FAKULTÄT FÜR !INFORMATIK

Semantic Integration of Engineering Environments Using an Engineering Knowledge Base

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der Sozial- und Wirtschaftswissenschaften

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Mag. Thomas Moser
Matrikelnummer 0125850

am
Institut für Softwaretechnik und Interaktive Systeme

Betreuung:
Betreuer: Ao. Univ.Prof. Dr. Stefan Biffl
Zweitbetreuer: Univ.Prof. Dr. Oscar Pastor

Wien, 15.12.2009

Verfasser	Betreuer	Zweitbetreuer

Es ist zum Erstaunen, wie leicht und schnell Homogenität oder Heterogenität des Geistes und Gemüts zwischen Menschen sich im Gespräch kundgibt: An jeder Kleinigkeit wird sie fühlbar.

Arthur Schopenhauer

Danksagung

Zuallererst möchte ich mich bei meinem Betreuer ao. Univ. Prof. Dr. Stefan Biffl bedanken, dessen unermüdlicher Innovationsdrang und dessen konstruktives Feedback der vergangenen drei Jahre maßgeblichen Erfolg an der erfolgreichen Fertigstellung dieser Arbeit hat. Zusätzlich möchte ich meinem Zweitbetreuer Univ. Prof. Dr. Oscar Pastor von der Universität Valencia für die unkomplizierte Zusammenarbeit, sowie für die wichtigen neuen Impulse, die er dieser Arbeit gegeben hat, danken.
Außerdem möchte ich mich bei meinen Kolleginnen und Kollegen am Institut für Softwaretechnik und interaktive Systeme für die konstruktive Zusammenarbeit der vergangenen Jahre bedanken. Zu allererst bei meinem Kollegen und Sitznachbar Richard Mordinyi, außerdem bei Amin Anjomshoaa, Erik Gostischa-Franta, Matthias Heindl, Prof. Dr. Gerti Kappel, Monika Lanzenberger, Kamil Matousek, Marcus Mor, Andreas Pieber, Prof. Dr. Andreas Rauber, Alexander Schatten, Wikan Danar Sunindyo, Prof. Dr. A Min Tjoa, Dindin Wahyudin, Florian Waltersdorfer und Dietmar Winkler. Zusätzlich zur Zusammenarbeit am Institut, möchte ich auch Eva Kühn vom Institut für Computersprachen und Munir Merdan vom Institut für Automatisierungs- und Regelungstechnik für die konstruktive Zusammenarbeit auf Ebene der TU Wien danken. Im Bereich der Zusammenarbeit mit Industriepartnern möchte ich mich bei besonders bei Alexander Mikula (Frequentis), Heinz Roth (senactive) und Heinrich Steininger (logi.cals) für die praktisch relevanten Herausforderungen und Evaluierungen bedanken. Gleichzeitig möchte ich den vielen Studierenden danken, die in den letzten drei Jahren Praktika und/oder Diplomarbeiten in unserer Arbeitsgruppe absolviert haben, und ohne deren wertvolle Beiträge eine umfangreiche Arbeit in dieser Zeit unmöglich zu beenden gewesen wäre.
Neben dem Beitrag der universitären Umgebung am Erfolg dieser Arbeit, möchte ich auch den Beitrag meines privaten Umfelds würdigen. Zuallererst danke ich meiner Familie, meinen Eltern Elisabeth und Reinhard, meiner Schwester Christine und meinem Bruder Stephan für die Geduld und die Unterstützung, die sie mir in den vergangenen – nicht nur beruflich sondern auch privat leider nicht immer leichten – Jahren entgegen gebracht haben. Zu guter Letzt danke ich meinen Freundinnen und Freunden, die nicht nur während der letzten Jahre zu jeder Tages- und Nachtzeit für mich da waren, sondern auch alle meine Hochs und Tiefs gebührend miterleben durften bzw. mussten. Danke für Alles Alex, Andi, Basti, Benni, Caro, Gigi, Isi, Kathi, Markus, Niko, Peter, Pia, Sophie, Terry, Tomschi, und Vanni!

Eidesstattliche Erklärung

Mag. Thomas Moser
Skodagasse 23/14
1080 Wien

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.

Wien, am 15. Dezember 2009 ____________________________

Abstract

Software-intensive systems in business IT and industrial automation have become increasingly complex due to the need for more flexible system re-configuration, business and engineering processes. Systems and software engineering projects depend on the cooperation of experts from several engineering domains and organizations, who work in engineering environments with a wide range of semantically different terms, models, and tools that were not designed to cooperate seamlessly. Current semantic engineering environment integration is often ad hoc and fragile, making the evolution of tools and re-use of integration solutions across projects unnecessarily inefficient and risky.

When designing an engineering project environment, project managers and engineering domain experts need to semantically integrate a given set of engineering tools and project data models to allow efficient adaptations to new or changed requirements, as well as for re-use in new engineering projects. Current alternative solutions like standards for data models and tools in the development process, data-driven tool integration, and complete transformation between tool data work in principle, but pose their own challenges for engineering, such as inefficient and complex data access and query definitions, solutions which are not robust enough, or take considerable effort to develop and modify.

This work introduces the Engineering Knowledge Base (EKB) framework for engineering environment integration in multi-disciplinary engineering projects. The EKB stores explicit engineering knowledge to support access to and management of engineering models across tools and disciplines by providing a) data integration based on mappings between local and domain-level engineering concepts; b) transformations between local engineering concepts; and c) advanced applications built on these foundations, e.g., end-to-end analyses. As a result experts from different organizations may use their well-known tools and data models, and can, in addition, access data from other tools in their syntax.

Semantic integration research has focused on finding general approaches for schema integration, which can be used in many contexts. However, these general approaches do not take into account the specifics of a domain and therefore tend to be inefficient and often fail to solve specific problems that are hard to solve in general. In this work, we build on domain-specific knowledge of engineering processes, models and analyses to enable designing semantic integration methods and tools. Key contributions of this work are the industrial application and proof-of-concept of the proposed semantic integration approach, as well as design guidelines for semantic integration in the engineering domain.

The research results have been evaluated in two industrial application domains: distributed business systems and services and software-intensive production automation systems, regarding feasibility, effort, robustness, performance, scalability, and usability. The evaluation results indicate an effort reduction of more than 20% for re-use in new engineering projects and finding defects earlier in the engineering process.

Zusammenfassung

Software-intensive Systeme in der Informatik und der industriellen Automatisierungsbranche erleben in den letzten Jahren ein stetiges Ansteigen der Komplexität aufgrund höherer Flexibilitätsanforderungen. Experten aus verschiedenen Ingenieurswissenschaften verwenden typischerweise eine Vielzahl an semantisch unterschiedlichen Termen, Modellen und Werkzeugen, welche ursprünglich nicht für eine Zusammenarbeit entwickelt wurden.

Um eine Ingenieursumgebung herzustellen müssen Projektmanager und Experten der verschiedenen Ingenieurswissenschaften eine vorgegebene Menge an Ingenieurswerkzeugen und Datenmodellen semantisch integrieren, sowohl um flexibel auf neue oder geänderte Anforderungen reagieren zu können, als auch um derartige Integrationslösungen für ähnliche neue System- und Softwareentwicklungsprojekte weiterverwenden zu können. Aktuelle verwendete Ansätze, wie die Verwendung von Standards für Datenmodelle, datenbasierte Werkzeugintegration, oder vollständige Transformation zwischen Werkzeugdatenmodellen, funktionieren prinzipiell, stellen aber neue Herausforderungen wie das Beherrschen komplexer und ineffizienter Datenzugriffe, oder den robusten Umgang mit fehlerhaften Datenmodellen dar.

Diese Arbeit stellt das Engineering Knowledge Base (EKB) Framework vor, welches sich mit der Integration von Ingenieursumgebungen aus mehreren Ingenieurswissenschaften befasst. Die EKB speichert explizites Ingenieurswissen um Zugriff und Verwalten von Modellen werkzeugübergreifend und über die Grenzen verschiedener Ingenieurswissenschaften hinweg zu ermöglichen. Dazu speichert die EKB a) Verbindungen zwischen toolspezifischen und gemeinsamen Konzepten, und ermöglicht so b) die Transformation. Diese Grundlagen ermöglichen dann anschließend auch c) fortgeschrittene Anwendungen wie z.B. End-to-End Analysen. Dadurch ermöglicht die EKB die Beibehaltung bekannter Werkzeuge und Datenmodell, und erlaubt zusätzlich den Zugriff auf Daten anderer Werkzeuge in anerkannter Darstellungsform. Hauptbeitrag der Arbeit ist einerseits die industrielle Anwendung und eine Machbarkeitsstudie des vorgestellten generischen Ansatzes, andererseits eine Reihe von generischen Richtlinien für semantische Integration im Bereich der Ingenieurswissenschaften.

Die Ergebnisse dieser Forschungsarbeit wurden in zwei Anwendungsgebieten, einerseits verteilte betriebliche IT Systeme und Services und andererseits Software-intensive Produktionsautomatisierungssysteme, im Bezug auf Machbarkeit, benötigter Aufwand, Robustheit, Performanz, Skalierbarkeit und Benutzerfreundlichkeit evaluiert. Erste Ergebnisse der Evaluation zeigen dass einerseits für die Wiederverwendbarkeit in anderen System- und Softwareentwicklungsprojekten mit einer Aufwandsreduktion von mehr als 20% zu rechnen ist, und dass andererseits Fehler in System- und Softwareentwicklungsprozessen früher entdeckt werden.

Table of Contents

List of Figures

List of Tables

List of Listings

Chapter 1

1 Introduction

Software-intensive systems in business IT and industrial automation become increasingly complex due to the need for flexibility of business processes, system re-configuration, and engineering processes (Schäfer and Wehrheim 2007). Such systems and software engineering projects bring together experts from several engineering domains and organizations, who work in a **heterogeneous engineering environment** with a wide range of models, processes, and tools that were originally not designed to cooperate seamlessly. A core question is how to integrate data models across tools and domain boundaries. Current semantic engineering environment integration is often ad hoc and fragile, making the evolution of tools and re-use of integration solutions across projects risky (Halevy 2005; Noy, Doan et al. 2005).

In order to reach the common goal of developing software products in the engineering team, it is important to **share** the necessary **knowledge** for common work processes **between engineering domain experts** (Schäfer and Wehrheim 2007). However, this knowledge is often only implicitly available and therefore inefficient to share, resulting in time-consuming repetitive tasks; often it is hard or even impossible to create and maintain common shared knowledge repositories. A **method and platform for making expert knowledge explicit and efficiently shareable** is needed in order to support quality and project managers in their data analyses based on engineering knowledge and concrete data in the engineering tool models, which is currently achieved using inefficient or fragile approaches.

Weakly integrated software engineering tools, methods, and processes lead to higher delays and risks when putting a system in operation and avoidable downtime during operation and maintenance. Typical system integration challenges are 1.) **technical heterogeneities**, e.g., tools from different sources may use a range of technologies that become expensive and error-prone to integrate in traditional point-to-point ways; and 2.) **semantic heterogeneities**, e.g., project participants may use different terms for common concepts in the engineering and application domains. In this thesis, the focus lies on managing semantic heterogeneities, while the handling of technical heterogeneities which is by now well achieved in related work, is the basis for this thesis, and therefore summarized in the related work.

There are two aspects on systematic engineering environment integration which are important in the context of this thesis: 1. the **design and creation of the engineering project environment**; and 2. **engineering project work** using the engineering environment.

The first aspect deals with the allocation of a tool box to **integrate a given set of engineering tools and project data models**, while keeping the solutions flexible for adaptation to new or changed requirements, as well as for re-use in new projects. The major problem here is the difficulty to re-use hardcoded integration or transformation scripts for other projects, i.e., the so-called "ossification" of IT solutions (Aaen 2003). The most basic approach would be the identification of general engineering concepts which can be re-used in other projects (Doan, Noy et al. 2004; Doan and Halevy 2005). This however rises issues such as a) which technology in the area of knowledge management or semantic web would support representation and re-use of these general engineering concepts; and b) how this extended functionality can be provided without the need to change existing and well-accepted notations and tools, as well as without the need to agree on a common data model beforehand. The second issue is especially important for this thesis, since typical data-driven integration approaches focus on a **common data model** agreed on by all participants, however this is **hard** to achieve **or even impossible** for multi-organizational engineering projects, as well as hard to re-use for other projects with different project partners.

The second aspect deals with the feasibility of supporting particular engineering process tasks in a given engineering environment architecture that go across tool and discipline boundaries. The major problems here are the currently high effort needed to **perform tasks** like change impact analyses or model checks **across domain boundaries**, risks of defects, or the need to use several tools that require re-entry of data in several tools. The most basic approach would be to extend the functionality of the existing tools, data models and interfaces, however this approach has to be as transparent in daily usage, i.e., invisible for normal application, or provide good usability and/or expert roles for special applications.

The target audiences of this thesis belong to the following three stakeholder classes: engineering domain experts, knowledge beneficiaries and ontology experts. Engineering domain experts, e.g., engineers, want to effectively and efficiently follow their engineering process. However, often problems like **high effort to perform typical engineering tasks** like change impact analyses or model checks across domain boundaries or risk of defects hinder them in following their engineering process. Knowledge beneficiaries, e.g., project or quality managers, want to monitor, control and improve engineering processes. This intention is often complicated by the needed high effort for performing cross domain process analyses, as well as by the impossibility to easily re-use these analyses in other projects. Finally, ontology experts, e.g., semantic technology expert with general engineering know-how, want to design and validate semantic solutions in an engineering application context.

An engineering environment integration approach needs to be effective and efficient, i.e., there should be few errors in the integrated knowledge as well as in the knowledge tasks, and the integration effort should be at least equal or considerably lower than the effort needed for achieving similar results with different approaches. Additionally, the approach needs to be robust against poor data quality and changes in the environment,

as well as flexible regarding the effort needed to adapt the solution to changed requirements.
An existing alternative solution is the **usage of standards** (e.g., RUP, sysML) for platforms, data models, modeling languages and tools in the development process (Kruchten 2000; Weilkiens 2008). This works well, if the standard is defined in an early phase of the project and if all project partners adhere to the standard, however, it is **hard to define and maintain** standards for cross-domain engineering, and even harder or nearly impossible for a larger number of project partners to agree on a standard, which usually takes longer than the time horizon of the project. Another alternative solution is the use of a common repository for collecting and storing the data of the single participants (Bernstein and Dayal 1994). This is a typical solution for modern **data-driven tool integration** which well solves the challenges of persistency and versioning of data, but poses new challenges since the stored data often is **hard to access and query** (Hohpe and Woolf 2004; Trowbridge, Roxburgh et al. 2004).The ultimate alternative solution is the **complete transformation** between data models of tools (Assmann, Dörr et al. 2005), i.e., the translation of engineering model parts of one tool for work in another tool. While the advantage of this solution is the seamless cooperation between project partners using well-known and established tools and notations, the feasibility of this approach is **hard to verify** and the **effort** required for establishing the needed transformations is **considerably high**.

This work proposes the **Engineering Knowledge Base** (EKB) framework for supporting **engineering environment integration in multi-disciplinary engineering projects**. Since standards are hard to apply in projects where experts from different organizations participate, who have invested into different kinds of local standards or approaches, these experts may use their well known local tools and data model, and additionally can access data from other tools in their local syntax. The EKB is located on top of a common repository and stores explicit engineering knowledge to support access and management of engineering models across tools and disciplines by providing a) data integration by exploiting **mappings** between local and common engineering concepts; b) **transformations** between local engineering concepts by following these mappings; and c) **advanced applications** using these foundations, e.g., end-to-end analyses. Only a selection of the most relevant data elements to achieve interaction between engineering tools and experts is stored in the EKB in order to avoid time-consuming and effort-intensive transformations of full engineering models. As a result experts from different organizations may use their well known local tools and data model, and additionally can access data from other tools in their local syntax.
By now, Semantic Integration research has focused on finding general approaches for schema integration which can be used in many contexts. However, these general approaches do not take into account the specifics of a domain and therefore tend to be inefficient and often fail to solve specific problems that are hard to solve in general. In this work, we build on domain-specific knowledge of engineering processes, models

and analyses to enable designing semantic integration methods and tools. Since the engineering project participants by now already work together, they already use common knowledge for their project tasks. By using the EKB framework we make this existing knowledge explicit and machine-understandable, and therefore can automate on project level tasks that build on this explicit and machine-understandable knowledge. Key contributions of this work are the **industrial application** and **proof-of-concept** of the proposed semantic integration approach, as well as **design guidelines for Semantic Integration** in the engineering domain.

