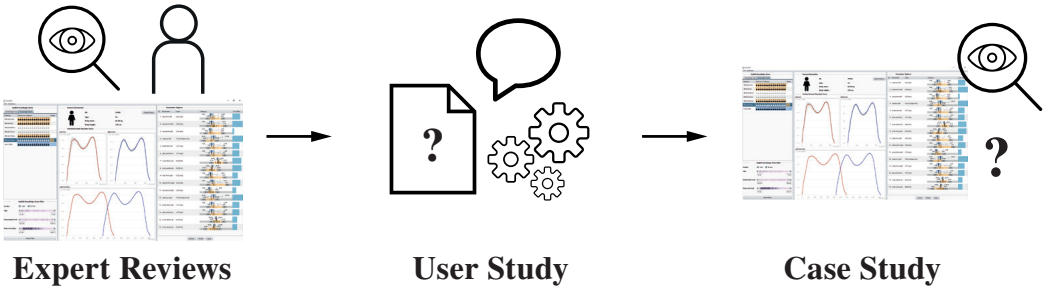


Figure 13.1: **Graphical Overview of Chapter 13** – Illustration of the topics which are covered in this chapter with regard to the validation strategy and results of KAVAGait, the prototype of the second research domain: clinical gait analysis.

To validate the KAVAGait prototype and provide evidence for its effectiveness, we followed a threefold research approach (see Figure 13.1) consisting of moderated expert reviews, user studies (Lazar et al., 2010) and a case study with a national expert. All of the insights were documented in detail to ensure reproducibility (Smuc et al., 2015) and used to improve our research prototype. All materials used, such as interview guidelines and tasks, are included in Appendix C as well as further supplement materials on https://phaidra.fhstp.ac.at/detail_object/o:1928.

13.1 Expert Reviews

In the first step, we conducted iterative expert reviews to eliminate usability issues in the basic functionality and appearance of the interface.

Participants

To validate the visual interface design, we invited three usability experts for this step (see Table 13.1). Each of them has between two to four years of experience in this field. Two of them are between 20 and 29 years of age and one is between 30 and 39 years of age. All of them have a Master's degree and advanced or expert knowledge in usability.

Person	Gender	Age	Know-ledge	Years in Field	Education
E1	f	20-29	3	2.5	MSc
E2	f	30-39	3	1	MSc
E3	m	20-29	3	3.5	MSc

Table 13.1: **Usability Experts** – Overview of usability experts who participated in the expert reviews of the KAVAGait prototype. (Gender: f := female, m := male; Knowledge: 1 := basic, 2 := skilled, 3 := advanced, 4 := expert).

Design and Procedure

Each usability expert received a short introduction to the basic features and the workflow of the system. Next, each expert walked through each feature individually and critiqued usability issues.

Apparatus and Materials

As evaluation material, we generated builds of KAVAGait in different development states and used them for the iterative expert review sessions. The review sessions were performed on a 15″ notebook with a full HD screen resolution and an external mouse for navigation. Each expert review was documented by short notes on paper by the facilitator.

Results

The basic color scheme of the prototype was found to be easily recognizable. Only the coloring of the categories was pointed out as being not well differentiated from the other elements (see Figure 12.2). The visualization metaphors (boxes, folders and sheets) for the knowledge tree visualization was developed in conjunction with one usability expert to represent a familiar structure to the analysts. The experts suggested that it is necessary that the interface automatically applies the entered parameter if the user left the focus of a filtering input box. Overall, all of the usability experts provided positive feedback on the overall interface structure of the prototype.

All of the experts' suggestions were included for a redesign and revision of the prototype in order to prevent the domain users from having basic interface issues.

13.2 User Study

A user study with six gait analysis experts was performed in October 2016 as formative evaluation of usability (Cooper et al., 2007). Each test took 1.5 hours on average and encompassed four analysis tasks, the system usability scale questionnaire (SUS) (Brooke, 1996), and a semi-structured interview built around 13 main questions to be answered. The user studies goals (G) and non-goals (NG) are defined as:

G1: Testing the functionality of the research prototype.

G2: Testing the visualization techniques for comprehensibility in relation to the domain.

G3: Testing the utility of knowledge storage and representation in the system.

NG1: Comparison of the research prototype with another analysis system.

NG2: Conducting performance tests.

Participants

We invited six gait analysis experts (see Table 13.2) to participate in the user study. One participant was also a member of the focus group for the user-centered design process and gave feedback on sketches and early prototypes (see Chapter 12). All subjects work in the field of clinical gait analysis as physical therapists, biomedical engineers or sports scientists. Additionally, all of them are involved in different gait analysis or physical therapy research projects and two of them are also working as physical therapists in a hospital.

Person (Gender)	Organization	Age	Knowledge	Years in Field	Education
P1 (f)	F	40-49	2	15	MSc
P2 (m)	F	40-49	3	15	PhD
P3 (m)	R&F	30-39	2	5	PhD
P4 (f)	R	40-49	1	1.5	MSc
P5 (f)	F&H	30-39	2	10	MSc
P6 (f)	F&H	20-29	2	7	MSc

Table 13.2: **Study Participants** – Data of the user study participants describing inter alia their education, their years in field and their knowledge in gait analysis. (Gender: f := female, m := male; Organization: R := research, F := faculty, H := hospital; Knowledge: 1 := basic, 2 := skilled, 3 := advanced, 4 := expert)

Design and Procedure

At the beginning of the user study, each participant was asked about a general impression of the user interface and which functions could be recognized. This step took approximately five minutes. Subsequently, each participant had to solve four guided **analysis tasks**. The first three included a step-wise introduction to the system and the last one was a combined analysis tasks to determine the understanding of the prototype's workflow (this step was also required for the subsequent SUS questionnaire). Each analysis task was read to the participant at the beginning of the task, and for reference, each task was handed over to the participant in printed form. For the analysis tasks, participants spent approximately 40 minutes. After the analysis task session, each participant had to fill out a **SUS** questionnaire in less than five minutes. The SUS is a standardized, technology-independent questionnaire to evaluate the usability of a system (Brooke, 1996). During this test, the participants were asked ten questions on a five-level Likert scale from strongly agree to strongly disagree. The ten questions are: 1) I think that I would like to use this system frequently; 2) I found the system unnecessarily complex; 3) I thought the system was easy to use; 4) I think that I would need the support of a technical person to be able to use this system; 5) I found the various functions in this system were well integrated; 6) I thought there was too much inconsistency in this system; 7) I would imagine that most people would learn to use this system very quickly; 8) I found the system very cumbersome to use; 9) I felt very confident using the system; 10) I needed to learn a lot of things before I could get going with this system. Finally, we performed **semi-structured interview** sessions with an average duration of 40 minutes. For this, we used an interview guideline consisting of 13 major questions addressing general system usability, filtering, using the EKS, and individual visual metaphors used in KAVAGait.

Apparatus and Materials

We used a silent and clean room without any distractions to perform the three parts of the user study for all participants under the same conditions. The four **analysis tasks** were performed on a 15″ notebook with full HD screen resolution and an external mouse. As datasets for the analysis tasks, we used two anonymous clinical gait analysis samples recorded by an AUVA clinical gait laboratory. The provided datasets included one healthy and one patient with a gait abnormality. To achieve the best outcome, we asked the participants to apply thinking aloud (Nielsen, 2010) during the whole analysis task part. For further analysis of the user test, we recorded the screen and the participant using the notebook's internal webcam. In parallel, the facilitator took notes in our pre-defined test guideline. The **SUS** questionnaire and semi-structured interview were conducted on paper in the participants' native language (German). For the detailed questions, we used small images in the **semi-structured interview** guidelines to support the participants in recalling the respective parts of KAVAGait. All test tasks and interview guidelines used are included in Appendix C.

Results on Analysis Tasks

During the four analysis tasks, all participants described their solutions to the tasks at hand by thinking aloud. They expressed problems as well as benefits of KAVAGait during their work. One major problem during the test was that the participants tried to find out which rectangle in the graphical summary, displayed in the knowledge table, relates to which twin box plot in the parameter explorer [P2, P3, P5, P6]. In relation to the graphical summary, all participants understood the gray, the empty outlined black and the filled black square shape to describe the matching of the single parameters in relation to 'no data available', 'out of the range' or 'is in range'. Three out of six participants suggested that the coloring of the box plots while comparing norm data with norm data should change to blue for both box plots. Four out of six participants stated during the third analysis task that they confirm the automated matching result after comparing to the GRF data. In general, all participants stated that the automated matching can be used as a guided entry point for further analysis using the twin box plots to analyze the patients' data in detail. Additionally, all participants handled the tasks depending on the knowledge exploration or own knowledge externalization very well. All of them understood the knowledge tree organization metaphors (boxes, folders and sheets) visualizing categories, classes and single persons.

Results on System Usability Scale (SUS)

By applying the SUS, we were able to obtain a good indication concerning the handling of our KAVAGait prototype. The results show a SUS value of 80 points on average out

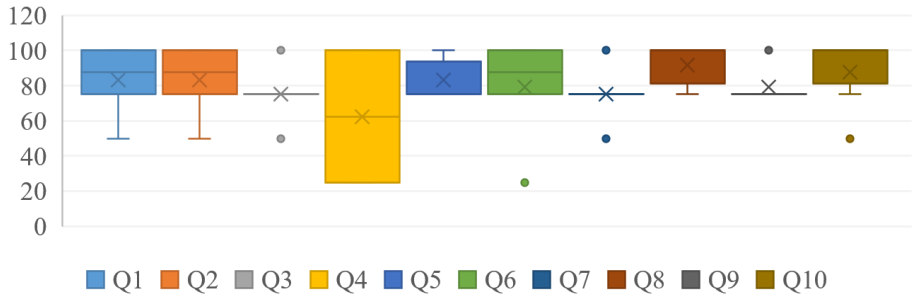


Figure 13.2: **Results of the SUS** – Illustration of the SUS questionnaires' results divided into the ten contained questions (Q1 to Q10).

of 100, which can be interpreted as good without significant usability issues according to the SUS description (Bangor et al., 2009). By dividing the SUS result into the ten questions (see Figure 13.2) it is apparent that the mean and median values for each question equal or higher then 80% out of one.

Since Sauro (2011) compared 500 SUS scores and identified the average score as 68 points. Thus, he showed that only 36% of the compared tests reached an SUS score higher than 74, and only 10% of the systems reached an SUS score greater than 80, which shows us that our system receives a grade of 'B' at this implementation state. During the test, some participants again addressed the need of a brief introduction of the included visualization techniques and the systems workflow by a technician. This results in the broad difference of SUS question four. It is very interesting that the participants voted very high for question nine and ten because they refer to the confidence using the system and the learning how to use it.

Results on Semi-structured Interviews

Here, we present the results of the performed interviews structured along 13 main questions. All participant statements quoted in this section have been translated from German to English by the authors.

Were the filter possibilities understandable? In general, all participants agreed that the filter possibilities of the system are understandable. P5 stated, "*definitely, a restriction of the data makes sense*". "*You only have to understand the effects of the filters*" [P6]. Similarly, P6 argued that she is not sure whether clinicians should or should not use this option.

Have the various visualization options contributed to your understanding? Five out of six participants indicated that the various visualization options contributed to the understanding very well. P1 and P6 stated that the line plot and box plots were understandable. P6, however, was confused with the 'Category Difference' because she

did not have any previous statistical knowledge in this regard and would therefore need more practice. P5 stated, "*it's all pretty clear. It is easy to find out which patients are decisive and its possible to make restrictions*". Additionally, P5 said that the box plots are good, but it would be helpful if both boxplots were blue when comparing norm data to norm data.

Have you understood the use of the knowledge base? All participants indicated that they have understood the use of the knowledge base, but P2 needed some additional explanation. P3 described that the distribution of the individual parameters of the persons of a class can be viewed based on the matching assignment by an experts and it also allows a great overview of the data. P5 stated that, "*it helps to reduce the results and it is very helpful when all have a similar system. You can also assume that a shared database would be good*". P4 and P5 stated that it is possible to compare a new data set with the data already stored in the EKS and then classify it. P3 added, "*if I were a clinician in a laboratory, I would add patients*".

Were the different possibilities of knowledge representation helpful for your decisions? In general, all participants have classified these options as very helpful. P1 said, "*it was very good for quick decision-making*". "*It confirms, or helps, on which one I should concentrate first, so where I set first*" [P5]. P3, P4 and P6 described the matching with the data sets as well as the line plots as very helpful. P3 added that the twin box plots are very well suited for the overview of the loaded patient to the norm data category and the selected category. This allows the user to click through the data himself, or to rely on the computed matches of the system. The reason for the classification of the loaded patient can also be found on the basis of the knowledge database. P6 said in relation to the twin box plots that she has "*no prior knowledge of what deviations I have to look for*".

How was the expert knowledge represented in the system? P1, P3, P5 and P6 immediately refer to the categories in the knowledge tree. P3 stated, "*through the categories, anybody has assigned the patients in groups*". P2, P4, and P5 also refereed to the different representations in the knowledge table. The bar charts and graphical summaries show the matching to the individual categories included in the EKS. In addition, P2, P3, P4, and P5 reported in relation to the twin box plots that they can be used to derive the sharp distinction, interval range, variance, differences and relationships for the analysis. P3 added that everything that is not based on the externalized knowledge stored in the EKS is a finding of the analyst.

Was saving of knowledge based on newly assigned patients or range adjustments of individual values understandable? All participants have confirmed to have understood the saving process of patients in the EKS. P1 stated that she would not store new patients in categories. She would leave it to other experts, since she did not work on clinical gait analysis on a daily basis. P3 added that it would be a 'nice to have' feature if the category, in which the patient is stored, appears subsequently. P5 stated, "*a sep-*

arate area for patients not yet diagnosed would be helpful". This refers to the saving of patients including them into the matching calculations. Additionally, annotations for the patient data would be helpful for the future.

Were the symbols in the knowledge tree structure of the EKS helpful and understandable? All subjects answered this question with yes. The subjects P2, P5 and P6 initially did not assign any further meaning to the various symbols, but after some practicing in the knowledge tree, they realized that they were different levels of the EKS. P3 stated that the size of the symbols in the 'Knowledge Tree' and their metaphors are a good principle to illustrate the different levels. P5 added that the different colors are also very good for distinguishing the levels.

Was the graphical summary helpful for the parameters in the category? Four out of six participants have classified the graphical summary as very helpful. Two participants stated that they had difficulties at the very beginning but they were eliminated quickly by a brief explanation. P2 and P6 described that the colors for the distinction are very good, and the individual boxes and their representations contribute to the understanding of the matching results. P2 added that a numbering of the individual parameters in the graphical summary could be helpful in order to retrieve them in the twin box plots.

Was the bar chart for the matching helpful? All participants graded the matching criterion as very helpful, as it provides a very good indication for the analysis starting point. Likewise, it is very helpful to narrow the walking problems of the patient added P5. P6 stated, "*it was the first thing I oriented myself*".

Were the shades of the individual values in the range slider helpful and if 'yes', for what? Five out of six participants immediately recognized the meaning of the shading, and one participant did not pay attention to the shading. P1, P2, P3 and P5 described it as a good basis for the assessment of the range of values (e.g., how the patients are distributed in the class). The darker the shading is the more patients are in this area [P1]. P3 and P5 added that this also gives an overview of the skewness of the distribution in order to recognize deviations. In addition, P5 also stated that it can also be seen how well a patient fits into the class or it helps to assess outliers. P6 said that she can explain the shape of the box plots based on the patient distribution.

Have you ever worked with box plots before this user study and if 'yes', why? All participants have confirmed that they know box plots from the statistics in connection with statistical tests or statistical data evaluation.

Were the visualization of the norm data category and the selected category understandable as (twin) box plot and why? Five out of six participants have indicated that they have understood the visualization immediately. Additionally, one participant required a brief additional description. P5 said, "*yes, is clearly described, contains lots of information in a short time, but its clear*". P3 added that during the comparison of the norm data and the norm data as selected class, both should be displayed in blue.

Was the comparison of the newly loaded patient, in comparison with the norm data category and the selected category helpful? Five out of six subjects have indicated that the comparison is helpful. P2, P3 and P4 added that the parameters of the patient can be compared immediately with the parameters of the norm data category and the selected category. It is also possible to see how well the individual parameters of the categories differ [P2, P3 & P4]. This makes it possible to recognize in which category the patient suits best [P4]. P6 has indicated that you need to know how to interpret the presented data.

Combined Results

All participants attending the user study confirmed a very clear system design. The used visualization metaphors are described as well interpretable, the color scheme is conclusive and the calculated matching acts trustworthy. The integrated filter options for data reduction make sense and are understandable. However, it was questioned whether clinicians should use such filtering options, as these may bias their interpretation. The various visualization options included in KAVAGait as well as the line plots and box plots contributed very well and were found to be understandable. Only one participant was confused in relation to the calculation method of the 'Category Difference'. She lacked statistical background knowledge with regard to Equation 12.4, but she understood the visualized result. Generally all participants told us that they readily understood the EKS, including the ability to compare newly loaded patients to the EKS for categorization. Additionally, a single system would be very beneficial as well as a shared EKS. The knowledge representation options ('Knowledge Tree', 'Knowledge Table', HRS and ITBP) were described as very helpful and well suited for quick decision making. They described that it was easy to get an overview of the loaded patient in relation to the 'Norm Data Category' and the 'Selected Category' based on the ITBPs. Thus, the analyst has the ability, to navigate through the data herself or to rely on the EKS based matches. Additionally, the participants referred to the categories of the EKS, including patient data of prior analysis, as well as to the different representations in the 'Knowledge Table'. Four out of six participants reported that based on the ITBP, the sharp distinction, interval range, variance, differences and relationships can be derived for the analysis. All participants noted that they have understood the saving process to the EKS. The category in which a patient is stored should appear subsequently. As 'nice to haves', a separated saving area for not yet diagnosed patients and the ability for annotations would be helpful. The separated category should not be included in the automated matching calculations. Furthermore, all participants reported that they understood the symbols represented in the 'Knowledge Tree' of the EKS but most of them did not assign any further meaning to them. Based on the different colors it was easy for the participants to distinguish the different levels of the EKS.

In relation to the 'Knowledge Table', the participants classified the 'Graphical Summary' as very helpful. Only two participants had small issues at the beginning by interpreting this visualization metaphor. Based on the coloring and the boxes for the parameters it was easy to understand the matching results. The matching criteria was indicated as helpful and good starting and orientation point for analysis. Some participants argued that a connection based on sequence numbers between the 'Graphical Summary' and the ITBPs would be very helpful. (This suggestion and other suggestions were subsequently implemented as shown in Figure 12.2.) The range slider shading and its usability for range assessment was quickly recognized by most of the participants (e.g., used for filtering, category parameter exploration and adjustment). The shading gave an overview of the underlying data deviation and outliers. Additionally, it helps to explain the ITBPs shape.

All participants had prior exposure to statistical box plots and therefore readily understood the ITBPs. They noted that this metaphor contains a lot of information but it is clearly structured. Additionally, some participants stated that the usage of the same color would be better understandable during the selection of the 'Norm Data Category' for comparison in the ITBP Section 12.3 in the future. The comparison of a (newly loaded) patient with the 'Norm Data Category' and a 'Selected Category' based on the ITBPs was described as helpful. It is possible to see the differences of the individual category parameter, but one has to know how to interpret them. In general, the KAVAGait system was described as very innovative and helpful for the analysts by providing automated analysis and pointing out possible reasons to be respected in clinical decision making.

Based on the combined insights from the four analysis tasks, the SUS and the interviews, we compiled and rated a list of found issues inspired by Nielsen's severity ratings (Nielsen, 2010). This list includes 'feature requests' (FR), 'severities' (SE) and the 'effort' for their implementation (see Table 13.3). All of these adjustments were included before the following case study.

13.3 Case Study

Before starting with the case study, we included and implemented all suggestions gathered during the previous studies (see Section 13.2) for improvement.

Participants

For our case study, we invited one leading national expert for gait rehabilitation to test and comment on our novel KAVAGait system. The expert performed substantial gait analyses over a period of more than 10 years in practice. Thus, the expert has the ability

Description	FR	SE	Effort
ITBP: Include Id numbers in relation to the fields of the graphical summary	3	-	2
ITBP: Change the coloring to only blue when comparing the 'Norm Data Category' with the 'Norm Data Category'	1	-	2
Tool tips: Add tool tips to all elements for a better description	-	2	1
Buttons: Using the same naming for the same functionality	-	2	1
Buttons: Using the same order in each menu	-	2	1
EKS: Create a separated area for not diagnosed patients (should not be included in the matching)	1	-	3

Table 13.3: **Severity Rating** – List of identified feature requests and severities (FR: 1 := nice to have, 2 := good feature, 3 := enhances usability; SE: 1 := minor, 2 := big, 3 := disaster; Effort: 1 := min, 2 := average, 3 := max).

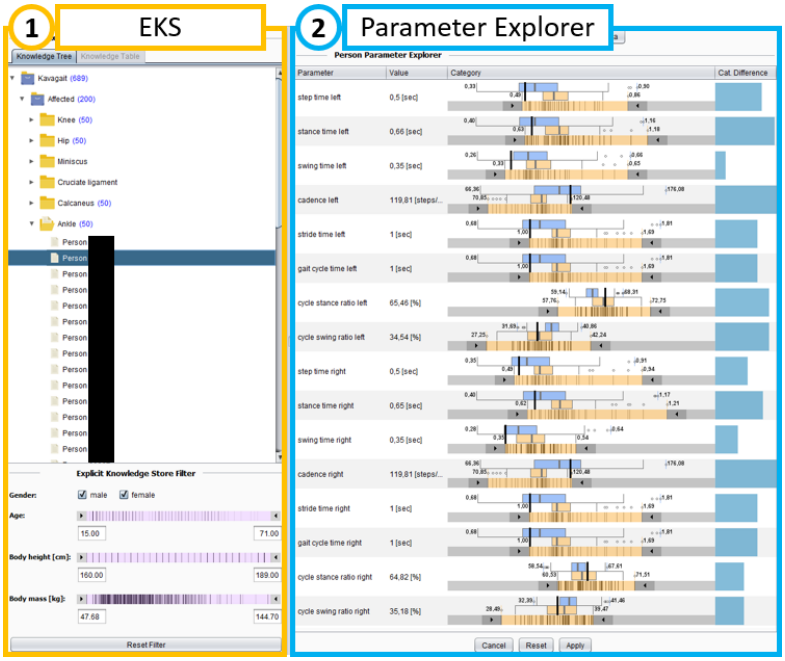
to identify gait patterns based on the representation of GRF and the calculated STPs as quantitative values (numbers).

Design and Procedure

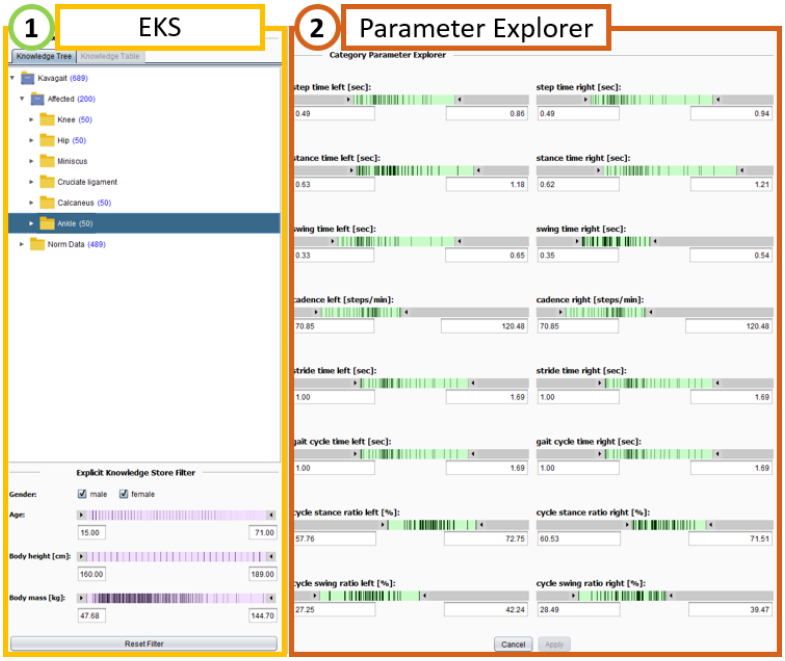
At first, the expert received a short introduction, in the form of a presentation of the basic features and the workflow of the system. Next, the expert was walked through each feature individually by an example and was asked to critically comment the prototype. Additionally, the expert could choose different patients from our data pool to explore them and tell us new insights gained using KAVAGait.

Apparatus and Materials

We met in the expert's office room to perform our case study. As materials we used a short presentation of the KAVAGait system and a build of the prototype including 489 patients in the 'Norm Data Category' and 50 patients for each out of four patient categories (ankle, calcaneus, hip and knee associated gait abnormalities). It is important to note that all the patient data included for analysis are anonymized by the data provider. The case study was performed on a 15″ notebook with a full HD screen resolution and an external mouse for navigation. The suggestions and comments stated by the expert were documented by the presenter and one observer.



(a) Patient exploration



(b) Category exploration

Figure 13.3: **Interface of KAVAGait** – User interface for EKS exploration and adjustment in relation to stored single 'Patients'. a) Structure while selecting a single 'Patient' for comparison and adjustment, b) structure while selecting a 'Category' for comparison

Results

The expert initially noted that a clinician normally focuses on two major aspects. Firstly, they are looking for asymmetries included in the data between the left and the right foot and secondly they are looking for deviations with respect to the norm data, collected by performing thousands of clinical gait analyses over the years. The walking speed of the patients is not of interest because it depends on the time of day and their constitution.

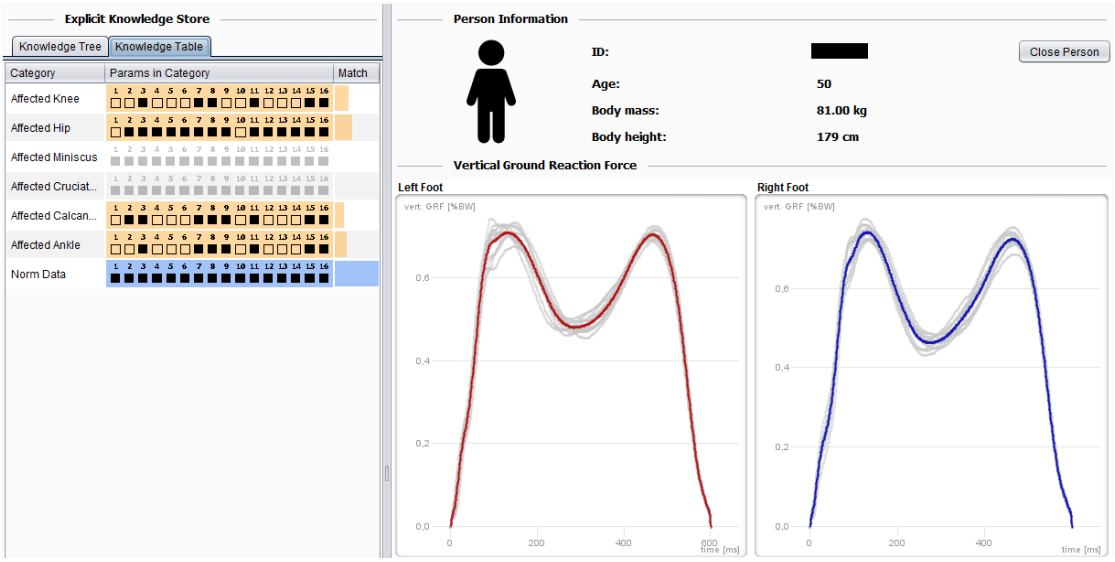
EKS Category Exploration: During the exploration of the category data stored in the EKS (used view see Figure 13.3b), the expert told us an interesting observation provided by KAVAGait. It was very interesting to see that the 'Step time' standard deviation is different between the left and the right foot in the norm data. In contrast, the 'Stance time' seems to be homogeneous. This insight was not expected by the expert.

EKS Patient Exploration: Next, the expert randomly selected a patient stored in the 'Norm Data Category'. This category normally includes only the best trials of a patient (used view see Figure 13.3a). For a fast comparison, the expert would prefer an initial sorted representation of the ITBPs by parameters (e.g., Stance time left/right, Swing time left/right). Generally, the expert found an added value in relation to the ITBP representation for the parameters. A key statement was: "*You do not have to hit the mean value directly, it is more important if the parameter values for the left and the right foot are looking similar*" by the expert. By randomly selecting a patient from the 'Knee Category' for exploration, the expert immediately was able to store the STPs. By orienting on the mean values represented from the 'Norm Data Category' and the 'Knee Category' the expert stated that the values did look quite reasonable, most of the patient values also matched with the 'Norm Data Category'. An additional suggestion was to have the ability to select two groups for comparison. In this case, the expert was interested on a comparison between the 'Knee Category' and the 'Ankle Category' because she had to change the selection. The explorability of the related mean curves of the selected patient would also be interesting. Additionally, a separation of male and female datasets in the EKS can be helpful to activate and deactivate these groups directly.

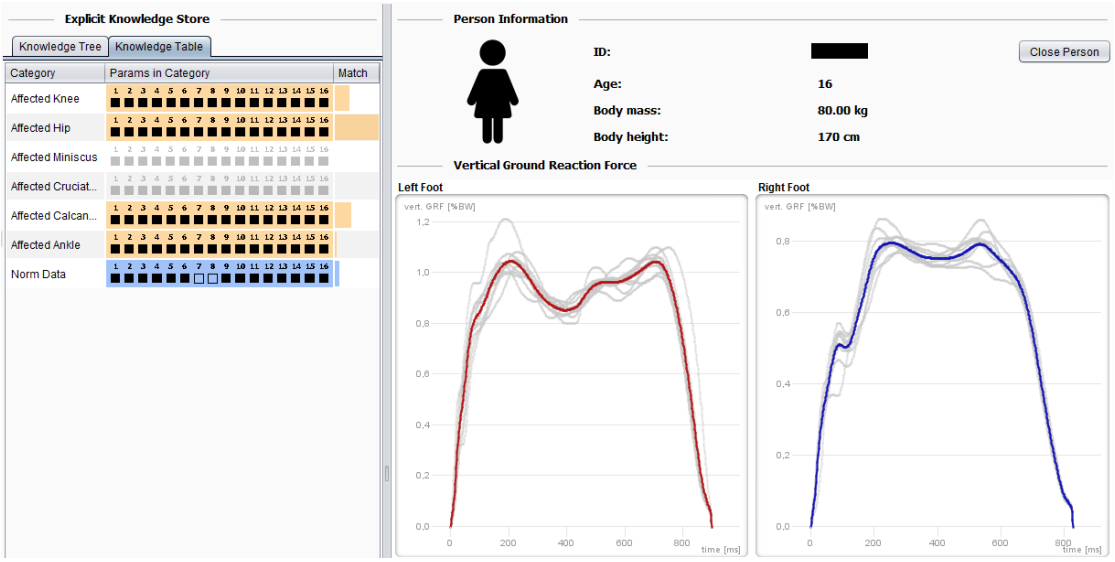
New Patient Data Exploration: To test the analysis abilities of KAVAGait, the expert loaded successively four clinical gait analysis records (1x norm, 2x hip, 1x ankle) which are illustrated in Figures 13.4 and 13.5. The loaded analysis files can contain gait abnormalities on the left, the right or both sides. The representation of the individual steps in light gray as background information from which the mean value line plots are calculated is described as very helpful. The expert analyzed each loaded patient file exactly in relation to the F_v, the calculated parameters represented in ITBPs, and the system-internal computed matching based on the EKS. The expert described the matching results in detail because several times more than one gait pattern category was indicated by the matching criteria provided in the 'Knowledge Table' of KAVAGait. This can be attributed to the fact that the patients usually undergo a therapy for a specific

problem. However, KAVAGait also recognizes abnormalities in other joints caused by specific problems. The expert stated that the community has always been aware that one abnormality can lead to others, but it has never been so well presented, as in KAVAGait. Likewise, the expert noted that a patient had to walk with a walking stick because the contact times for the right leg are considerably longer than for the left one. In addition, the expert noted that the walking aid cannot be crutches because the related F_v curve would look almost perfect and only the standardized force per se would be substantially less than 1 N/kg (force divided by body weight).

Concluding Discussion: In our data we observed a bias towards male gait data as more males work in industries where the majority of gait-related accidents occur. This is displayed when filtering by gender. The body mass index, the body height or the body mass should normally not influence the stored data for comparison if the EKS contains enough data (e.g., connection to the AUVA database). Additionally, the expert described that the system is very good for the comparison of new patient data to stored data. A future feature could be to visualize and store the vertical loading rates of the F_v for comparison. It could also be helpful to compare the patient with earlier treatments to see the evolution of rehabilitation over time. The KAVAGait system is well suited for educational training as well as for analysts who do not perform clinical gait analysis every day. For experts who are working every day in this area, the system can be a good supplement during the analysis process. Overall, the expert described KAVAGait as a very good and helpful analysis tool.

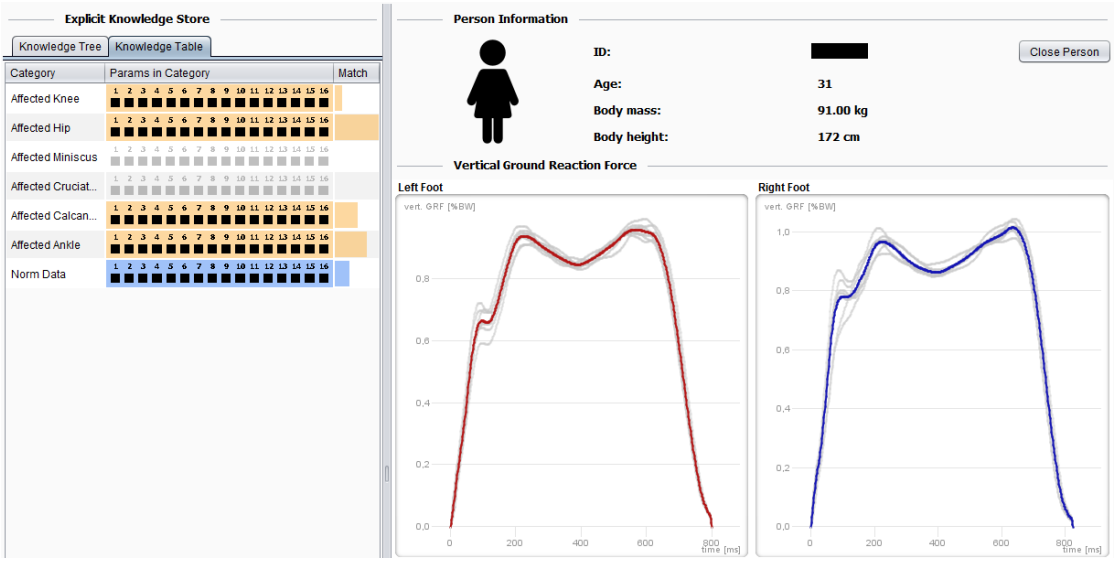


(a) Normal gait

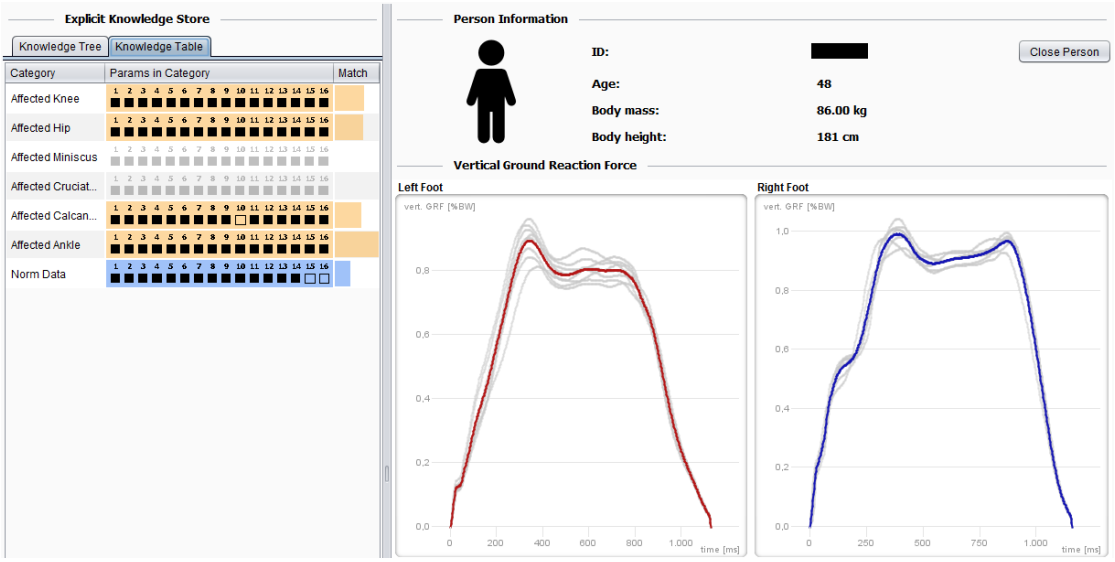


(b) Hip associated gait abnormality

Figure 13.4: **Presented Datasets** – Illustrations of the gait analysis data sets presented and discussed during the case study containing patterns of (a) a normal gait and (b) a hip associated gait abnormality.



(a) Hip associated gait abnormality



(b) Ankle associated gait abnormality

Figure 13.5: **Presented Datasets** – Illustrations of the gait analysis data sets presented and discussed during the case study containing patterns of (a) a hip associated gait abnormality and (b) a ankle associated gait abnormality.

CHAPTER 14

Reflection & Discussion

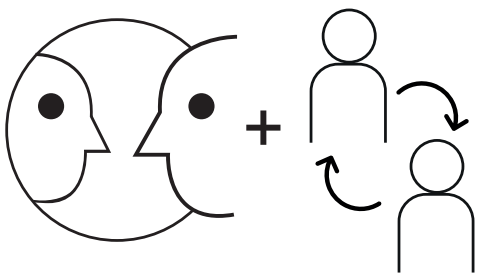


Figure 14.1: **Graphical Overview of Chapter 14** – Illustration of the reflection and discussion topic which is covered in this chapter in relation to the validation of KAVAGait, the prototype of the second research domain: clinical gait analysis.

Following the design study methodology by Sedlmair et al. (2012b), the reflection (including retrospective analysis and lessons learned) is the third contribution of a design study for the improvement of current guidelines (see Figure 14.1). The following paragraphs describe the reflection in line with the initially stated requirements from Section 12.1 (R1 – R4).

R1 Data: The data structure provided by a force plate represents the 3D ground reaction forces which are organized as synchronized time series data. Especially, the gait analysis setting, providing the data for KAVAGait consists of two force plates, one for the left and one for the right foot. Using this information as basis, we calculated eight STPs (spatio-time parameters) for each foot which were used for automated and visual comparison. Based on these calculations, the clinicians get the ability to gain new insights by comparing a single patient to different data sets (a norm data category or specific gait pattern categories). For data comparison, we designed on the one hand the graphical summary, showing the analyst how a single patient parameter is related to the parameter set of a category. On the other hand, we designed the ITBP for detailed intercategory parameter comparison between the 'Norm Data Category', a 'Selected Category' of a specific gait pattern and the patient to analyze. One future prospect can be to include several discrete key ground reaction force (GRF) parameters which are typically used for gait performance analyses in clinics (Benedetti et al., 1998). Another challenging future direction can be to provide machine learning or techniques for automated patient categorization based different unsupervised (e.g., (Christian et al., 2016; Nüesch et al., 2012; Pradhan et al., 2015)) or supervised (e.g., (Janssen et al., 2011; Pauk and Minta-Bielecka, 2016; Sangeux et al., 2015)) approaches.

R2 Visual Representation: In general, the decision for an interface providing a clear structure and understandable visual representations was well received. It was easy for the domain experts in our validation to understand the handling and to work with it. Additionally, all experts appreciated the prototype's wide range of coordinated features, which they regarded as useful for data exploration and analysis while not being overloaded. A particularly interesting outcome of the tests from the visualization design perspective is, that the 'Hatching Range-Slider' are very useful for a parameter overview and range settings during patient data exploration as well as for the EKS exploration and adaption. Additionally, the ITBPs were considered as very good for intercategory comparison in relation to a single patient parameter. This way, it is possible to see how well the patient develops in the direction of the 'Norm Data Category' for example. Another particularly notable outcome is that the 'Graphical Summary' and the 'Matching Criteria', represented in the 'Knowledge Table', were described as very interesting by the national gait rehabilitation expert. Now it is possible to see how a specific gait pattern affects to other abnormalities for example is a very helpful insight. This insight can be discovered easily with the KAVAGait system but was not brought to attention in the current setting. In the future it will be one of our goals to extend the visualization to all three force components of the provided GRFs and to add the discrete key GRF parameters to further improve the automated analysis and exploration quality. Additionally, the integration of the direct comparison of left and right

foot (i.e., gait asymmetry indices (Cabral et al., 2016)) in the twin box plots can be another future direction.

R3 Workflow: All included filter methods and the dynamic query concept providing a very fast response in KAVAGait were very well received. In general, the participants described the filtering, analysis and exploration abilities as intuitive and the usage and adaption of the EKS as easy to use. As further improvement before the case study, we added an ID for each of the 16 STP to easier relate between the graphical summary and the ITBPs depending on the sorting abilities in the 'Parameter Exploration' view. Our national expert mentioned that KAVAGait is a very easy to use tool which suits several use cases. These use cases include the support for clinical experts, assistance for less experienced clinicians, and learning and training opportunities for students for example. Based on the insights we gained during our validation studies we found that for the participants and the national expert, the visual representations of the expert knowledge and the handling of the EKS was easy to understand and to use. To further improve the workflow, the ability to annotate the patients' data would be a helpful future prospect.

R4 Expert Knowledge: As previously mentioned, the two visualization metaphors of the EKS (the'Knowledge Tree' and the 'Knowledge Table') were well received by the participants and the case study member. The knowledge organization as boxes (classes) folders (categories) and sheets (patients) was well received by most of the participants in relation to file sorting. Based on the counter after each class and category (see Figures 12.2.1 and 12.2.3), it was easy to understand how meaningful the data are for comparison and to get an overview of the included data. Reflecting on the insights gained in the validation process, it can be concluded that the analysts appreciated and could benefit from the externalized expert knowledge by sharing and adapting EKS elements during the analysis process. As a future step, explicit knowledge should also be used in the visual analytics workflow to train machine learning based analysis methods with newly gained insights.

Relation to Knowledge-assisted VA Model

Since KAVAGait was described in detail in Chapter 12, now we describe its functionalities in relation to our new 'Knowledge-assisted VA Model' (see Chapter 3). KAVAGait supports analysts during diagnosis and clinical decision making. Users can load patient gait data containing *ground reaction forces* (GRF) measurements (see Figure 14.2). These collected GRF data are visualized as line plots in the center of the interface. Additionally, 16 'Spatio-Temporal Parameters' (STP) (e.g., step time, stance time, cadence) are calculated, visualized, and used for automated patient comparison and categorization based on the novel 'Interactive Twin-Box-Plots' (ITBP). Since one primary goal

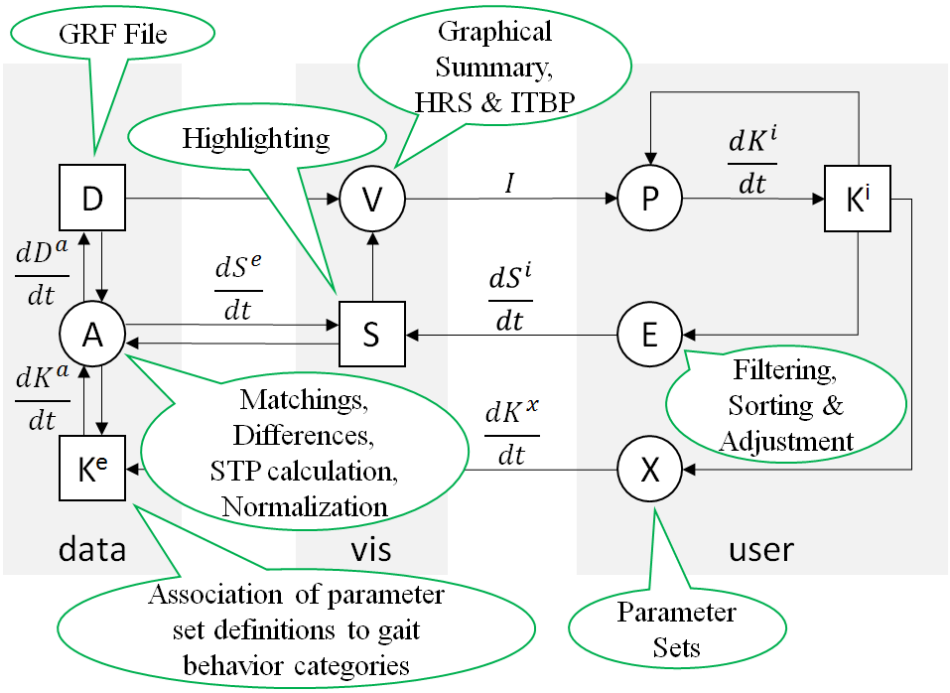


Figure 14.2: **Instantiation of Knowledge-assisted VA Model KAVAGait** – Illustrating the different prototype specific elements in relation to the related components and processes of the'Knowledge-assisted VA Model'. (Important abbreviations included in the green bubbles: GRF := ground reaction force, HRS := hatching range-slicer, ITBP := interactive twin-box-plot, STP := spatio-temporal parameters)

during clinical gait analysis is to assess whether a recorded gait measurement displays normal gait behavior or if not, which specific gait abnormality are present. Thus, the system's 'Explicit Knowledge Store' (EKS) contains several categories of gait abnormalities (e.g., knee, hip and ankle) as well as a category including healthy gait pattern data, which are used for analysis and comparison by default. However, analysts can apply their expertise (implicit knowledge) as specification $\boxed{K^i} \rightarrow \textcircled{E} \rightarrow \boxed{S}$, to filter entries by patient data (e.g., age, height, weight). Automated data analysis of newly loaded patient data is provided for categories (e.g., automatically calculated category matching) influencing the systems specification: $\{ \boxed{D}, \boxed{K^e}, \boxed{S} \} \rightarrow \textcircled{A} \rightarrow \boxed{S}$. The EKS stored explicit knowledge and the automated data analysis methods are strongly intertwined with the visual data analysis system in KAVAGait. Thus, the combined analysis and visualization pipeline consists of the following process chain, and supports the analysts while interactive data exploration $\{ \boxed{D}, \{ \boxed{D}, \boxed{K^e}, \boxed{S} \} \rightarrow \textcircled{A} \rightarrow \boxed{S} \} \rightarrow \textcircled{V}$. Based on the visualization, the generated image is perceived by the analyst, gaining implicit knowledge $\textcircled{V} \xrightarrow{I} \textcircled{P} \rightarrow \boxed{K^i}$, influencing the analysts perception $\boxed{K^i} \rightarrow \textcircled{P}$. As

data exploration and analysis is an iterative process, the analyst gains further implicit knowledge based on the adjusted visualization and driven by the specification. To generate explicit knowledge, the analyst can include the STPs of analyzed patients based on his/her clinical decisions to the EKS, which can be described as the extraction of implicit knowledge $\boxed{K^i} \rightarrow \textcircled{X} \rightarrow \boxed{K^e}$. Moreover, KAVAGait provides the ability to interactively explore and adjust the systems EKS, whereby the explicit knowledge can be visualized in a separated view $\boxed{K^e} \rightarrow \textcircled{A} \rightarrow \boxed{D} \rightarrow \textcircled{V}$. Two different options (one for a single patient and one for a category) are provided for the adjustment of the stored explicit knowledge by the analysts' implicit knowledge $\textcircled{P} \rightarrow \boxed{K^i} \rightarrow \textcircled{X} \rightarrow \boxed{K^e}$.

Contributions of KAVAGait

In relation to the design study presented in this part, the main contributions are:

- During the process of problem characterization and abstraction, a common language and understanding between domain experts (clinicians) and visualization researchers was established. Additionally, we summarized the background of clinical gait analysis and abstract specifics of data, users, and tasks to be considered in VA design.
- We presented the concept and the implementation of KAVAGait as systematically designed, developed and evaluated instantiation of an knowledge-assisted VA solution for the handling of a vast amounts of data for clinical gait analysis. The systems input data can be described as time-oriented data consisting of quantitative values on an ordinal time scale with a single granularity in milliseconds.
- New knowledge-assisted visualization approaches were used to generate understandable 'Graphical Summaries' of the data to gain a fast overview of the parameter matchings. Additionally, the novel 'Interactive Twin-Box-Plots' (ITBP) were developed providing parameter based intercategory comparisons.
- We validated the visualization design of the knowledge-assisted VA prototype for clinical gait analysis based on expert reviews, user studies and a case study. This way, we test if the used visual data representations are effective to support the domain experts while solving their analysis tasks.

Lessons Learned

During this design study, we learned that explicit knowledge opens the possibility to support clinicians during clinical decision making. Additionally, KAVAGait could also be used to share the knowledge of domain experts and for educational support. In contrast to other analysis systems (e.g., based on MatLab), KAVAGait uses analytical and

visual representation methods to provide a scalable and problem-tailored visualization solution following the visual analytics agenda (Keim et al., 2010a; Thomas and Cook, 2005). For keeping up with the large number of patients stored in the EKS, clinical gait analysts need to continuously adapt the systems settings during the clinical decision making process. Supporting such interactive workflows is a key strength of visualization systems. Clinical gait analysis in particular profits from extensive interaction and annotation because it is a very knowledge-intensive job. By providing knowledge-oriented interactions, externalized knowledge can subsequently be used in the analysis process to support the clinicians.

Transferability

The knowledge generation loop (see Figure 12.7) can be generalized for other domains taking into account domain-specific data structures and patterns of interest. On a general level, the workflow for knowledge generation and extraction is mostly similar and always includes the system user as an integral part of the loop (Endert et al., 2014). Our newly developed visual metaphors provide an easy way to inspect variability of the data (e.g., standard deviation range), allow to identify outliers in the data, and provides a easy to understand summarization of the data and automated matching results (as demonstrated in Figure 12.2.1.a). Additionally, based on the ITBPs (see Figure 12.6) it is possible to perform intercategory and patient comparisons by details on demand to find similarities in the data and to gain new insights.

Part IV

Application of the Knowledge-assisted VA Model & Future Directions

CHAPTER 15

KAVA Model Application & Comparison

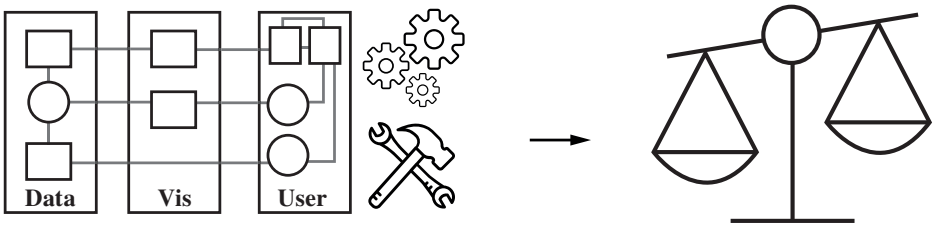


Figure 15.1: **Graphical Overview of Chapter 15** – Illustration of the topics which are covered in this chapter with regard to the model application and the comparison.

This chapter (see Figure 16.1) describes the application of the new 'Knowledge-assisted VA Model' with regard to four visualization prototypes (see Section 15.1). Second, a comparison between the 'Knowledge Generation Model of VA' is performed (see Section 15.2) to further demonstrate its applicability.

15.1 Application of the KAVA Model

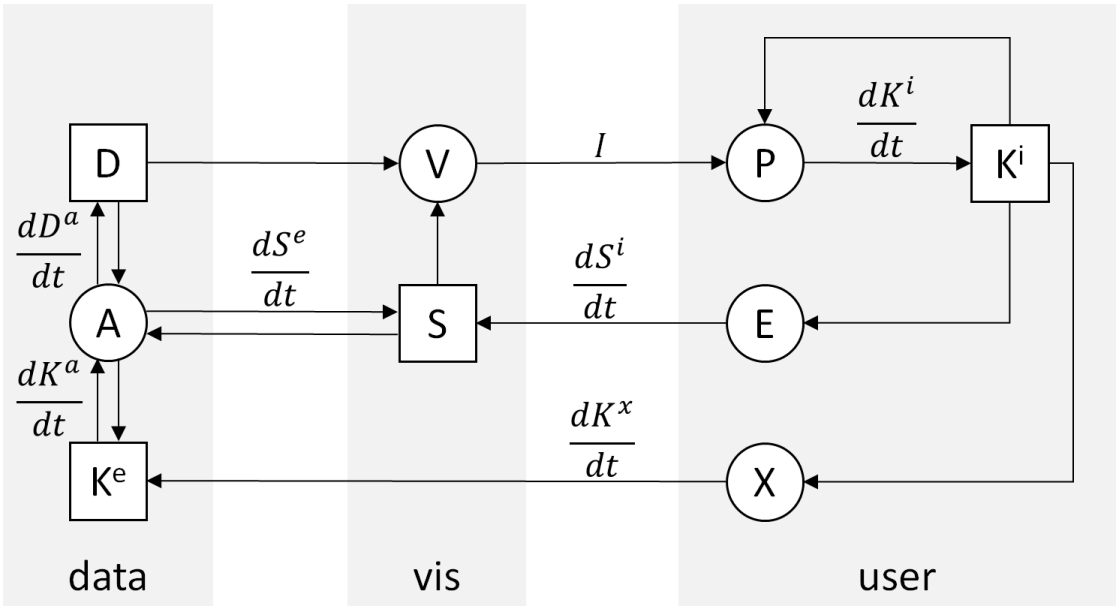


Figure 15.2: **The Knowledge-assisted VA Model** – Illustration of the 'Knowledge-assisted VA Model' conceptually grounded on Van Wijk (2005), which can be used to describe different visualization systems in relation to the contained components and processes and their connections.

To demonstrate the application of the novel 'Knowledge-assisted VA Model' (see Figure 15.2), described in detail in Chapter 3, four research prototypes are characterized. First, the two design studies prototypes KAMAS (see Part II) and KAVAGait (see Part III) included into this doctoral thesis are utilized. Second, two different systems: SEEM by Gove et al. (2014) and Gnaeus by Federico et al. (2015) are characterized, to demonstrate the general applicability of the 'Knowledge-assisted VA Model'.

Application on the Design Studies Prototypes

In general, the basic functionalities of KAMAS (see Part II) and KAVAGait (see Part III) can be described along the 'Simple Visualization Model' by Van Wijk (2005) which builds the conceptual grounding for the new 'Knowledge-Assisted VA Model'. However, based on Van Wijk's model, automated data analysis A the extraction of explicit knowledge K^e as well as the involvement of both can not be described. Due to the description of the explicit knowledge K^e generation by the user or by automated data analysis A and the analysis support by explicit knowledge K^e as well as integration of automated data analysis methods A, the novel 'Knowledge-assisted VA Model' is

needed, which includes the model by Van Wijk (2005). As reminder for the two design study prototypes (KAMAS and KAVAGait), their description is combined with a short summary.

Application on KAMAS

Figure 15.3: **Interface of KAMAS** – User interface of the KAMAS prototype with its three main areas. 1) The (1.a) tree structure of the 'Knowledge Database' (KDB) and the (1.b) 'Test Menu' used for logging during the user study. 2) The 'Rule Explorer', including the (2.a) 'Rule Overview Table' with the 'Graphical Summaries', the (2.b) 'Connection Line' to the (2.c) 'Rule Detail Table', representing the included calls and the (2.d) 'Arc-Diagram' for pattern recognition support. Additionally, on the bottom there are different filtering options (2.e–h). 3) The 'Call Explorer' interface, including (3.a) the table of calls available in the loaded file and different filtering options (3.b–d).

KAMAS is a 'Knowledge-assisted Malware Analysis System' supporting analysts (IT-security experts) in behavior-based malware analysis in order to learn about previously unknown samples of malicious software (malware) or malware families (see Figure 15.3). Thereby, the analysts are exploring preprocessed call sequences (rules), containing system and API calls to find out if the observed samples are malicious or not. If a sample is malicious, the system can be used to determine the related malware family. The analysts are supported by an integrated 'Knowledge Database' (KDB) storing explicit knowledge in the form of rules to ease the analysis process and to share it with colleagues. Thus, rule highlighting based on the KDBs included rules (comparing loaded rules with stored rules) is performed. For the integration of new knowledge into

the KDB, the analyst can, on the one hand, add whole rules and on the other hand, the analyst can add a selection of interesting calls (see Chapter 7).