The research results have been evaluated in **two industrial application domains**, distributed business systems and services and software-intensive production automation systems, regarding effort, feasibility, performance, scalability, robustness and usability. The evaluation is based on prototypes for a set of specific use cases of the two industrial application domains, as well as on empirical studies of beneficiary roles as proof-of-concept. Major results of this work are the **feasibility of the EKB framework**, i.e., the process, method and tool support is usable and useful across engineering domains, as well as better accuracy, effectiveness and efficiency. In addition, **defects are found earlier** in the engineering process, resulting in risks like errors or inconsistent entries in data models being mitigated earlier and more efficiently. Initial evaluation results indicate an effort reduction of more than 20% for re-use in new engineering projects and finding defects earlier in the engineering process. In addition, the engineers found the method useable and useful; furthermore, new kinds of analysis could be performed easily.

The remainder of this work is structured as follows: Chapter 2 summarizes related work on technical system integration, semantic heterogeneity, model-driven architecture, ontologies, semantic integration, and semantic web services. Chapter 3 describes the research approach by identifying the research issues, specifying the research methods and introducing the application scenarios. Chapter 4 introduces the Engineering Knowledge Base (EKB) framework, describes usage scenarios and the generic framework architecture, as well as specifies the evaluation aspects. Chapter 5 summarizes the results of applying the EKB framework to the System-Wide Information Sharing (SWIS) application scenario, while chapter 6 summarizes the results for the EKB framework application to the other application domain, namely to the Production Automation domain using the Simulation of Assembly Workshops (SAW) application scenario. Chapter 7 presents the results of the EKB framework evaluation and discusses these results with regard to the specified research issues. Finally, chapter 8 concludes this thesis and gives an outlook on future research perspectives.

Chapter 2

2 Related Work

This chapter summarizes related work on technical system integration, semantic heterogeneity, model-driven architecture, ontologies, semantic integration and semantic web services. In the first section, technical system integration challenges, types and architectures are introduced. Then, in the second section, problems, challenges and origins of semantic heterogeneity are explained, as well as solution approaches such as schema matching. In the next section, a brief introduction to model-driven architecture is given. In the fourth section, ontologies are introduced, including methods for Ontology Alignment and some examples for typical usage scenarios for ontologies in Software Engineering. In the fifth section, the research field of Semantic Integration is introduced, the different available approaches are classified and explained, and in addition application scenarios for the usage of ontologies for Semantic Integration are given. Finally, the sixth section summarizes related work on Semantic Web Services and service matchmaking approaches.

2.1 *Technical System Integration*

This section summarizes related work on technical system integration by giving a short overview followed by shortly explaining technical system integration challenges, types and architectures. Technical integration typically deals with technical heterogeneities, e.g., tools from different sources may use a range of technologies that become expensive and error-prone to integrate in traditional point-to-point ways, and is the basis for dealing with semantic heterogeneities.

2.1.1 Overview

System integration is the task to combine numerous different systems to appear as one big system. There are several levels at which system integration could be performed (Balasubramanian, Gokhale et al. 2006), but there is so far no standardized integration process that explains how to integrate systems in general.
System integration can require changes (Hohpe and Woolf 2004) in the actual business policy of a company not only due to the emerging communication needs between multiple computer systems but also due to the communication requirements which have

to be established between business units. Therefore, integration can have strong implications on the company as improper integration solutions can lead to considerable inefficiency. Another integration challenge is to keep sufficient control over the involved applications as in most cases integration developers have only limited control over these applications, e.g., legacy systems.

Typical integration solutions focus only on either the heterogeneity on service level or the heterogeneity on network level. In order to cope with technological heterogeneity on service level a homogeneous middleware technology approach (Gail, David et al. 2003) could be used for syntactical transformation between services, while the semantic heterogeneity of services could be addressed by means of a common data schema (Halevy 2005). Heterogeneity on network level may be addressed by using so called adapters transforming messages between each used combination of middleware technologies. However, in order to provide an effective continuous integration solution in this environment, both integration levels (i.e. service and network level) need to be addressed in a mutual way.

The derived limitations for such kinds of integration approaches are on the one hand the need for a common data schema (Halevy 2005), which is often a hard and time consuming procedure, if not even impossible in integration scenarios with several different stakeholders. On the other hand, the need for integration over heterogeneous middleware technologies with different APIs, transportation capabilities, or network architecture styles implies the development of static and therefore inflexible wrappers between each combination of middleware technologies, and thus increases the complexity of communication. Traditional approaches for integration of business services can be categorized (Chappel 2004) into: Hub and spoke vs. distributed integration and coupled vs. separated application and integration logic. In the following, using current technology concepts for each category a brief discussion about their advantages and disadvantages with respect to the described scenario is given.

Application servers (Gail, David et al. 2003) are capable of interoperating through standardized protocols, but tightly couple integration logic and application logic together. Additionally, as the name suggests a server based architecture style is used for integration and as such has proven to be inconvenient for the scenario. Traditional EAI brokers (Chappel 2004), some of them built upon application servers, use a hub-and-spoke architecture. This approach on the one hand has the benefit of centralized functions such as the management of business rules or routing knowledge, but on the other hand does not scale well across business unit or departmental boundaries, although it offers clear separations between application, integration and routing logic. Message-oriented Middleware (Piyush and Michael 2005) is capable of connecting application in a loosely coupled manner but requires low-level application coding intertwining integration and application logic. The resulting effort and complexity of implementing an integration platform with the support for any kind of existing middleware technologies and protocols therefore is considerably high. To enable transparent service integration, the Enterprise Service Bus (ESB) provides the infrastructure services for message exchange and routing as the infrastructure for Service Oriented Architecture (SOA) (Mike and Willem-Jan 2007). It provides a distributed integration platform and clear separation of business logic and integration logic. It offers routing services to navigate the requests to the relevant service provider based on a routing path specification. Routing may be (Chappel 2004) itinerary-based, content-based, conditional-based defined manually (Satoh, Nakamura et al. 2008) or dynamic (Xiaoying, Jihui et al. 2007). In both cases the drawback is the minimal support for considering all functional and non-functional requirements of all service connections in the system. Dynamic configuration focuses mainly on creating a route for a special business case. Using manual configuration, a system integrator has to rely on his expertise, thus the high number of service interactions may get complex and the configuration error-prone. This may lead to routes that are configured in a way in which their influence on other business interactions is not fully known. As a consequence, business interactions may mutually violate their non-functional business requirements, such as message delivery within a specific time frame otherwise the message may be still useful but not up-to-date any more. Additionally, dynamic configuration may not cope with e.g. node failures fast enough due to missing routing alternatives, therefore possibly violating the same type of non-functional business service requirements.

2.1.2 Integration Challenges

The integration of heterogeneous systems is not an easy task. There are a lot of challenges which must be handled to reach the aim of a functioning coherent integrated system (e.g. applications are running on different platforms and are located on different places). Current system integration technologies partially provide great techniques for dealing integration tasks, but implicate also numerous limitations. Gorton et al. (Gorton, Thurman et al. 2003) defined some challenges for system integration regarding the integration of different applications, which must be solved. These challenges can occur as a result of ever-changing technologies applications are developed with and focuses on the requirements pretended for the realization of an integration solution.

2.1.2.1 Scale

Due to the high amount of digital data sources and the increasing number of modern applications depending on rapid access to multiple data sources, scalability of integration solutions to handle numerous different data sources is a crucial task. Integration solutions should be able to rapidly merge different data from disparate data formats to provide a transparent access to this data from different applications. Therefore modern integration solutions must have a look to scalability to handle numerous data source and have to provide a flexible transformation mechanism to convert data from one format to another (Gorton and Liu 2004).

2.1.2.2 Dynamic configuration

Integration techniques often must handle different heterogeneous data sources by means of adapters. An adapter converts data from one specific format to another specific data format. But often no appropriate adapter for a data source is available and so a new adapter must be created to achieve the needed tasks. Furthermore the development of an adapter is not as easy as it sounds. It could lead to high costs for development and it is important to consider the time an adapter needs to convert the relevant data of a data source. If there are many requests for accessing the data source, the adapter must be built with main focus on performance. For system integration purposes modern integration technologies have to minimize cost and time factors for the integration of data sources. At best an integration technology automatically creates suitable adapters for the integrated data sources to establish access to the data from participating applications (Kramer and Magee 1985).

2.1.2.3 Finding Relevant Data

Finding relevant data out of a mass of data for a specific application is becoming a real problem due to the increasing amount of integration solutions with big infrastructures and enormous existing data. Most traditional data sources do not possess with semantic search functions where data of interest can be indicated and easy located Modern integration techniques should be able to automatically find the relevant data from the data sources by extracting semantics of the data sources and linking the appropriate data to the participating applications (Gorton, Thurman et al. 2003).

2.1.3 Integration Types

System integration techniques focus on different levels to combine participating heterogeneous system. There are multiple types of system integration techniques which differ at the level where the integration is done. Basically two groups of integration types exist. The first group of integration techniques focuses on the design of an integration layer (Hohpe and Woolf 2004; Trowbridge, Roxburgh et al. 2004). This group contains the following types:

- ***Business Process integration:*** Process integration is an orchestration of interactions between multiple systems by defining a business process model outside of the applications.

- ***Portal integration:*** Portal integration represents an overall user-interface of multiple applications so that the user gets a comprehensive view of all the underlying applications.

- ***Entity aggregation:*** Entity aggregation extends the portal integration so that not only users but also applications can deal with the integration by providing a unified data view.

The second group of integration techniques focuses on the mechanism how the systems are connected together (Hohpe and Woolf 2004; Trowbridge, Roxburgh et al. 2004). This group consists of the following types:

- ***Data integration:*** Data integration is an approach to make the high amount of data, containing in different data sources, accessible so that all other systems can use all the data.

- ***Functional integration:*** By means of functional integration the participating systems are combined together by providing special interfaces. Via this interfaces the systems can access among each other to use the underlying data source.

- ***Presentation integration:*** With presentation integration all participating applications interacts with the host application via the user interface. Applications can access the functionality of another application through a presentation byte stream by simulating users input and get the required information back by reading the output from the display.

2.1.3.1 Data Integration

The data integration mechanism integrates systems at the logical data layer. The idea is to provide an overall interface for accessing different data sources of multiple applications. In an enterprise many applications exist which keep large amounts of information in data stores like flat files or databases. Other applications which want to use the information connect directly to these data stores. An advantage of data integration is that the applications which held the data sources must not be changed to provide an interface on where the other applications get access to the underlying data. Another advantage is that a user, who needs some data from different data sources, must not care about the location of the wanted data. The user does not need to know which application stores the specific data. The data integration approach gives users the ability

to specify *what* data they want, instead of determining *how* to obtain the data (Levy 2000). But the integration solution within data integration uses a strong binding to the data structure of the underlying applications. This means that in case of changing one data model of any application, also the integration solution has to be changed to meet the modified specifications and to access the data source furthermore. In general data integration is easy to develop, because no application logic of the integrated applications is used (Hohpe and Woolf 2004; Trowbridge, Roxburgh et al. 2004).
Organizations have to care about the possibilities to share the data between the different heterogeneous applications. There are some techniques for such integration. Data integration can be realized by means of shared databases, maintain data copies, and file transfer. Each of these techniques gives an answer to the question, on how to integrate multiple applications that are not designed to work in correlation and are not constructed to change information among each other.

2.1.3.2 Functional Integration

Functional integration is also known as application integration and integrates systems at the logical business layer. This means that the business logic of an application which keeps data in data stores is shared, so that other applications can use the data store across the application without direct access to it. The individual applications will be connected via interfaces and specifications allocated by the integrated application. But often some of the participating legacy applications don't provide any interfaces or specifications and an integration of such applications is hard or rather not possible (Hohpe and Woolf 2004).
To realize an integration of multiple applications by means of functional integration, two preconditions are needed. First: availability of the business function which is used for the integration in the business logic of the source application. If this condition is not given the source application must be modified to implement the needed business functionality. Second: remote access to the source applications API is needed. If an application only supports local API calls and middleware must be created which receives remote API calls and transforms them into local calls, accepted by the application. The Implementation of a functional integration solution can be realized by means of distributed objects, message-oriented middleware or service-oriented architectures (Trowbridge, Roxburgh et al. 2004).

2.1.4 Integration Architectures

For the integration of different systems there exist several ways how these systems could be connected together to build one big corporate system. Generally there exist three main possible integration architectures to establish the integration of systems. The difference between these architectures is the way how senders and receivers are connected together. In the following section the three various architectures are described, starting with the basic Point-to-Point connection, following by the more complex Hub connection and finally the Bus connection (Hohpe and Woolf 2004; Trowbridge, Roxburgh et al. 2004).

2.1.4.1 *Point-to-Point architecture*

The Point-to-Point communication is the simplest way of connecting participating systems among each other (see Figure 1). Each system has respectively a direct connection to all other systems. To establish the communication some precondition has to be given. The first requirement to send a message from sender to receiver is that the sending system must know where the receiving system is located because a sender could be connected to more than one system. Furthermore each involved system can only deal with specific message formats and so the sender must transform a message from one format into another format that could be handled by the receiver. That is a big disadvantage of such integration architecture. Each system needs a separate integration solution to all other involved systems. Generally each system in a Point-to-Point integration has a direct connection to all other systems and requires a specific message transformation for any connection. If systems supported message format changes, the message transformer of all associated systems that communicate with the changed entity must be updated.

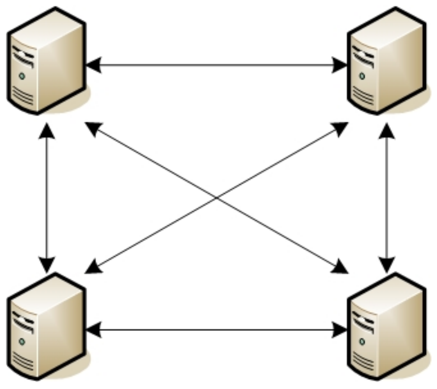


Figure 1: Point-to-Point integration architecture.

Finally a Point-to-Point integration is easy to handle if just a few systems are connected together, but for more and more systems the effort to maintain such integration increases very fast. With *n* participating systems $n \cdot \frac{n-1}{2}$ different integration solutions exist. So it is obviously that a Point-to-Point integration is quite reasonable for small organizations with few systems (Hohpe and Woolf 2004; Trowbridge, Roxburgh et al. 2004).

2.1.4.2 *Hub/Broker architecture*

This kind of integration architecture connects all involved systems via a central point, namely the hub. Figure 2 shows the basic design of the hub integration architecture.

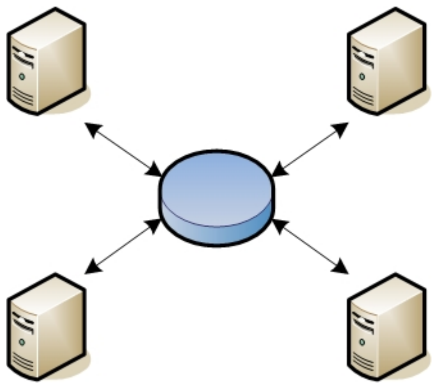


Figure 2: Hub integration architecture.

The hub controls the whole communication between senders and receivers. All participating systems do not have to care about the location of the receiver and do not have to know the message format supported by the receiver. A sender forwards his message to the hub and the hub takes the message, transforms the message and sends the transformed message to the correct receiver. This technique is often called the "hub & spoke" integration architecture (Hohpe and Woolf 2004).
The hub architecture follows a broker pattern. The task of a broker is to decouple sender systems from the receiver systems by coordinating the communication between them. Systems are loosely coupled if only few common variables are used by the systems or if the common variables are less addicted to other influencing factors (Pinelle and Gutwin 2005). By using hub architectures, the single systems are separated from each other and so a loose coupling takes place. The decoupling of the participating systems is achieved by three main tasks (Trowbridge, Roxburgh et al. 2004):

- ***Routing:*** Routing is the task of determining the location of the receiving system of a message and performing the routing via direct or indirect communication.

- ***Endpoint registration:*** Endpoint registration is used by the involved systems to register themselves with the broker. After registration the system is public and can be found by other systems.

- ***Transformation:*** Each participating application uses its own specific data format. To make it possible that applications can communicate with each other, the messages must be converted to the right format. The transformation is the mechanism to convert a message from one format to another format.

2.1.4.3 Bus architecture

In this architecture, all participating systems are connected via a special component the so-called bus (see Figure 3). The easiest communication mode of a bus is the broadcast communication. A system sends its message to the bus and the bus forwards the message to all other connected systems. Therefore the systems themselves must decide if a message is addressed to them or not (Hohpe and Woolf 2004; Trowbridge, Roxburgh et al. 2004).

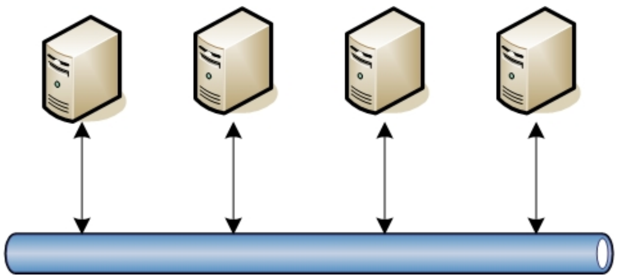


Figure 3: Bus integration architecture.

Generally, an integration bus provides a common communication mechanism to connect heterogeneous systems. To achieve the integration, the involved systems must follow some agreements. Trowbridge et al. (Trowbridge, Roxburgh et al. 2004) defined three criteria for the participating systems to be able for a connection to the bus:

- ***Message schema:*** All connected systems must support the correct structure of the messages.
- ***Command message:*** Command messages are used for reliable invocation of a procedure provided by another application. A command message is a normal message with a command in it.
- ***Shared Infrastructure:*** To build a bus architecture a predefined infrastructure is needed for sending messages from sender to receiver, e.g. message router, publish/subscribe mechanism. These different types of shared infrastructures are described below.

2.2 *Semantic Heterogeneity*

This section summarizes related work on semantic heterogeneity by explaining challenges and origins of semantic heterogeneity, as well as by presenting solution approaches such as schema matching. The management of semantic heterogeneities using semantic integration techniques is the main focus of this thesis; hence this section provides basic definitions of semantic heterogeneities.

2.2.1 Challenges and Origins

When database schemas for the same domain are developed by independent parties, they will almost always be quite different from each other. These differences are referred to as semantic heterogeneity. Semantic heterogeneity also appears in the presence of multiple XML documents, web services and ontologies. Or, more general, whenever there is more than one way to structure a body of data. In order for multiple data systems to cooperate with each other, they must understand each other's schema. Without such understanding, the multitude of data sources amounts to a digital version of the Tower of Babel (Halevy 2005).

Enterprises today are increasingly facing data management challenges that involve accessing and analyzing data residing in multiple sources, such as database systems, legacy systems, ERP systems and XML files and feeds. For example, in order for an enterprise to obtain a single view of customer data, they must tap into multiple databases. Similarly, to present a unified external view of their data, either to cooperate with a third party or to create an external facing web site, they must access multiple sources. There are many reasons for which data in enterprises resides in multiple sources. First, many data systems were developed independently for targeted business needs, but when the business needs changed, data needs to be shared between different parts of the organization. Second, enterprises acquire many data sources as a result of mergers and acquisitions (Halevy 2005).

The problem of reconciling schema heterogeneity has been a subject of research for decades, but solutions are few. The fundamental reason that makes semantic heterogeneity so hard is that the data sets were developed independently, and therefore varying structures were used to represent the same or overlapping concepts (Bergamaschi, Castano et al. 1999; Doan and Halevy 2005). In many cases, we are trying to integrate data systems that were developed for slightly (or vastly) different business needs. Hence, even if they model overlapping domains, they will model them in different ways. Differing structures are a byproduct of human nature people think differently from one another even when faced with the same modeling goal. From a practical perspective, one of the reasons that schema heterogeneity is difficult and time consuming is that it requires both domain and technical expertise: you need a person that understands the business meaning of each of the schemas being reconciled and people skilled in writing transformations (e.g., SQL or XQuery experts). While schema heterogeneity is challenging for humans, it is drastically more challenging for programs. A program is only given the two schemas to reconcile but those schemas are merely symbols. They do not capture the entire meaning or intent of the schemas; those are only in the minds of the designers (Doan and Halevy 2005).

It is often argued that the way to resolve semantic heterogeneity is though standard schemas. However, experience has shown that standards have very limited success and only in domains where the incentives to agree on standards are very strong. Even then, while data providers may share their data using a standard, their own data systems still employ their original schemas (and the cost of changing those systems is prohibitive). Hence, semantic heterogeneity needs to be resolved at the step where the data provider exposes its data to its counterparts (Halevy 2005).

2.2.2 Solution Approaches

Resolving schema heterogeneity is inherently a heuristic, human assisted process. Unless there are very strong constraints on how the two schemas you are reconciling are different from each other, one should not hope for a completely automated solution. The goal is to reduce the time it takes a human expert to create a mapping between a pair of schemas, and enable them to focus on the hardest and most ambiguous parts of the mapping (Rahm and Bernstein 2001).
A solution approach is to query multiple data sources in real-time. While the users of these systems still see a single schema (whether relational or XML), queries are translated on the fly to appropriate queries over the individual data sources, and results are combined appropriately from partial results obtained from the sources. Consequently, answers returned to the user are always based on fresh data. In any data sharing architectures, reconciling semantic heterogeneity is the key (Doan and Halevy 2005). No matter whether the query is issued on the fly, or data is loaded into a warehouse, or whether data is shared through web services or in a peer-to-peer fashion, the semantic differences between data sources need to be reconciled. Typically, these differences are reconciled by semantic mappings. These are expressions that specify how to translate data from one data source into another in a way that preserves the semantics of the data, or alternatively, reformulate a query posed on one source into a query on another source. Semantic mappings can be specified in a variety of mechanisms, including SQL queries, XQuery expressions, XSLT scripts, or even Java code (Halevy 2005).

2.2.2.1 Schema Matching

As would be expected, people have tried building semi-automated schema matching systems by employing a variety of heuristics (Rahm and Bernstein 2001). Below a few of these as well as their limitations are listed. The process of reconciling semantic heterogeneity typically involves two steps. In the first, called schema matching, correspondences between pairs (or larger sets) of elements of the two schemas that refer to the same concepts or objects in the real world are identified. In the second phase, these correspondences are exploited to create the actual schema mapping expressions.

The following classes of heuristics have been used for schema matching (Halevy 2005):

- ***Schema element names:*** Element names (e.g., table and attribute names) carry some information about their intended semantics. Hence, by looking at the names, possibly stemming the words first we can obtain clues for the schema matcher. The challenges involved in using names are that the use of synonyms is very common as is the use of *hypernyms* (words that are specializations or generalizations). Furthermore, we often see that same word being used with different meanings (*homonyms*). In addition, we often see abbreviations and concatenations of words appearing in element names (Gangemi, Guarino et al. 2003).

- ***Data types:*** Schema elements that map to each other are likely to have compatible data types, but this is certainly not a rule. However, in many schemas the data types are underspecified (e.g., CDATA for XML). In practice, considering data types is a useful heuristic for ruling out certain match candidates.

- ***Data instances:*** Elements from two schemas that match each other often have similar data values. Similarities can arise in several ways: (1) values drawn from the same small domain, e.g., brands of cars or names of countries, (2) significant occurrences of the same values, or (3) patterns of values (e.g., phone numbers). Data instances are extremely useful when available, but one cannot rely on their availability.

- ***Schema structure:*** Matching elements in a schema are typically related to other related schema elements. For example, in an object-oriented hierarchy, it's often the case that if two classes match each other, then the children of these classes will also (at least partially) match. However, relying on such a heuristic can be very fragile, and one of the main challenges is to find an initial match that drives the similarity of its neighbors.

- ***Integrity constraints:*** Considering integrity constraints on single attributes or across attributes can be useful for generating matches. For example, if two attributes are known to be keys in their respective schemas, then that provides additional evidence for their similarity.

While each of these heuristics is useful, experience has shown that taking any of them in isolation leads to a brittle schema matching solution. Hence, research has focused on building systems that combine multiple heuristics (Rahm and Bernstein 2001; Do and Rahm 2002; Doan, Madhavan et al. 2004). Despite these ideas, commercial products often rely on completely manual specification of semantic mappings. They help by offering visual interfaces that enable designers to draw the lines between elements of disparate schemas, while the details of the mapping can often be generated on the back end. These tools already save a significant amount of time, but they do not suggest initial mappings to the designer.

2.2.2.2 Leveraging Past Experience

One of the fundamental reasons that the schema matching solutions described above are brittle is that they only exploit evidence that is present in the two schemas being matched, ignoring past experience (Halevy 2005). These schemas often lack sufficient evidence to be able to discover matches. However, looking more closely at schema matching tasks, it is evident that these tasks are often repetitive. Specifically, we often find that we repeatedly map schemas in the same domain into a common mediated schema. A human expert, after seeing many schemas in a particular domain, is able to map schemas much faster because he/she has seen many variations on how concepts in the domain are represented in schemas. The challenge, therefore, is to endow the schema matcher with the same capabilities: leverage past experience. For example, once the system has been given several mappings in the domain of used cars, it should be able to predict mappings for schemas it has not seen before. As it sees more schemas in a particular domain, its predictions should become more accurate, and it should be more robust in the presence of variations (Noy, Doan et al. 2005).
The paradigm of learning from past experience of performing schema matching tasks is only in its infancy. It is interesting to take a step back and consider what one can learn from the past in this context. We assume the past is given to us as a collection of schemas in a particular domain, mappings between pairs of schemas in that collection, and to the extent possible, data instances. The schemas can come from anywhere, and can involve very closely related domains, not necessarily modeling the same data. We often refer to such a collection of schemas as a corpus, in analogy to the use of corpora of documents underlying Information Retrieval (IR) and web-search engines (Frakes and Baeza-Yates 1992). Of course, while the corpora in IR involve collections of words, here we are managing semantically richer elements, such as schemas and their instances. The goal of analyzing a corpus of schemas and mappings is to provide hints about deeper domain concepts and at a finer granularity.

2.3 *Model-Driven Architecture*

This section gives an overview of the Model-Driven Architecture techniques. To understand this comprehensive topic, the first section presents some elementary explanations about models and meta models. Then the layered architecture of MDA is

figured and finally the benefits of using the MDA technique compared to a traditional software development process are listed.
Model-Driven Architecture (MDA) developed by the Object Management Group (OMG) is an approach for modern software development, by using a layered architecture for software system specifications and development (Malveau 2000). The defined system specifications describe the software system at different abstraction levels. Each level provides a special view of the system. MDA is used for separating business and application logic from the underlying platform technologies[1]. In other words, MDA is the separation of the specification of system functionality from the actual implementation of the specified functionalities (Miller and Mukerji 2001). All defined specifications are expressed as models.

2.3.1 Models and Meta Models

This chapter describes two fundamental parts used in Model Driven Architecture, models and meta models. *“A model is a coherent set of formal elements describing something (for example, a system, bank, phone, or train) built for some purpose that is amenable to a particular form of analysis.”* (Mellor, Clark et al. 2003) Another definition for a model comes from Stachowiak (Stachowiak 1973). He specifies that a model is essentially a scale, detailedness and/or functionality shortened and accordingly abstract representation of the original system. In short, a model is a replication of the real world. It must be noted, that a model is just a representation of an original system and not a copy. For example if someone builds a true to detail object according to an original one so that the replication equals the original in every little detail, the replication is a copy and not a model. It is obviously that a model has to concentrate and represent just some particular details of the original. Models are a basic part in Model Driven Architecture.
Selic (Selic 2003) has appointed five key characteristics an engineering model must conform to a certain degree:

- ***Abstraction*** is the most important characteristic of a model. A model is always a shortened representation of a system that it specifies. Abstraction means that the model is not a one to one replication of a system, but reflects only the relevant properties of a regarding system. This means that irrelevant details are unattended in the model. Therefore abstraction is almost the only method to deal with the complexity of an always increasing sophisticated functionality of software systems.

[1] http://www.omg.org/mda

- ***Understandability*** is also an important characteristic for a model. If a model is suppressed in a language which needs much intellectual knowledge to understand it, a model will provide no benefit. A model must present their information in a form (e.g. a notation) that it could be understood without significant intellectual effort. For that reason a model is a good model when not much intellectual effort is needed to understand the content provided by the model.

- ***Accuracy:*** Useful models must be accuracy. This means that a model must provide the modeled system in such a way that it offers a concise representation of the system's features the model is interested in.

- ***Predictiveness:*** With models it should be possible to exactly predict the interests the modeled system focuses on without non suggesting properties, by experimentations or formal analysis. Predictiveness relies on the accuracy characteristic of a model and the modeling form.

- ***Inexpensiveness:*** The last characteristic a model should possess is inexpensiveness. The construction and analysis of a model should be essentially cheaper than the construction and analysis of the system itself. It would be very inefficient and uneconomical for building models if the modeling of a system costs more than the creation of the actually system.

There are some more concepts which occur in relation to the MDA approach, Platform-Independent Models (PIM) and Platform-Specific Models (PSM). A PIM represents a formal specification of systems structure and characteristic, without including technical details. The Platform-Independent Models are constructed for an implementation on different platforms. A PSM specifies how to realize the defined functions of a PIM on a specific platform. It represents enough details and information (e.g. software architecture) to generate a complete coded application (Mellor and Balcer 2002). But it is still defined as a model. Out of a Platform-Specific Model the code for the whole implementation of a software system can be created (Miller and Mukerji 2001).

2.3.2 Model Driven Architecture Layered Model

The Model Driven Architecture approach is based on a layered architecture. Generally, the MDA architecture consists of four layers: the M3-layer which represents a meta-meta model, the M2-layer, representing a meta model, the M1-layer, depicting a concrete model and the M0-layer which illustrates the reality. In Figure 4 these four layers of the MDA architecture are displayed (Gašević, Djurić et al. 2006).

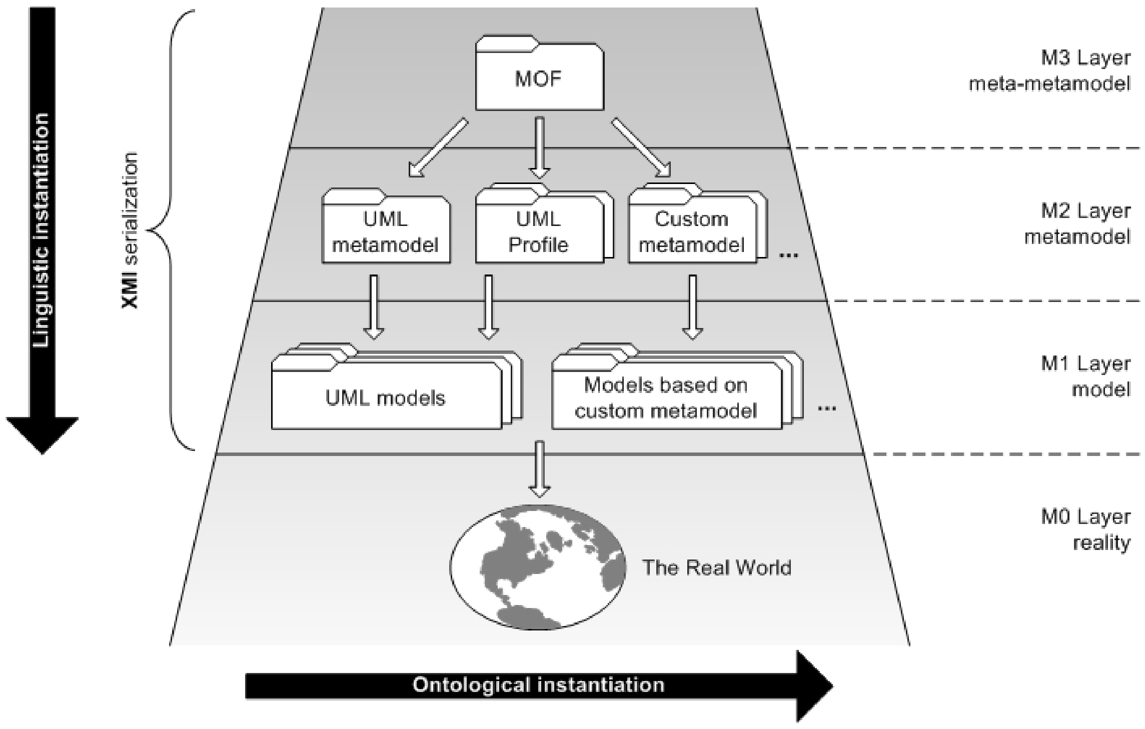


Figure 4: The MDA four-layer architecture (Gašević, Djurić et al. 2006).