Data Visualization and Exploration: If the analyst loads a cluster file of malware samples D into the system, the contained rules and calls are visualized V based on the specification S: $\{ \boxed{D} , \boxed{S} \} \rightarrow \textcircled{V}$. If there is no specification S prepared in the first visualization cycle (e.g., zooming, filtering, sorting), all read-in data are visualized. Based on the generated visualization V the image I is prepared. This image I is perceived by the analyst, gaining new implicit knowledge K^i: $\textcircled{V} \xrightarrow{I} \textcircled{P} \rightarrow \boxed{K^i}$, which also influences the users perception P: $\boxed{K^i} \rightarrow \textcircled{P}$. Depending on the gained implicit knowledge K^i, the analyst has now the ability to interactively explore E the visualized malware data by the systems provided methods (e.g., zooming, filtering, sorting), which are affecting the specification S: $\boxed{K^i} \rightarrow \textcircled{E} \rightarrow \boxed{S}$. During this interactive process, the analyst gains new implicit knowledge K^i based on the adjusted visualization V, driven by the specification S.

Explicit Knowledge and Automated Analysis Support: A KDB storing explicit knowledge K^e is integrated into KAMAS supporting the analyst. Based on the explicit knowledge K^e, automated data analysis methods A are comparing the rules included in the loaded cluster file of malware samples D based on the specification S with the stored explicit knowledge K^e. Thereby, the specification S gets adapted to highlight known rules $\{ \boxed{D} , \boxed{K^e} , \boxed{S} \} \rightarrow \textcircled{A} \rightarrow \boxed{S}$. To generate new explicit knowledge K^e, the analyst has the ability to add rules of interest to the KDB by extracting X his/her implicit knowledge K^i: $\boxed{K^i} \rightarrow \textcircled{X} \rightarrow \boxed{K^e}$. Additionally, the explicit knowledge K^e is visualized to the user $\boxed{K^e} \rightarrow \textcircled{A} \rightarrow \boxed{D} \rightarrow \textcircled{V}$ and can be turned on and off partially or completely by interaction $\textcircled{E} \rightarrow \boxed{S}$.

Application on KAVAGait

KAVAGait is a 'Knowledge-assisted VA System for Clinical Gait Analysis' (see Figure 15.4), whereby the analysts (clinicians) are supported during analysis and clinical decision making. Therefore, the analysts have the ability to load patient gait data containing ground reaction forces (GRF) measurements. The collected GRF data are visualized as line plots, describing the force over time. Additionally, 16 'Spatio-Temporal Parameters' (STP) (e.g., step time, stance time, cadence) related to the patients gait are calculated, visualized, and used for automated patient comparison and categorization. Since one primary goal during clinical gait analysis is to assess whether a recorded gait measurement displays 'normal gait' behavior or if not, which specific 'gait abnormality' is present. Thus, the system's internally 'Explicit Knowledge Store' (EKS) contains several categories of 'gait abnormalities' (relating to e.g., knee, hip, ankle) and also a category including data of 'normal gaits'. Each category consists of automatically calculated parameter ranges $[min, max]$ as category description based on the included patient

Figure 15.4: **Interface of KAVAGait** – User interface of the KAVAGait prototype with its three main areas for gait analysis. 1) The (1.a) table structure of the 'Explicit Knowledge Store' (EKS) and the (1.b) 'Explicit Knowledge Store Filters' used for filtering the included data in the EKS. 2) The patient explorer including the (2.a) 'Person Information', the (2.b) visualization of the vertical ground GRF (F_v) for each foot on an separated scale and the (2.c) visualization of the combined F_v from both feet. 3) Shows the 'Parameter Explorer' visualizing the 16 calculated spatio-temporal parameters (STPs) of the loaded person in relation to the 'Norm Data Category' and a second 'Selected Category'.

data containing the 16 calculated STPs. Based on this category descriptions, automated data analysis of new loaded patient data is provided (e.g., automatically calculated category matching). This automated data analysis supports the analysts while interactive data exploration. By adding the analysis result to the EKS, new explicit knowledge is generated. Additionally, KAVAGait provides the ability to interactively explore and adjust the system internally stored explicit knowledge (see Chapter 12).

Automated Data Analysis Visualization and Exploration: If the analyst loads a gait analysis file of a patient D into the system, first, the contained GRFs are visualized V based on the specification S: $\{ \boxed{D}, \boxed{S} \} \rightarrow \textcircled{V}$. Second, in KAVAGait, the EKS, storing the explicit knowledge K^e, and the automated data analysis methods A (e.g., matching calculations) are strongly intertwined with the analysis system and are forming its main characteristic: $\{ \boxed{D}, \boxed{K^e}, \boxed{S} \} \rightarrow \textcircled{A} \rightarrow \boxed{S}$. This pipeline immediately

calculates the 16 STPs based on the loaded GRFs D and the matching to the different EKS categories, affecting the specification S. The combination of both former described procedures can be expressed as the initial analysis and visualization pipeline: $\{ \boxed{D} , \{ \boxed{D} , \boxed{K^e} , \boxed{S} \} \rightarrow \textcircled{A} \rightarrow \boxed{S} \} \rightarrow \textcircled{V}$. However, if the specification S is not influenced by the analyst (e.g., zooming, filtering, sorting), all EKS data are used for analysis and comparison. Based on the generated visualization V, the image I is perceived by the analyst, gaining implicit knowledge K^i: $\textcircled{V} \xrightarrow{I} \textcircled{P} \rightarrow \boxed{K^i}$, which also influences the analysts perception P: $\boxed{K^i} \rightarrow \textcircled{P}$. Depending on the implicit knowledge K^i, the analyst has now the ability to interactively explore E the visualized GRFs and STPs by using the systems provided methods (e.g., zooming, filtering, sorting), influencing the specification S: $\boxed{K^i} \rightarrow \textcircled{E} \rightarrow \boxed{S}$. During this iterative process, the analyst gains further implicit knowledge K^i based on the adjusted visualization V, driven by the specification S.

Explicit Knowledge Generation and Adjustment: To generate explicit knowledge K^e, the analyst has the ability to include the STPs of analyzed patients based on his/her clinical decisions to the EKS, which can be described as the extraction X of implicit knowledge K^i: $\boxed{K^i} \rightarrow \textcircled{X} \rightarrow \boxed{K^e}$. This explicit knowledge K^e can be visualized V in a separated view, whereby the explicit knowledge K^e is automatically transformed A into a dataset D: $\boxed{K^e} \rightarrow \textcircled{A} \rightarrow \boxed{D} \rightarrow \textcircled{V}$. Different views are providing the adjustment of the stored explicit knowledge K^e by the analysts implicit knowledge K^i: $\boxed{K^e} \rightarrow \textcircled{V} \xrightarrow{I} \textcircled{P} \rightarrow \boxed{K^i} \rightarrow \textcircled{X} \rightarrow \boxed{K^e}$.

Application Summary on KAMAS and KAVAGait

As demonstrated above, KAMAS and KAVAGait are both 'Knowledge-assisted VA Systems' for different domains (IT-security and clinical gait analysis), supporting the analysts during their work. Thus, both systems are fulfilling the criteria: $|K^i| \geq 0, |K^e| > 0, A > 0, V > 0$ to be referred as KAVA system, described in detail in Section 3.3. The main difference between KAMAS and KAVAGait is obvious by the integration and the handling of the knowledge database (designated as KDB or EKS). On the one hand, in KAMAS , the KDB can be partially or fully turned on and off, thus the analyst does not have to use the (full) knowledge support, depending on his/her tasks or interests (e.g., searching for malicious rules, comparison to a single malware family). On the other hand, in KAVAGait, the EKS is fully intertwined with the analysis system to provide the necessary comparisons and matchings, supporting the analyst during clinical decision making. A further benefit of KAVAGait is the ability to visualize and adapt the explicit knowledge K^e in a separated view. Based on the explicit knowledge K^e, both systems are providing the ability of sharing knowledge to others, thus they also can be used as training system for example. Generally, all finesses of the both described systems can be expressed using the 'Knowledge-assisted VA Model'.

Application on two Research Prototypes from the Literature

As former described, the application of the new 'Knowledge-assisted VA Model' is also demonstrated on two prototypes of other researchers to show its appropriateness. First, it is demonstrated on SEEM (Gove et al., 2014), a system for malware comparison for the IT-security domain. Second, Gnaeus (Federico et al., 2015) is describing a system for electronic health record analysis in the domain of health care.

Application on SEEM

Figure 15.5: **Interface of SEEM** – Illustration of the user interface of SEEM developed by Gove et al. (2014), used for the comparison of malware samples attributes.

SEEM is a 'Similarity Evidence Explorer for Malware' developed by Gove et al. (2014) to support analysts (IT-security experts) performing the comparison between the attributes of malware samples.

Data Visualization and Exploration: At first, the analyst loads a new malware sample into SEEM for comparison (Gove et al., 2014). The visualization V provides histograms

and a Venn diagram list to visually compare the data D based on the systems specification S (e.g., focusing on the most similar samples): $\{ \boxed{D}, \boxed{S} \} \rightarrow \textcircled{V}$. During this visual comparison process, the analyst gains new implicit knowledge K^i based on the perceived image I: $\textcircled{V} \xrightarrow{I} \textcircled{P} \rightarrow \boxed{K^i}$, which also influences the users perception P: $\boxed{K^i} \rightarrow \textcircled{P}$. By using different filtering methods provided by SEEM, samples which do not have similarities can be faded out for example. Additionally, the string relation ship matrix of SEEM can be sorted "*by lexicographical order and noticed lots of misspellings and typographical errors*"(Gove et al., 2014): $\boxed{K^i} \rightarrow \textcircled{E} \rightarrow \boxed{S}$. Thereby, the systems specification gets adapted.

Explicit Knowledge Generation: SEEM also provides a kind of knowledge base to store explicit knowledge K^e, which is extracted from the implicit knowledge K^i of the analysts. This happens by adding tags to the (most) similar samples while the comparison process by the analyst: $\boxed{K^i} \rightarrow \textcircled{X} \rightarrow \boxed{K^e}$. This way, the gained implicit knowledge of the analysts can be made available for others.

System Categorization: In general, it was not possible to find out in detail if SEEM supports the analyst by using included automated data analysis methods A. In the system description by Gove et al. (2014), only the possibility of visual comparisons were described. But, the system supports the analyst by extracting implicit knowledge K^i to generate explicit knowledge K^e which can be used by others for further analysis. Based on this insights, SEEM can be categorized as a knowledge-assisted visualization system ($|K^i| \geq 0, |K^e| > 0, A = 0, V > 0 \rightarrow$ KAV)

Application on Gnaeus

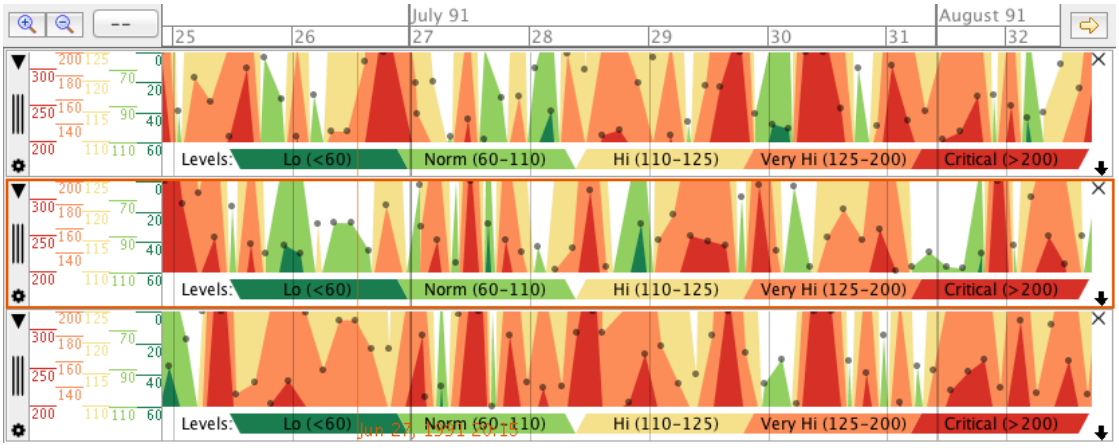


Figure 15.6: **Interface of Gnaeus** – Illustration of the user interface of Gnaeus developed by Federico et al. (2015), used for knowledge-assisted VA of electronic health records.

Gnaeus is a 'Guideline-based Knowledge-assisted VA System' to visually and interactively explore electronic health records (EHRs) for cohorts (see Figure 15.6) (Federico et al., 2015). For each patient, EHRs are containing a vast amount of multivariate time-oriented data. Evidence-based clinical practice guidelines (CPGs) are sets of statements and recommendations used to improve health care. Thereby, trustworthy comparison between treatment options in relation to risks and benefits according to patient's status and domain knowledge underneath the clinical praxis are provided. The formalization is provided in Gnaeus as computer-interpretable guidelines (CIGs).

Automated Data Analysis and Visualization: If the analyst (e.g., a clinician) loads new patient data (EHR data) D into the system, the data D are visualized V according to the systems specification S: $\{ \boxed{D}, \boxed{S} \} \rightarrow \textcircled{V}$. Based on the included explicit knowledge K^e, automated data analysis methods A (in the case of Gnaeus its a rule based reasoning engine) are specified S to analyze the EHR data D which influences the specification S: $\{ \boxed{D}, \boxed{K^e}, \boxed{S} \} \rightarrow \textcircled{A} \rightarrow \boxed{S}$. Additionally, Gnaeus computes knowledge-based temporal-abstractions of the EHR data (Shahar, 1997): $\{ \boxed{D}, \boxed{K^e} \} \rightarrow \textcircled{A} \rightarrow \boxed{D}$. The explicit knowledge K^e is usually acquired from domain experts and organized in the form of CIGs (values of multivariate time oriented-data). The explicit knowledge K^e and the automated data analysis methods A are strongly intertwined with the visualization process V of Gnaeus, which leads to a combination of the former described pipelines similar to KAVAGait: $\{ \boxed{D}, \{ \boxed{D}, \boxed{K^e}, \boxed{S} \} \rightarrow \textcircled{A} \rightarrow \boxed{S} \} \rightarrow \textcircled{V}$. Moreover, CIGs (explicit knowledge K^e) can also be directly visualized: $\boxed{K^e} \rightarrow \textcircled{A} \rightarrow \boxed{D} \rightarrow \textcircled{V}$.

Knowledge Generation and Exploration: Based on the visualization process V, the image I is generated, which is perceived P by the analyst to gain implicit knowledge K^i of the automatically analyzed data D: $\textcircled{V} \xrightarrow{I} \textcircled{P} \rightarrow \boxed{K^i}$. The gained implicit knowledge K^i also influences the analysts perception P: $\boxed{K^i} \rightarrow \textcircled{P}$. Moreover, the analyst has the ability to further explore E the presented EHR data D by several interaction methods (e.g., zooming, filtering) and to switch between visualization techniques provided by Gnaeus: $\boxed{K^i} \rightarrow \textcircled{E} \rightarrow \boxed{S}$, whereby also knowledge-assisted interactions are supported: $\boxed{K^e} \rightarrow \textcircled{A} \rightarrow \boxed{S}$.

System Categorization: Gnaeus supports the analyst during the exploration E of EHR data by automated data analysis methods A, which are based on explicit knowledge K^e (organized as CIGs). Based on the gained insights K^i, the analyst has the ability to interactively explore E the loaded EHR data D. Therefore, the analyst can use the provided interaction methods and change the visualization V techniques. Since in Gnaeus the explicit knowledge K^e is based on guide lines (CIGs) acquired by domain experts, the system provides no pipeline to extract X the implicit knowledge K^i of the analyst to make it available as K^e. Nevertheless, the system has to be specified as KAVA based on the systems included components ($|K^i| \geq 0, |K^e| > 0, A > 0, V > 0$).

Summary

Based on this new 'Knowledge-assisted VA Model', it is now possible to distinguish between four types of visualization systems (VIS, KAV, VA, KAVA) and two system types without visualization (A, KAA) for their categorization (see Chapter 3). For the demonstration of this ability, we described the functionality of four different systems in this chapter: KAMAS (see Part II) and KAVAGait (see Part III), which were implemented during this doctoral thesis, and two systems from literature: SEEM (Gove et al., 2014) and Gnaeus (Federico et al., 2015). Based on the categorization, we demonstrated that KAMAS , KAVAGait and Gnaeus are KAVA systems ($|K^i| \geq 0, |K^e| > 0, A > 0, V > 0$). In contrast, SEEM can be categorized as a KAV system ($|K^i| \geq 0, |K^e| > 0, A = 0, V > 0$), because it does not use automated data analysis methods. Based on such a general description, the identification of system differences and their comparability is supported. Thereby, designers get the ability to find out, which elements are currently not integrated in their system and how to integrate them in a general level in the future.

15.2 Description of the Knowledge Generation Model

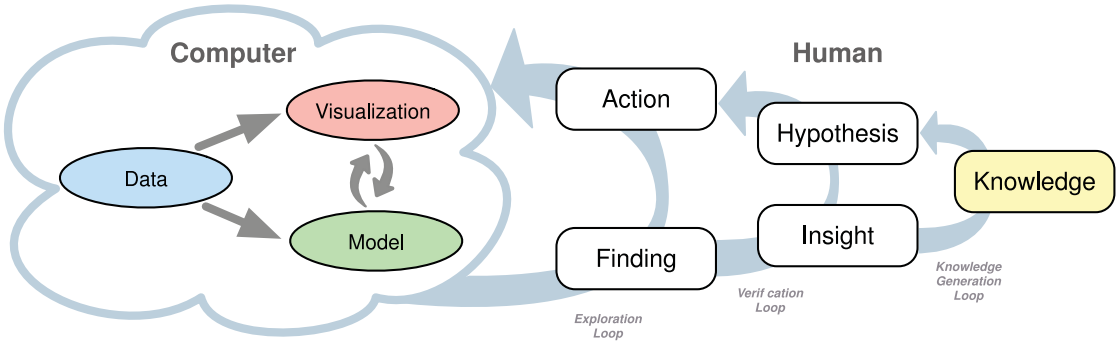


Figure 15.7: **Knowledge Generation Model for VA** – Illustration of model by Sacha et al. (2014) describing the different stages of the humans reasoning process for gaining new knowledge by the visualized data.

As demonstrated in Section 15.1 the 'Knowledge-assisted VA Model' is well applicable for describing existing systems as well as to categorize system types. In this section, the 'Knowledge Generation Model for VA' by Sacha et al. (2014) is now compared and described along the new 'Knowledge-assisted VA Model'. Thereby, the three different loops are described based on the new model.

The model by Sacha et al. (2014) is an extension to the 'VA Process' by Keim et al. (2010a) with regard to the different stages of the human reasoning process for gaining new knowledge from the represented data (see Chapter 2). Therefore, the analyst's

knowledge generation process is illustrated on the right side of Figure 15.7 and consists three loops: 1) 'exploration loop'; 2) 'verification loop' and 3) 'knowledge generation loop'. In general, knowledge generation is a very complex process where the analyst performs numerous reasoning activities, hypothesis generation and testing, in combination with direct system interaction. Therefore, the three loops are tightly intertwined where lower level loops are directed by higher level loops.

Exploration Loop: This loop describes the direct 'Actions' with a VA system and may affect all the components of the system, such as the data selections, preprocessings, visual mappings, navigation, and adaptions of the underlying computations. Each interaction causes an observation that has to be made by the analyst in order to spot a 'Finding' which are visual patterns of interest, such as outliers, sequences, or clusters. Comparing this description to the new 'Knowledge-assisted VA Model', the described 'Exploration Loop' is driven by the users implicit knowledge K^i. The performed exploration E influences the systems specification S which adjusts the visualization process V and/or the systems internally provided automated analysis methods A: $\boxed{K^i} \rightarrow \textcircled{E} \rightarrow \boxed{S} \rightarrow \{ \textcircled{A}, \textcircled{V} \}$.

Verification Loop: The analyst needs to understand and interpret spotted patterns by applying his/her implicit knowledge K^i by gaining new 'Insights' and switches to a 'Verification Phase'. Insights are visual findings (based on the perception P) that are enriched with human interpretation that contribute to validate, refine, or reject a given 'Hypothesis'. In this phase, the analyst is building and improving a mental model of the problem domain. Given a very concrete hypothesis, the analysts will seek to verify or falsify evidences. In contrast, a vague or open hypothesis leads to more exploratory analysis. This concept can also be characterized with the new 'Knowledge-assisted VA Model'. The new gained insights are depending on the perceived P image I which is prepared by the visualization process P. Thereby, the analyst gains new insights of the represented data D in relation to the hypothesis: $\textcircled{V} \xrightarrow{I} \textcircled{P} \rightarrow \boxed{K^i}$. Based on the gained implicit knowledge K^i the analysts perception P is influenced (verification or falsification): $\boxed{K^i} \rightarrow \textcircled{P}$. If the hypothesis are vague or open, the analyst has to perform further exploration E, similar to the process of the 'Exploration Loop'.

Knowledge Generation Loop: The entire verification process is driven by the analyst's implicit knowledge K^i. There are several types of knowledge and we can distinguish between two general phases of externalizing (explicit knowledge K^e) and internalizing knowledge (implicit knowledge K^i). Hypothesis and assumptions about the data D are defined and formed based on implicit knowledge K^i and trusted and verified insights are internalized as new implicit knowledge K^i. Moreover, VA allows the analyst to provide feedback to the system in order to

incorporate the analysts knowledge into the entire process. This can be also achieved by extracting the analysts implicit knowledge K^i to the system where it is made available as explicit knowledge K^e in a computerized form. Seen from the view of the 'Knowledge-assisted VA Model', the analysts implicit knowledge K^i can be extracted X and included into the system as explicit knowledge K^e: $\boxed{K^i} \rightarrow \textcircled{X} \rightarrow \boxed{K^e}$. This explicit knowledge K^e is then included into the VA process to influence the automated data analysis methods A and/or to change the systems specification: $\boxed{K^e} \rightarrow \{ \textcircled{A}, \boxed{S} \}$. Additionally, based on the experts implicit knowledge K^i, the systems specification S can be manipulated directly. Depending on specification S, the automated data analysis methods A and the visualization are adjusted: $\boxed{K^i} \rightarrow \textcircled{E} \rightarrow \boxed{S} \rightarrow \{ \textcircled{A}, \textcircled{V} \}$. Additionally, it is also possible to perform indirect adjustments for the explicit knowledge K^e and the data D: $\boxed{S} \rightarrow \textcircled{A} \rightarrow \{ \boxed{D}, \boxed{K^e} \}$.

As demonstrated above, all three loops described by Sacha et al. (2014), can also be recreated by the new 'Knowledge-assisted VA Model'. In general, the 'Knowledge Generation Model for VA' fits better for the description of the performed operations by the user. In contrast, the new 'Knowledge-assisted VA Model' can be used to describe the system's characteristics. Based on a combination of both models, the designer gets the ability to describe the user processes at a more detailed level with respect to the included components and processes to generate a detailed system abstraction.

CHAPTER 16

Discussion & Future Directions

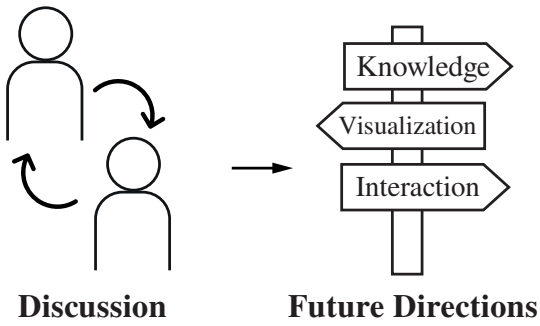


Figure 16.1: **Graphical Overview of Chapter 16** – Illustration of the topics which are covered in this chapter with regard to the discussion of the new 'Knowledge-assisted VA Model' as well as the future directions.

This chapter (see Figure 16) includes the discussion (see Section 16.1) of the new 'Knowledge-assisted VA Model' in relation to the model's application, the costs and the possibilities to be used as systems architectural blueprint as well as its limitations. Additionally, the answers to the research questions which were defined in Chapter 1 are discussed Second, future directions (see Section 16.2) in relation to the model as well as to knowledge-assisted VA are included. Finally, this chapter contains the conclusion (see Section 16.4) to sum up this doctoral thesis.

16.1 Discussion

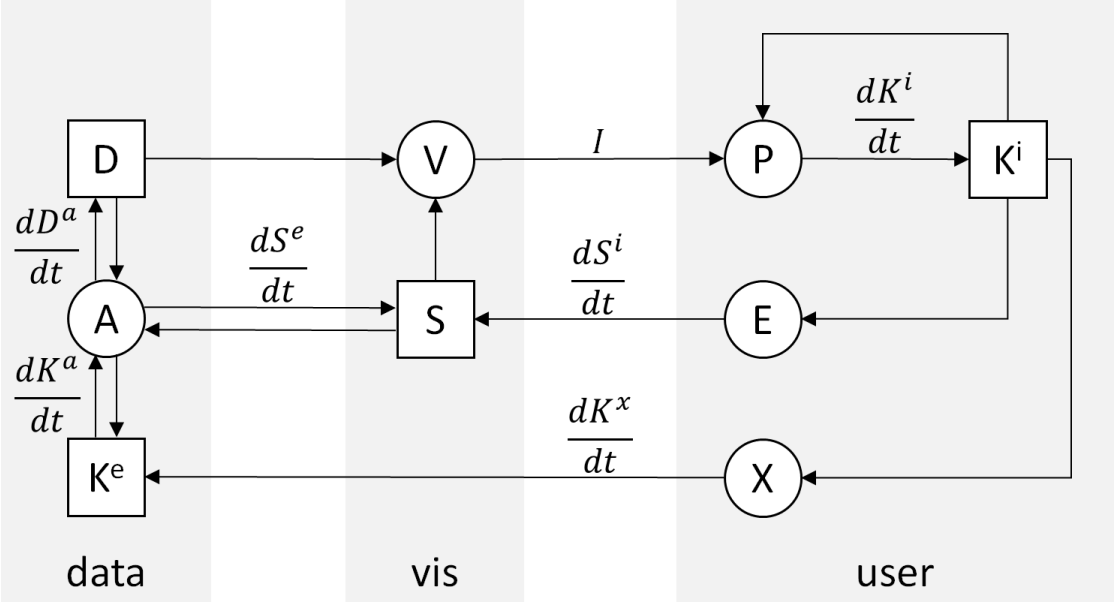


Figure 16.2: **The Knowledge-assisted VA Model** – Illustration of the 'Knowledge-assisted VA Model' conceptually grounded on Van Wijk (2005), which is discussed in relation to its descriptive power, the generative power and the evaluative power (Beaudouin-Lafon, 2004) as well as to answer the general research questions.

In this section, we discuss the novel 'Knowledge-assisted VA Model' (see Figure 16.2) in terms of its: 1) descriptive power; 2) evaluative power; and 3) generative power, described by Beaudouin-Lafon (2004), in order to demonstrate its practical applicability. As Beaudouin-Lafon (2004) pointed out, a good theoretical model needs to "strike a balance between generality (for descriptive power), concreteness (for evaluative power) and openness (for generative power)", which are contradicting goals. The novel 'Knowledge-assisted VA Model' is a rather high-level model focusing slightly more onto generality. Additionally, the limitations as well as the answers to the research questions of this doctoral thesis are included.

Descriptive Power: Describing the Characteristics of Existing Approaches

As described in Chapter 3, the 'Simple Visualization Model' by Van Wijk (2005) was used as conceptual grounding to generate the 'Knowledge-assisted VA Model'. To describe the externalization X of the users implicit knowledge K^e in a computerized form and the knowledge generation by automated data analysis A, we added several new

components and processes. Based on this new model, four types of visualization systems (VIS, KAV, VA, KAVA) and two system types without visualization (A, KAA) can be described and distinguished (see Equation 16.1).

$$\textbf{System Types} := \begin{cases} |K^i| \geq 0,\ |K^e| = 0,\ A = 0,\ V > 0 \ \Rightarrow\ \textbf{VIS} \\ |K^i| \geq 0,\ |K^e| > 0,\ A = 0,\ V > 0 \ \Rightarrow\ \textbf{KAV} \\ |K^i| \geq 0,\ |K^e| = 0,\ A > 0,\ V > 0 \ \Rightarrow\ \textbf{VA} \\ |K^i| \geq 0,\ |K^e| > 0,\ A > 0,\ V > 0 \ \Rightarrow\ \textbf{KAVA} \\ |K^i| > 0,\ |K^e| = 0,\ A > 0,\ V = 0 \ \Rightarrow\ \textbf{A} \\ |K^i| > 0,\ |K^e| > 0,\ A > 0,\ V = 0 \ \Rightarrow\ \textbf{KAA} \end{cases} \tag{16.1}$$

To demonstrate the 'descriptive power' (Beaudouin-Lafon, 2004) of the new 'Knowledge-assisted VA Model' (see Figure 16.2) six different visualization systems are described, including the two design study prototypes: KAMAS (see Part II) and KAVAGait (see Part III), which were implemented during this doctoral thesis, and two other visualization systems: SEEM (Gove et al., 2014) and Gnaeus (Federico et al., 2015).