The meta-meta model layer (M3-layer) is the topmost level of the MDA architecture. This layer is represented by the Meta Object Facility (MOF). MOF builds an industry standard environment to export models from one application and import it to another application, transferred over a network and transformed into different formats. MOF represents a basis to define other modeling languages, like UML (Unified Modeling Language), IDL (Interface Definition Language) used in CORBA or CWM (Common Warehouse Meta model). Even MOF is described in MOF and can be subdivided into EMOF (Essential MOF) and CMOF (Complete MOF). EMOF is a simple language for defining meta models and is useful for meta modelers. CMOF is an extension for EMOF with support and management of metadata. In generally the M3-layer provides a specification of modeling languages and is primarily used to express meta models of the M2-layer (Seidewitz 2003).
The meta model layer (M2-layer) contains the actual meta models (model of model) defined by the MOF. This layer represents an instance of the M3-layer. UML is one of numerous meta modeling languages. The Unified Modeling Language technique is used to help system architects, software engineers and software developers by providing tools for better analysis, design and implementation of software-based systems or miscellaneous modeling challenges. The model layer (M1-layer) contains representations of the real world in terms of models. Such a model is an instance of meta models defined in the M2-layer (e.g. UML model of a software system). The reality layer (M0-layer) represents an instance of the models defined in the M1-layer.

This layer contains actual objects of the real world, like persons, buildings, etc (Miller and Mukerji 2001; Mellor, Kendall et al. 2004).

2.3.3 Benefits of Model Driven Architecture

Developing software by means of the Model Driven Architecture approach provides some improvements of the software development process. Kleppe et al. (Kleppe, Warmer et al. 2003) have researched the benefits of MDA and categorized them into four classifications: Productivity, Portability, Interoperability, and Maintenance and Documentation. These benefits are explained in relation to a traditional software development life cycle with their containing problems.

2.3.3.1 Productivity

Specifications between the requirement, analysis, design and implementation phase are represented in terms of text and diagrams. This means that phase 1 through 3 produce many text documents and diagrams for the later software implementation. The written specifications are lacking maintained and so they present no exact mapping of the created implementation. This becomes a serious problem due to permanent changes at the code level. Introducing the changes to the documents and diagrams is very time-consuming and therefore hard to maintain (Kleppe, Warmer et al. 2003; Mellor, Kendall et al. 2004).
Model Driven Architecture attempts to solve the problem of creating mass of documents and diagrams by using a Platform-Independent Model (PIM). By means of a PIM the determined requirements and capabilities for a software product are represented in the form of a model. This primarily created PIM will be later on transformed into a Platform-Specific Model (PSM) which comprehends specific information about the underlying platform. A PSM conforming to another platform can be easily generated out of the defined PIM. But the extensive creation of the PIM is the only disadvantage of using MDA for reaching a higher productivity. It looks very easy, but much effort is needed to produce a correct PIM for further processing. Often only a high skilled specialist can achieve the creation of the abstract PIM. But once the PIM was created, the productivity benefits to generate PSMs for different platforms are very high if tools to automatically transform the specified PIM into a PSM are used (Miller and Mukerji 2001).

2.3.3.2 Portability

Portability describes the possibility to use the same program or model on different platforms without modification. In a Model Driven Architecture portability can be obtained by using Platform-Independent Models (PIMs). A PIM is defined in a platform independent manner and therefore can be used on different platforms without any modifications. A key benefit of portability in MDA with a PIM is, that independent of new developed platform technologies the created PIM can be used furthermore. With a specific transformation tool, according to the new platform technology, the PIM is transformed into a functioning PSM without altering the original PIM (Frankel 2003; Mellor, Kendall et al. 2004).

2.3.3.3 Interoperability

Interoperability deals with the problem that a specific system should be able to interact with other existing systems developed in another technology. It is crucial that the different systems support a common working to gain a result. The multiple generated PSMs for different platforms out of one common PIM may have particular similarities and therefore some correlations, so-called bridges in MDA. The different PSMs are not able to directly communicate among each other, but by means of the bridges a communication can be established. A bridge transforms the concepts according to one platform into the concepts according to the other participating platform. Within MDA, interoperability is achieved by additionally generating the required bridges between the generated PSMs (Miller and Mukerji 2001; Kleppe, Warmer et al. 2003).

2.3.3.4 Maintenance and Documentation

In a traditional software development process the numerous created documents are often very hard to maintain. After creation of the source code out of the requirements the documents are neglected. If changes in the source code are made, the documents are often not updated to meet the altered requirements. Therefore in a traditional software development process changes must be updated multiple times on different places. Within MDA, changes are only updated on a single place in the PIM. Out of the PIM the different PSMs are generated and out of a PSM the source code is generated. Therefore each source code is an exact representation of the PIM and no inconsistencies between source code and specifications can occur. Generally a PIM illustrates a form of a high-level documentation used by any underlying software system (Frankel 2003; Kleppe, Warmer et al. 2003).

2.4 *Ontologies*

The proposed semantic integration approaches use ontologies as major modeling method, therefore in this section the term ontology will be explained. An overview on the basic concepts of ontologies is given. Furthermore the main operational areas of ontologies are described. Then, an overview of the available languages for describing ontologies is presented. After focusing on guidelines for creating ontologies, the research field of ontology alignment is described in detail. Finally, some applications of ontologies in Software Engineering are presented.

2.4.1 Definition and Overview

First of all, a definition of an ontology is presented: *"An ontology is a formal, explicit specification of a shared conceptualization. A 'conceptualization' refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. 'Explicit' means that the type of concepts used, and the constraints on their use are explicitly defined. 'Formal' refers to the fact that the ontology should be machine readable, which excludes natural language. 'Shared' reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group."* (Studer, Benjamins et al. 1998)
In general, ontologies are a main part of the semantic web technology and are used like a knowledge representation of the real world or only part of it. Ontologies are formal models of a specific application domain, and primarily used to facilitate the exchange and partitioning of knowledge. More precisely, an ontology is a data model that represents a set of concepts within a domain and their relationships. The word ontology has its origin from the Greek words ontos (=being) and logos (=word). From a philosophical point of view an ontology refers to the subject of existence, that is the study of being as such (Gašević, Djurić et al. 2006). Gruber (Gruber 1993) defines an ontology as an explicit specification of a conceptualization. Where a conceptualization illustrates an abstract, simplified picture of the world used for representation and designation. Each knowledge representation follows a certain degree of conceptualization, either explicitly or implicitly. Moreover ontologies can effectively support software development processes, primarily by providing a continuous data model (Calero, Ruiz et al. 2006).
According to Powers (Powers 2003) ontologies consist of four main components: classes, relations between classes, properties of classes, and constraints on relationships between the classes and properties of the classes. But additionally an ontology also consists of individuals which represents instances of concrete types. A class represents concepts of a domain, for example the concept "vehicle" with his specifications: car, motorcycle, bus, etc. (a set of objects with common properties). A relation represents an association between class concepts of the domain. A property represents an attribute to

describe objects in the ontology. And the last component, a constraint defines statements for a relation between classes or properties that cannot be formally expressed by the other main components.
In the following, the main components of an ontology (individuals, classes, attributes and relations) are described in detail (Gruber 1993; Gruber 1995):

- ***Individuals (instances):*** The individuals build the basic components of an ontology and are similar to object instances in the object oriented programming. Individuals represent concrete types like house or car, and additionally more discrete types like numbers or words.

- ***Classes (concepts):*** The classes represent abstract groups, sets, or collections of objects and are similar to abstract objects in the object oriented programming. Classes can contain other classes or individuals or a combination of classes and individuals. The single ontologies can vary among each other on the conditions they support. They distinguish whether classes can contain other classes, or whether a class can belong to itself and so on. Also restrictions can be made to prevent that an ontology can have an invalid state, like whether an individual inherits from two disjunctive classes.

- ***Attributes:*** Attributes represent properties, features and characteristics of an object in an ontology. An attribute consists of at least a name and a value, whereas the value can be a normal value type and also a complex data type.

- ***Relations:*** Relations specifies how the various objects are related together. A relation between objects in the ontology is described by means of attributes. Together, all the specified relations characterize the semantic of an ontology. Generally different types of relations exist: the subsumption relation (is-subtype-of, is superclass-of, whereas the objects are members of a common group of objects), the is-a relation (tree structure with child and parent objects, whereas each object is a child of a parent) and the meronymy relation (part-of relation).

2.4.2 Ontology Languages

Ontologies can be expressed in different languages. Gómez-Pérez et al. (Goméz-Pérez, Fernandez-Lopez et al. 2003) divide the logical ontology languages into traditional ontology languages and web-based ontology languages. Whereas traditional languages are developed in the early 1990s for artificial intelligence purposes, the web-based ontology languages are developed at the beginning of the web-age to use the characteristics of the internet. In this section we focus on the web-based languages shown in Figure 5.

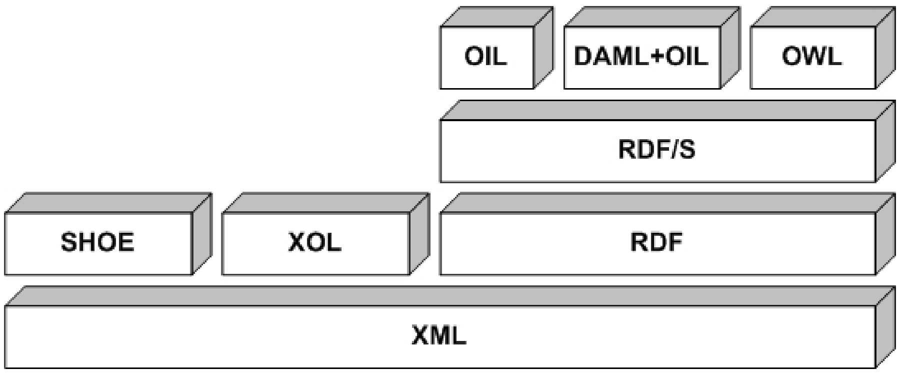


Figure 5: Web-based ontology languages (Goméz-Pérez, Fernandez-Lopez et al. 2003).

The syntax of the web-based ontology languages is based on common web markup languages like HTML or XML. In the following enumeration the web-based ontology languages shown in the diagram are described shortly (Goméz-Pérez, Fernandez-Lopez et al. 2003):

- ***SHOE:*** The Simple HTML Ontology Extension language uses frames and rules and was developed as an extension to the HTML markup language. With SHOE it is possible to describe a webpage in a semantic manner.

- ***XOL:*** The XML-based Ontology Language was developed to include primitives based on the OKBC (Open Knowledge Base Connectivity) protocol. OKBC is a protocol to access knowledge bases stored in different knowledge systems.

- ***RDF:*** The Resource Description Framework language is used for defining web-resources in a semantically way. RDF was developed by the World Wide Web Consortium (W3C).

- ***RDF/S:*** The RDF Schema extends the Resource Description Framework and represents an easy language to specify domain-ontologies. With RDF/S the declarations defined in RDF can be structured hierarchically into classes and instances. Furthermore it is possible to precisely specify the relations between the particular properties. RDF/S builds a basis for the next three described ontology languages: OIL, DAML+OIL and OWL.

- ***OIL:*** The Ontology Inference Layer adds a frame-based knowledge representation to the underlying RDF/S and supports formal semantics provided by Description Logics (Fensel, Van Harmelen et al. 2001).

- ***DAML+OIL:*** The DARPA Agent Markup Language is a communication language for software agents (Rebstock and Paulheim 2008) and builds in combination with OIL the basis for OWL. DAML+OIL uses an object oriented approach and therefore it is designed to specify the structure of a specific domain in terms of classes and properties (Horrocks 2002).

- ***OWL:*** The Web Ontology Language is a semantic markup language used to create ontologies constructed in a formal representation language. Unlike of just providing information to humans, ontologies written in OWL can be used by applications to process the content of information[2]. OWL is best suitable for the description of relations between classes, properties and other individuals (Gašević, Djurić et al. 2006). There exist three different types of OWL which differ in the capability of expression, OWL Full, OWL DL and OWL Lite. OWL Full provides all OWL language constructs and additionally offers the use of RDF constructs. OWL DL is a subtype of the OWL language constructs with some restrictions (e.g. a class must not be an instance of another class) and without support for RDF constructs. OWL Lite represents a minimal subset of the OWL language construct with several restrictions and was developed as easy to implement language.

2.4.3 Designing Ontologies

In this section design criteria for the development of ontologies are described. Such criteria become very crucial, because if we represent something of the real world in the form of an ontology, it is essential to make suitable design decisions. For the development of well designed ontologies a set of objective criteria is needed, which corresponds to the scope of the resulting items. Therefore Gruber (Gruber 1995) appointed five main criteria and principles that have to be considered for the creation of ontologies. They are significant for ontologies used for knowledge sharing and interoperation between applications in a shared manner. These criteria serve as guiding principles and help to evaluate the developed ontology design (Noy 2004).

- ***Clarity:*** In general, an ontology has to clearly represent the intended sense of the environment it is used for. The ontology must be specified in an objective manner and should not depend on social or computational impacts. Furthermore an ontology must be completely defined. Not only the essential capabilities but also additional sufficient capabilities are preferred to get a complete definition and not just a partial definition about the environment. These definitions contained in an ontology should be described with formal languages (McGuire, Kuokka et al. 1993).

[2] http://www.w3.org/TR/owl-features

- ***Coherence:*** It is crucial to develop an ontology in a way that it is coherent. Coherence means, that ontologies should support various implications which conform to the definitions. Therefore it is necessary that the specified conventions are logically conforming to each other. But the term coherence is not only limited to inferences which should satisfy the definitions, it should also relate to any concepts that are described in an informally way. Such informal described concepts are documents and samples specified within a formal language. If a derived concept out of the specified conventions does not conform to the definitions, the ontology is not coherent (Gruber 1995).

- ***Extendibility:*** This criterion focuses on the possibility of further development and enhancement of an ontology. An ontology must be designed for arbitrary expandability, and therefore should provide a conceptual basis for later appending of anticipating tasks. It is crucial that the implementation of an ontology is designed for featuring a monotonically extension. Generally, within an existing ontology new items should be added by using the available vocabulary of the ontology without altering the previous containing definitions. If it is not possible that a new item is specified in the same scheme as the underlying ontology, the ontology must be able to deal items written in another format. But the original scheme should not be modified (McGuire, Kuokka et al. 1993).

- ***Minimal encoding bias:*** An ontology should be designed in a decoupled manner, in other words, the conceptualization should not depend on a specific encoding format. The design of an ontology should not match only one particular case of notation or implementation, if so, an encoding bias exist. Because of the reuse of developed ontologies the encoding bias must be as small as possible (Gruber 1993; Gruber 1995).