System Categorization: As demonstrated in Chapter 15, KAMAS and KAVAGait are both fulfilling the criteria: $|K^i| \geq 0, |K^e| > 0, A > 0, V > 0$ to be referred as KAVA system. Additionally, SEEM can be categorized as a KAV system ($|K^i| \geq 0, |K^e| > 0, A = 0, V > 0$) and Gnaeus can be specified as KAVA system, based on the systems included components ($|K^i| \geq 0, |K^e| > 0, A > 0, V > 0$).

System Comparison: By using the 'Knowledge-assisted VA Model', each system can be described in relation to the components integrated in the model's three main areas: 1) data; 2) visualization and 3) user (see Figure 16.2) and their connections. Based on this general description, the identification of differences between systems and their comparability is supported.

For Example: The main difference between the described systems is obvious by the integration and the handling of the knowledge database. In KAMAS, the KDB can be partially or fully turned on and off, depending on the analysts tasks or interests (e.g., comparing a sample to a malware family). In contrast, KAVAGaits EKS is fully intertwined with the analysis system, providing comparison and matching to support the analyst during clinical decision making. In Gnaeus, explicit knowledge K^e is based on guidelines which are defined by domain experts. Nevertheless, the system does not provide the extract X of implicit knowledge K^i to make it available as explicit knowledge K^e. SEEM is categorized as knowledge-assisted visualization (KAV) system, because no automated data analysis methods A are included. However, the extraction X of implicit knowledge K^i for explicit knowledge K^e generation is supported.

System Extension: In relation to a system description, which is based on the 'Knowledge-assisted VA Model', designers get the ability to find out, which elements are currently

not integrated in their system (e.g., explicit knowledge K^e, knowledge extraction X, automated data analysis methods A). If such elements are needed in the future, it is possible to discover how they can be integrated in a general level and which elements have to be connected.

Model Comparison: Finally, to demonstrate the 'descriptive power' of the new developed 'Knowledge-assisted VA Model', we compared our model to the 'Knowledge Generation Model for VA' by Sacha et al. (2014). Thereby, we found out that all three loops (exploration, verification and knowledge generation), described by Sacha et al. (2014), can also be described and reproduced by the new 'Knowledge-assisted VA Model'. On the one hand, the 'Knowledge Generation Model for VA' is better applicable for the description of the performed operations on the user side. On the other hand, the new 'Knowledge-assisted VA Model' fits better to the describing a system's characteristics. In combination with a description based on the 'Knowledge Generation Model for VA', the designer can describe the processes at a more detailed level with respect to the included components and processes (loops) by using the 'Knowledge-assisted VA Model' to generate corresponding system abstraction and vice versa.

Evaluative Power: Costs of Knowledge-assisted VA

The novel 'Knowledge-assisted VA Model', which is introduced in this doctoral thesis, can also be used to evaluate systems/approaches in terms of their cost optimization. As specified by Van Wijk (2005) we are assuming that a community of n homogeneous users are using the visualization V to visualize a dataset m times. Therefore, each user needs k exploration steps per session and a time t. Additionally, in "*the real world, the user community will often be highly varied, with different K_0's and also with different aims*" (Van Wijk, 2005). Depending on this, he described four different types of costs which now can be extended by the generation of explicit knowledge K^e whereby l knowledge generation steps will be fulfilled:

Initial Development Costs $C_i(S_0)$: Relates to implementation and the acquiration of new hardware.

Initial Costs per User $C_u(S_0)$: Refers to the learning and the adapting of the system.

Initial Costs per Session $C_s(S_0)$: Depends on the data converting and the initial specification.

Perception and Exploration Costs C_e: In relation to watch and understand the visualization as well as its modification for exploration.

Knowledge Extraction and Computation Costs C_k: Relates to the extraction of the implicit knowledge of the user, the generation of knowledge by automated analysis methods, and the computation of both.

Based on this five major cost elements, the total costs can be now calculated as described in Equation16.2:

$$C = C_i + nC_u + nmC_s + nmkC_e + nmlC_k \tag{16.2}$$

Thus, the new return of the former described costs can be described by the value $W(\Delta K^i, \Delta K^e)$. In this case, the generated implicit knowledge $\Delta K^i = K^i(t) - K^i(0)$ and the generated explicit knowledge $\Delta K^e = K^e(t) - K^e(0)$ per session, has to be multiplied by the total number of sessions. The total number of sessions is defined by number of users n and the number of visualization uses m (see Equation16.3) which returns the knowledge gain G. At this point, it is important to note that ΔK^i and ΔK^e are evaluated for itself and not added. However, it can be assumed that a high value in ΔK^i can yield a high value at ΔK^e through the extraction of the implicit knowledge. Additionally, the total profit can be described by $F = G - C$ (see Equation16.4)

$$G = nmW(\Delta K^i, \Delta K^e) \tag{16.3}$$

$$F = nm(W(\Delta K^i, \Delta K^e) - C_s - kC_e - lC_k) - C_i - nC_u \tag{16.4}$$

Van Wijk (2005) described that high values for n, m, $W(\Delta K^i, \Delta K^e)$ and low values for C_i, C_u, C_s, C_e, C_k, k and l are positive. This tells us that a great KAVA system is used by many users, gaining a high value of knowledge and extracting it to the system without spending time and money to hardware and training. It is important to note that users which are gaining a lot of implicit knowledge K^i during the data exploration, have the ability to generate more explicit knowledge K^e. By including this K^e into the system, the user gets the ability to use the explicit knowledge generated by himself, by others and by automated analysis methods for his/her exploratory purposes to achieve his/her goals. This leads us to the assumption that the VA process is not only improved but also accelerated, which was confirmed by our two performed qualitative user studies (see Chapters 8 and 13). Additionally, by sharing the generated knowledge, the users get the opportunity to learn from others and to improve and gain new insights.

From the view of interaction costs (in approximately a combination of C_e, $C_u(S_0)$, $C_s(S_0)$), which are described by Lam (2008) as: "*less is more*", can be optimized by reducing the effort of execution and evaluation. Thereby, the knowledge-assisted VA process moves parts of the specification effort from the 'user' side of the model (see Figure 16.2) to the 'visualization' and 'data' side (automated data analysis A and specification S based on the explicit knowledge K^e). Additionally, automated analysis methods are supporting the user by analyzing the data based on S and K^e. Based on this support, the analyst has the ability to gain new implicit knowledge K^i which can be extracted as K^e to adjust S and A.

Chen and Golan (2016) described that the most generic cost is energy in terms of the computer (e.g., run an algorithm, create a visualization) as well as the human (e.g.,

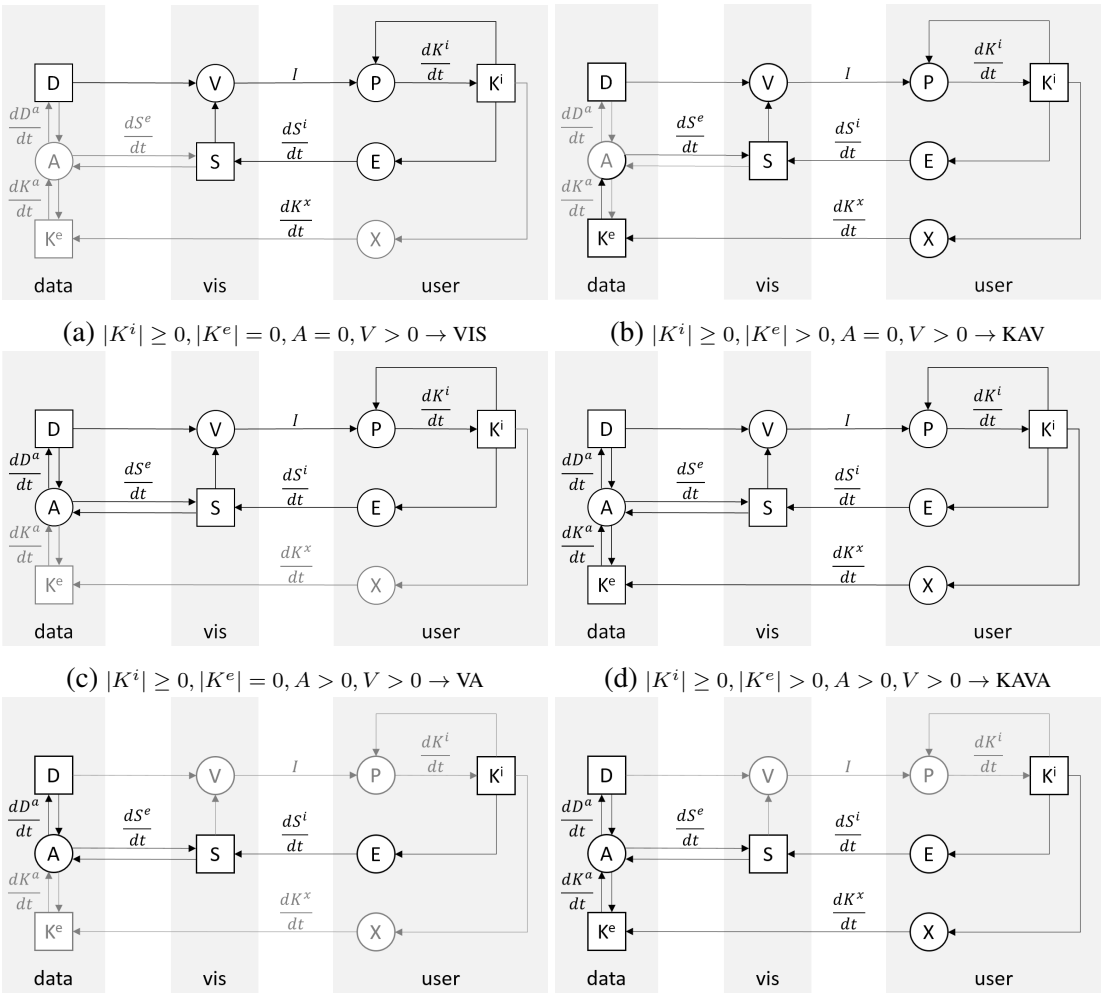
read data, view visualization, decision making). A measurement of the computer energy consumption is a common practice, but the measurement of the user activities are mostly not really feasible (Chen and Golan, 2016). Therefore, time t can be a point for the measurement as well as the amount of performed exploration steps k and knowledge generation steps l. Additionally, Crouser et al. (2017) described that a model currently cannot elaborate how much a user is doing, its only possible to measure how often the human is working. In the case of our new KAVA model (see Figure 16.2), we can measure how often the specification S was done by the human (e.g., by exploration E) or by the computer (e.g., automated analysis methods A) as well as the generation of explicit knowledge K^e).

Generative Power: Model as Architectural Template

The novel 'Knowledge-assisted VA Model' (see Figure 16.2), described in detail in Chapter 3, is a high-level blueprint for the design of four types of visualization systems: VIS, KAV, VA and KAVA, systems (see Figure 16.3.a–d) and two system types without providing visualizations: A, KAA (see Figure 16.3.e and 16.3.f) can be designed. Therefore, the designer needs to acquire the following domain-specific information (e.g., by using the design triangle (Miksch and Aigner, 2014)):

- What kind of data have to be analyzed?
- Who are the users of the system?
- Which visual metaphors are appropriate and supportive for the users?
- What are the tasks and goals to be achieved by using the visualization system?
- How can expert knowledge be supportive integrated into the system?
- Which automated data analysis methods should be integrated?

Based on the answers of the questions above, the designer has the ability to choose s suitable blueprint for a future system. This blueprint does not guide the designer through the systems underlying source code design. In contrast, its supportive by understanding the needed connection between the different system components (e.g., data D, visualization V, explicit knowledge K^e) and the user (e.g., exploration E, perception P, implicit knowledge K^i) which have to be established. Additionally, it answers the questions: *What should happen in a {VIS, KAV, VA, KAVA, A, KAA} system, if the user {...}*?This is demonstrated on three example questions:



(a) $|K^i| \geq 0, |K^e| = 0, A = 0, V > 0 \rightarrow$ VIS

(b) $|K^i| \geq 0, |K^e| > 0, A = 0, V > 0 \rightarrow$ KAV

(c) $|K^i| \geq 0, |K^e| = 0, A > 0, V > 0 \rightarrow$ VA

(d) $|K^i| \geq 0, |K^e| > 0, A > 0, V > 0 \rightarrow$ KAVA

(e) $|K^i| \geq 0, |K^e| = 0, A > 0, V = 0 \rightarrow$ A

(f) $|K^i| \geq 0, |K^e| > 0, A > 0, V = 0 \rightarrow$ KAA

Figure 16.3: **Supported System Types** – Illustration of the four visualization system blueprints (a–d) and the two system blueprints without visualization (e and f) supported by the 'Knowledge-assisted VA Model' for interactive data analysis. It is important to note that gray drawn elements are not included in the specific system type.

Example Question 1: *What should happen in an VIS system, if the user interactively explores the data?* Based on the exploration E, the specification S has to be influenced. This specification S accordingly adapts the visualization process V, which is generating the image I. The resulting image I is perceived P by the user who generates implicit knowledge K^i: $(E) \rightarrow [S] \rightarrow (V) \xrightarrow{I} (P)$ (see Figure 16.3a).

Example Question 2: *What should happen in a VA system, if the user would like to perceive the analyzed data?* The automated data analysis methods A have to

process the data D, and to adapt the specification S: $\boxed{D} \rightarrow \textcircled{A} \rightarrow \boxed{S}$. Based on the adapted specification S, the visualization process V creates the image I, which is perceived P by the user who generating implicit knowledge K^i: $\boxed{S} \rightarrow \textcircled{V} \xrightarrow{I} \textcircled{P} \rightarrow \boxed{K^i}$ (see Figure 16.3c).

Example Question 3: *What should happen in a KAVA system, if the user would like to extract his/her implicit knowledge and integrate it as explicit knowledge?* By using the provided interaction methods, the user has to extract $\textcircled{X}$ his/her implicit knowledge $\boxed{K^i}$. Thereby, the implicit knowledge $\boxed{K^i}$ gets externalized, modeled and stored as explicit knowledge $\boxed{K^e}$ in the systen, which subsequently influences the automated data analysis methods $\textcircled{A}$ and the specification $\boxed{S}$ of the system: $\boxed{K^i} \rightarrow \textcircled{X} \rightarrow \boxed{K^e} \rightarrow \{\ \textcircled{A}, \boxed{S}\ \}$ (see Figure 16.3d).

Limitations

As Beaudouin-Lafon (2004) mentioned, a good theoretical model needs to "*strike a balance between generality (for descriptive power), concreteness (for evaluative power) and openness (for generative power)*", which are contradicting goals. Our 'Knowledge-assisted VA Model' is a rather high-level model focusing slightly more onto generality. The new 'Knowledge-assisted VA Model' represents a high-level system blueprint, which can be used for a generalized system description from the viewpoint of the components to be used, the included processes and their connections. In this respect, the model does not provide the system architecture in the level of detail which is required directly for the implementation (e.g., design patterns, algorithms, data structures). Since this model is concentrated on high-level systems architecture, a further limitation is the possible depth of describing the user cognitive processes, perception and implicit knowledge generation. Therefore, other established models like the 'Knowledge Generation Model for VA' by Sacha et al. (2014) might be used. Moreover, the new 'Knowledge-assisted VA Model' does not distinguish directly between single and multi user systems as well as the way how the explicit knowledge and the analysis data are collected, prepared, stored or made available – since its a high-level systems blueprint. The model also provides an approach to calculate the costs and profit of knowledge-assisted VA systems, but its not providing any procedure to measure and quantify the quality of the integrated explicit knowledge. Similar to the model by Van Wijk (2005) the input data did not change over time, they appear as a static entity throughout exploration and visualization. Based on this assumption, the new model does not describe dynamic datasets or data sources directly (e.g., different types of (time-oriented) streaming data). Based on this assumption, the model can be expanded by the integration of dynamic datasets or data sources $D(t)$ (e.g., different types of (time-oriented) streaming data) in the future. To do so, an adaption of the mathematical formalism, including the aspect of time

into the input data (and the other related elements) is needed. Based on this, it might be possible to describe real time data stream analysis and present its important insights.

Responding the Research Questions

According on the insights gained during this doctoral theses, the initially posed research questions (see Section 1.4) can be answered in detail. To answer these questions, two design study prototypes were developed, tested and evaluated: 1) KAMAS, see Part II; and 2) KAVAGait, Part III. Based on the findings, the 'Knowledge-assisted VA Model' (see Part I) was developed, which can be used to describe, evaluate and generate new visualization systems. These visualization systems can be characterized along 4 system types: 1) VIS; 2) KAV; 3) VA; and 4) KAVA, depending on their included components and processes which is demonstrated in Section 15.1).

Main Question: *How can the visual analytics process benefit from explicit knowledge of analysts / domain experts?* During the two performed design studies: 1) KAMAS, see Part II; and 2) KAVAGait, Part III, we found that explicit knowledge opens the possibility to support the analysts while performing their analysis tasks to gain new insights from the data. By extracting the implicit knowledge of domain experts, it can be made available as explicit knowledge system internally. This explicit knowledge supports the analyst during the interactive data exploration process. Based on this, the analyst gets the possibility to find new patterns and to gain new implicit knowledge (insights) about the data. These insights can be extracted and stored internally in the system as new explicit knowledge which is than shared and also available for further analysis. Thereby, the user gets the ability to use the explicit knowledge generated by himself, by others and by automated analysis methods for his/her exploratory purposes to achieve his/her goals. Moreover, depending on the explicit knowledge stored in a visualization system, analysts get the opportunity to learn from others and to improve themselves. In relation to VA, the stored explicit knowledge can also be used to specify (parametrize) the automated data analysis methods supporting the analyst during data exploration and analysis processes. Based on the statements of the participants in our user and case studies, the integration of explicit knowledge can not only improve the data analysis, it also accelerates it. Thus, by summarizing our findings based on the new 'Knowledge-assisted VA Model', the visual analytics process benefits on several ways:

1. By including explicit knowledge into the automated data analysis process $\boxed{K^e} \rightarrow \textcircled{A}$, the loaded datasets can be analyzed automatically in relation to the actual analyst's needs.

2. The system's specification $\boxed{K^e} \rightarrow \textcircled{A} \rightarrow \boxed{S}$ can be adapted by the explicit knowledge to guide the analyst gaining new insights.
3. By using the explicit knowledge included by the analyst himself or by others, the analyst gets the ability to faster gain new implicit knowledge which can be extracted and stored as new explicit knowledge $\boxed{K^e} \rightarrow \ldots \rightarrow \textcircled{V} \xrightarrow{I} \textcircled{P} \rightarrow \boxed{K^i} \rightarrow \textcircled{X} \rightarrow \boxed{K^e}$.

Sub Question 1: *How can explicit knowledge be visually represented effectively in a visual analytics system?* As demonstrated in Sections 7.3 and 12.3 explicit knowledge can be generally represented in two different forms:

1. Using explicit knowledge to enhance the visualization of data.
2. Representing the explicit knowledge itself as data.

By the usage of (1) explicit knowledge as data visualization enhancement, a) one or more further visual variables (e.g., (Carpendale, 2003; Mackinlay, 1986; Munzner, 2014, pp. 94)) can be added for a directly representation of the explicit knowledge in the visualization (e.g., colored highlighting). For example, in KAMAS , we highlighted patterns based on different color gradations in relation to the used explicit knowledge for direct comparisons of value or string sequences (see Sections 7.3 represented in the graphical summary). Additionally, b) explicit knowledge can also be used for the parameterization of the visualization (e.g., automated zooming to areas of interest or selection of the appropriated visualization granularity). For example: in KAVAGait the explicit knowledge contains a value range, it is possible to display if the compared value is in or out of range (see Sections 7.3) in relation to the represented graphical summary. Generally, both representations are supporting the analyst during the decision making process. From a bird's perspective, explicit knowledge can be seen as one or more data variables combined with the loaded dataset to support the analysts while data exploration and analysis. Thus, 2) the representation explicit knowledge is also very important for the analysts. Thereby, they get the ability to explore and analyze the contained explicit knowledge directly as well as to adjust it for example. This was also noted as very helpful during both performed user studies (see Sections 8 and 13). It is important to note, that the level of detail for the representation of explicit knowledge depends on its internal data structure and complexity.

Sub Question 2: *Is it possible to generalize the interaction with knowledge-assisted visualization methods for different application scenarios?* For the demonstration of the generalizability, the novel 'Knowledge-assisted VA Model' conceptually

grounded on the 'Simple Visualization Model' by Van Wijk (2005) was developed (see Part I). This model can be used to describe and categorize systems (see Chapter 15), to evaluate them as well as to be used as a high-level blueprint for the generation of new visualization systems (see Section 16.1). By using the model as high-level blueprint, needed system components and processes can be described together with interactions in a high-level form. As has been shown in both performed validations (see Chapters 8 and 13), the same interactions can be applied to knowledge-assisted visualization methods (KAV and KAVA systems) as to application scenarios without knowledge integration (VIS and VA systems). A knowledge-assisted visualization system has to provide further interactions to extract the users implicit knowledge and to integrate it as explicit knowledge into the system. Additionally, the possibilities to visualize, explore and adapt the internally stored explicit knowledge should be made available. Generally, explicit knowledge should be integrated into the system in such a way that the user can easily recognize, use and interact with it.

Sub Question 3: *How can analysts during the exploration of a large amount of data benefit from knowledge-assisted visual analytics methods?* As found in the validations (see Chapters 8 and 13) of the two implemented prototypes: 1) KAMAS (see Part II); and 2) KAVAGait (see Part III), explicit knowledge supports the analysts by gaining new insights of the visualized data (e.g., finding unknown patterns, verify or falsify hypothesis). On the one hand, it supports the analysts to recognize known patterns faster in the loaded data. On the other hand, it helps the analyst to concentrate on the interactive exploration and analysis of new problems to gain new insights and find new patterns more quickly. By combining the explicit knowledge with automated data analysis methods, the data can be filtered automatically to provide specific pre-analyzed visualizations in relation to specific tasks which have to be explored. It can also be used to provide analysis suggestions to the analysts to guide them through the analysis process. A further benefit of knowledge-assisted visualization systems lies in the ability to learn and get trained by the extracted knowledge from others implicit knowledge or by explicit knowledge which is generated by automated data analysis methods.

Summary

As discussed above, the novel 'Knowledge-assisted VA Model' can be used to describe, categorize, evaluate and generate new visualization systems with and without the integration of explicit knowledge. Since the model is a high-level systems blueprint, it can be used to describe existing systems in relation to the contained components and processes and their relations as well as to categorize them along the six different visualization system types: 1) VIS; 2) KAV; 3) VA and 4) KAVA, as well as systems

without a visual representation: 1) A; 2) KAA. This depends on the generation of implicit knowledge, its extraction as explicit knowledge and the usage of automated data analysis methods. Based on these criteria, it's also possible to use this novel model as high-level blueprint for systems design. The discussed cost and profit functions are supporting the designers to quantify the use and efficiency of the system from a birds eye perspective.

The two performed design studies: 1) KAMAS, see Part II; and 2) KAVAGait, Part III, demonstrated the benefits of knowledge-assisted VA systems. The extraction and integration of implicit expert knowledge into the system as explicit knowledge supports the analysts by gaining new insights (e.g., finding new patterns, hypothesis verification, solving tasks) and learning from the explicit knowledge which was integrated by others. Additionally, by combining automated data analysis methods with explicit knowledge, the data can be analyzed automatically based on the gained insights of domain experts. By providing the ability to integrate, explore and adjust explicit knowledge during the systems runtime, all insights gained by the analysts can be included directly into the automated data analysis and visualization process. This way, new found patterns are highlighted automatically or matching criteria are recalculated in realtime. Generally, by providing knowledge-oriented interaction methods, the externalized knowledge can subsequently be used in the iterative exploration and analysis process to improve the analysts performance and helps to improve the data analysis.

16.2 Future Directions

In this section, we are proposing future directions which provide particular guidance for visual analytics professionals. The findings are extracted from the elaboration of our novel theoretical and mathematical model to describe knowledge-assisted visualization (see Part I), the two performed case studies (KAMAS, see Part II; and KAVAGait, see Part III), and the model's application and discussion (see Part IV).

Supporting Different Data Sources and Streaming Data

Data analysis is based on a wide range of base data collected by different data providers and different analysis methods including static data sources as well as dynamic data sources. As data analysis is getting more and more complex, there is a need to combine different data sources to solve specific domain problems. This involves not only supporting different data formats but also handling the resulting heterogeneous data in a suitable way. The novel 'Knowledge-assisted VA Model' considers the input data as a static entity throughout exploration and visualization. During the data exploration, it is not considered that the input dataset can be changed or adapted over time, because the model is currently designed to support one or more static data sources. Based on

this assumption, the model should be expanded by the integration of dynamic datasets or data sources $D(t)$ (e.g., different types of (time-oriented) streaming data). For example, analyzing one or more data streams automatically in real time and presenting important events or findings that can be explored and analyzed in parallel. Moreover, the combination of static and dynamic data sources should be considered.

Involve Expert's Knowledge through Interaction

As confirmed in the performed user studies (see Chapters 8 and 13), interactive data exploration and analysis can be improved by the integration of explicit knowledge. In the future, it might be interesting to provide explicit knowledge not only for the data to be explored, it could also be interesting to generate and provide explicit knowledge based on the performed interactions of the users in order to improve the performed data analysis. Examples could be: How to parameterize the filters for certain problems? Which area of the visualized data are interesting? Which interactions have to be performed to make certain patterns visible in the data? Moreover, the explicit separation between the knowledge in relation to the system's usage and knowledge with regard to the data could be equally interesting.

Intertwine Knowledge with Automated Data Analysis Methods

In the knowledge discovery process (KDD) the central role of prior knowledge is well known (Fayyad et al., 1996). Applications of artificial intelligence (AI) techniques in data analysis are aiming at automatic information extraction by using explicit domain knowledge (Hand, 1998) and knowledge-based systems are enabling the integration of explicit knowledge into the reasoning process (Perner, 2006). Thus, approaches using knowledge-based data analysis have obvious advantages in contrast to other systems (Zupan et al., 2006). To support analysts during interactive data exploration, analysis and decision making, it will be necessary to combine automated data analysis methods with explicit knowledge which is extracted from the human user's implicit knowledge. During the iterative data exploration and analysis process, new implicit knowledge is gained by the human which in turn can be fed into the system as new explicit knowledge iteratively to influence and improve the automated data analysis methods.

Prevent Misinterpretations by Inaccurate Explicit Knowledge

Applying a knowledge database containing explicit knowledge to a visualization or analysis system, always goes with the desire to make the system better and more accurate. Thus it has to support the analysts to improve sense-making of the visualized data and finding unknown pattern. This also creates the problem that, when we assume that

new explicit knowledge always causes an improvement. Misleading or negative knowledge can arise in many ways. On the one hand, it can be generated by inexperienced or untrained users, and on the other hand, it can also be caused by incorrectly edited datasets or incorrectly parameterized automatic data analysis methods. To overcome such problems on the computational side, predefined, tested and documented datasets in combination with explicit knowledge (a related knowledge database) has to be provided to test if a system works correctly. To overcome the issues on the humans' side, validation methods for explicit knowledge have to be established.