- ***Minimal ontological commitment:*** To satisfy the purposed knowledge sharing tasks, an ontology needs to fulfill the minimal ontological commitment. On the other side for a versatile usage of the ontology it is crucial that the ontology requires as few assumptions about the underlying modeled world as possible. Therefore a basic ontology, based on a minimal ontological commitment, can be used from many different parties for many different models due to the individual configuration and instantiation of such an ontology. It is always advisable to minimize the ontological commitment of ontologies by defining only elementary conditions of the represented knowledge to allow the most models using such a minimized ontology (Gruber 1995).

2.4.4 Ontology Alignment

This section summarizes related work from the research field Ontology Alignment. The first subsection gives an overview and defines the used terms, the second subsection introduces different methods and techniques, the third subsection deals with similarity measurements used for the identification of possible alignments, the fourth subsection introduces a generic ontology mapping process, and finally the fifth subsection describes three different ontology alignment tools (FOAM, PROMPT, GLUE) in more details.

2.4.4.1 Definition

It is difficult to find concrete definitions of the term ontology alignment. There are also terms like matching, mapping, merging, which are mostly used synonymous in literature. Ehrig has done a differentiation and definition of these terms (Ehrig 2007). In the following the most important and needed terms for this work will be defined. In general ontology alignment, ontology mapping and so on is used to connect partially different ontologies of one certain domain and overcomes therewith the problem of semantic heterogeneity. Ontology alignment tries to find a corresponding entity in one ontology for each entity of another ontology, with the same or at least a similar meaning (see Figure 6).

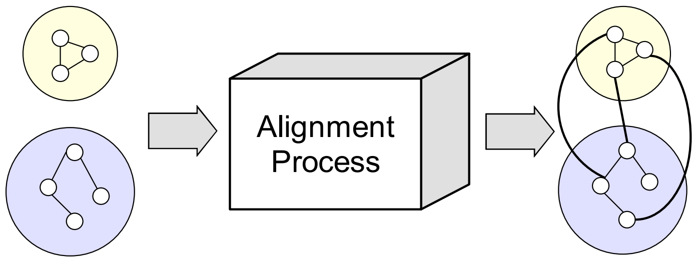


Figure 6: Ontology Alignment (Ehrig 2007).

Due to the fact that the terms matching and mapping are often used as synonyms for alignment and vice versa, they will also be regarded. Ehrig sees the difference between alignment and mapping as follows: *"Whereas alignment merely identifies the relation between ontologies, mappings focus on the representation and the execution of the relations for a certain task."* (Ehrig 2007) Ontology mapping is used to transform instances of one ontology into instances of another. The instructions, how this has to be done, are called mapping axioms, which are stored apart from the ontology. A mapping does not change the ontologies and often this can only be done in one direction. A typical use case for mapping is a query of an user formulated for one ontology representation, which will be translated or rewritten and forwarded to other ontologies.

The same happens to the answers on the way back. Another example for the need of ontology mapping is instance data sharing (Ehrig 2007).
Ontology matching handles the search for two corresponding entities, which do not have to be the same, but they must have a certain degree of similarity. *"Matching corresponds to our definition of general alignment, however, where a fixed relation between the aligned entities expresses the kind of match."* (Ehrig 2007) Hence for matching there has to be one specific kind of relation between two entities. Ontology Merging, however, concentrates on merging ontologies with overlapping domains (see Figure 7). Two or more ontologies will be merged into one ontology, which will replace the others.

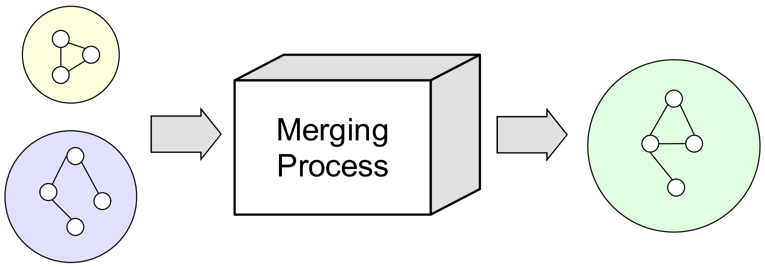


Figure 7: Ontology Merging (Ehrig 2007).

Ontology integration deals with integrating one or more ontologies into an existing one. This does not assume that something has to be merged. Furthermore the integrated ontologies will not be changed but at most extended. This makes sense if the ontologies derive from different domains.

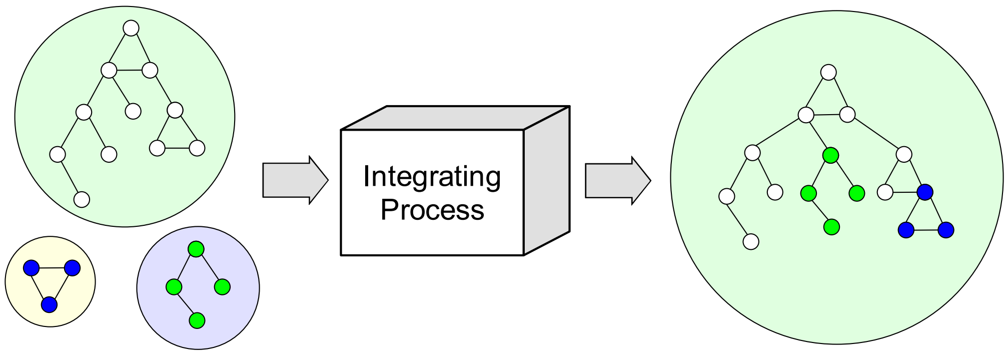


Figure 8: Ontology Integrating (Ehrig 2007).

2.4.4.2 *Methods and Techniques*

To align two or more ontologies, one has to discover connections between the entities (concepts, relations or instances) of the ontologies. These entities can be equal, similar or different. According to Noy (Noy 2005), alignments between ontologies can be detected by using information sources like:

- A common reference ontology (upper ontologies, background knowledge)
- Lexical information: String normalization (upper and lower Case, blanks, delimiters); String distance (Hamming-distance, edit distance); Language-based (Stemming); Semantic-based (WordNet); Soundex; Thesaurus
- Ontology structure
- User input
- External resources (Wordnet, synonym databases, dictionaries)
- Prior matches

The following diverse methods are used to find alignments (Noy, Doan et al. 2005):

- Heuristic and Rule-based methods (FOAM, Prompt): Structure analysis and lexical analysis methods (Ehrig and Sure 2005)
- Graph analysis (Anchor Prompt): Ontologies are seen as graphs and corresponding sub graphs are compared (Noy and Musen 2001)
- Machine-learning (GLUE): Statistics of data content; Using multiple learners; Using instance and value information (Doan, Madhavan et al. 2004)
- Probabilistic approaches: Combining results produced by heuristic-based mappings
- Reasoning, theorem proving

Furthermore Ontology Alignment or Mapping uses techniques, which have been developed for Schema matching (XML schemas, relational schemas etc.). The main difference between schemas and ontologies is that schemas do not provide semantics for their data.

2.4.4.3 Similarity

Similarity is a very important issue for ontology alignment, because most of the approaches in this field are based on similarity to align entities. It is used to compare ontologies respectively sets of entities. A comparison results into a numeric value, which states how similar two elements are. There are three layers of similarity, the data layer, the ontology layer and the context layer, which build upon each other (Ehrig 2007). Additionally there is the domain knowledge (see Figure 9).

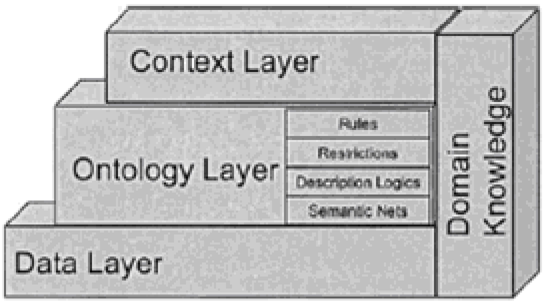


Figure 9: Similarity Layers (Ehrig 2007).

On the data layer only simple or complex data types, like integer and string, are considered for the comparison of entities. However, on the ontology layer semantic relations between the entities are regarded for comparison. On the context layer the external context of the use of the entities is important. The domain-specific knowledge can be situated on every layer and at this point the domain-specific vocabulary is the decisive factor. In the following some examples of the measures of these layers will be given (Ehrig 2007).

- ***Equality:*** Sometimes entities need to be equal, for example data values that are used as identifiers.
- ***Syntactic Similarity:*** The edit distance of two strings states how many atomic actions (i.e. addition, deletion, replacement, moving position) have to be done to change one string into the other.
- ***Distance-Based Similarity for Numeric Values:*** The arithmetic difference between numeric values is used to compute the similarity.
- ***Dice coefficient:*** Sets of entities are compared via the overlap of the sets individuals
- ***Label Similarity:*** String similarity, for example syntactic similarity, is used to compare labels, which are human identifiers for entities. Also dictionaries and databases for synonyms are used to compare labels, but in case of homonyms, failures are very probably.
- ***Extensional Concept Similarity:*** Two concepts are similar, if their instances are similar.

2.4.4.4 A Generic Ontology Mapping Process

Based on the fact that there has been defined a generic mapping process, which is said to subsume all the other approaches, it will be described first (Ehrig and Staab 2004).

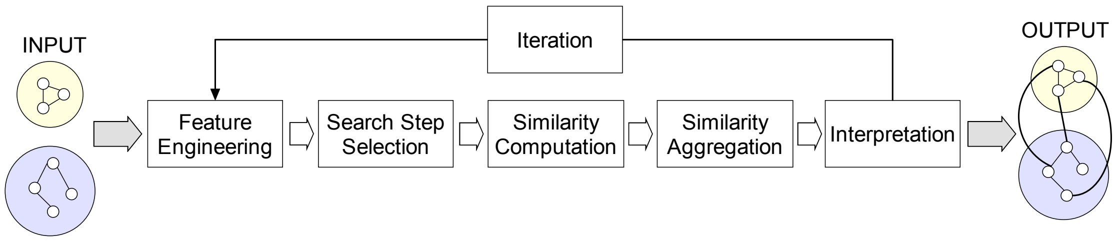


Figure 10: General mapping process (Ehrig and Staab 2004).

First of all two ontologies are needed as input. In the first step, feature engineering, features will be selected, which describe a specific entity (concept, attribute, relation). The following features of ontologies are used to detect alignments (Noy 2004):

- Concept names and descriptions
- Class hierarchy (relationships)
- Property definitions (domains, ranges, restrictions)
- Instances of classes
- Class descriptions

After that it is possible to restrict the search space by choosing the entities for a comparison. For the next step, similarity computation, similarity functions, as described in section 2.4.4.3, for strings, objects and sets of objects, as well as analysis of dissimilarity and so on are applied. Then the similarity values for a candidate pair of entities have to be aggregated to get one single value. These values will be used for mapping the entities of the ontologies. There are several possibilities like thresholds, relaxation labeling or combining structural and similarity criteria. After these steps it is possible to iterate over the whole process, for a better using of the structure of ontologies, because similarities of related entities are able to influence similarities of other entities (Ehrig and Staab 2004; Sure, Ehrig et al. 2006).

2.4.4.5 Ontology Alignment Tools

Generally there are two different types of tools for working with ontologies, ontology development tools and ontology alignment, mapping or merging tools. One common

development tool is Protégé[3] (see also section 4.1.1.4). Protégé is a Java-based free, open source ontology editor and knowledge-base framework, where ontologies can be modeled via the Protégé-Frames or the Protégé-OWL editors. There are many plug-ins available, which range from visualization to mapping tools. Today there are many approaches for ontology mapping or merging. There are console- and web-based tools as well as tools with graphical user interface. They reach from completely manual to fully automatic processes. In the majority of cases ontology mapping is done manually, although this is a very time and effort consuming work. Hence there are more and more semi-automatic ontology mapping approaches, which try to support users by making suggestions or providing visualizations.
In the following three different approaches for ontology mapping are described. Each approach applies a different method. FOAM is a heuristic based tool, which applies different similarity functions for detecting alignments. PROMPT also uses heuristics, but has an additional function called Anchor-PROMPT, which analyses the structure of the graph to find alignments. And GLUE is a machine learning approach, which originates from the research area of schema mapping.

2.4.4.6 FOAM

FOAM, Framework for Ontology Alignment and Mapping, is based on NOM, Naïve Ontology Alignment, and was developed by Ehrig and Sure at the University of Karlsruhe (Ehrig and Sure 2005; Sure, Ehrig et al. 2006). It is a fully or semi-automatically framework for aligning two or more ontologies. It has been implemented as a Java application (console or programming environment) and also a web service is available. It works with OWL-DL ontologies and uses similarity computation to find alignments. Foam is based on the general alignment process and applies heuristic measures, more precisely a wide range of similarity functions, to compute similarities of labels, structure and instances. Similarity functions for strings, objects and sets of objects and also equality are implemented , which can be defined as defined in (Ehrig and Staab 2004):

- ***Object Equality:*** based on existing logical assertions, especially assertions from previous iterations:
- ***Explicit Equality:*** checks whether a logical assertion already forces two entities to be equal:
- ***String Similarity:*** measures the similarity of two strings on a scale from 0 to 1 based on Levenshtein's edit distance, *ed* (Ehrig and Staab 2004):

[3] http://protege.stanford.edu

- ***SimSet:*** for many features it must be determined to what extend two sets of entities are similar. Multidimensional scaling measures how far two entities are from all other entities and assumes that if they have similar distances to all other entities, they must be very similar:

After computing the similarities, they are aggregated. In the next step a threshold is applied to reduce similarity failures and bijectivity is accomplished. Iteration over the process is needed, because this allows using already computed pairs. FOAM starts the process with the basic comparison methods based on labels and string similarity and thereafter all functions are used.
The following extensions have been implemented in FOAM (Sure, Ehrig et al. 2006):

- ***QOM – Quick Ontology Mapping (Ehrig and Staab 2004):*** Improves the efficiency of the alignment of bigger ontologies, by restricting the amount of compare pairs. Only very similar pairs or pairs close by already finished alignments are permitted.

- ***APFEL (Ehrig, Staab et al. 2005):*** APFEL is an approach to solve the problem of the selection of features by machine learning. There are feature combinations, which are not useful for the remaining process. Machine learning is used to eliminate these combinations and thus improve the alignment results. Therefore an adequate amount of training data is needed.

- ***Interactive Integration (Sure, Ehrig et al. 2006):*** It is possible to interact with the user via questioning to minimize the effort. The user will be only asked for candidate pairs, which are highly cross-linked within the ontology.

- ***Adaptive Integration (Sure, Ehrig et al. 2006):*** This allows parameters of the process to be automatically chosen. Bigger ontologies need other specifications than small ones. Therefore the process can be used for diverse tasks.

2.4.4.7 PROMPT

PROMPT is a semi-automatic approach for ontology merging and mapping. It has been developed by Noy and Musen at Stanford University in 2000 and is available as a plug-in for Protégé (Noy and Musen 2000). Later on in literature it is called iPROMPT. It needs two ontologies as input and produces a single merged ontology as output by making suggestions and guiding the user through the process. It also detects inconsistencies, which occur due to user actions, and is able to suggest possible solutions. During the process of merging, it logs the identified mappings to create a declarative mapping specification between the source ontologies.

Figure 11 illustrates the flow of the iPROMPT algorithm. First an initial list of matches based on class names is created. After that the user can choose an operation out of the suggestion list of iPROMPT or do an operation manually. iPROMPT provides explanations like why an operation has been suggested to be done first or why it has been moved. After choosing an operation iPROMPT will execute the operation, perform automatic updates, find conflicts and present new suggestions to the user. An evaluation has revealed that human experts followed 90 % of PROMPT's suggestions (Noy and Musen 2000). The original PROMPT is a heuristic-based tool. It works with lexical comparison of entity labels. As similarity measure only equality is applied. The entities with identical labels are presented to the user successively. There is no iteration necessary because the similarity computation does not build on previously computed alignments (Noy and Musen 2000; Ehrig 2007).

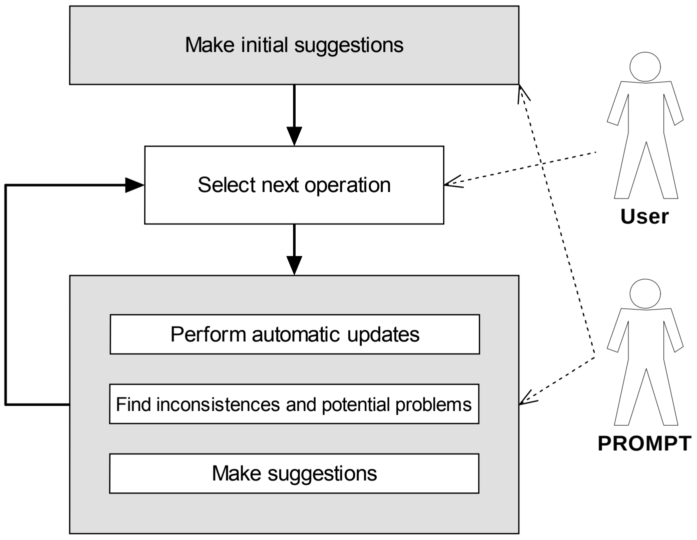


Figure 11 iPROMPT Algorithm (Noy and Musen 2003).

iPROMPT works with simple lexical-distance measures, to detect similar labels. But it is also designed for other algorithms to be easily plugged in, like WordNet to find synonyms, the Foam algorithm etc. (Noy and Musen 2003). Anchor-PROMPT analyses the graph structure of the ontologies. It traverses paths between anchor-points. Anchor-points are entities, which have been identified as equal, because their labels are identical. They can be chosen by the user manually or generated automatically. New alignments are detected along the paths by comparing the labels to find similar ones, which will be suggested to the user. Pairs with the same position have higher similarity values. With Anchor-PROMPT iteration is needed to recalculate the corresponding similarities after the user's choice to present new suggestions (Noy and Musen 2001; Noy and Musen 2003; Ehrig 2007).

2.4.4.8 GLUE

GLUE has been developed by Doan and colleagues (Doan, Madhavan et al. 2004), who already have done research in schema mapping. It is a semi-automatic approach that implements machine learning techniques for ontology mapping. It needs a large number of instances for learning and it is not possible to align relations and instances directly. The architecture of GLUE is made up of three parts: the Distribution Estimator, the Similarity Estimator and the Relaxation Labeler.
GLUE pays only attention to the taxonomy of ontologies and checks every possible concept pair. The Distribution Estimator takes the tree structure and the data instances as input to find similar concepts. This is done by computing the joint distribution of two concepts by applying machine leaning techniques to the instances. There are different types of information like instance names, value formats, etc., which can be used by learners for predictions. Therefore a multi-strategy learning approach is used (Doan, Madhavan et al. 2004).

2.4.5 Ontologies in Software Engineering

The emerging field of semantic web technologies promises new stimulus for Software Engineering research. However, since the underlying concepts of the semantic web have a long tradition in the knowledge engineering field, it is sometimes hard for software engineers to overlook the variety of ontology-enabled approaches to Software Engineering.
Happel and Seedorf (Happel and Seedorf 2006) propose a simple classification schema that allows a better differentiation among the various ideas of using ontologies in Software Engineering. The Ontology Driven Architecture (ODA) note at W3C serves as a starting point to elaborate a systematic categorization of the approaches and to derive more clearly defined acronyms (Tetlow, Pan et al. 2005). Happel and Seedorf propose two dimensions of comparison to achieve a more precise classification. First, they distinguish the role of ontologies in the context of Software Engineering between usage at run-time and development time. Second, they look at the kind of knowledge the ontology actually compromises. Here, they distinguish between the problem domain that the software system tries to tackle, and infrastructure aspects to make the software or its development more convenient. Putting these two dimensions together, the result is the matrix shown in Figure 12. The four categories for ontology usage in Software Engineering are (Happel and Seedorf 2006):

- ***Ontology-driven development (ODD)*** subsumes the usage of ontologies at development time that describe the problem domain itself. Prime example are the approaches in the context of MDD, see section 2.4.5.1.

- ***Ontology-enabled development (OED)*** also uses ontologies at development time, but for supporting developers with their tasks. For example, requirements engineering (see section 2.4.5.2) or test case generation (see section 2.4.5.3) can be put in here.

- ***Ontology-based architectures (OBA)*** use ontologies as a primary artifact at run-time. The ontology makes up a central part of the application logic. Business rule approaches are an example for this kind of application.

- ***Ontology-enabled architectures (OEA)*** finally, leverage ontologies to provide infrastructure support at the run-time of a software system. An example are semantic web services (see section 2.6), where ontologies add a semantic layer on top of the existing web service descriptions, adding functionality for the automatic discovery, matching and composition of service-based workflows.

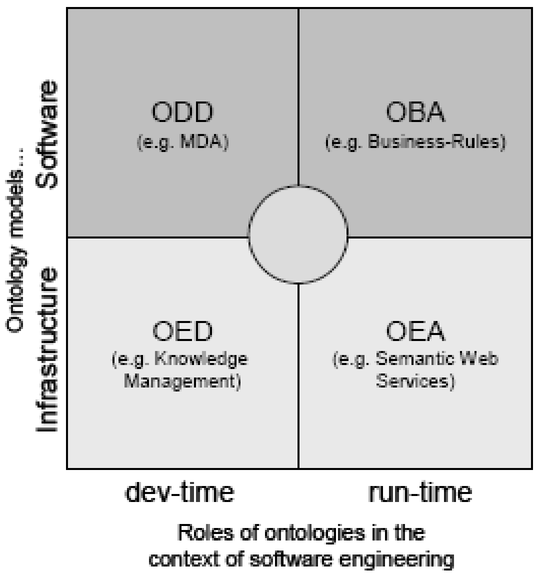

Ontology models...
Software
Infrastructure
ODD
(e.g. MDA)
OBA
(e.g. Business-Rules)
OED
(e.g. Knowledge Management)
OEA
(e.g. Semantic Web Services)
dev-time
run-time
Roles of ontologies in the context of software engineering

Figure 12: Ontology Usage in Software Engineering (Happel and Seedorf 2006).

2.4.5.1 Ontologies and Model-Driven Development

The current MDA-based infrastructure provides an architecture for creating models and meta models, define transformations between those models, and managing meta data. Though the semantics of a model is structurally defined by its meta model, the mechanisms to describe the semantics of the domain are rather limited compared to knowledge representation languages (Tetlow, Pan et al. 2005). MDA–based languages do not have a knowledge-based foundation to enable reasoning. Other possible shortcomings include validation and automated consistency checking. However, this is addressed by the Object Constraint Language (OCL).

There are several alternatives for integrating MDA-based information representation languages and ontology languages, which are exemplified in (Kiko and Atkinson 2005). Whereas some regard the UML as ontology representation language by defining direct mappings between language constructs (Cranefield 2002), others employ the UML as modeling syntax for ontology development (Baclawski, Kokar et al. 2002). In most cases, MDA-compliant languages and RDF/OWL are regarded as two distinct technological spaces sharing a "semantic overlap" where synergies can be realized by defining bridges between them (Gaševic, Djuric et al. 2004). The Ontology Definition Metamodel (ODM) (OMG 2006) is an effort to standardize the mappings between knowledge representation and conceptual modeling languages. It specifies a set of MOF meta models for RDF Schema and OWL among others, informative mappings between those languages, and profiles for a UML-based notation.
Software modeling languages and methodologies can benefit from the integration with ontology languages such as RDF and OWL in various ways, e.g. by reducing language ambiguity, enabling validation and automated consistency checking (Tetlow, Pan et al. 2005). Ontology languages provide better support for logical inference, integration and interoperability than MOF-based languages. UML-based tools can be extended more easily to support the creation of domain vocabularies and ontologies. Since ontologies promote the notion of identity, ODM and related approaches simplify the sharing and mediation of domain models

2.4.5.2 Ontologies in Requirements Engineering

The phase of requirements engineering deals with gathering the desired system functionality from the customers. Since the involved software engineers are often no domain experts, they must learn about the problem domain from the customers. A different understanding of the concepts involved may lead to an ambiguous, incomplete specification and major rework after system implementation. Therefore it is important to assure that all participants in the requirements engineering phase have a shared understanding of the problem domain. Moreover, change of requirements needs to be considered because of changing customer's objectives.
Ontologies can be used for both, to describe requirements specification documents (Mayank, Kositsyna et al. 2004; Decker, Ras et al. 2005) and formally represent requirements knowledge (Lin, Fox et al. 1996; Wouters, Deridder et al. 2000). In most cases, natural language is used to describe requirements, e.g. in the form of use cases. However, it is possible to use normative language or formal specification languages which are generally more precise and pave the way towards the formal system specification. Because the degree of expressiveness can be adapted to the actual needs, ontologies can cover semi-formal and structured as well as formal representation (Wouters, Deridder et al. 2000).

In contrast to traditional knowledge-based approaches, e.g. formal specification languages, ontologies seem to be well suited for an evolutionary approach to the specification of requirements and domain knowledge (Wouters, Deridder et al. 2000). Moreover, ontologies can be used to support requirements management and traceability (Lin, Fox et al. 1996; Mayank, Kositsyna et al. 2004). Automated validation and consistency checking are considered as a potential benefit compared to semi-formal or informal approaches providing no logical formalism or model theory. Finally, formal specification may be a prerequisite to realize model-driven approaches in the design and implementation phase.

2.4.5.3 Ontologies for Test Case Generation

Software tests are an important part of quality assurance (Abran, Moore et al. 2004). However, the writing of test cases is an expensive endeavor that does not directly yield business value. It is also not a trivial task, since the derivation of suitable test cases demands a certain amount of domain knowledge.
Ontologies could help to generate basic test cases since they encode domain knowledge in a machine processable format. A simple example for this would be regarding cardinality constraints. Since those constraints define restrictions on the association of certain classes, they can be used to derive equivalency classes for testing (Knublauch, Oberle et al. 2006).
Ontologies may not be the first candidate for such a scenario, since there are formalisms like OCL that are specialized for such tasks. However, once domain knowledge is available in an ontology format anyway (e.g. due to one of the various other scenarios described in this thesis), it might be feasible to reuse that knowledge.
Nguyen et al. (Nguyen, Perini et al. 2008) describe a framework for automated test case generation for multi-agent systems. They use agent interaction ontologies that define content semantics of agent interactions to generate test inputs, guide the exploration of the input space during test case generation, and verify messages exchanged between agents with respect to the agent interaction ontology. Experimental results show that whenever the interaction ontology has non trivial size, the proposed method achieves a higher coverage of the ontology classes than manual test case derivation. It also overcomes manual derivation in terms of revealed faults, as well as portion of input space explored during testing.

2.4.5.4 Advantages of Ontologies in Software Engineering

Since modeling ontologies is a tedious and costly task, it is always important to demonstrate the advantages one can gain by applying ontologies in Software Engineering. This is underlined by the fact that most of the formal foundations of

ontologies have been in place for a long time, without enjoying a wide-spread adoption by software engineers.
So clearly the current advent of logic-based formalisms in the context of the semantic web effort is an important factor. Activities by the W3C and others have helped to flesh out standards like RDF or OWL that receive increasing attention by tool builders and users. In a certain sense, the importance of standardization here can be compared to the situation of visual modeling in Software Engineering before UML (Happel and Seedorf 2006).
Another important factor is the flexibility of ontologies. With information integration as a major use case, ontologies are well-suited to combine information from various sources and infer new facts based on this. Also, the flexibility allows extending existing ontologies very easy, thus fostering the reuse of existing work. This is further promoted by the "web"-focus of current ontology approaches. Due to the fact that software systems also get increasingly web-enabled and must thus cope with data from heterogeneous sources that may not be known at development time, software engineers seek technologies that can help in this situation. Thus, experts in the field like Grady Booch are expecting semantic web technology to be one of the next big things in the architecture of web-based applications (Booch 2006). Also, the web makes it easier to share knowledge. Having URIs as globally unique identifiers, it is easy to relate one’s ontology to someone else's conceptualization. This in turn encourages interoperability and reuse.
Regarding more Software Engineering-specific advantages, ontologies make domain models first order citizens. While domain models are clearly driving the core of every software system, their importance in current Software Engineering processes decreases after the analysis phase. The core purpose of ontologies is by definition the formal descriptions of a domain and thus encourages a broader usage throughout the whole Software Engineering lifecycle (Uschold and Gruninger 2004).

2.4.6 Ontologies vs. Metamodeling

To be widely adopted by users and to succeed in real-world applications, knowledge engineering and ontology modeling must catch up with mainstream software trends. It will provide a good support in software tools and ease the integration with existing or upcoming software tools and applications, which will add values to both sides. To be employed in common applications, software knowledge management must be taken out of laboratories and isolated high-tech applications and put closer to ordinary developers (Cranefield 2002).
Djurić et al. (Djurić, Gašević et al. 2005) propose an approach for ontology modeling in the context of MDA and Semantic Web, as presented in Figure 13.

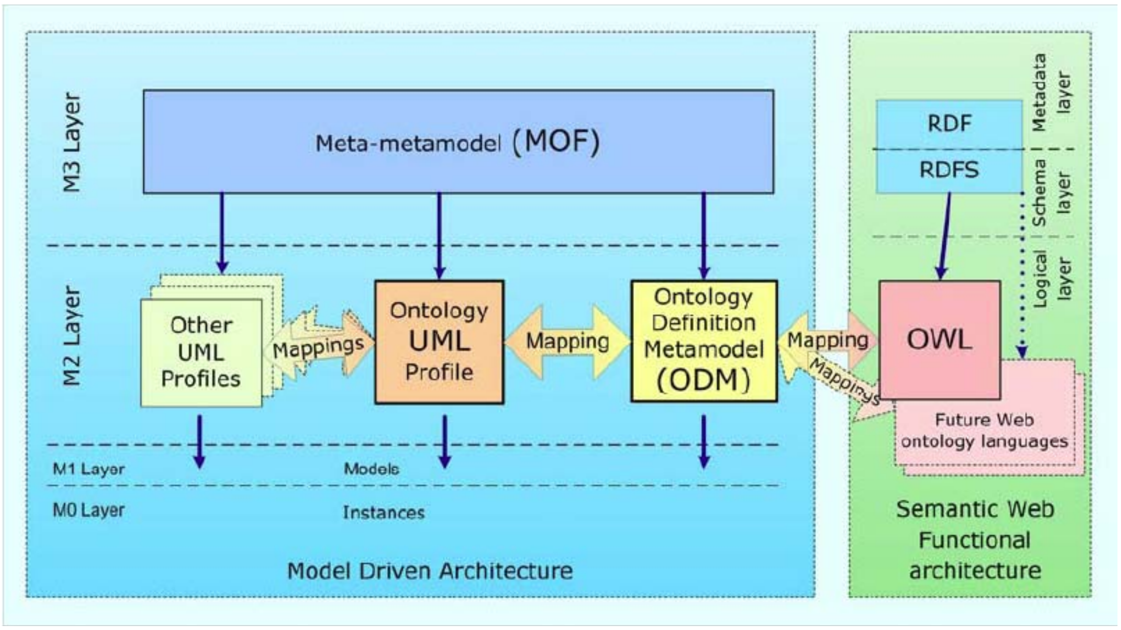


Figure 13: Ontology Modeling in MDA and Semantic Web (Djurić, Gašević et al. 2005).

The proposed Ontology Definition Metamodel (ODM) should be designed to comprehend common ontology concepts. A good starting point for ODM construction is OWL since it is the result of the evolution of existing ontology representation languages, and is going to be a W3C recommendation. It is at the Logical layer of the Semantic Web, on top of RDF Schema (Schema layer). In order to make use of graphical modeling capabilities of UML, an ODM should have a corresponding UML Profile. This profile enables graphical editing of ontologies using UML diagrams as well as other benefits of using mature UML CASE tools (Biffl, Ferstl et al. 2009). Both UML models and ODM models are serialized in XMI format so the two-way transformation between them can be done using XSL Transformation. OWL also has representation in the XML format, so another pair of XSL Transformations should be provided for two-way mapping between ODM and OWL. For mapping from the Ontology UML Profile into another, technology-specific UML Profiles, additional transformations can be added to support usage of ontologies in design of other domains and vice versa (Djurić, Gašević et al. 2005).