Validation of Extracted and Stored Explicit Knowledge

Generally, it is possible for every system user to extract and generate explicit knowledge. On the one hand, it is not distinguished between experts and novices and, on the other hand, there is no distinction between the specializations of experts (e.g., in IT-security, experts can be specialized on different malware families; in clinical gait analysis, clinicians can be specialized on different gait abnormalities). Wang et al. (2009) described three potential issues which can occur by extracting new explicit knowledge into a knowledge database: 1) 'Duplicated Knowledge', can occur when experts examine the same data; 2) 'Partially Overlapped Knowledge', mostly similar rules with subtle but important differences; and 3) 'Conflicted Knowledge', it occurs when existing knowledge differs to new knowledge based on new policies (Wang et al., 2009). This shows that it is not only necessary to provide the ability of extracting implicit expert knowledge. Quite the opposite, it shows that mechanisms has to be created to avoid such issues. For example, by checking new extracted knowledge against the system internally stored explicit knowledge with automated analysis methods. Thereby, a kind of a four-eyes principle can be applied to permit new entries or to provide a kind of voting hierarchy, which allows analysts not only to add new knowledge, also to agree or disagree to the existing one.

Novel Evaluation Methods for Implicit and Explicit Knowledge

For the measurement of the assistance and expressiveness of implicit and explicit knowledge it is imperative to investigate and design novel evaluation methods. Such methods, should offer the possibility to measure how much implicit knowledge is generated on the basis of the stored explicit knowledge. Similarly, methods should be developed to measure how much explicit knowledge was extracted from the users implicit knowledge. Moreover, it is necessary to elaborate methods to measure the knowledge flows between the user and the system as well as to assess how effectively an approach supports reasoning and data analysis. For example, the 'Nested Workflow Model' by Federico et al. (2016) points in this direction.

Supporting Collaboration and Multi User Systems

When analysts are solving real world problems, normally they have a vast amount of complex and heterogeneous data at their disposal. The solving of such complex exploration and analysis tasks leads to the assumption that the development, implementation and evaluation of collaborative knowledge-assisted visualization systems becomes more and more important (Blumenstein et al., 2015a; Von Landesberger et al., 2011). Such systems will support the users by sharing their implicit knowledge by extracting it as explicit knowledge into a knowledge database. Additionally, it supports the users while collaboration during the exploration and analysis of datasets finding the best solution to solve their problems (Munzner, 2014). To create such systems, some important points have to be considered(Blumenstein et al., 2015a; Wagner et al., 2016a): How can the vast amount of data be synchronized in an appropriate way if everyone works on their own device? How to create a history of the performed interactions and interim results created by different users? How to represent explicit knowledge in such a system? Should all users have the same permissions or does a collaborative system need a session master?

16.3 Contributions

In general, the contributions presented in this doctoral thesis can be split into 3 main elements: the new 'Knowledge-assisted VA Model' and the two specific knowledge-assisted VA methods (KAMAS and KAVAGait) used for demonstrating the model's application.

Knowledge-assisted VA Model: The main contributions of this doctoral thesis are based on the generalization of the knowledge-assisted VA process. Therefore, we provide a detailed description of the theoretical generalization resulted in the new 'Knowledge-assisted VA Model'. It can be found in Part I and the application of the new model is demonstrated in detail in Part IV. Thus, the main contributions are:

- We provided a mathematical abstraction and theoretical modeling of the VA process based on the introduction of the new 'Knowledge-assisted VA Model'.
- We illustrated the possibilities of explicit knowledge integration and extraction, the integration of automated data analysis methods as well as the combination of both. This supports the data exploration, analysis and implicit knowledge gaining as well as the knowledge extraction and its sharing with other users.
- We demonstrated the utility of the model by showing its: 1) descriptive power, to describe the functionalities of existing approaches and to categorize them in

relation of the included components, processes and their connections (e.g., on KAMAS and KAVAGait); 2) generative power, to inspire and enable design of innovative approaches using the new model as a high-level system blueprint; and 3) evaluative power, to express the costs and benefits of knowledge-assisted processes and systems.

KAMAS: The first subset of contributions is based on the design study of our new 'Knowledge-assisted Malware Analysis System' for behavior-based malware analysis, which is illustrated in detail in Part II:

- We provided a detailed problem characterization and abstraction to establish a common understanding between domain experts and visualization researchers. Thereby, we are describing the data to be visualized, the future system users and their tasks to be solved in detail.

- We presented a detailed literature review with regard to visualization systems for malware analysis in combination introducing the 'Malware Visualization Taxonomy' for system categorization. In general, the taxonomy divides the categorization of malware visualization systems into three categories: 1) Individual Malware Analysis; 2) Malware Comparison; and 3) Malware Summarization.

- We presented the concept and the implementation of KAMAS as systematically designed, developed and evaluated instantiation of an knowledge-assisted VA solution for the handling of a vast amounts of data in behavior-based malware analysis. The calls have a nominal value and can be described as time-oriented data on an ordinal time scale with instances as time primitives.

- We show that applying knowledge-assisted VA methods allows domain experts to externalize their implicit knowledge and profit from this explicit knowledge during their analysis workflow.

- For the visualization of the explicit knowledge we provided a three-step color map for knowledge highlighting in combination with a graphical summary helping the analyst while comparing different rules. Additionally, for the exploration of the explicit knowledge, we provided an intended list.

- To provide evidence for the effectiveness of the developed methods, we provide a rigorous and reproducible validation of the introduced techniques with malware analysis experts.

KAVAGait: The second subset of contributions was achieved by a 'Knowledge-assisted VA System for Clinical Gait Analysis' which are expressed in detail in Part III:

- During the process of the problem characterization and abstraction, a common language and understanding between domain experts (clinicians) and visualization researchers was established. Additionally, we summarized the background of clinical gait analysis and abstract specifics of data, users, and tasks to be considered in VA design.

- We presented the concept and the implementation of KAVAGait as systematically designed, developed and evaluated instantiation of an knowledge-assisted VA solution for the handling of a vast amounts of data for clinical gait analysis. The systems input data can be described as time-oriented data consisting of quantitative values on an ordinal time scale with a single granularity in milliseconds.

- New knowledge-assisted visualization approaches were used to generate easily understandable 'Graphical Summaries' of the data to gain a fast overview of the parameter matchings. Additionally, the novel 'Interactive Twin-Box-Plots' (ITBP) were developed providing parameter based intercategory comparisons.

- We validated the visualization design of the new implemented knowledge-assisted VA prototype for clinical gait analysis based on expert reviews, user studies and a case study. This way, we test if the used visual data representations are effective to support the domain experts while solving their analysis tasks.

16.4 Conclusion

The main goal of this doctoral theses was the integration of explicit knowledge into the VA process. To fulfill this overall goal, we generated a high level 'Knowledge-assisted VA Model' to describe the process of knowledge-assisted VA, conceptually grounded on the 'Simple Visualization Model' by Van Wijk (2005) (see Part I). The new model contains all components and processes as well as the needed connections, to be included in a knowledge-assisted VA system: 1) implicit knowledge extraction; 2) automated data analysis methods; 3) explicit knowledge based specification 4) explicit knowledge visualization and 5) implicit knowledge generation. To demonstrate the applicability of the new model, we used a problem-driven research approach to study knowledge-assisted VA systems for time-oriented data in the context of two real world problems (KAMAS in Part II and KAVAGait in Part III).

The first case study relates to the domain of IT-security by implementing and evaluating KAMAS, a 'Knowledge-assisted VA System for Behavior-based Malware Analysis'. Therefore, we started with a problem characterization and abstraction (see Chapter 6), grounded on a threefold research approach (literature research, focus group meetings and semi-structured interviews) to find out which data are used, who are the users and which tasks have to be fulfilled by behavior-based malware analysis and how these

can be covered with KAMAS. Based on this abstraction, we found out that the data are nominal values on an ordinal time scale with instants as time primitives. These instances are building the parse tree of a cluster grammar as a simple directed acyclic graph with nominal data attributed to the nodes. The users are IT-security experts and the main tasks can be summarized as: 1) selecting different rules; 2) categorizing them by their task and storing them in the database as well as 3) manual adaption and/or tuning of rules. During the semi-structured interviews, we elaborated problem appropriated and known interactive data visualization techniques for this domain. Grounded on these insights, the KAMAS prototype was implemented (see Chapter 7) consisting of a knowledge database (KDB) for storing new rules, supporting the analysts during interactive data analysis and exploration as well as including newly gained knowledge and providing it for others. The arc-diagram stated as very interesting during the problem characterization did not provide the assumed insights in the evaluation phase (see Chapter 8). In contrast, the simple looking connection line was found to be particularly useful as it was leading the analyst from the overview table (containing the rules) to the detail table. The implemented graphical summary, colored in relation to the knowledge contained in the KDB, was very helpful for the analysts. In general, all participants evaluated KAMAS as a suitable and innovative solution for behavior-based malware analysis.

Under consideration of the insights grounded on the first case study, we performed a second one related to the domain of clinical gait analysis. Therefore, we implemented KAVAGait, a 'Knowledge-assisted VA System for Clinical Gait Analysis'. Again, we started with a problem characterization and abstraction (see Chapter 11), based on focus group meetings set in context with domain-specific literature, to find out which data needs to be visualized, who is using it and which tasks need to be supported. The input data are discrete quantitative values with an ordinal timescale and a single granularity (ms). The users of the system are domain experts (physicians, physical therapists, biomedical engineers or movement scientists), which are also called clinicians. The main tasks of a clinician in gait rehabilitation are: 1) assessing gait performance; 2) analyzing and interpreting the acquired data and 3) using this information for clinical decision making. During the interviews, we elaborated that clinicians mostly are using line charts for data visualization. Moreover, based on their statistical knowledge, they are familiar with box plots. According to these insights, the KAVAGait prototype was implemented (see Chapter 12). To do so, the KDB used for KAMAS was extended and renamed to the explicit knowledge store (EKS) to store 16 spatio-temporal parameters (STP) per patient. In relation to these STPs, the graphical summary was extended to provide an overview, if a new loaded patient fits to a category of the EKS or not. Supporting clinical decision-making, automatically calculated matching criteria were added to the visualization. For a detailed overview and comparison of the calculated STPs of a newly loaded patient to the EKS, we created the twin box plot visualization. This technique

allows the comparison of the newly loaded patient STPs to the normal gait category and to a selected category of a gait abnormality from the EKS. During the evaluation of KAVAGait (see Chapter 13), we found out that the system was generally well perceived and valued. The participants stated that they now could compare several patients and include them based on their findings in an EKS, which helps to gain new insights of the data. Additionally, they mentioned that the system is also supportive to share their insights and learn from others.

In addition to the two performed case studies, the models applicability (see Part IV) was also elaborated on two examples of the literature (SEEM (Gove et al., 2014) and Gnaeus (Federico et al., 2015)) to demonstrate its descriptive power. Additionally, it was also exposed that it is possible to describe existing models of the visualization community on the 'Knowledge Generation Model for VA' by Sacha et al. (2014). As indicated in this work, the new 'Knowledge-assisted VA Model' can be used to describe, evaluate, and generate novel and innovative interactive data visualization systems with and without the integration of explicit knowledge. Since the model is a high-level system's blueprint, it can be used to describe existing systems in relation to the contained components and processes and their relations and to categorize them along the four different visualization system types: VIS, KAV, VA and KAVA, as well as two systems without a visual representation: A, KAA. It is also possible, to use the model as high-level blueprint for systems design. In this sense, the designer gets supported on how to design the generation of implicit knowledge, its extraction as explicit knowledge, and the usage of automated data analysis methods in combination with interactive data visualization. This leads to a possible future direction, how the integrated explicit knowledge could be used interactively to enhance the quality of automated data analysis methods (e.g., (Fayyad et al., 1996; Zupan et al., 2006)). Therefore, our new model can be used as blueprint in system design. The extraction and integration of implicit knowledge into a system as explicit knowledge can support the analysts by gaining insights and learning from explicit knowledge integrated by others. By providing the ability to integrate, explore, and adjust explicit knowledge during runtime of a system, the analysts' insights can be included directly into the automated data analysis and visualization process. As confirmed in the performed user studies (see Chapters 8 and 13), interactive data exploration and analysis can be improved by the integration of explicit knowledge. In the future, explicit knowledge should not only be provided for the data to be explored, it should also be provided for the interactions to be performed during data exploration and analysis. Therefore, separated knowledge bases in relation to the system's usage and for data analysis should be integrated. The integration of a knowledge database into a visualization or analysis system, always goes with the desire to make the system better and more accurate, supporting the analysts to improve sense-making. But this can also become a problem by creating misleading or negative knowledge. To overcome such problems, predefined test cases are needed to figure out if a system works correctly,

and validation methods for explicit knowledge have to be established. Additionally, for the evaluation of such new knowledge-assisted VA systems, new evaluation methods have to be investigated. One approach in this direction is the extended and discussed cost and profit functions grounded on Van Wijk (2005). These functions are supporting the designers in quantifying the usage and efficiency of the system from a bird's-eye perspective. In the future, it is necessary to elaborate and introduce evaluation methods which consider not only explicit knowledge according to our model. Additionally, methods are needed which can also measure knowledge flows and assess how effectively an approach supports reasoning and data analysis. For example, the 'Nested Workflow Model' (Federico et al., 2016) points in this direction.

Part V

Appendix, Bibliography, Lists of Figures & Tables

APPENDIX A

Used Search Terms for the Survey on Malware Visualization Systems

Detailed Description of the used Search Terms

1. Areas of this Survey

To specify the visualization aspect of this paper, the covered and not covered areas are listed below.

- **Covered Areas**
  - malware (different devices, hosts)
  - malicious data
  - malicious code

- **Not Covered Areas**
  - intrusion detection
  - phishing / spam
  - threat landscape

2. Words of the first literature search

The following list shows the used keywords in relation to the used search engines and to the additional search parameters:

- **Google Scholar:**
  - malware, visual analytics
  - visual analytics, IT security
  - IT security, visualization
  - malware, IT security, visualization
  - visualization, ,malware
  - time-oriented data, visualization, malware
  - time-oriented, visual analytics, malware
  - malware, visual analytics, IT security
  - malware, visual analytics, VAST, security
  - malicious, visualization
  - visual analytics, malicious
  - visual analysis, malicious

- **Academic Research Microsoft:**
  - malware detection
  - malware, visual analytics
  - time-oriented data, malware
  - time-oriented data, malware, visual analytics

- **IEEE Xplore (normal search):**
  - malware, visual analytics
  - malware detection, visualization
  - malware, time-oriented
  - malware, security, time-oriented, visualization

- **IEEE Xplore (advanced search):**
  - *Document Title*: malware, *Full Text & Metadata*: visual analytics

- *Document Title*: visual analytics, *Full Text & Metadata*: malware
- *Document Title*: survey, *Full Text & Metadata*: Visualization, malware
- *Document Title*: survey, visual analytics, malware
- *Document Title*: survey, *Full Text & Metadata*: visual analytics, malware
- *Document Title*: survey, *Full Text & Metadata*: time oriented data, malware
- *Document Title*: behavior based, malware, visualization
- *Document Title*: behavior based, malware, visual analytics
- *Document Title*: visualization, *Abstract*: malicious
- *Document Title*: malicious, *Abstract*: visual analytics
- *Document Title*: malicious, *Abstract*: visual analysis
- *Document Title*: survey, *Full Text*: time-oriented data, malware
- *Document Title*: malicious, *Abstract*: visualization
- *Document Title*: visual analytics, *Abstrac*t: malicious
- *Document Title*: visual analysis, *Abstrac*t: malicious

- **ACM digital library:**
  - malware, visualization
  - malware, visual analytics
  - malware, time-oriented
  - IT security, visualization
  - IT security, visual analytics

3. Authors and Keywords for the second literature search

The names of the authors which were used for the research are shown in following list, followed by a list with the keywords which were used for the combination.

- **List of the used authors:**
  - Philipp Trinius
  - Torsten Holz
  - Jan Göbel
  - Felix C. Freiling
  - John Donahue
  - Anand Paturi
  - Srinivas Mukkamala
  - Asaf Shabtai
  - Daniel A. Quist
  - Denis Klimov
  - Josh Saxe
  - David Mentis
  - Carsten Willems
  - Chan Lee Yee
  - Lee Chuan
  - Dong Hwi Lee
  - In Soo Song

- **Used keywords for the combination with the author names:**
  - malware
  - malicious
  - visualization
  - visual analytics
  - visual analysis
  - time-oriented data

4. VizSec Conference

Additional we extended our search by the VizSec-Paper-Search: http://vizsec.dbvis.de/

- **Searched in Abstracts and Titles from 2004 till 2014**
  - malicious
  - malware
  - spam

APPENDIX B

Material for the User Study on KAMAS (in German)

KAVA-RuleTree Usability Study

Allgemeines

Die Systemtests werden voraussichtlich im Zeitraum zwischen 04.10.2015 und 20.10.2015 durchgeführt. Es werden voraussichtlich 5-6 ProbandInnen getestet wobei diese alle aus der IT-Security Domäne kommen. Hierbei streben wir an, 3 ProbandInnen aus der Forschung und 2-3 ProbandInnen aus der Wirtschaft zu haben.

Methode

Um eine gute und aufschlussreiche Evaluierung des KAVA-Time RuleTree Prototypen zu garantieren, werden mehrere verschiedene Testansätze zeitgleich bzw. hintereinander durchgeführt.

- Erhebung der persönlichen Daten
- Usertest (Aufgabenbasiert) + Logging + Video
- System Usability Scale (SUS)
- Semi-strukturiertes Interview

Basierend auf den Erkenntnissen dieser Tests werden verschiedene Auswertungen durchgeführt und dokumentiert.

Vorgehensweise

JedeR ProbandIn wird auf einem Handelsüblichen PC den Test durchführen Der Test dauert ca. eine Stunde und wird von einem Testleiter begleitet. Neben den Aufzeichnungen die durch den Testleiter erstellt werden, wird während dem Systemtest auch noch ein Logfile mitgeschrieben und auch ein Bildschirmvideo erstellt (gegebenenfalls kann auch der Proband gefilmt werden um die Gesichtsausdrücke während der einzelnen Aufgaben festzuhalten).

- Testdauer: ca. 1 Stunde
- Anwesende Personen: TestleiterIn und ProbandIn
- System: Handelsüblicher PC mit 22“ oder 24“ Monitor

Forschungsfrage

1. Profitiert der Analyst von extern gespeicherten Wissen bei der Analyse von Behavior Based Malware Analysedaten?
2. Ist die Art der Wissensspeicherung verständlich und nachvollziehbar?
3. Ist die Visualisierung des externen Wissens verständlich für den Probanden / die Probandin?

Ziele

- Testen des Forschungsprototypen auf seine Funktionalität.
- Testen der Visualisierungstechniken auf Verständlichkeit in Bezug auf die Domäne.
- Testen der Effektivität der Wissensspeicherung und Repräsentation im System.

Nicht Ziele

- Vergleich des Prototypen mit einem anderen Analysesystem.
- Performancetests

Fachhochschule St. Pölten GmbH, Matthias Corvinus-Straße 15, 3100 St. Pölten

_____

_____

_____

Im Folgenden <Befragte/r> genannt

Zustimmungserklärung Interview Projekt KAVA-Time

Im Rahmen des FWF Projektes KAVA-Time werden Interviews und Softwaretests in der Zeit vom 04.10.2015 bis 30.10.2015 durch MitarbeiterInnen der FH St. Pölten durchgeführt und aufgezeichnet (Video-, Tonaufnahme und Notizen).

Der/die obengenannte Befragte stimmt hiermit ausdrücklich zu, dass das mit ihm/ihr geführte Interview und die Softwaretests ausgewertet und FH-intern für Lehre und Forschung verwendet und in anonymisierter Form auch zB für wissenschaftliche Publikationen veröffentlicht werden dürfen.

St. Pölten, am _____

_____

Unterschrift

KAVA-RuleTree Usability Study: Road Map

Willkommen (ca. 5 min)

Hallo, mein Name ist <NAME> und ich werde Sie heute durch den etwa ein stündigen Test unseres Systems begleiten. Vor dem Beginn des Tests werde ich Ihnen nähere Informationen zum Testablauf vorlesen. Diese Maßnahme ist insofern notwendig, da ich sicherstellen möchte, dass alle Testpersonen die gleichen Informationen erhalten.

Bei diesem Test wird ein interaktives Tool zur Exploration von System- und API-Calls sowie zum Analysieren von Call-Sequenzen (werden auch Regeln genannt) auf seine Nutzbarkeit getestet. Wichtig ist an dieser Stelle anzumerken dass nicht Sie als Person getestet werden sondern das System. Ich möchte Sie bitten ehrliche Aussagen zu dem System zu tätigen, sei es positiv oder negativ, beides ist für uns sehr hilfreich, um die Entwicklung des Systems bestmöglich fortzusetzen.

Zu Beginn werde ich Ihnen ein paar Fragen zu Ihrer Person, Ausbildung, beruflichen Werdegang und der Erfahrung in Bezug auf Malware Analyse stellen. Anschließend werde ich Ihnen 5 Aufgaben vorlesen und Sie werden versuchen diese zu lösen. Lassen Sie sich bei den Aufgaben so viel Zeit wie sie benötigen und sprechen Sie ihre Gedanken einfach laut aus.

Ich werde Sie während des Tests beobachten und dazu Notizen machen. Während der Testphase unseres Systems werden die Interaktionen aufgezeichnet (protokolliert), so dass wir diese im Nachhinein genau analysieren können. Ebenso wird von diesem Test ein Video aufgenommen. Dieses Video hilft uns bei der Testanalyse und der Verbesserung der Applikation. Alle Aufzeichnungen dieses Tests werden vertraulich für Forschungszwecke in diesem Projekt behandelt und nicht an außenstehende Personen weitergegeben. Diesbezüglich möchte ich Sie bitten unsere Einverständniserklärung zu unterzeichnen.

Einverständniserklärung von Probandin oder Probanden unterzeichnen lassen!

Falls während des Tests Fragen auftreten, lassen Sie es mich bitte sofort wissen. Um die Testanalyse zu vereinfachen, möchte ich Sie bitten, bei den Aufgaben Ihre Gedankenweg und jeden Schritt den Sie unternehmen werden, laut auszusprechen (Bspw. Ich klicke jetzt auf <XY> um eine Sortierung der Daten nach <YZ> vorzunehmen …).

Gibt es noch Fragen?

<Starten der Kamera für den Test>

Ich werde nun die Kamera für die Aufzeichnung des Tests starten.

Personenbezogene Fragen

1. Als erstes beginnen wir mit den zuvor besprochenen Personenbezogenen Fragen:

Name: _____

Geschlecht: _____

Alter: _____

Ausbildung: _____

Beruf: _____

Tätigkeit: _____

2. Wie viele Jahre Berufserfahrung haben Sie in der IT?

◯ 0 – 4 ◯ 5 – 9 ◯ 10 – 14

◯ 15 – 19 ◯ 20 – 24 ◯ 25 – 29

◯ 30 – 34 ◯ 35 – 39 ◯ 40 - 44

3. Haben Sie Erfahrung in der verhaltensbasierten Malwareanalyse?

◯ Ja

◯ Nein

4. Wie würden Sie sich selbst in Bezug auf Ihre / deine Fertigkeiten in diesem Feld einstufen?

◯ Anfänger ◯ Fortgeschritten ◯ Erfahren ◯ Experte

5. Mit welchen Interfaces arbeiten Sie bei der Malwareanalyse (mehrere Möglichkeiten)

◯ Konsolenprogramme (z.B.: Shell, CMD)

◯ Texteditoren (z.B.: VI, VIM, NANO, Notepad++)

◯ Interaktive Interfaces (z.B.: Interfaces die die Analysedaten als Text darstellen und weitere Optionen zur Analyse zur Verfügung stellen.)

◯ Grafisch aufbereitete Interfaces (z.B.: Interfaces die auch grafisch aufbereitete Statistiken oder ähnliches in Bezug auf die Analysedaten zur Verfügung stellen.)

◯ Grafisch aufbereitete interaktive Interfaces (z.B.: Interfaces die eine Interaktion mit den grafisch aufbereiteten Daten zulassen um weitere Einsichten über die Analysedaten zu gewinnen)

6. Können Sie mir ein paar Programme nennen?

Besten Dank für die Informationen. Beginnen wir nun mit den Aufgaben in unserem zu testenden System. Bitte sprechen Sie alle Gedankengänge laut aus während der Lösung der Aufgaben.

Test der Software (ca. 30 min)

Kennenlernen des RuleTree-Systems zur interaktiven Exploration von behavior based Malware Analysedaten.

Vor ihnen sehen Sie nun das Interface des Analysesystems. Bitte sehen Sie sich das Interface einmal an um einen ersten Eindruck zu gewinnen.

Was ist der erste Eindruck?

Danke für die erste Einschätzung des Systems. Sehen wir uns nun das System einmal in Detail bezüglich der gebotenen Optionen und Funktionalitäten an.

Welche Funktionalitäten können Sie nach ihrem ersten Eindruck ableiten?

<!!! NICHT SAGEN!!!: Auf der rechten Seite sehen Sie eine Tabelle mit 3 Spalten in der die einzelnen System- und API-Calls dargestellt werden die im Analyse File vorkommen. Darunterliegend sind einige verschiedene Filter zu sehen.>

Aufgabe 1:

Bitte versuchen Sie alle Calls die zwischen 5 und 30 mal in dem Analysefile vorkommen und in denen der Wortlaut „file“ vorkommt zu filtern. Wie viele Einträge haben Sie gefunden?

◯ 9 Stk. (richtig gelöst) ◯ Falsch Antwort Anzahl Versuche: _____

Welche 3 der 9 gefundenen Calls kommen am häufigsten vor?

◯ SetFilePointer ◯ NtQueryInformationFile ◯ NtSetInformationFile

Welche Features haben Sie benutzt um die 3 häufigsten Calls zu finden?

◯ Balken ◯ Sortieren der Spalte ◯ Nummern

Welcher Call kommt am Häufigsten vor und wie oft?

◯ NtSetInformationFile ◯ 23 mal

Woran haben Sie das erkannt?

◯ Balken ◯ Sortieren der Spalte ◯ Nummern

Bitte versuchen Sie nun alle Calls zu finden die zwischen 5 und 30 in dem Analysefile vorkommen und in denen der Wortlaut „Open“ vorkommt. Wir wollen nur die großgeschriebenen finden.

Wie viele Calls haben sie gefunden?

◯ 1 Stk. (richtig gelöst) ◯ Falsche Antwort Anzahl versuche:_____

Haben Sie Auswirkungen der Selektionen auf der rechten Seite in der Mitte des Bildschirms bemerkt?

◯ Ja ◯ Nein

Warum?

Aufgabe 2:

Nachdem Sie nun alle Regeln in der Mitte des Bildschirmes gefiltert haben die den zuvor gesuchten Call „NtOpenFile“ beinhalten (rechter Bildschirm). Werden wir nun in diesem Bereich weitere Filteroptionen benutzen.

Wie viele Regeln werden derzeit in der Mitte des Bildschirmes (Regelvisualisierung) angezeigt?

◯ 9 Stk. (richtig gelöst) ◯ Falsche Antwort Anzahl versuche:_____

Was können Sie in der Tabelle alles Ablesen?

◯ Occurrence ◯ Verteilung ◯ Regel ◯ Länge der Regel

Bitte filtern Sie nun nach Regeln die eine Auftrittshäufigkeit zwischen 30 und 90 haben und eine Länge zwischen 3 und 5. Wie viele Regeln sind nun sichtbar?

◯ 5 Stk. (richtig gelöst) ◯ Falsche Antwort Anzahl versuche:_____

Wie viele Samples sind in diesem grafisch dargestellten Analysefile zusammengepackt?

◯ 16 Stk. (richtig gelöst) ◯ Falsche Antwort Anzahl versuche:_____

Welche weiteren Optionen können Sie unter der Sample Anzahl erkennen.

◯ multiples only ➔ nur Regeln anzeigen deren Anzahl ein Vielfaches der Sampleanzahl ist.

◯ equal only ➔ nur Regeln anzeigen deren Calls gleichmäßig auf alle Samples verteilt sind.

Anmerkungen:

Aufgabe 3:

<RESET DER FILTER> Bitte filtern Sie nun nach allen Calls die zwischen 1 und 50 mal vorkommen. Jeder der Calls soll den Wortlaut „file“ enthalten.

◯ Sofort geschafft ◯ herumprobiert und geschafft ◯ nicht geschafft

Nachdem Sie das nun erledigt haben, möchten Sie nun auch nur die Regeln betrachten deren Auftrittshäufigkeit ein Vielfaches der Sampleanzahl ist.