Parreiras et al. (Parreiras, Staab et al. 2007) illustrated variations on the principle idea of using meta modeling technical space (MMTS) with different ontological technical spaces (OTSs). The basic patterns they identified in their work is that next to existing technical spaces of established meta modeling frameworks, new technical spaces are positioned that either enrich or exploit the software engineering capabilities by or for ontology technologies.

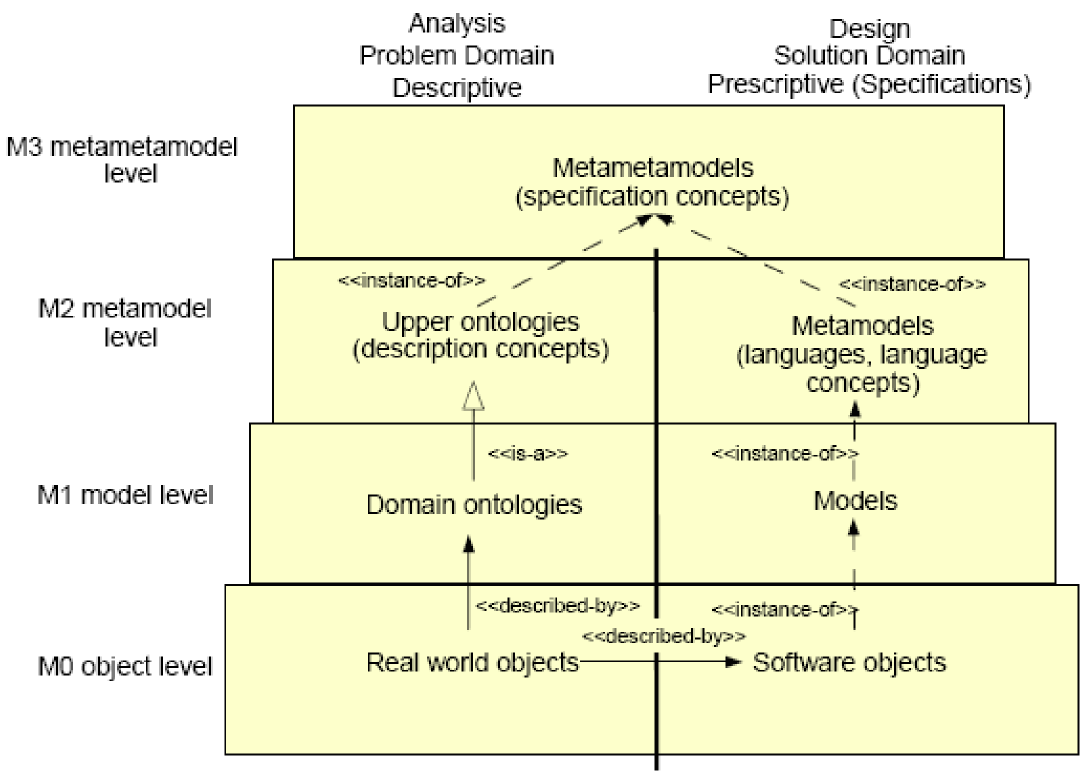


Figure 14: The ontology-aware meta-pyramid (Aßmann, Zschaler et al. 2006).

Aßmann et al. (Aßmann, Zschaler et al. 2006) discuss the role of descriptive and structural models, in particular ontologies, in the model-driven process. They extend the MDA layered architecture (see Figure 4) to the so-called "ontology-aware" meta pyramid as shown in Figure 14.
Aßmann et al. (Aßmann, Zschaler et al. 2006) identified several benefits of the ontology-ware meta pyramid. First of all, it suggests a more concrete model-driven software development process. The designer starts from standardized analysis models, ontologies, which may have been defined long before project start. These domain and business models are refined towards design models, avoiding the risks of a self-made domain analysis. Secondly, ontologies as analysis models offer more common vocabulary for software architect, customer, and domain expert. This should improve the understanding of the parties that order and construct software. Then, the standardization of the ontologies improves the interoperability of applications, because applications that use the ontology contain a common core of common vocabulary. Finally, the ontology-aware meta-pyramid distinguishes conceptual from behavioral models. It seems to be convenient to centre software modeling on concepts of a domain, or structure of a domain, while adding behavior to it step by step.

2.5 *Semantic Integration*

The proposed Engineering Knowledge Base (EKB) framework can be seen as a semantic integration approach, hence this section introduces the research field of Semantic Integration, the different available approaches are classified and explained, and in addition application scenarios for the usage of ontologies for Semantic Integration are given.

2.5.1 Overview

Semantic Integration is defined as the solving of problems originating from the intent to share data across disparate and semantically heterogeneous data (Halevy 2005). These problems include the matching of ontologies or schemas, the detection of duplicate entries, the reconciliation of inconsistencies, and the modeling of complex relations in different sources. (Noy, Doan et al. 2005) Over the last years, semantic integration became increasingly crucial to a variety of information-processing applications and has received much attention in the web, database, data-mining and AI communities. One of the most important and most actively studied problems in semantic integration is establishing semantic correspondences (also called mappings) between vocabularies of different data sources. (Doan, Noy et al. 2004)
Goh (Goh 1996) identified three main categories of semantic conflicts in the context of data integration that can appear: confounding conflicts, scaling conflicts, and naming conflicts. The use of ontologies as a solution option to semantic integration and interoperability problems has been studied over the last 10 years. Wache et al. (Wache, Vögele et al. 2001) reviewed a set of ontology-based approaches and architectures that have been proposed in the context of data integration and interoperability.

2.5.2 Classification of Approaches

Doan and Halevy (Doan and Halevy 2005) summarize the research on semantic integration in the database community. There, the matching of two database schemas requires deciding if any two elements of both schemas match, meaning that they refer to the same real-world concept. Typical challenges include the efficient extraction of semantic information, unreliable clues for matching schema elements (e.g., element names, types, data values, schema structures and integrity constraints), incomplete schema and data clues, and subjective matching depending on the application. Rule-based matching techniques use hand-crafted and/or probabilistic rules to exploit schema information for the identification of mappings. Rule-based matching techniques are relatively inexpensive and fairly fast since the typically operate only on schemas and not on data instances. But this is also their main drawback, as they cannot exploit data instances effectively, even though the instances can encode a wealth of information. Additionally, in many cases effective matching rules are simply too difficult to hand craft. Learning-based matching techniques consider a variety of machine learning techniques to exploit both schema and data information. There is also a growing realization that schema- and data-related evidence in two schemas being matched often is inadequate for the matching process, leading to the inclusion of external evidences beyond the two current schemas to the matching process. The key idea here is that a matching tool must be able to learn from past matches (Halevy 2005).

2.5.3 Ontologies for Semantic Integration

Noy (Noy 2004) identified three major dimensions of the application of ontologies for supporting semantic integration: the task of finding mappings (semi-)automatically, the declarative formal representation of these mappings, and reasoning using these mappings. There exist two major architectures for mapping discovery between ontologies. On the one hand, the vision is a general upper ontology which is agreed upon by developers of different applications. Two of the ontologies that are built specifically with the purpose of being formal top-level ontologies are the Suggested Upper Merged Ontology (SUMO) (Niles and Pease 2001) and DOLCE (Gangemi, Guarino et al. 2003). On the other hand, there are approaches comprising heuristics-based or machine learning techniques that use various characteristics of ontologies (e.g., structure, concepts, instances) to find mappings. These approaches are similar to approaches for mapping XML schemas or other structured data (Bergamaschi, Castano et al. 1999; Cruz, Huiyong et al. 2004). The declarative formal representation of mappings is facilitated by the higher expressive power of ontology languages which provide the opportunity to represent mappings themselves in more expressive terms. There exists a large spectrum of how mappings are represented. Bridging axioms relate classes and properties of the two source ontologies and can be seen as translation rules referring to the concepts of source ontologies and e.g., specifying how to express a class in one ontology by collecting information from classes in another ontology. Another mapping representation is the declarative representation of mappings as instances in an ontology. This ontology can then be used by tools to perform the needed transformations. Then a mapping between two ontologies constitutes a set of instances of classes in the mapping ontology and can be used by applications to translate data from the source ontology to the target. Naturally, defining the mappings between ontologies, either automatically, semi-automatically, or interactively, is not a goal in itself. The resulting mappings are used for various integration tasks: data transformation, query answering, or web-service composition, to name a few. Given that ontologies are often used for reasoning, it is only natural that many of these integration tasks involve reasoning over the source ontologies and the mappings (Noy, Doan et al. 2005).

Rosenthal et al. (Rosenthal, Seligman et al. 2004) extend the concept of semantic integration to semantics management, which has the goals of easing data sharing for both new and old systems, of ensuring that needed data is actually collected, and of maximizing over time the business value of an enterprise's information systems. To reach these goals, new areas of useful semantic agreements need to be produced proactively, helping enterprises to satisfy new requirements and also reducing costs by reducing unneeded semantic and representation diversities. Additionally, not only the needs of technology-savvy system integrators need to be considered, but also other roles (e.g., enterprise owners, architects, end users and developers) need assistance to have a greater shared understanding of what the data means. Finally, the definition of "semantics" need to be broadened, to describe what data instances are collected and desired (as in publish/subscribe systems), not just concept definitions and relationships.

Uschold and Gruninger (Uschold and Gruninger 2004) identified four main categories of ontology application to provide a shared and common understanding of a domain that can be communicated between people and application systems (Fensel 2003): Given the vast number of non-interoperable tools and formats, a given company or organization can benefit greatly by developing their own neutral ontology for authoring, and then developing translators from this ontology to the terminology required by the various target systems. To ensure no loss in translation, the neutral ontology must include only those features that are supported in all of the target systems. The trade-off here is loss of functionality of some of the tools; since certain special features may not be usable. While it is safe to assume there will not be global ontologies and formats agreed by one and all, it is nevertheless possible to create an ontology to be used as a neutral interchange format for translating among various formats. This avoids the need to create and maintain $O(N^2)$ translators and it makes it easier for new systems and formats to be introduced into an existing environment. In practical terms, this can result in dramatic savings in maintenance costs - it has been estimated that 95% of the costs of enterprise integration projects is maintenance (Pollock 2002).

There is a growing interest in the idea of "Ontology-Driven Software Engineering" in which an ontology of a given domain is created and used as a basis for specification and development of some software. The benefits of ontology-based specification are best seen when there is a formal link between the ontology and the software. This is the approach of Model-Driven Architecture (MDA) (Miller and Mukerji 2001) created and promoted by the Object Modeling Group (OMG) as well as ontology software which automatically creates Java classes and Java Documents from an ontology. A large variety of applications may use the access functions of the ontology (Parreiras, Staab et al. 2007). Not only does this ensure greater interoperation, but it also offers significant cost reduction for software evolution and maintenance. A suite of software tools all based on a single core ontology are semantically integrated for free, eliminating the need to develop translators. To facilitate search, an ontology is used as a structuring device for an information repository (e.g., documents, web pages, names of experts); this supports the organization and classification of repositories of information at a higher level of abstraction than is commonly used today Using ontologies to structure information repositories also entails the use of semantic indexing techniques, or adding semantic annotations to the documents themselves. If different repositories are indexed to different ontologies, then a semantically integrated information access system could deploy mappings between different ontologies and retrieve answers from multiple repositories (Happel and Seedorf 2006).

2.6 *Semantic Web Services*

This section summarizes related work on Semantic Web Services and present approaches for Service Matchmaking.

2.6.1 Overview

The promise of Web Services and the need for widely accepted standards enabling them are by now well recognized, and considerable efforts are underway to define and evolve such standards in the commercial realm. In particular, the Web Services Description Language (WSDL) (Christensen, Curbera et al. 2001) is already well established as an essential building block in the evolving stack of Web Service technologies, allowing the specification of the syntax of the input and output messages of a basic service, as well as of other details needed for the invocation of the service. WSDL does not, however, support the specification of workflows composed of basic services. In this area, the Business Process Execution Language for Web Services (BPEL4WS) (Juric 2006), has the most prominent status. With respect to registering Web services, for purposes of advertising and discovery, Universal Description, Discovery and Integration (UDDI) (Bellwood, Clement et al. 2002) has received the most attention to date.
At the same time, recognition is growing of the need for richer semantic specifications of Web Services, so as to enable fuller, more flexible automation of service provision and use, support the construction of more powerful tools and methodologies, and promote the use of semantically well-founded reasoning about services. Because a rich representation language permits a more comprehensive specification of so many different aspects of services, they can provide a better foundation for a broad range of activities, across the Web service lifecycle. Furthermore, richer semantics can help to provide fuller automation of activities as verification, simulation, configuration, supply chain management, contracting, and negotiation of services. (Martin, Paolucci et al. 2005)

To meet this need, researchers have been developing languages, architectures and related approaches for so called Semantic Web services (McIlraith, Son et al. 2001). The Ontology Web Language for Services (OWL-S) (Martin, Ankolekar et al. 2004), which seeks to provide the building blocks for encoding rich semantic service descriptions in a way that builds naturally upon OWL (Bechhofer, van Harmelen et al. 2004), the Semantic Web language, supplies Web Service providers with a core set of markup language constructs for describing the properties and capabilities of their Web Services in unambiguous, computer-interpretable form. OWL-S markup of Web Services facilitates the automation of Web Service tasks, including automated Web Service discovery, execution, composition and interoperation.
Sivashanmugam et al. (Sivashanmugam, Verma et al. 2003) propose an approach for adding semantics in WSDL and UDDI. Semantics are added to WSDL using extensibility in elements and attributes supported by the WSDL specification. Using this extensibility existing and extended WSDL constructs (i.e., operations and message parts) are mapped to ontologies. The use of ontologies allows representing Web Service descriptions in a machine-interpretable form like. Additionally, new WSDL tags for Web Service preconditions and effects are added. Semantic discovery using UDDI is enabled by storing the semantic annotation of Web Services in the existing structures of UDDI and by providing an interface to construct queries that use these semantic annotations.
WSDL-S (Miller, Verma et al. 2004) is another approach for annotating current Web Service standards with semantic descriptions. In WSDL-S, the expressivity of WSDL is enriched with semantics by employing concepts similar to those in OWL-S while being agnostic to the semantic representation language. The advantage of this approach to adding semantics to WSDL is multi-fold. First, users can, in an upwardly compatible way, describe both the semantics and operation level details in WSDL- a language that the developer community is familiar with. Second, by externalizing the semantic domain models, a language-agnostic approach to ontology representation is taken. This allows Web service developers to annotate their Web services with their choice of modelling language (such as OWL, or legacy models developed in UML or other knowledge representation languages). This is significant because the ability to reuse existing domain models expressed in modelling languages like UML can greatly alleviate the need to separately model semantics. Finally, it is relatively easy to update the existing tooling around WSDL specification to accommodate our incremental approach. Moreover, the externalization of the semantic domain models still allows for richer representations of domain concepts and relationships in languages such as OWL, thereby bringing together the best of both worlds. Use of expressive mapping representation and techniques can further enable this approach to deal with significant types of syntactic, structural, representational and semantic heterogeneity. (Akkiraju, Farrell et al. 2005)
The Web Service Modeling Ontology (WSMO) (Lausen, Polleres et al. 2005) is a framework for Semantic Web Services which refines and extends the Web Service

Modeling Framework (WSMF) (Fensel and Bussler 2002) to a meta-ontology for Semantic Web services. WSMF defines a rich conceptual model for the development and the description of Web Services based on two main requirements: maximal decoupling and strong mediation. WSMO is accompanied by a formal language, the Web Service Modeling Language (WSML) that allows annotating Web Services according to the conceptual model. Also an execution environment (WSMX) (Haller, Cimpian et al. 2005) for the dynamic discovery, selection, mediation, invocation, and inter-operation of Semantic Web services based on the WSMO specification is included. (Feier, Roman et al. 2005).

2.6.2 Service Matchmaking Approaches

Software components discovery and Web Service discovery can be classified into two categories: signature matching and semantic matching.