◯ multiples only sofort angekreuzt ◯ herumprobiert und geschafft ◯ nicht geschafft

Im nächsten Schritt wollen Sie nur die Regeln sehen deren auftreten gleich in allen Samples verteilt ist.

◯ equal only sofort angekreuzt ◯ herumprobiert und geschafft ◯ nicht geschafft

Wie würden Sie die Grafik im Bereich der Spalte „Rule“ interpretieren?

◯ Zusammenfassung des Textes ◯ Eine Art Fingerprint ◯ Sonstiges

Anmerkungen zu Sonstiges:

Klicken Sie nun eine der gefilterten Regeln an. Welche Informationen können Sie jetzt ablesen?

◯ Occurrence ◯ Verteilung ◯ Rule Image ◯ Length ◯ ID

◯ Die einzelnen enthaltenen Calls in einer eigenen Tabelle.

Welche Bedeutung können Sie aus dieser Verbindungslinie zwischen den beiden Tabellen ablesen?

Sonstiges:

Hätten Sie eine andere Form der Darstellung erwartet?

◯ ja ◯ nein

Sonstiges:

Aufgabe 4:

Bitte versuchen Sie nun eine Regel in die Wissensdatenbank einzufügen ***(<Hinweis: Drag & Drop>).***

◯ Sofort geschafft ◯ herumprobiert und geschafft ◯ nicht geschafft

War es für Sie klar, dass Sie gerade erfolgreich ein Element in die Wissensdatenbank eingefügt haben?

◯ ja ◯ nein

Anmerkung:

Nachdem Sie nun eine oder mehrere Regeln zu der Struktur hinzugefügt haben, ist ihnen da etwas in der Visualisierung aufgefallen?

◯ Regeln werden in unterschiedlichen Farben angezeigt ◯ nichts aufgefallen

Aufgabe 5:

<SYSTEMNEUSTART> Versuchen Sie bitte nach allen Calls zu suchen die mit „Nt“ beginnen. Nun grenzen Sie bitte noch alle Regeln aus die nicht ein Vielfaches der Sampleanzahl sind und über keinerlei Gleichverteilung der Calls verfügen. Im nächsten Schritt wollen Sie nur noch alle Regeln sehen mit einer Auftrittshäufigkeit von max 64.

Wie viele bekannte und teilweise bekannte Regeln finden Sie hier?

◯ Bekannte: <_____> ◯ Teilweise bekannte <_____>

Ist das Farbschema für Sie passend oder haben Sie etwas anderes erwartet?

◯ ja ◯ nein

Anmerkung:

Filtern Sie nun noch nach Regeln die mindestens 10 Calls enthalten müssen. Wählen Sie eine dieser Regeln aus und versuchen Sie in dieser Regel ein Muster zu erkennen. Sie können auch gerne verschiedene Regeln betrachten um sich einen besseren Überblick zu verschaffen.

Wie haben Sie die Muster in den Regeln erkannt?

◯ Gedanklicher Abgleich der Namen ◯ Selbst nach Muster basierend auf den Strings gesucht

◯ Die Bögen an der Seite zeigen die Muster automatisch auf

Würden sie diese Bögen als hilfreich einstufen?

◯ ja ◯ nein

Warum?:

Fragebogen zum Softwarehandling → SUS (max 2 min)

Bitte füllen sie diese Fragebogen schnell nach ihrem 1. Eindruck nach aus.

1. Ich denke, dass ich das System gerne häufig benutzen würde.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

2. Ich fand das System unnötig komplex.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

3. Ich fand das System einfach zu benutzen.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

4. Ich glaube, ich würde die Hilfe einer technisch versierten Person benötigen, um das System benutzen zu können.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

5. Ich fand, die verschiedenen Funktionen in diesem System waren gut integriert.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

6. Ich denke, das System enthielt zu viele Inkonsistenzen.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

7. Ich kann mir vorstellen, dass die meisten Menschen den Umgang mit diesem System sehr schnell lernen.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

8. Ich fand das System sehr umständlich zu nutzen.

Stimme überhaupt nicht zu 1	**2**	**3**	**4**	**Stimme voll zu 5**
○	○	○	○	○

9. Ich fühlte mich bei der Benutzung des Systems sehr sicher.

Stimme überhaupt nicht zu 1	**2**	**3**	**4**	**Stimme voll zu 5**
○	○	○	○	○

10. Ich musste eine Menge lernen, bevor ich anfangen konnte das System zu verwenden.

Stimme überhaupt nicht zu 1	**2**	**3**	**4**	**Stimme voll zu 5**
○	○	○	○	○

Semi-strukturiertes Interview (ca. 15 min)

Waren die Filtermöglichkeiten verständlich?

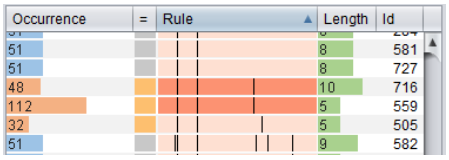
Haben die verschiedenen Visualisierungsmöglichkeiten zum Verständnis beigetragen?

Haben Sie die Verwendung der Knowledge Datenbank verstanden?

Haben Ihnen die verschiedenen Möglichkeiten der Wissensrepräsentation bei der Findung ihrer Entscheidungen geholfen?

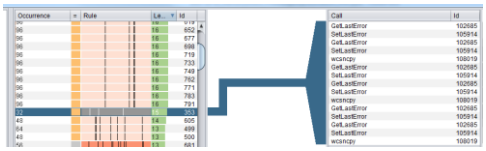
Wie wurde das Expertenwissen im System repräsentiert?

Waren die Balkendiagrame in den Tabellen hilfreich? Wofür?

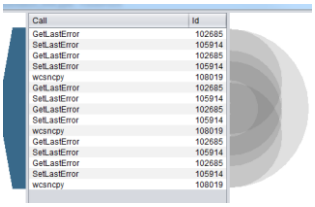


Waren die Histogramme bei den Regeln hilfreich? Wofür?

War die Verbindungslinie zwischen den Tabellen hilfreich? Wofür?



Waren die Bögen (Arc-Diagram) neben der Detailtabelle hilfreich? Wofür?



Wie ist ihr Gesamteindruck zu der Software?

Herzlichen Dank dass sie an diesem Softwaretest teilgenommen haben.

APPENDIX C

Material for the User Study on KAVAGait (in German)

KAVA-GAIT Usability Study

Allgemeines

Die Systemtests werden voraussichtlich im Zeitraum zwischen 10.10.2016 und 31.10.2016 durchgeführt. Es werden voraussichtlich 5-8 ProbandInnen getestet wobei diese alle aus der Physiotherapie Domäne kommen. Hierbei streben wir an, 4-6 ProbandInnen aus der Forschung und 1-2 ProbandInnen aus der Wirtschaft zu haben. Die ProbandInnen aus dem wirtschaftlichen Sektor werden Sinne einer Case Study befragt.

Methode

Um eine gute und aufschlussreiche Evaluierung des KAVA-Time KAVAGait Prototypen zu garantieren, werden mehrere verschiedene Testansätze zeitgleich bzw. hintereinander durchgeführt.

- Erhebung der persönlichen Daten
- Usertest (Aufgabenbasiert) + Video / Case Study + Video
- System Usability Scale (SUS)
- Semi-strukturiertes Interview

Basierend auf den Erkenntnissen dieser Tests werden verschiedene Auswertungen durchgeführt und dokumentiert.

Vorgehensweise

JedeR ProbandIn wird auf einem Handelsüblichen PC den Test durchführen. Der Test dauert ca. eine Stunde und wird von einem Testleiter begleitet. Neben den Aufzeichnungen die durch den Testleiter erstellt werden, wird während dem Systemtest auch noch ein Bildschirmvideo erstellt (gegebenenfalls kann auch der Proband gefilmt werden um die Gesichtsausdrücke während der einzelnen Aufgaben festzuhalten).

- Testdauer: ca. 1 Stunde
- Anwesende Personen: TestleiterIn und ProbandIn
- System: Handelsüblicher PC mit 22“ oder 24“ Monitor

Forschungsfrage

1. Profitiert der Analyst von extern gespeicherten Wissen bei der Analyse von aufgezeichneten Ganganalysedaten?
2. Ist die Art der Wissensspeicherung verständlich und nachvollziehbar?
3. Ist die Visualisierung des externen Wissens verständlich für den Probanden / die Probandin?

Ziele

- Testen des Forschungsprototypen auf seine Funktionalität.
- Testen der Visualisierungstechniken auf Verständlichkeit in Bezug auf die Domäne.
- Testen der Effektivität der Wissensspeicherung und Repräsentation im System.

Nicht Ziele

- Vergleich des Prototyps mit einem anderen Analysesystem.
- Performancetests

Fachhochschule St. Pölten GmbH, Matthias Corvinus-Straße 15, 3100 St. Pölten

_____

_____

_____

Im Folgenden <Befragte/r> genannt

Zustimmungserklärung Interview Projekt KAVA-Time

Im Rahmen des FWF Projektes KAVA-Time werden Interviews und Softwaretests in der Zeit vom 10.10.2016 bis 31.10.2016 durch MitarbeiterInnen der FH St. Pölten durchgeführt und aufgezeichnet (Video-, Tonaufnahme und Notizen).

Der/die obengenannte Befragte stimmt hiermit ausdrücklich zu, dass das mit ihm/ihr geführte Interview und die Softwaretests ausgewertet und FH-intern für Lehre und Forschung verwendet und in anonymisierter Form auch z.B. für wissenschaftliche Publikationen veröffentlicht werden dürfen.

St. Pölten, am _____

_____

Unterschrift

KAVA-GAIT Usability Study: Road Map

Willkommen (ca. 5 min)

Hallo, mein Name ist <NAME> und ich werde Sie heute durch den etwa einstündigen Test unseres Systems begleiten. Vor dem Beginn des Tests werde ich Ihnen nähere Informationen zum Testablauf vorlesen. Diese Maßnahme ist insofern notwendig, da ich sicherstellen möchte, dass alle Testpersonen die gleichen Informationen erhalten.

Bei diesem Test wird ein interaktives Tool zur Exploration von aufgezeichneten Bodenreaktionskräften aus einer Ganganalyse, sowie zum Analysieren von Beeinträchtigungen auf seine Nutzbarkeit getestet. Wichtig ist an dieser Stelle anzumerken, dass nicht Sie als Person getestet werden sondern das System. Ich möchte Sie bitten ehrliche Aussagen zu dem System zu tätigen, sei es positiv oder negativ, beides ist für uns sehr hilfreich, um die Entwicklung des Systems bestmöglich fortzusetzen.

Zu Beginn werde ich Ihnen ein paar Fragen zu Ihrer Person, Ausbildung, beruflichen Werdegang und der Erfahrung in Bezug auf Ganganalyse stellen. Anschließend werde ich Ihnen eine kurze Einführungspräsentation für das Tool zeigen und anschließend 4 Aufgaben vorlesen die Sie bitte zu lösen versuchen. Lassen Sie sich bei den Aufgaben so viel Zeit wie sie benötigen und sprechen Sie ihre Gedanken einfach laut aus. Nicht sie werden getestet, sondern das System auf seine Verständlichkeit und Benutzerfreundlichkeit.

Ich werde Sie während des Tests beobachten und dazu Notizen machen. Während der Testphase unseres Systems werden die Interaktionen aufgezeichnet (protokolliert), sodass wir diese im Nachhinein genau analysieren können. Ebenso wird von diesem Test ein Video aufgenommen. Dieses Video hilft uns bei der Testanalyse und der Verbesserung der Applikation. Alle Aufzeichnungen dieses Tests werden vertraulich für Forschungszwecke in diesem Projekt behandelt und nicht an außenstehende Personen weitergegeben. Diesbezüglich möchte ich Sie bitten unsere Einverständniserklärung zu unterzeichnen.

Einverständniserklärung von Probandin oder Probanden unterzeichnen lassen!

Falls während des Tests Fragen auftreten, lassen Sie es mich bitte sofort wissen. Um die Testanalyse zu vereinfachen, möchte ich Sie bitten, bei den Aufgaben Ihren Gedankengang und jeden Schritt den Sie unternehmen werden, laut auszusprechen (Bspw. Ich klicke jetzt auf <XY> um eine Sortierung der Daten nach <YZ> vorzunehmen …).

Gibt es noch Fragen?

<Starten der Kamera für den Test>

Ich werde nun die Kamera für die Aufzeichnung des Tests starten.

Personenbezogene Fragen

Als erstes beginnen wir mit den zuvor besprochenen Personenbezogenen Fragen:

Name: _____

Geschlecht: _____

Alter: _____

Ausbildung: _____

Beruf: _____

Tätigkeit: _____

Wie viele Jahre Berufserfahrung haben Sie in der Ganganalyse?

_____

Haben Sie Erfahrung in der instrumentierten Ganganalyse?

◯ Ja ◯ Nein

Wenn Ja, mit welchen Systemen haben Sie bereits des Öfteren gearbeitet.

◯ Videobasierte Haltungs- und Bewegungsanalyse (2D)

◯ Videobasierte Haltungs- und Bewegungsanalyse (3D)

◯ Elektromyographie

◯ Kraftmessplatte

◯ Druckverteilungsmessplattform oder ähnliches

◯ klinische instrumentierte 3D Ganganalyse (inkludiert eine Kombination von Punkt 1-4)

◯ Sonstiges:

_____

_____

_____

Haben sie Erfahrung im Interpretieren von Bodenreaktionskräften die während einer Ganganalyse (mittels Kraft- oder Druckmessplattform) erhoben wurden?

◯ Ja ◯ Nein

Wenn ja, wie würden Sie sich selbst in Bezug auf Ihre Fähigkeit zur klinischen Interpretation von Bodenreaktionskräften einstufen?

◯ Anfänger ◯ Mäßig erfahren ◯ Erfahren ◯ Experte

Bitte schätzen sie Ihre IT-Kompetenz in Bezug zur instrumentierten Ganganalyse ein?

◯ Gering ◯ Mäßig ◯ Ausreichend ◯ Gut ◯ Hervorragend

Besten Dank für die Informationen. Beginnen wir nun mit der kurzen Einführung und anschließend mit den Aufgaben in unserem zu testenden System. Bitte sprechen Sie alle Gedankengänge laut aus während der Lösung der Aufgaben.

Teaser Präsentation „how it works“ (ca. 2,5 min)

Die wichtigsten 5 Features

<!!! ACHTUNG: Starten der Präsentation, des Präsentationsvideos!!!>

Test der Software (ca. 30 min)

Kennenlernen des KAVAGait-Systems zur interaktiven Exploration von aufgezeichneten Ganganalysedaten.

Vor Ihnen sehen Sie nun das Interface des Analysesystems. Bitte sehen Sie sich das Interface einmal an um einen ersten Eindruck zu gewinnen.

Was ist der erste Eindruck?

Danke für die erste Einschätzung des Systems. Sehen wir uns nun das System einmal in Detail bezüglich der gebotenen Optionen und Funktionalitäten an.

Welche Funktionalitäten können Sie nach ihrem ersten Eindruck ableiten?

<!!! NICHT SAGEN!!!:Links oben ist die Wissensdatenbank; links unten befindetn sich diverse Filter; Rest ist leer aber hat einen Button zum Patienten laden.>

Aufgabe 1 <Wissensexploration → Kategorien (Guided)>:

Bitte widmen Sie nun Ihre Aufmerksamkeit dem „Knowledge Tree“ der die Wissensdatenbank des Systems darstellt. Was sehen Sie auf den ersten Blick in dieser Struktur?

◯ Kisten ◯ Ordnersymbole ◯ Sheets

Was stellen Ihrer Meinung nach die Kisten, Ordnersymbole und Sheets in dieser Struktur dar?

Kisten (Kategorien): _____

Ordnersymbole (Klassen): _____

Sheets (Personen): _____

Bitte Klicken Sie nun auf ein Ordnersymbol (Klasse), was sehen Sie hier?

_____

_____

_____

Mittels der Struktur **zu ihrer „Rechten“** können Sie die einzelnen Wertebereiche der verschiedenen Klassen in den Kategorien nach Ihren persönlichen Vorgaben verändern / anpassen. Ist die Darstellung der Bereiche verständlich bzw. die Anwendung?

◯ Ja ◯ Nein

_____

_____

_____

Ist die Anwendung der Verstellung der Ober- bzw. Untergrenze der einzelnen Bereiche verständlich?

◯ Ja ◯ Nein

_____

_____

_____

Ist Ihnen die Veränderung der farblichen Hinterlegung des / der Balken aufgefallen die Sie angepasst haben?

◯ Ja ◯ Nein

_____

_____

_____

Nachdem Sie nun Ihre Änderungen durchgeführt haben, bestätigen Sie diese so, dass die Änderungen in der Datenbank gespeichert werden.

◯ Sofort geschafft ◯ herumprobiert und geschafft ◯ nicht geschafft

Erkennen sie in der Struktur des Knowledge Trees, nun dass der Datensatz händisch angepasst wurde?

◯ Ja (anhand des Dreieckigen Symbols) ◯ Nein

_____

_____

_____

Bitte versuchen Sie nun den Datenstamm wieder zurückzusetzen so dass alle Wertebereiche wieder automatisiert berechnet werden:

◯ Sofort geschafft ◯ herumprobiert und geschafft ◯ nicht geschafft

<u><!!! NICHT SAGEN!!!: Funktioniert mit der rechten Maustaste auf der Baumstruktur></u>

Aufgabe 2 <Wissensexploration → Personen (Guided)>:

Nachdem wir uns zuvor mit den Kategorien und Klassen der Wissensdatenbank beschäftigt haben, werden wir uns nun den einzelnen gespeicherten Personen widmen. Bitte klicken Sie auf die Kategorie „Examination“ und wählen sie die Klasse „Knee“ aus. Klicken Sie nun auf eine Person der Klasse. Was sehen sie hier? Nehmen Sie sich ruhig Zeit um sich alles anzusehen:

_____

_____

_____

_____

_____

_____

Der obere Boxplot repräsentiert immer die Klasse „**Norm Data**“, der untere Boxplot zeigt Ihnen immer die Klasse, der die Person zugeordnet ist. Die dicke schwarze Linie zeigt ihnen den Wert der Person in diesem Bereich. Ist diese Darstellung für die Exploration der Parameter hilfreich? Warum?

◯ Ja ◯ Nein

_____

_____

_____

Hinweisen: Jede Spalte hat einen Tooltik Text als Kurzbeschreibung

Was können sie aus der Spalte **„Category Difference“** (**Cat. Difference**) ableiten?

◯ Unterschied zwischen dem Parameter der gewählten und der „**Norm Data**“ Klasse

◯ Ein Längerer Balken zeigt einen größeren Unterschied an (Wert ist aussagekräftiger)

_____

_____

_____

_____

Verändern Sie nun den Wertebereich eines Parameters, was fällt Ihnen dabei auf?

◯ Boxplot der gewählten Klasse passt sich an ◯ Range-Slider wird wieder anders eingefärbt

_____

_____

_____

Bitte verwerfen Sie nun die von Ihnen durchgeführten Änderungen in dieser Ansicht:

◯ Sofort geschafft ◯ herumprobiert und geschafft ◯ nicht geschafft

<!!! NICHT SAGEN!!!: RESET Button drücken>

Bitte adaptieren Sie nun einen beliebigen Wert und speichern sie diesen in der Datenbank:

◯ Sofort geschafft ◯ herumprobiert und geschafft ◯ nicht geschafft

<!!! NICHT SAGEN!!!: SAVE Button drücken>

Erkennen Sie in der Struktur des Knowledge Trees nun, dass der Datensatz händisch angepasst/verändert wurde?

◯ Ja (anhand des Dreieckigen Symbols) ◯ Nein

_____

_____

_____

Bitte versuchen Sie nun den Datenstamm wieder zurückzusetzen, so dass alle Wertebereiche wieder automatisiert berechnet werden:

◯ Sofort geschafft ◯ herumprobiert und geschafft ◯ nicht geschafft

Aufgabe 3 <Laden und explorieren von Patientendaten (Guided)>:

Im nächsten Schritt werden wir nun einen Patienten laden und dessen aufgezeichnete Ganganalysedaten und die Berechnungsergebnisse explorieren. Wir nehmen an, dass Sie soeben mit der Durchführung der Messungen fertig geworden sind und nun die gespeicherten Daten Ihres Patienten in das System laden möchten. **Der Patient ist männlich, 53 Jahre alt, hat 86 kg und ist 178 cm groß**. Bitte laden sie jetzt den Patienten:

◯ Sofort geschafft ◯ herumprobiert und geschafft ◯ nicht geschafft

Sie haben nun erfolgreich ihren ersten Patienten im System geladen, was sehen Sie alles am Monitor:

◯ Knowledge Table (◯ Category, ◯ Parameter in Category, ◯ Match)

◯ Filter (◯ Alter, ◯ Größe, ◯ Gewicht)

◯ Personeninformationen

◯ vertikale Bodenreaktionskräfte (◯ links, ◯ rechts und ◯ zusammengesetzt)

_____

_____

_____

Widmen wir nun einmal unsere Aufmerksamkeit dem Knowledge Table. Was sehen sie hier?

◯ Kategorien ◯ Parameter der Kategorien ◯ Match

_____

_____

_____

_____

_____

_____

Im nächsten Schritt möchten Sie mittels der Filter die Personendaten einschränken. Die Filter können verwendet werden um die **Matches** für den geladenen Patienten genauer zu berechnen. Sie Stellen nun den Filter für das Alter auf 48 bis 56 Jahre:

◯ Sofort geschafft ◯ herumprobiert und geschafft ◯ nicht geschafft

Den Filter für die Körpergröße auf 165 bis 186 cm:

◯ Sofort geschafft ◯ herumprobiert und geschafft ◯ nicht geschafft

Und den Filter für das Körpergewicht auf 60 bis 110 kg:

◯ Sofort geschafft ◯ herumprobiert und geschafft ◯ nicht geschafft

Ist ihnen während des Einstellens der Filter etwas aufgefallen? Hat sich in irgendeiner Ansicht etwas verändert?

◯ In Knowledge Table

◯ Match Balken hat sich verändert

_____

_____

_____

Würden Sie das vom System vorgeschlagenen Analyseergebnis bestätigen (und warum)?

◯ Ja ◯ Nein

_____

_____

_____

_____

_____

_____

Klicken Sie nun bitte auf die Klasse mit dem größten **Matching** Ergebnis, was ist jetzt passiert?

◯ Parameter Explorer hat sich geöffnet

_____

_____

_____

Klicken Sie nun auf eine Kategorie mit einem geringeren **Matching** Ergebnis, was sagt Ihnen das?

◯ Parameter Explorer hat sich verändert

_____

_____

_____

Gehen Sie bitte wieder zurück auf Klasse mit dem größten **Matching** Ergebnis, was können sie in dieser Ansicht jetzt tun?

<!!! NICHT SAGEN!!!: Die einzelnen gemessenen und errechneten Parameter des geladenen Patienten mit denen aus der Datenbank vergleichen (in diesem Fall aber nur mit Healthy)>

_____

_____

_____

_____

_____

Bestätigen die einzelnen errechneten Parameter der gewählten Klasse Ihr Analyseergebnis (warum)?

◯ Ja ◯ Nein

_____

_____

_____

_____

Aufgabe 4 <Selbstständige Exploration „ankle1.csv“ frischer Prototyp>:

<!!! ACHTUNG: Für diesen Test muss das System mit dem „ankle1.csv“ gestartet werden!!!>

Bitte laden sie erneut den Patienten der vorhergehenden Analyse. Der Patient hatte die folgenden persönlichen Daten: **Geschlecht: männlich, Alter: 48 Jahre, Größe: 174 cm, Gewicht: 86 kg.**

◯ Sofort geschafft ◯ herumprobiert und geschafft ◯ nicht geschafft

Welche drei Vorschläge der Wissensdatenbank sind am meisten ausgeprägt für den geladenen Patienten?

◯ Knee ◯ Calcaneus ◯ Ankle

Stellen sie nun bitte die Filter auf folgende Werte ein:

Geschlecht: männlich

◯ Sofort geschafft ◯ herumprobiert und geschafft ◯ nicht geschafft

Alter: 40 bis 65

◯ Sofort geschafft ◯ herumprobiert und geschafft ◯ nicht geschafft

Gewicht: 75 bis MAX

◯ Sofort geschafft ◯ herumprobiert und geschafft ◯ nicht geschafft

Was fällt ihnen nun auf wenn sie sich den „Knowledge Table“ ansehen

◯ Match ist bei Klasse „Ankel“ am größten

◯ 1 leeres Feld bei Klasse „Calcaneus“

◯ 2 leere Felder bei Klasse „Healthy“

_____

_____

_____

_____

_____

Was könnten die leeren Felder bei den Klassen „Calcaneus“ und „Healthy“ bedeuten?

◯ Parameter liegt nicht im Bereich / Parameter liegt auserhalb

_____

_____

_____

Bitte setzen Sie nun die Filter zurück und wählen sie nur männliche Patienten bei den Filteroptionen aus.

◯ Sofort geschafft ◯ herumprobiert und geschafft ◯ nicht geschafft

Versuchen Sie nun die Ausreißer der gewählten Klasse im Parameter Explorer zu eliminieren. Ausreißer sind mit einem Kreis im zugehörigen Boxplot gekennzeichnet.

◯ Sofort geschafft ◯ herumprobiert und geschafft ◯ nicht geschafft

Nachdem Sie nun mit Ihren Einstellungen fertig sind, speichern Sie bitte diese in der Wissensdatenbank. Der Patient wird automatisch mitgespeichert.

◯ Sofort geschafft ◯ herumprobiert und geschafft ◯ nicht geschafft

Herzlichen Dank, wir haben nun diesen Teil des Tests abgeschlossen.

Fragebogen zum Softwarehandling → SUS (max 2 min)

Bitte füllen sie diese Fragebogen schnell nach ihrem 1. Eindruck nach aus.

1. Ich denke, dass ich das System gerne häufig benutzen würde.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

2. Ich fand das System unnötig komplex.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

3. Ich fand das System einfach zu benutzen.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

4. Ich glaube, ich würde die Hilfe einer technisch versierten Person benötigen, um das System benutzen zu können.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

5. Ich fand, die verschiedenen Funktionen in diesem System waren gut integriert.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

6. Ich denke, das System enthielt zu viele Inkonsistenzen.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

7. Ich kann mir vorstellen, dass die meisten Menschen den Umgang mit diesem System sehr schnell lernen.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

8. Ich fand das System sehr umständlich zu nutzen.

Stimme überhaupt nicht zu 1	**2**	**3**	**4**	**Stimme voll zu 5**
○	○	○	○	○

9. Ich fühlte mich bei der Benutzung des Systems sehr sicher.

Stimme überhaupt nicht zu 1	**2**	**3**	**4**	**Stimme voll zu 5**
○	○	○	○	○

10. Ich musste eine Menge lernen, bevor ich anfangen konnte das System zu verwenden.

Stimme überhaupt nicht zu 1	**2**	**3**	**4**	**Stimme voll zu 5**
○	○	○	○	○

Semi-strukturiertes Interview (ca. 15 min)

Waren die Filtermöglichkeiten verständlich?

Haben die verschiedenen Visualisierungsmöglichkeiten zum Verständnis beigetragen?

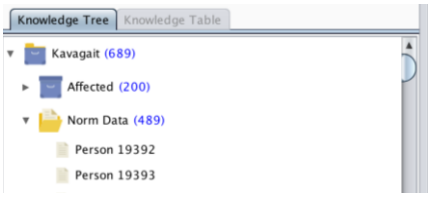
Haben Sie die Verwendung der Wissensdatenbank verstanden?

Haben Ihnen die verschiedenen Möglichkeiten der Wissensrepräsentation bei der Findung ihrer Entscheidungen geholfen?

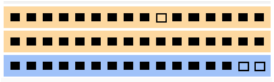
Wie wurde das Expertenwissen im System repräsentiert?

War das Speichern von Wissen basierend auf neu zugeordneten Patienten oder Bereichsanpassungen der einzelnen Werte verständlich?

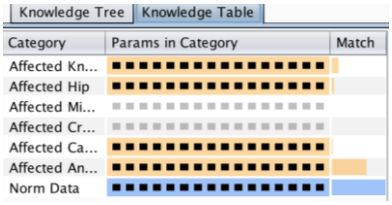
Waren die Symbole in der Baumstruktur der Wissensdatenbank hilfreich und verständlich?



Waren die Fingerprints für die Parameter in der Kategorie hilfreich?



Waren die Balkendiagramme für das Matching hilfreich?



Waren die Schattierungen der einzelnen Werte im Range-Slider hilfreich? Wofür?

Haben Sie vor diesem Test schon einmal mit Boxplots gearbeitet? Warum?

War die Darstellung der Norm Daten und der ausgewählten Klasse als Boxplots verständlich? Warum?

War der Vergleich des neu geladenen Patienten, der Norm Daten und der ausgewählten Klasse hilfreich? Wofür?

Herzlichen Dank, dass sie an diesem Softwaretest teilgenommen haben.

Bibliography

Aaltonen, A. and Lehikoinen, J. (2005). Refining visualization reference model for context information. *Personal and Ubiquitous Computing*, 9(6):381–394.

Ahlberg, C. and Shneiderman, B. (1994). Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pages 313–317. ACM.

Ahlberg, C., Williamson, C., and Shneiderman, B. (1992). Dynamic queries for information exploration: An implementation and evaluation. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pages 619–626. ACM.

Aigner, W., Hoffmann, S., and Rind, A. (2013). EvalBench: A software library for visualization evaluation. *Computer Graphics Forum*, 32(3pt1):41–50.

Aigner, W., Miksch, S., Schumann, H., and Tominski, C. (2011). *Visualization of Time-Oriented Data*. Springer, London.

Amar, R. A. and Stasko, J. T. (2005). Knowledge precepts for design and evaluation of information visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):432–442.

Anderson, B., Storlie, C., and Lane, T. (2012). Improving malware classification: Bridging the static/dynamic gap. In *Proc. ACM Workshop on Security and Artificial Intelligence, AISec*, pages 3–14. ACM.

Andrews, D. F. (1972). Plots of high-dimensional data. *Biometrics*, 28(1):125–136.

Andrienko, G., Andrienko, N., Bak, P., Keim, D., and Wrobel, S. (2013). *Visual analytics of movement*. Springer, Berlin.

Andrienko, N. and Andrienko, G. (2005). *Exploratory Analysis of Spatial and Temporal Data: A Systematic Approach*. Springer, Berlin, New York.

Ankerst, M., Keim, D. A., and Kriegel, H.-P. (1996). Circle segments: A technique for visually exploring large multidimensional data sets. In *Proc. Visualization '96, Hot Topic Session*, San Francisco, CA.

Baker, R. (2013). *Measuring walking: a handbook of clinical gait analysis*. Mac Keith Press, London.

Bangor, A., Kortum, P., and Miller, J. (2009). Determining what individual SUS scores mean: Adding an adjective rating scale. 4(3):114–123.

Barghouti, N. S., Mocenigo, J. M., and Lee, W. (1997). Grappa: a GRAPh PAckage in java. In *Int. Symp. on Graph Drawing*, pages 336–343. Springer.

Bayer, U., Kruegel, C., and Kirda, E. (2006). TTAnalyze: A tool for analyzing malware. In *15th Ann. Conf. Europ. Inst. Computer Antivirus Research, EICAR*.

Bazrafshan, Z., Hashemi, H., Fard, S., and Hamzeh, A. (2013). A survey on heuristic malware detection techniques. In *Conf. on Information and Knowledge Technology*, pages 113–120.

Beaudouin-Lafon, M. (2004). Designing interaction, not interfaces. In *Proc. Working Conf. Advanced Visual Interfaces, AVI*, pages 15–22. ACM.

Becker, R. A. and Cleveland, W. S. (1987). Brushing scatterplots. *Technometrics*, 29(2):127–142.

Benedetti, M., Catani, F., Leardini, A., Pignotti, E., and Giannini, S. (1998). Data management in gait analysis for clinical applications. *Clinical Biomechanics*, 13(3):204–215.

Bernard, J., Wilhelm, N., Krüger, B., May, T., Schreck, T., and Kohlhammer, J. (2013). MotionExplorer: Exploratory Search in Human Motion Capture Data Based on Hierarchical Aggregation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2257–2266.

Bernard, J., Wilhelm, N., Scherer, M., May, T., and Schreck, T. (2012). TimeSeriesPaths : Projection-Based Explorative Analysis of Multivariate Time Series Data. *Journal of WSCG20*, 20(2):97–106.

Bertin, J. (1983). *Semiology of Graphics*. University of Wisconsin Press.

Bertin, J. (1999). Readings in Information Visualization. pages 62–65. Morgan Kaufmann, San Francisco, CA, USA.

Bertini, E. and Lalanne, D. (2009). Surveying the Complementary Role of Automatic Data Analysis and Visualization in Knowledge Discovery. In *Proc. of the ACM SIGKDD: Integrating Automated Analysis with Interactive Exploration*, VAKD '09, pages 12–20, New York. ACM.

Bhuyan, M. H., Bhattacharyya, D., and Kalita, J. (2011). Surveying port scans and their detection methodologies. *The Computer Journal*, 54(10):1565–1581.

Bischoff, K. M. (1992). Design, implementation, use, and evaluation of ox: An attribute-grammar compiling system based on yacc, lex, and c. *Computer Science Technical Reports (TR92-31)*.

Blumenstein, K., Niederer, C., Wagner, M., Schmiedl, G., Rind, A., and Aigner, W. (2016). Evaluating Information Visualization on Mobile Devices: Gaps and Challenges in the Empirical Evaluation Design Space. In *Proc. of the Workshop on Beyond Time And Errors (BELIV)*, pages 125–132, Baltimore, MD, USA. ACM.

Blumenstein, K., Wagner, M., and Aigner, W. (2015a). Cross-Platform InfoVis Frameworks for Multiple Users, Screens and Devices: Requirements and Challenges. In *DEXiS Workshop on Data Exploration for Interactive Surfaces. In conjunction with ACM ITS*, pages 7–11.

Blumenstein, K., Wagner, M., Aigner, W., von Suess, R., Prochaska, H., Püringer, J., Zeppelzauer, M., and Sedlmair, M. (2015b). Interactive Data Visualization for Second Screen Applications: State of the Art and Technical Challenges. In Schulz, H.-J., Urban, B., and Freiherr von Lukas, U., editors, *Proceedings of the International Summer School on Visual Computing*, pages 35–48, Rostock, Germany. Frauenhoferverlag.

Bou-Harb, E., Debbabi, M., and Assi, C. (2014). Cyber scanning: A comprehensive survey. *IEEE Communications Surveys Tutorials*, 16(3):1496–1519.

Brehmer, M. and Munzner, T. (2013). A multi-level typology of abstract visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2376–2385.

Brooke, J. (1996). SUS–a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7.

Cabral, S., Resende, R. A., Clansey, A. C., Deluzio, K. J., Selbie, W. S., and Veloso, A. P. (2016). A global gait asymmetry index. *Journal of applied biomechanics*, 32(2):171–177.

Cammarano, M., Dong, X., Chan, B., Klingner, J., Talbot, J., Halevy, A., and Hanrahan, P. (2007). Visualization of heterogeneous data. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1200–1207.

Cappozzo, A., Della Croce, U., Leardini, A., and Chiari, L. (2005). Human movement analysis using stereophotogrammetry: Part 1. theoretical background. *Gait & Posture*, 21(2):186–196.

Card, S. K. and Mackinlay, J. (1997). The structure of the information visualization design space. In *IEEE Symposium on Information Visualization (InfoVis)*, pages 92–99.

Card, S. K., Mackinlay, J. D., and Shneiderman, B. (1999). *Readings in Information Visualisation. Using Vision to Think.: Using Vision to Think*. Morgan Kaufman Publ Inc, San Francisco, Calif.

Carpendale, M. (2003). Considering visual variables as a basis for information visualisation. University of Calgary, http://prism.ucalgary.ca/handle/1880/45758.

CERT (2017). ProcDOT - CERT.at. http://www.cert.at/downloads/software/procdot.html, [cited: 2017-01-29].

Chen, C. (2005). Top 10 unsolved information visualization problems. *IEEE Computer Graphics and Applications*, 25(4):12–16.

Chen, M., Ebert, D., Hagen, H., Laramee, R., Van Liere, R., Ma, K.-L., Ribarsky, W., Scheuermann, G., and Silver, D. (2009). Data, information, and knowledge in visualization. *IEEE Computer Graphics and Applications*, 29(1):12–19.

Chen, M. and Golan, A. (2016). What may visualization processes optimize? *IEEE Transactions on Visualization and Computer Graphics*, 22(12):2619–2632.

Chen, M. and Hagen, H. (2010). Guest editors' introduction: Knowledge-assisted visualization. *IEEE Computer Graphics and Applications*, 30(1):15–16.

Chernoff, H. (1973). The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, 68(342):361–368.

Chi, E. H. (2000). A taxonomy of visualization techniques using the data state reference model. In *IEEE Symposium on Information Visualization (InfoVis)*, pages 69–75.

Chi, E. H. (2002). Expressiveness of the Data Flow and Data State Models in Visualization Systems. In *Proc. of the Working Conference on Advanced Visual Interfaces*, AVI '02, pages 375–378, New York, NY, USA. ACM.

Chi, E. H.-H. and Riedl, J. (1998). An operator interaction framework for visualization systems. In *IEEE Symposium on Information Visualization (InfoVis)*, pages 63–70.

Chimani, M., Gutwenger, C., Jünger, M., Klein, K., Mutzel, P., and Schulz, M. (2007). The open graph drawing framework. In *Int. Symp. on Graph Drawing*, pages 23–26.

Christian, J., Kröll, J., Strutzenberger, G., Alexander, N., Ofner, M., and Schwameder, H. (2016). Computer aided analysis of gait patterns in patients with acute anterior cruciate ligament injury. *Clinical Biomechanics*, 33:55–60.

Christodorescu, M., Jha, S., and Kruegel, C. (2008). Mining specifications of malicious behavior. In *India Software Eng. Conf.*, pages 5–14. ACM.

Cleveland, W. S. and McGill, R. (1984). Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 79(387):531–554.

Collins, C., Viegas, F., and Wattenberg, M. (2009). Parallel tag clouds to explore and analyze faceted text corpora. In *Symp. on Visual Analytics Science and Technology*, pages 91–98.

Combi, C., Keravnou-Papailiou, E., and Shahar, Y. (2010). *Temporal Information Systems in Medicine*. Springer, New York.

Conti, G. (2007). *Security data visualization: graphical techniques for network analysis*. No Starch Press.

Conti, G., Dean, E., Sinda, M., and Sangster, B. (2008). Visual reverse engineering of binary and data files. In Goodall, J. R., Conti, G., and Ma, K.-L., editors, *Int. Workshop on Visualization for Cyber Security (VizSec)*, LNCS 5210, pages 1–17. Springer.

Cooper, A., Reimann, R., and Cronin, D. (2007). *About Face 3: The Essentials of Interaction Design*. Wiley, Indianapolis, IN, 3rd edition.

Crouser, R. J., Franklin, L., Endert, A., and Cook, K. (2017). Toward theoretical techniques for measuring the use of human effort in visual analytic systems. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):121–130.

Cuckoo Foundation (2017). Automated Malware Analysis - Cuckoo Sandbox. https://cuckoosandbox.org/, [cited: 2017-01-29].

Diehl, S. (2007). *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Springer, Berlin.

Donahue, J., Paturi, A., and Mukkamala, S. (2013). Visualization techniques for efficient malware detection. In *Proc. IEEE Int. Conf. Intelligence and Security Informatics, ISI*, pages 289–291.

Dornhackl, H., Kadletz, K., Luh, R., and Tavolato, P. (2014). Malicious behavior patterns. In *IEEE International Symposium on Service Oriented System Engineering*, pages 384–389.

Duarte, F., Sikansi, F., Fatore, F., Fadel, S., and Paulovich, F. (2014). Nmap: A novel neighborhood preservation space-filling algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2063–2071.

Egele, M., Scholte, T., Kirda, E., and Kruegel, C. (2012). A survey on automated dynamic malware-analysis techniques and tools. *ACM Computing Surveys*, 44(2):6:1–6:42.

Ellson, J., Gansner, E. R., Koutsofios, E., North, S. C., and Woodhull, G. (2004). Graphviz and dynagraph - static and dynamic graph drawing tools. In Jünger, M. and Mutzel, P., editors, *Graph Drawing Software*, Math.and Vis., pages 127–148. Springer.

Endert, A., Hossain, M. S., Ramakrishnan, N., North, C., Fiaux, P., and Andrews, C. (2014). The human is the loop: new directions for visual analytics. *Journal of Intelligent Information Systems*, 43(3):411–435.

Erickson, W., Lee, C., and von Schrader, S. (2016). 2014 Disability Status Report: United States. Technical report, Cornell University Yang Tan Institute of Employment and Disability (YTI), Ithaca, NY.

Fabian, R. (2013). Data-Oriented Design. http://www.dataorienteddesign.com/dodmain/dodmain.html, [cited: 2015-11-11].

Falconer, S., Bull, R., Grammel, L., and Storey, M. (2009). Creating visualizations through ontology mapping. In *Int. Conf. on Complex, Intelligent and Software Intensive Systems (CISIS)*, pages 688–693.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37.

Federico, P., Amor-Amorós, A., and Miksch, S. (2016). A nested workflow model for visual analytics design and validation. In *Proc. of the Workshop on Beyond Time And Errors (BELIV)*, BELIV, pages 104–111, New York. ACM.

Federico, P., Hoffmann, S., Rind, A., Aigner, W., and Miksch, S. (2014). Qualizon Graphs: Space-efficient time-series visualization with qualitative abstractions. In *Proc. Int. Working Conf. Advanced Visual Interfaces, AVI*, pages 273–280. ACM.

Federico, P., Unger, J., Amor-Amorós, A., Sacchi, L., Klimov, D., and Miksch, S. (2015). Gnaeus: Utilizing clinical guidelines for a knowledge-assisted visualisation of EHR cohorts. In Bertini, E. and Roberts, J. C., editors, *Proc. of the EuroVis Workshop on Visual Analytic (EuroVA)*, pages 79–83. Eurographics.

Felt, A. P., Finifter, M., Chin, E., Hanna, S., and Wagner, D. (2011). A survey of mobile malware in the wild. In *Proc. 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM*, pages 3–14. ACM.

Fink, G., North, C., Endert, A., and Rose, S. (2009). Visualizing cyber security: Usable workspaces. In *Int. Workshop on Vis. for Cyber Sec.*, pages 45–56.

FireEye, Inc. (2017). Cyber Security & Malware Protection. https://www.fireeye.com, [cited: 2017-01-29].

Fischer, F., Mansmann, F., and Keim, D. A. (2012). Real-time visual analytics for event data streams. In *Proceedings of the Annual Symposium on Applied Computing*, SAC '12, pages 801–806. ACM.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.

Gelenbe, E., Gorbil, G., Tzovaras, D., Liebergeld, S., Garcia, D., Baltatu, M., and Lyberopoulos, G. (2013). Security for smart mobile networks: The NEMESYS approach. In *IEEE Global High Tech Congress on Electronics*, pages 63–69.

Gilson, O., Silva, N., Grant, P., and Chen, M. (2008). From web data to visualization via ontology mapping. *Computer Graphics Forum*, 27(3):959–966.

Goodall, J. R., Komlodi, A., and Lutters, W. G. (2004). The work of intrusion detection: Rethinking the role of security analysts. In *Proc. of the 10th Americas Conf. on Info. Systems*, pages 1421–1427, NY.

Gotz, D., Stavropoulos, H., Sun, J., and Wang, F. (2012). ICDA: A platform for intelligent care delivery analytics. *AMIA Annual Symp. Proceedings*, 2012:264–273.

Gove, R., Saxe, J., Gold, S., Long, A., and Bergamo, G. (2014). SEEM: A scalable visualization for comparing multiple large sets of attributes for malware analysis. In *Int. Workshop on Visualization for Cyber Security (VizSec)*, pages 72–79. ACM.

Grégio, A. R. A., Baruque, A. O. C., Afonso, V. M., Filho, D. S. F., Geus, P. L. d., Jino, M., and Santos, R. D. C. d. (2012). Interactive, visual-aided tools to analyze malware behavior. In *Computational Science and Its Applications, ICCSA*, LNCS 7336, pages 302–313. Springer.

Grégio, A. R. A. and Santos, R. D. C. (2011). Visualization techniques for malware behavior analysis. In *Proc. Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense X*, volume 8019, pages 801905:1–801905:9. SPIE.

Han, J., Hu, X., and Cercone, N. (2003). A Visualization Model of Interactive Knowledge Discovery Systems and Its Implementations. *Information Visualization*, 2(2):105–125.

Han, K., Kang, B., and Im, E. G. (2014a). Malware analysis using visualized image matrices. *The Scientific World Journal*, 2014:15.

Han, K., Lim, J. H., and Im, E. G. (2013). Malware analysis method using visualization of binary files. In *Proc. Research in Adaptive and Convergent Systems, RACS*, pages 317–321.

Han, K. S., Lim, J. H., Kang, B., and Im, E. G. (2014b). Malware analysis using visualized images and entropy graphs. *Int. Journal of Information Security*, pages 1–14.

Hand, D. J. (1998). Intelligent data analysis: Issues and opportunities. *Intell. Data Anal.*, 2(1–4):67 – 79.

Haroz, S., Kosara, R., and Franconeri, S. (2016). The connected scatterplot for presenting paired time series. *IEEE Transactions on Visualization and Computer Graphics*, 22(9):2174–2186.

Havre, S., Hetzler, E., Whitney, P., and Nowell, L. (2002). ThemeRiver: Visualizing thematic changes in large document collections. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):9–20.

Heer, J., Bostock, M., and Ogievetsky, V. (2010). A tour through the visualization zoo. *Communications of the ACM*, 53(6):59.

Heer, J., Card, S. K., and Landay, J. A. (2005). Prefuse: A toolkit for interactive information visualization. In *Conf. on Human Factors in Comp. Systems*, pages 421–430. ACM.

Heer, J., Kong, N., and Agrawala, M. (2009). Sizing the Horizon: The Effects of Chart Size and Layering on the Graphical Perception of Time Series Visualizations. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1303–1312. ACM.

Herman, I., Melancon, G., and Marshall, M. (2000). Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43.

Hex-Rays SA. (2017). IDA: About. https://www.hex-rays.com/products/ida/, [cited: 2017-01-29].

Huber, P. J. (1985). Projection pursuit. *The Annals of Statistics*, 13(2):435–475.

Idika, N. and Mathur, A. P. (2007). A survey of malware detection techniques. Technical Report nr. 48, Purdue University.

IKARUS (2017). IKARUS Security Software. https://www.ikarussecurity.com/at/, [cited: 2017-01-29].

Inselberg, A. and Dimsdale, B. (1991). Parallel coordinates. In Klinger, A., editor, *Human-Machine Interactive Systems*, Languages and Information Systems, pages 199–233. Springer.

Janetzko, H., Sacha, D., Stein, M., Schreck, T., Keim, D. A., and Deussen, O. (2014). Feature-driven visual analytics of soccer data. In *IEEE Conf. on Visual Analytics Science and Technology (VAST)*, pages 13–22.

Jankun-Kelly, T. J., Ma, K.-L., and Gertz, M. (2002). A model for the visualization exploration process. In *IEEE Visualization*, pages 323–330.

Jankun-Kelly, T. j., Ma, K. l., and Gertz, M. (2007). A Model and Framework for Visualization Exploration. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):357–369.

Janssen, D., Schöllhorn, W. I., Newell, K. M., Jäger, J. M., Rost, F., and Vehof, K. (2011). Diagnosing fatigue in gait patterns by support vector machines and self-organizing maps. *Human Movement Science*, 30(5):966–975.

Javed, W., McDonnel, B., and Elmqvist, N. (2010). Graphical perception of multiple time series. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):927–934.

Joe Security LLC (2017). Automated Malware Analysis - Joe Sandbox Products. http://www.joesecurity.org/joe-security-products, [cited: 2017-01-29].

Johnson, J. (2014). *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines*. Morgan Kaufmann, Amsterdam ; Boston, 2 edition.

Jouault, F. and Kurtev, I. (2006). Transforming models with ATL. In Bruel, J.-M., editor, *Satellite Events at the MoDELS 2005 Conference*, number 3844 in Lecture Notes in Computer Science, pages 128–138. Springer.

Kadlec, B., Tufo, H., and Dorn, G. (2010). Knowledge-assisted visualization and segmentation of geologic features. *IEEE Computer Graphics and Applications*, 30(1):30–39.

Kancherla, K. and Mukkamala, S. (2013). Image visualization based malware detection. In *Proc. 2013 IEEE Symp. Computational Intelligence in Cyber Security, CICS*, pages 40–44.

Keim, D. (2000). Designing pixel-oriented visualization techniques: theory and applications. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):59–78.

Keim, D. (2002). Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8.

Keim, D., Kohlhammer, J., Ellis, G., and Mansmann, F., editors (2010a). *Mastering the information age: solving problems with visual analytics*. Eurographics, Goslar.

Keim, D., Kriegel, H.-P., and Ankerst, M. (1995). Recursive pattern: a technique for visualizing very large amounts of data. In *, IEEE Conference on Visualization*, pages 279–286, 463.

Keim, D. A., Mansmann, F., Schneidewind, J., Thomas, J., and Ziegler, H. (2008). Visual Analytics: Scope and challenges. In Simoff, S. J., Böhlen, M. H., and Mazeika, A., editors, *Visual Data Mining*, LNCS 4404, pages 76–90. Springer, Berlin.

Keim, D. A., Mansmann, F., and Thomas, J. (2010b). Visual Analytics: How Much Visualization and How Much Analytics? *SIGKDD Explor. Newsl.*, 11(2):5–8.

Kerren, A., Purchase, H. C., and Ward, M. O., editors (2014). *Multivariate Network Visualization*. LNCS 8380. Springer, Cham.

Kielman, J., Thomas, J., and May, R. (2009). Foundations and frontiers in visual analytics. *Information Visualization*, 8(4):239–246.

Kirtley, C. (2006). *Clinical gait analysis: theory and practice*. Elsevier, Edinburgh. OCLC: 845846752.

Kromer, L., Wagner, M., Blumenstein, K., Rind, A., and Aigner, W. (2016). Performance Comparison between Unity and D3.js for Cross-Platform Visualization on Mobile Devices. In *Proceedings of the Forum Media Technology*, pages 47–52. CEUR-WS.

Kulyk, O., Kosara, R., Urquiza, J., and Wassink, I. (2007). Human-centered aspects. In Kerren, A., Ebert, A., and Meyer, J., editors, *Human-Centered Visualization Environments*, number 4417 in Lecture Notes in Computer Science, pages 13–75. Springer.

Lam, H. (2008). A framework of interaction costs in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1149–1156.

Lam, H., Bertini, E., Isenberg, P., Plaisant, C., and Carpendale, S. (2012). Empirical studies in information visualization: Seven scenarios. *IEEE Transactions on Visualization and Computer Graphics*, 18(9):1520–1536.

Lammarsch, T., Aigner, W., Bertone, A., Gartner, J., Mayr, E., Miksch, S., and Smuc, M. (2009). Hierarchical temporal patterns and interactive aggregated views for pixel-based visualizations. In *Information Visualisation, 2009 13th International Conference*, pages 44–50.

Lammarsch, T., Aigner, W., Bertone, A., Miksch, S., and Rind, A. (2011). Towards a concept how the structure of time can support the visual analytics process. In Miksch, S. and Santucci, G., editors, *Proceedings of the International Workshop on Visual Analytics held in Europe (EuroVA)*, pages 9–12. Eurographics, Eurographics.

Lamping, J. and Rao, R. (1994). Laying out and visualizing large trees using a hyperbolic space. In *Proc. of the Annual ACM Symposium on User Interface Software and Technology*, UIST '94, pages 13–14, New York. ACM.

Lanzenberger, M., Miksch, S., and Pohl, M. (2005). Exploring highly structured data: a comparative study of stardinates and parallel coordinates. In *Proc. Int. Conf. Information Visualisation*, pages 312–320.

Laxman, S. and Sastry, P. S. (2006). A survey of temporal data mining. *Sadhana*, 31(2):173–198.

Lazar, J., Feng, J. H., and Hochheiser, H. (2010). *Research Methods in Human-Computer Interaction*. Wiley, Chichester, West Sussex, U.K, 1 edition.

LeBlanc, J., Ward, M., and Wittels, N. (1990). Exploring n-dimensional databases. In *Proc. IEEE Conf. Visualization*, pages 230–237.

Lee, D., Song, I. S., Kim, K., and Jeong, J.-h. (2011). A study on malicious codes pattern analysis using visualization. In *Int. Conf. on Information Science and Applications*, pages 1–5.

Long, A., Saxe, J., and Gove, R. (2014). Detecting malware samples with similar image sets. In *Int. Workshop on Visualization for Cyber Security (VizSec)*, pages 88–95. ACM.

Luh, R., Schramm, G., Wagner, M., and Schrittwieser, S. (2017). Sequitur-based Inference and Analysis Framework for Malicious System Behavior. In *Workshop for Formal Methods in Software Engineering (ForSE), (ICISSP)*, pages 632–643, Porto, Portugal. SCITEPRESS Digital Library.

Mackinlay, J. (1986). Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141.

Mackinlay, J., Hanrahan, P., and Stolte, C. (2007). Show me: Automatic presentation for visual analysis. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1137–1144.

McKenna, S., Staheli, D., Fulcher, C., and Meyer, M. (2016). Bubblenet: A cyber security dashboard for visualizing patterns. *Computer Graphics Forum*, 35(3):281–290.

McKenna, S., Staheli, D., and Meyer, M. (2015). Unlocking user-centered design methods for building cyber security visualizations. In *Visualization for Cyber Security (VizSec), 2015 IEEE Symposium on*, pages 1–8.

McLachlan, P., Munzner, T., Koutsofios, E., and North, S. (2008). LiveRAC: Interactive Visual Exploration of System Management Time-series Data. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1483–1492. ACM.

Meyer, M., Munzner, T., and Pfister, H. (2009). MizBee: A multiscale synteny browser. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):897–904.

Microsoft (2016). Process Monitor. https://technet.microsoft.com/en-us/sysinternals/bb896645, [cited: 2016-02-18].

Miksch, S. and Aigner, W. (2014). A matter of time: Applying a data-users-tasks design triangle to visual analytics of time-oriented data. *Computers & Graphics, Special Section on Visual Analytics*, 38:286–290.

Mistelbauer, G., Bouzari, H., Schernthaner, R., Baclija, I., Kochl, A., Bruckner, S., Sramek, M., and Groller, M. (2012). Smart super views: A knowledge-assisted interface for medical visualization. In *IEEE Conf. on Visual Analytics Science and Technology (VAST)*, pages 163–172.

Mühlbacher, T., Piringer, H., Gratzl, S., Sedlmair, M., and Streit, M. (2014). Opening the black box: Strategies for increased user involvement in existing algorithm implementations. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1643–1652.

Munzner, T. (2009). A nested model for visualization design and validation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):921–928.

Munzner, T. (2014). *Visualization Analysis and Design*. AK Peters.

Nam, J. E., Maurer, M., and Mueller, K. (2009). A high-dimensional feature clustering approach to support knowledge-assisted visualization. *Computers & Graphics*, 33(5):607–615.

Nataraj, L., Karthikeyan, S., Jacob, G., and Manjunath, B. S. (2011a). Malware images: Visualization and automatic classification. In *Int. Workshop on Visualization for Cyber Security (VizSec)*, pages 4:1–4:7. ACM.

Nataraj, L., Yegneswaran, V., Porras, P., and Zhang, J. (2011b). A comparative assessment of malware classification using binary texture analysis and dynamic analysis. In *Proc. ACM Workshop on Security and Artificial Intelligence, AISec*, pages 21–30.

Nazemi, K., Stab, C., and Kuijper, A. (2011). A Reference Model for Adaptive Visualization Systems. In Jacko, J. A., editor, *Human-Computer Interaction. Design and Development Approaches*, number 6761 in LNCS, pages 480–489. Springer.

Nevill-Manning, C. G. and Witten, I. H. (1997). Identifying hierarchical structure in sequences: A linear-time algorithm. *J. Artif. Int. Res.*, 7(1):67–82.

Nielsen, J. (2010). *Usability Engineering*. Kaufmann, Amsterdam, nachdr. edition.

Nigg, B. and Herzog, W. (2007). *Biomechanics of the musculo-skeletal system*. Wiley, Hoboken NJ, 3rd ed. edition.

Nüesch, C., Valderrabano, V., Huber, C., von Tscharner, V., and Pagenstert, G. (2012). Gait patterns of asymmetric ankle osteoarthritis patients. *Clinical Biomechanics*, 27(6):613–618.

Panas, T. (2008). Signature visualization of software binaries. In *Proc. ACM Symp. Software Visualization, SoftVis*, pages 185–188.

PandaLab (2014). Quarterly Report, Q3/2014. Technical report. http://pandanews.de/pandalabs-3-quartalsbericht-2014/.

Paturi, A., Cherukuri, M., Donahue, J., and Mukkamala, S. (2013). Mobile malware visual analytics and similarities of attack toolkits (malware gene analysis). In *Proc. Int. Conf. Collaboration Technologies and Systems, CTS*, pages 149–154.

Pauk, J. and Minta-Bielecka, K. (2016). Gait patterns classification based on cluster and bicluster analysis. *Biocybernetics and Biomedical Engineering*, 36(2):391–396.

Perin, C., Vuillemot, R., and Fekete, J. D. (2013). SoccerStories: A kick-off for visual soccer analysis. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2506–2515.

Perlin, K. and Fox, D. (1993). Pad: An alternative approach to the computer interface. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 57–64, New York. ACM.

Perner, P. (2006). Intelligent data analysis in medicine-recent advances. *Artif. Intell. Med.*, 37(1):1–5.

Perry, J. and Burnfield, J. M., editors (2010). *Gait analysis: normal and pathological function*. SLACK, Thorofare, NJ, 2. ed edition. OCLC: 705606523.

Pickett, R. and Grinstein, G. (1998). Iconographic displays for visualizing multidimensional data. In *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, volume 1, pages 514–519.

Pike, W. A., Stasko, J., Chang, R., and O'Connell, T. A. (2009). The science of interaction. *Information Visualization*, 8(4):263–274.

Pirker, M. and Nusser, A. (2016). A work-flow for empirical exploration of security events. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, pages 477–480, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Pradhan, C., Wuehr, M., Akrami, F., Neuhaeusser, M., Huth, S., Brandt, T., Jahn, K., and Schniepp, R. (2015). Automated classification of neurological disorders of gait using spatio-temporal gait parameters. *Journal of Electromyography and Kinesiology*, 25(2):413–422.

Pretorius, A. J. and Van Wijk, J. J. (2009). What does the user want to see? What do the data want to be? *Information Visualization*, 8(3):153–166.

Purchase, H. C., Andrienko, N., Jankun-Kelly, T. J., and Ward, M. (2008). Theoretical Foundations of Information Visualization. In Kerren, A., Stasko, J. T., Fekete, J.-D., and North, C., editors, *Information Visualization*, Lecture Notes in Computer Science, pages 46–64. Springer. DOI: 10.1007/978-3-540-70956-5_3.

Purwantiningsih, O., Sallaberry, A., Andary, S., Seilles, A., and Azé, J. (2016). Visual analysis of body movement in serious games for healthcare. In *IEEE Pacific Visualization Symposium (PacificVis)*, pages 229–233.

Quist, D. and Liebrock, L. (2009). Visualizing compiled executables for malware analysis. In *Int. Workshop on Visualization for Cyber Security (VizSec)*, pages 27–32.

Quist, D. and Liebrock, L. (2011). Reversing compiled executables for malware analysis via visualization. *Information Visualization*, 10(2):117–126.

RDF Working Group (2017). Resource Description Framework (RDF). http://www.w3.org/RDF/, [cited: 2017-04-07].

Ribarsky, W. and Fisher, B. (2016). The Human-Computer System: Towards an Operational Model for Problem Solving. In *Hawaii Int. Conf. on System Sciences (HICSS)*, pages 1446–1455.

Rieck, K. (2016). Malheur - Automatic Analysis of Malware Behavior. http://www.mlsec.org/malheur/, [cited: 2016-02-18].

Rind, A., Aigner, W., Miksch, S., Wiltner, S., Pohl, M., Turic, T., and Drexler, F. (2011). Visual exploration of time-oriented patient data for chronic diseases: Design study and evaluation. In Holzinger, A. and Simonic, K.-M., editors, *Information Quality in e-Health*, number 7058 in Lecture Notes in Computer Science, pages 301–320. Springer, Berlin.

Rind, A., Aigner, W., Wagner, M., Lammarsch, T., and Miksch, S. (2014). User Tasks for Evaluation: Untangling the Terminology Throughout Visualization Design and Development. In Lam, H., Isenberg, P., Isenberg, T., and Sedlmair, M., editors, *Proc. of the Workshop on Beyond Time And Errors (BELIV)*, pages 9–15. ACM.

Rind, A., Aigner, W., Wagner, M., Miksch, S., and Lammarsch, T. (2015). Task Cube: A three-dimensional conceptual space of user tasks in visualization design and evaluation. *Information Visualization*, 15(4):288–300.

Rind, A., Federico, P., Gschwandtner, T., Aigner, W., Doppler, J., and Wagner, M. (2017a). Visual analytics of electronic health records with a focus on time. In Rinaldi, G., editor, *New Perspectives in Medical Records: Meeting the Needs of Patients and Practitioners*, TELe-Health, pages 65–77. Springer.

Rind, A., Haberson, A., Blumenstein, K., Niederer, C., Wagner, M., and Aigner, W. (2017b). PubViz: Lightweight visual presentation of publication data. In *Proc. Eurographics Conf. Visualization (EuroVis) – Short Paper*. Accepted.

Rohitab, B. (2016). API Monitor: Spy on API Calls and COM Interfaces (Freeware 32-bit and 64-bit Versions!). http://www.rohitab.com/apimonitor, [cited: 2016-02-18].

Roy Rosenzweig Center for History and New Media (2015). Zotero documentation. https://www.zotero.org/support/dev/web_api/v3/basics, [cited: 2015-04-07].

Sacha, D., Stoffel, A., Stoffel, F., Kwon, B. C., Ellis, G., and Keim, D. (2014). Knowledge generation model for visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1604–1613.

Sangeux, M., Rodda, J., and Graham, H. K. (2015). Sagittal gait patterns in cerebral palsy: The plantarflexor–knee extension couple index. *Gait & Posture*, 41(2):586–591.

Sauro, J. (2011). Measuring usability with the system usability scale (SUS): MeasuringU. http://www.measuringu.com/sus.php, [cited: 2016-03-29].

Saxe, J., Mentis, D., and Greamo, C. (2012). Visualization of shared system call sequence relationships in large malware corpora. In *Int. Workshop on Visualization for Cyber Security (VizSec)*, VizSec '12, pages 33–40. ACM.

Schreck, T., Bernard, J., Von Landesberger, T., and Kohlhammer, J. (2009). Visual Cluster Analysis of Trajectory Data with Interactive Kohonen Maps. *Information Visualization*, 8(1):14–29.

Sedlmair, M., Baur, D., Boring, S., Isenberg, P., Jurmu, M., and Butz, A. (2008). Requirements for a MDE system to support collaborative in-car communication diagnostics. In *Workshop on Beyond the Laboratory: Supporting Authentic Collaboration with Multiple Displays*.

Sedlmair, M., Frank, A., Munzner, T., and Butz, A. (2012a). RelEx: Visualization for actively changing overlay network specifications. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2729–2738.

Sedlmair, M., Meyer, M., and Munzner, T. (2012b). Design study methodology: Reflections from the trenches and the stacks. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2431–2440.

Shabtai, A., Klimov, D., Shahar, Y., and Elovici, Y. (2006). An intelligent, interactive tool for exploration and visualization of time-oriented security data. In *Int. Workshop on Visualization for Cyber Security (VizSec)*, pages 15–22. ACM.

Shahar, Y. (1997). A framework for knowledge-based temporal abstraction. *Artificial Intelligence*, 90(12):79 – 133.

Shaid, S. and Maarof, M. (2014). Malware behavior image for malware variant identification. In *Proc. 2014 Int. Symp. on Biometrics and Security Technologies, ISBAST*, pages 238–243.

Shaid, S.Z.M. and Maarof, M.A. (2014). Malware behaviour visualization. *Jurnal Teknologi*, 70(5):25–33.

Sharp, H., Rogers, Y., and Preece, J. (2007). *Interaction Design: Beyond Human-Computer Interaction*. Wiley, Chichester ; Hoboken, NJ, 2. edition.

Shiravi, H., Shiravi, A., and Ghorbani, A. (2012). A survey of visualization systems for network security. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1313–1329.

Shneiderman, B. (1992). Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99.

Shneiderman, B. (1996). The eyes have it: a task by data type taxonomy for information visualizations. In *IEEE Symposium on Visual Languages. Proceedings*, pages 336–343.

Siddiqui, M., Wang, M. C., and Lee, J. (2008). A survey of data mining techniques for malware detection using file features. In *Proceedings of the Annual Southeast Regional Conference on XX*, ACM-SE 46, page 509–510, New York, NY, USA. ACM.

Sikorski, M. and Honig, A. (2012). *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. No Starch Press, 1 edition.

Smuc, M., Schreder, G., Mayr, E., and Windhager, F. (2015). Should we dream the impossible dream of reproducibility in Visual Analytics evaluation? In Aigner, W., Rosenthal, P., and Scheidegger, C., editors, *EuroVis Workshop on Reproducibility, Verification, and Validation in Visualization (EuroRV3)*. Eurographics.

Spence, R. (2006). *Information Visualization: Design for Interaction*. Prentice Hall, New York, 2nd rev. edition.

Stoneburner, G., Goguen, A. Y., and Feringa, A. (2002). SP 800-30. risk management guide for information technology systems. Technical report, National Institute of Standards & Technology.

Tahir, N. M. and Manap, H. H. (2012). Parkinson disease gait classification based on machine learning approach. *Journal of Applied Sciences*, 12(2):180.

Thomas, J. J. and Cook, K. A. (2005). *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Ctr. Published: Paperback.

Tirosh, O., Baker, R., and McGinley, J. (2010). GaitaBase: Web-based repository system for gait analysis. *Computers in Biology and Medicine*, 40(2):201–207.

Tominski, C. (2011). Event-based concepts for user-driven visualization. *Int. Conf. on Information Visualisation*, 10(1):65–81.

Tory, M. and Staub-French, S. (2008). Qualitative analysis of visualization: A building design field study. In *Proc. Workshop BEyond Time and Errors: Novel evaLuation Methods for Information Visualization*, pages 7:1–7:8. ACM.

Trinius, P., Holz, T., Gobel, J., and Freiling, F. (2009). Visual analysis of malware behavior using treemaps and thread graphs. In *Int. Workshop on Visualization for Cyber Security (VizSec)*, pages 33–38.

Tufte, E. R. (2001). *Visual Display of Quantitative Information*. Bertrams, Cheshire, Conn, 2 edition.

Tweedie, L., Spence, R., Dawkes, H., and Hus, S. (1996). Externalising Abstract Mathematical Models. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '96, pages 406–412, New York, NY, USA. ACM.

Van Wijk, J. J. (2005). The value of visualization. In *IEEE Visualization*, pages 79–86.

Vögele, A., Krüger, B., and Klein, R. (2014). Efficient Unsupervised Temporal Segmentation of Human Motion. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SIGGRAPH/SCA)*, SCA '14, pages 167–176, Aire-la-Ville, Switzerland. Eurographics.

Von Landesberger, T., Kuijper, A., Schreck, T., Kohlhammer, J., van Wijk, J., Fekete, J.-D., and Fellner, D. (2011). Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum*, 30(6):1719–1749.

Wagner, M. (2013). *Simulation mittels Masse-Feder-Systemen: Parallelisierte Simulation einer Flüssigkeitsoberfläche mittels Masse-Feder-Systemen*. AV, Saarbrücken.

Wagner, M. (2015). Integrating Explicit Knowledge in the Visual Analytics Process. In *Doctoral Consortium on Computer Vision, Imaging and Computer Graphics Theory and Applications (DCVISIGRAPP)*, pages 9–18, Berlin. SCITEPRESS Digital Library.

Wagner, M., Aigner, W., Haberson, A., and Rind, A. (2015a). Literature review in visual analytics for malware pattern analysis. In *Proc. of the Forschungsforum der österreichischen Fachhochschulen (FFH)*. FH Hagenberg.

Wagner, M., Aigner, W., Rind, A., Dornhackl, H., Kadletz, K., Luh, R., and Tavolato, P. (2014). Problem characterization and abstraction for visual analytics in behavior-based malware pattern analysis. In *Int. Workshop on Visualization for Cyber Security (VizSec)*, pages 9–16. ACM.

Wagner, M., Blumenstein, K., Rind, A., Seidl, M., Schmiedl, G., Lammarsch, T., and Aigner, W. (2016a). Native Cross-platform Visualization: A Proof of Concept Based on the Unity3d Game Engine. In *Int. Conf. on Information Visualisation*, pages 39–44, Lisbon, Portugal. IEEE.

Wagner, M., Blumenstein, K., Wieländer, U., and Judmaier, P. (2015b). Genderorientierter Informatikunterricht: Plattformübergreifende Spieleentwicklung mit Unity3d. In Haag, J., Weißenböck, J., Gruber, W., and Freisleben-Teutscher, C. F., editors,

Game Based Learning - Dialogorientierung & spielerisches Lernen digital und analog: Beiträge zum Tag der Lehre an der FH St.Pölten am 15.10. 2015, pages 23–32, St. Pölten, Österreich. ikon Verlag.

Wagner, M., Federico, P., Rind, A., Amor-Amorós, A., Aigner, W., and Miksch, S. (2017a). The simple model of knowledge-assisted visualization. *IEEE Transactions on Visualization and Computer Graphics*. Under review.

Wagner, M., Fischer, F., Luh, R., Haberson, A., Rind, A., Keim, D. A., and Aigner, W. (2015c). A survey of visualization systems for malware analysis. In *Eurographics Conf. on Visualization (EuroVis) State of The Art Reports*, pages 105–125. Eurographics.

Wagner, M., Rind, A., Rottermanner, G., Niederer, C., and Aigner, W. (2016b). Knowledge-Assisted Rule Building for Malware Analysis. In *Proc. of the Forschungsforum der österreichischen Fachhochschulen (FFH)*, Vienna, Austria. FH des BFI Wien.

Wagner, M., Rind, A., Thür, N., and Aigner, W. (2017b). A knowledge-assisted visual malware analysis system: design, validation, and reflection of KAMAS. *Computers & Security*, 67:1–15.

Wagner, M., Sacha, D., Rind, A., Fischer, F., Luh, R., Schrittwieser, S., Keim, D. A., and Aigner, W. (2017c). Visual Analytics: Foundations and Experiences in Malware Analysis. In Othmane, L. B., Jaatun, M. G., and Weippl, E., editors, *Empirical Research for Software Security: Foundations and Experience*. CRC/Taylor and Francis. In press.

Wagner, M., Slijepčević, D., Horsak, B., Rind, A., Zeppelzauer, M., and Aigner, W. (2017d). KAVAGait – knowledge-assisted visual analytics solution for gait analysis: A design study. *IEEE Transactions on Visualization and Computer Graphics*. Under review.

Wang, C. and Ma, K.-L. (2008). Information and Knowledge assisted analysis and Visualization of large-scale data. In *Workshop on Ultrascale Visualization, UltraVis*, pages 1–8.

Wang, T. D., Plaisant, C., Quinn, A. J., Stanchak, R., Murphy, S., and Shneiderman, B. (2008). Aligning temporal data by sentinel events: Discovering patterns in electronic health records. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 457–466, New York.

Wang, X., Jeong, D. H., Dou, W., Lee, S.-W., Ribarsky, W., and Chang, R. (2009). Defining and applying knowledge conversion processes to a visual analytics system. *Computers & Graphics*, 33(5):616–623.

Wattenberg, M. (2002). Arc diagrams: visualizing structure in strings. In *IEEE Symposium on Information Visualization (InfoVis)*, pages 110–116.

Wattenberg, M. and Viegas, F. (2008). The word tree, an interactive visual concordance. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1221–1228.

Wegner, P. (1997). Why interaction is more powerful than algorithms. *Commun. ACM*, 40(5):80–91.

Wei, J. and Salvendy, G. (2004). The cognitive task analysis methods for job and task design: review and reappraisal. *Behaviour & Information Technology*, 23(4):273–299.

Wilhelm, N., Vögele, A., Zsoldos, R., Licka, T., Krüger, B., and Bernard, J. (2015). FuryExplorer: Visual-Interactive Exploration of Horse Motion Capture Data. In *Visualization and Data Analysis (VDA)*.

Wilkinson, L. (2005). *The Grammar of Graphics*. Springer, 2 edition.

Willems, C., Holz, T., and Freiling, F. (2007). Toward automated dynamic malware analysis using CWSandbox. *IEEE Sec. Privacy*, 5(2):32–39.

Willett, W., Heer, J., and Agrawala, M. (2007). Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1129–1136.

Wills, G. and Wilkinson, L. (2010). AutoVis: Automatic visualization. *Information Visualization*, 9(1):47–69.

Winter, D. A. (2005). *Biomechanics and motor control of human movement*. Wiley, Hoboken NJ, 3rd ed. edition.

Winter, D. A. (2009). *Biomechanics and motor control of human movement*. Wiley, Hoboken, NJ, 4. ed edition. OCLC: 318408191.

Wongsuphasawat, K. and Gotz, D. (2011). Outflow: Visualizing patient flow by symptoms and outcome. In *IEEE VisWeek Workshop on Visual Analytics in Healthcare, Providence, Rhode Island*, pages 25–28.

Wu, T., Wu, Y., Shi, C., Qu, H., and Cui, W. (2016). PieceStack: Toward better understanding of stacked graphs. *IEEE Transactions on Visualization and Computer Graphics*, 22(6):1640–1651.

Wu, Y. and Yap, R. H. C. (2013). Experiments with malware visualization. In Flegel, U., Markatos, E., and Robertson, W., editors, *Proc. Int. Conf. Detection of Intrusions and Malware, and Vulnerability Assessment, DIMVA*, LNCS 7591, pages 123–133. Springer.

Wüchner, T., Pretschner, A., and Ochoa, M. (2014). Davast: Data-centric system level activity visualization. In *Int. Workshop on Visualization for Cyber Security (VizSec)*, pages 25–32. ACM.

Xia, Y., Fairbanks, K., and Owen, H. (2008). Visual analysis of program flow data with data propagation. In Goodall, J. R., Conti, G., and Ma, K.-L., editors, *Vis. for Comp. Sec.*, LNCS 5210, pages 26–35. Springer.

Yao, D., Shin, M., Tamassia, R., and Winsborough, W. (2005). Visualization of automated trust negotiation. In *Int. Workshop on Visualization for Cyber Security (VizSec)*, pages 65–74.

Yee, C. L., Chuan, L. L., Ismail, M., and Zainal, N. (2012). A static and dynamic visual debugger for malware analysis. In *Asia-Pacific Conference on Communications*, pages 765–769.

Yoo, I. (2004). Visualizing windows executable viruses using self-organizing maps. In *Int. Workshop on Visualization for Cyber Security (VizSec)*, pages 82–89, New York, NY, USA. ACM.

Zhuo, W. and Nadjin, Y. (2012). MalwareVis: Entity-based visualization of malware network traces. In *Int. Workshop on Visualization for Cyber Security (VizSec)*, pages 41–47. ACM.

Zupan, B., Holmes, J. H., and Bellazzi, R. (2006). Knowledge-based data analysis and interpretation. *Artificial Intelligence in Medicine*, 37(3):163–165.

List of Figures

List of Tables

Curriculum Vitae

Personal Data

Address	Markus Wagner Mainburgstraße 2 A-3202 Hofstetten-Grünau, Austria
Date of Birth	June 21^{th}, 1983 in St. Pölten, Austria
Email Address	markuswagner83@gmail.com

Education

Since Mar 2014	PhD studies in Computer Science at Vienna University of Technology PhD Thesis: *Integrating Explicit Knowledge in the Visual Analytics Process: Model and Case Studies on Time-oriented Data* Supervisors: Wolfgang Aigner, Daniel A. Keim
Sept 2011 to June 2013	MSc studies in Game Engineering and Simulation at University of Applied Sciences Technikum Wien Master's Thesis: *Parallelisierte Simulation einer Flüssigkeitsoberfläche mittels Masse-Feder-Systemen* Supervisors: Andreas Monitzer, Markus Schordan
Sept 2008 to July 2011	BSc studies in Industrial Simulations at St. Pölten University of Applied Sciences

	Bachelor's Thesis: *Simulationsinterface zum Empfang, Entschlüsseln und Verteilen von Luftraumüberwachungsdaten für ein dreidimensionales Visualisierungssystem* Supervisor: Otto Reichel
Sept 2006 to July 2007	Foreman for Industrial Electronics at TGA of the AK Lower Austria
Sept 2004 to June 2005	High-voltage Technician at TGA of the AK Lower Austria
Sept 2004 to June 2005	Management and qualification examination for the Electrical Industry at TGA of the AK Lower Austria
Sept 2003 to June 2005	Foreman for Electrical Engineering at TGA of the AK Lower Austria
July 1997 to Jan 2002	Electrician Apprentice at Schubert Elektroanlagen GmbH in connection with the theoretic education at Vocational School of Electrical Engineering Stockerau

---

Job Experience

Since Mar 2014	Research Associate at St. Pölten University of Applied Sciences IC\M/T Institute of Creative Media Technologies Media Computing Research Group
Aug 2013 to Feb 2014	System Programmer at Medizinkraft Solutions GmbH & Co KG Industrial image processing (marker less motion detection), 3D-modeling, software design and document creation
Oct 2012 to July 2013	IT-Technician at Constantia Teich GmbH

	IT-Infrastructure
Feb 2011 to July 2011	Practical training at Cassidian Air Systems (Germany) Simulation programming
Aug 2008 to Jan 2011	Assistant at St. Pölten University of Applied Sciences Simulations- and electronics laboratory
June 2007 to Aug 2008	Technical Employee at Schubert Elektroanlagen GmbH Project planning and controlling in the field of Energy Technology
Jan 2002 to May 2007	Electrician at Schubert Elektroanlagen GmbH Switchgear manufacturing, mounting and equipment commissioning
July 1997 to Jan 2002	Electrician Apprentice at Schubert Elektroanlagen GmbH

Publications

Please see:	http://mc.fhstp.ac.at/publications/author/Wagner https://scholar.google.at/citations?user=J-XPsgUAAAAJ&hl=de

Book

1. Wagner, M. (2013). *Simulation mittels Masse-Feder-Systemen: Parallelisierte Simulation einer Flüssigkeitsoberfläche mittels Masse-Feder-Systemen*. AV, Saarbrücken

Book Chapter

1. Wagner, M., Sacha, D., Rind, A., Fischer, F., Luh, R., Schrittwieser, S., Keim, D. A., and Aigner, W. (2017c). Visual Analytics: Foundations and Experiences in

Malware Analysis. In Othmane, L. B., Jaatun, M. G., and Weippl, E., editors, *Empirical Research for Software Security: Foundations and Experience*. CRC/Taylor and Francis. In press

2. Rind, A., Federico, P., Gschwandtner, T., Aigner, W., Doppler, J., and Wagner, M. (2017a). Visual analytics of electronic health records with a focus on time. In Rinaldi, G., editor, *New Perspectives in Medical Records: Meeting the Needs of Patients and Practitioners*, TELe-Health, pages 65–77. Springer

Journal Article

1. Wagner, M., Rind, A., Thür, N., and Aigner, W. (2017b). A knowledge-assisted visual malware analysis system: design, validation, and reflection of KAMAS. *Computers & Security*, 67:1–15

2. Rind, A., Aigner, W., Wagner, M., Miksch, S., and Lammarsch, T. (2015). Task Cube: A three-dimensional conceptual space of user tasks in visualization design and evaluation. *Information Visualization*, 15(4):288–300

Conference Paper

1. Luh, R., Schramm, G., Wagner, M., and Schrittwieser, S. (2017). Sequitur-based Inference and Analysis Framework for Malicious System Behavior. In *Workshop for Formal Methods in Software Engineering (ForSE), (ICISSP)*, pages 632–643, Porto, Portugal. SCITEPRESS Digital Library

2. Rind, A., Haberson, A., Blumenstein, K., Niederer, C., Wagner, M., and Aigner, W. (2017b). PubViz: Lightweight visual presentation of publication data. In *Proc. Eurographics Conf. Visualization (EuroVis) – Short Paper*. Accepted

3. Blumenstein, K., Niederer, C., Wagner, M., Schmiedl, G., Rind, A., and Aigner, W. (2016). Evaluating Information Visualization on Mobile Devices: Gaps and Challenges in the Empirical Evaluation Design Space. In *Proc. of the Workshop on Beyond Time And Errors (BELIV)*, pages 125–132, Baltimore, MD, USA. ACM

4. Kromer, L., Wagner, M., Blumenstein, K., Rind, A., and Aigner, W. (2016). Performance Comparison between Unity and D3.js for Cross-Platform Visualization on Mobile Devices. In *Proceedings of the Forum Media Technology*, pages 47–52. CEUR-WS

5. Wagner, M., Blumenstein, K., Rind, A., Seidl, M., Schmiedl, G., Lammarsch, T., and Aigner, W. (2016a). Native Cross-platform Visualization: A Proof of Concept

Based on the Unity3d Game Engine. In *Int. Conf. on Information Visualisation*, pages 39–44, Lisbon, Portugal. IEEE

6. Wagner, M., Rind, A., Rottermanner, G., Niederer, C., and Aigner, W. (2016b). Knowledge-Assisted Rule Building for Malware Analysis. In *Proc. of the Forschungsforum der österreichischen Fachhochschulen (FFH)*, Vienna, Austria. FH des BFI Wien

7. Blumenstein, K., Wagner, M., Aigner, W., von Suess, R., Prochaska, H., Püringer, J., Zeppelzauer, M., and Sedlmair, M. (2015b). Interactive Data Visualization for Second Screen Applications: State of the Art and Technical Challenges. In Schulz, H.-J., Urban, B., and Freiherr von Lukas, U., editors, *Proceedings of the International Summer School on Visual Computing*, pages 35–48, Rostock, Germany. Frauenhoferverlag

8. Blumenstein, K., Wagner, M., and Aigner, W. (2015a). Cross-Platform InfoVis Frameworks for Multiple Users, Screens and Devices: Requirements and Challenges. In *DEXiS Workshop on Data Exploration for Interactive Surfaces. In conjunction with ACM ITS*, pages 7–11

9. Wagner, M. (2015). Integrating Explicit Knowledge in the Visual Analytics Process. In *Doctoral Consortium on Computer Vision, Imaging and Computer Graphics Theory and Applications (DCVISIGRAPP)*, pages 9–18, Berlin. SCITEPRESS Digital Library

10. Wagner, M., Aigner, W., Haberson, A., and Rind, A. (2015a). Literature review in visual analytics for malware pattern analysis. In *Proc. of the Forschungsforum der österreichischen Fachhochschulen (FFH)*. FH Hagenberg

11. Wagner, M., Fischer, F., Luh, R., Haberson, A., Rind, A., Keim, D. A., and Aigner, W. (2015c). A survey of visualization systems for malware analysis. In *Eurographics Conf. on Visualization (EuroVis) State of The Art Reports*, pages 105–125. Eurographics

12. Wagner, M., Blumenstein, K., Wieländer, U., and Judmaier, P. (2015b). Genderorientierter Informatikunterricht: Plattformübergreifende Spieleentwicklung mit Unity3d. In Haag, J., Weißenböck, J., Gruber, W., and Freisleben-Teutscher, C. F., editors, *Game Based Learning - Dialogorientierung & spielerisches Lernen digital und analog: Beiträge zum Tag der Lehre an der FH St.Pölten am 15.10. 2015*, pages 23–32, St. Pölten, Österreich. ikon Verlag

13. Rind, A., Aigner, W., Wagner, M., Lammarsch, T., and Miksch, S. (2014). User Tasks for Evaluation: Untangling the Terminology Throughout Visualization Design and Development. In Lam, H., Isenberg, P., Isenberg, T., and Sedlmair, M.,

editors, *Proc. of the Workshop on Beyond Time And Errors (BELIV)*, pages 9–15. ACM

14. Wagner, M., Aigner, W., Rind, A., Dornhackl, H., Kadletz, K., Luh, R., and Tavolato, P. (2014). Problem characterization and abstraction for visual analytics in behavior-based malware pattern analysis. In *Int. Workshop on Visualization for Cyber Security (VizSec)*, pages 9–16. ACM