Purtilo and Atlee (Purtilo and Atlee 1991) propose a signature-matching approach by specifying the invocation parameters. Zaremski and Wing (Zaremski and Wing 1995) describe exact and relaxed signature matching as a means for retrieving functions and modules from a software library. Wang and Stroulia (Wang and Stroulia 2003) provide a structure-matching-based signature matching for Web Service discovery. Signature matching is an efficient means for software components retrieval, but two software components with similar signatures may have completely different behaviors.

Semantic matching addresses this problem by comparing software components based on formal descriptions of the semantics of their behaviors. Zaremski and Wing (Zaremski and Wing 1997) extend their signature-matching work with a specification-matching scheme. Cho et al. (Cho, McGregor et al. 1998) use a protocol to specify interoperability of objects. Semantic matching identifies suitable services more precisely than signature-matching methods, but the cost of formally defining provided and required services is considerable.

Paolucci et al. (Paolucci, Kawamura et al. 2002) propose a DAML-S based approach for a declarative description of web services outside the representation capabilities of UDDI and WSDL. They provide an upper-level ontology of service profiles consisting of service actors, functional service attributes, and function service descriptions.

Trastour et al. (Trastour, Bartolini et al. 2001) define a set of requirements needed for service matchmaking based on Semantic Web techniques and evaluate a set of standard approaches (e.g., UDDI, ebXML) using these requirements. The potential complexity of the service descriptions, like attribute-value pairs or nested tree/graph style structures, requires a flexible and expressive metadata model. In order to support under-specified data structures like incomplete service advertisements, an approach needs to be able to express semi-structured data. Additionally, support for types and subsumption is needed to be able to work at different levels of generality. Finally, constraints need to be expressed to define and check the acceptable instances for service invocation.

Li and Horrocks (Li and Horrocks 2004) investigate how Semantic and Web Services technologies can be used to support service advertisement and discovery in e-Commerce. They describe the design and implementation of a service matchmaking prototype which uses a DAML-S based ontology and a Description Logic reasoner to compare ontology based service descriptions. By representing the semantics of service descriptions, the matchmaker enables to locate suitable web services automatically. The approach is evaluated using a realistic agent based e-commerce scenario. Although the initial classification of large numbers of service descriptions could be quite time consuming, subsequent matching of queries could be performed very efficiently.
Kolovski et al. (Kolovski, Parsia et al. 2005) provide a mapping of WS-Policy to OWL. WS-Policy (Bajaj, Box et al. 2006) provides a general purpose model and syntax to describe the policies of a Web service. It specifies a base set of constructs that can be used and extended by other Web service specifications to describe a broad range of service requirements and capabilities.WS-Policy's scope is limited to allowing endpoints to specify requirements and capabilities needed for establishing a connection. Its goal is not be used as a language for expressing more complex, applicationn-specific policies that take effect after the connection is established. Kolovski et al. (Kolovski, Parsia et al. 2005) show how standard OWL reasoners can be used to check policy conformance and perform an array of policy analysis tasks. The main advantage of representing Web Service policies using OWL is that OWL is much more expressive than WS-Policy and thus provides a framework for exploring richer policy languages.
Verma et al. (Verma, Akkiraju et al. 2005) present an approach for matching the non-functional properties of Web Services represented using WS-Policy (Bajaj, Box et al. 2006). To date, most policy matching has been done using syntactic approaches, where pairs of policies are compared for structural and syntactic similarity to determine compatibility. In their approach, the authors enhance the policies of a Web Service with semantics by creating the policy assertions based on terms from ontologies. The use of semantic terms enables richer representations of the intent of a policy and allows matching of policies with compatible intent, but dissimilar syntax. This approach of using semantic concepts and rules during policy matching leads to better Web Service matches that may not have been possible with syntax based matchers, or prior semantic based methods.
Oldham et al. (Oldham, Verma et al. 2006) present a framework and implementation of an innovative tool for the matching providers and consumers based on WS-Agreements. The WS-Agreement specification (Andrieux, Czajkowski et al. 2004) defines a language and protocol for capturing the relationship with agreements between two parties. An agreement between a service consumer and a service provider specifies one or more service level objectives (SLO) which state the requirements and capabilities of each party on the availability of resources and service qualities. WS-Agreement is more expressive than the previous policy standards because in addition to service level objectives, an agreement contains scopes for which the guarantee holds, conditions which must exist in order for the guarantee on the SLO to be valid, and business values,

such as penalties and rewards, which incur if the SLO is not satisfied. Oldham et al. (Oldham, Verma et al. 2006) utilize Semantic Web technologies to achieve rich and accurate matches. A key feature is the flexible approach for achieving user personalized matches using user-defined rules.

Chapter 3

3 Research Approach

This chapter describes the research approach by identifying the research issues, specifying the research methods and introducing the two application scenarios, namely System-Wide Information Sharing (SWIS) and Simulation of Assembly Workshops (SAW).
The scope of this work is an engineering team consisting of experts from several engineering disciplines, who work on engineering process tasks with role-specific tools and systems that encapsulate engineering models and project data. As the engineers work together to deliver a product to the end user, they inevitably have to form common concepts on deliverables at interfaces between their work tasks. Such common concepts can be found in elements of requirements, design, and defect descriptions, which concern more than one role. Typical requirements for such engineering process tasks are low delay, i.e., in-time availability of information from other engineering tools and low effort for achieving the information exchange between the engineering tools.

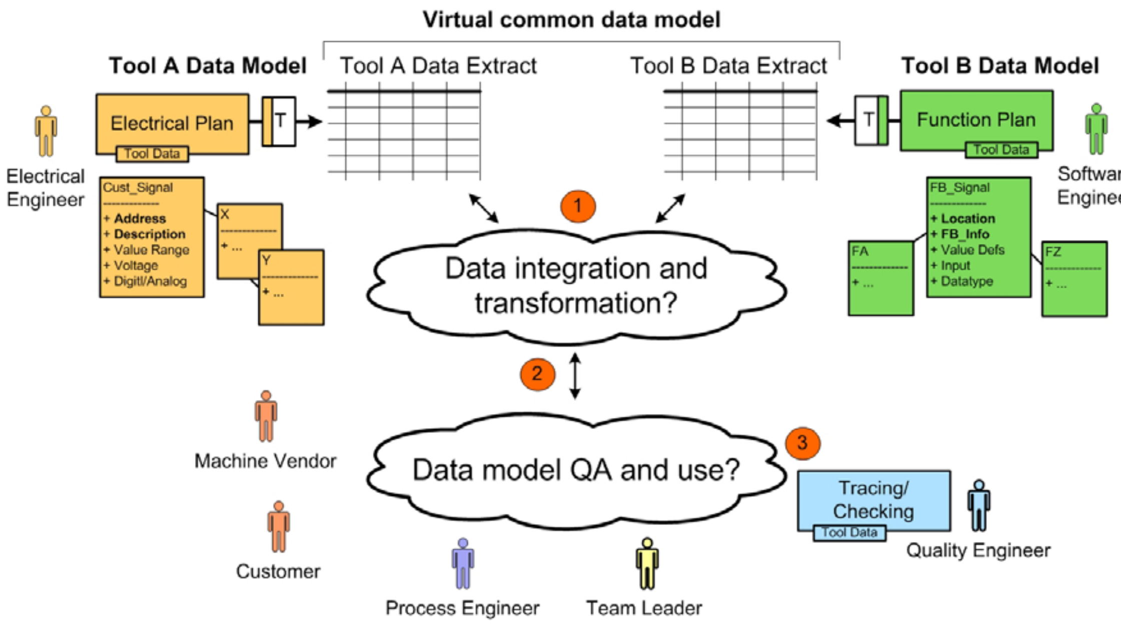


Figure 15: Overview of the research challenges.

As shown in Figure 15, each engineering role (e.g., electrical engineer or software engineer) has a tailored tool set that works on data relevant to the engineer's tasks. In order to support the data exchange between these engineering tools, an additional component is needed. In a typical process step in the engineering process an engineer exports data from his tool to a transfer document (e.g., PDF of data table) and integrates

this document in a common repository accessible by a set of partner engineering tools. The major challenges here are on the one hand side in the identification and description of tool data that should be extracted from tools and made available to other tools. On the other hand side, the data integration itself poses another huge challenge, since it is often not possible to agree on a common data schema agreed on by all tools, and additionally all engineers working with the tools want to stick with their well-known terms and notations. Finally, the re-use of at least parts of integration solutions for other projects with different project partners is mostly not possible. In order to support data exchange between these sets of partner engineering tools, transformations between the different notations and format of the particular partner engineering tools is needed. The major challenges of the transformation process are both the adaptation of transformation instructions to new or changed tool data structures which normally requires time-consuming manual human work, as well as the runtime performance of these transformation instructions. Using these foundations, i.e., export, integration and transformation, additional methods like Quality Assurance (QA) support or other advanced methods like model consistency checking or end-to-end analyses are allowed. Currently there is high effort needed to perform typical engineering project tasks like model checks across tool boundaries or end-to-end analyses, which may also lead to risks of defect since data needs to be manually re-entered in several different tools.
For these tasks, we propose to use the novel Engineering Knowledge Base (EKB) framework. In comparison to a simple data storage such as a common repository, a knowledge base stores information (i.e., the original data plus meta-data describing links between data elements or annotations of data elements using machine-understandable syntax which can be used to automate time-consuming tasks and support human experts in doing their work. The EKB stores explicit engineering knowledge to support access to and management of engineering models across tools and disciplines by providing (1) **data integration by exploiting mappings** between local and common engineering concepts; (2) **transformations** between local engineering concepts; and (3) **advanced applications** using these foundations, e.g., end-to-end analyses.

3.1 *Research Issues*

This section identifies the research issues addressed in this thesis. The key research item of this thesis is the Engineering Knowledge Base (EKB) framework, which aims at enabling effective and efficient data integration and transformation between heterogeneous engineering experts' data models without the need for a common data schema, and at additionally providing advanced methods like end-to-end analyses. The application area of the EKB as key research item is the engineering of distributed, flexible and complex systems, which traditionally were designed inflexible, but in order to follow the trend of engineering flexible systems, new engineering approaches are needed.

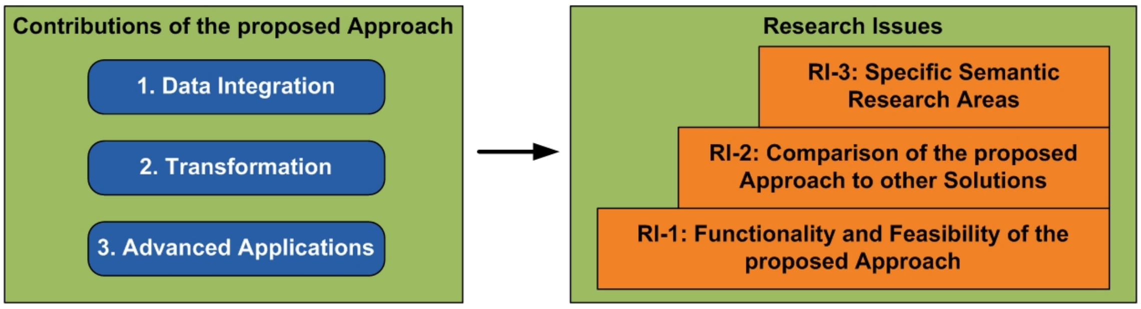

Contributions of the proposed Approach
1. Data Integration
2. Transformation
3. Advanced Applications
Research Issues
RI-3: Specific Semantic Research Areas
RI-2: Comparison of the proposed Approach to other Solutions
RI-1: Functionality and Feasibility of the proposed Approach

Figure 16: Overview of the research issues.

Figure 16 shows the three major research issue categories, which were derived from the three major contributions of the proposed approach, namely (1) Data Integration, (2) Transformation, and (3) Advanced Applications. The first research issue category addresses the **functionality and feasibility of the proposed approach (RI-1)**, by dealing with the foundations for data integration and transformation, with Quality Assurance support, with the support for traceability across engineering domains, and with the support for end-to-end testing. The second research issue category deals with the **comparison of the proposed EKB framework to two different kinds of alternative solutions (RI-2)**, namely solutions that primarily rely on implicit knowledge, such as common repositories and data warehouses, as well as solutions that also rely on explicit knowledge, such as other ontology-based approaches. Since the process of applying the EKB framework uses ontologies as modeling methods, the third research issue category deals with two **semantic web specific research areas (RI-3)**, namely the usage of Ontology Alignment methods for providing the required mappings, as well as a conceptual approach for structuring big ontologies in order to increase usability and maintainability

The following subsections describe the single research issues more detailed with respect to the three major research issue categories: functionality and feasibility of the proposed engineering environment integration approach, comparison of the proposed engineering environment integration approach to traditional approaches that use only implicit knowledge and other approaches which use explicitly represented knowledge, and specific semantic research areas of the proposed engineering environment integration approach.

3.1.1 Functionality and Feasibility of the Proposed Approach

In this thesis, we apply the Engineering Knowledge Base (EKB) framework to two application scenarios from two different application domains. The first research issue category deals with the general functionality and feasibility of the EKB architecture and processes. As precondition for these research issue, we needed to ensure that a) the knowledge is complete enough for relevant process steps, and b) the knowledge can be accessed by tools or processes, e.g., by means of an API.

3.1.1.1 Foundations for data integration and transformation

The basic process of applying the EKB framework (refer to section 4.3.2) consists of the following steps: As a first step for the preparation of the engineering environment, the involved models are analyzed to identify overlapping concepts between pairs of engineers and their tools. Once the overlapping engineering concepts are identified, these concepts need to be described and modeled. Once this domain model is complete and revised, the local tool-specific concepts need to be described and modeled. After the tool models is complete and revised, the final step of mapping local tool-specific concepts to overlapping common concepts needs to be performed. These mapping allows the creation of transformation instructions. Based on these transformations, more complex applications can be implemented which use the integrated data of the virtual common data model to perform advanced tasks like tracing of artifacts, consistency checking across tool boundaries, change impact analyses or end-to-end analyses.
Based on this, we derive the following research issues (refer also to research challenges 1-3 in Figure 15).

RI-1.1. Feasibility of the proposed engineering environment integration approach. Investigate, whether the proposed engineering environment integration approach is capable of supporting and automating typical engineering process steps. This research issue primarily is addressed in sections 4.3, 5.2.3, and 6.2.1.

RI-1.2. Foundations for tool support for automation of engineering process steps. Investigate to what extent (e.g., effort saved during process execution) the explicit and machine-understandable semantic modeling of common domain knowledge helps to automate time-consuming engineering process steps. This research issue primarily is addressed in the sections 4.3.3, 5.5, 6.6, and 6.7.

To address changing business needs, IT systems have to be built in shorter cycles and more flexible, which puts pressure on quality management capabilities to effectively evaluate the quality of more complex systems (with new sources of defects). The general challenges of engineering of distributed, flexible and complex systems – weak integration between engineering disciplines and the need for more flexible systems and engineering processes – lead to the following research issues regarding quality assurance support, support for traceability across engineering domains, and

3.1.1.2 Quality assurance support

New software development approaches, such as the EKB framework, are expected to bring benefits to software development like faster or more efficient development. However, from a software quality point of view the question remains whether the means for quality assurance (QA) are comparable to or better than with a traditional approach; e.g., the complexity introduced by the EKB framework architecture may make QA actually harder. From the goal to measure and ensure stakeholder-oriented quality of the product and the development process, we derive the following research issues (refer also to research challenge 4 in Figure 15). These research issues primarily are addressed in section 6.5.

RI-1.3. Explicit modeling of stakeholder requirements. To what extent can domain-specific stakeholder value elements be explicitly modeled in the EKB framework as input to QA?

RI-1.4. Tool support and QA for requirements transformation. To what extent can the EKB framework transform the explicit quality requirements models into a running system without significant sources of defects like manual interaction; better quality measurement and feedback on intermediate models during systems development?

RI-1.5. Stakeholder-level quality measurement. How can the required quality levels be measured and assured in the EKB framework life cycle; i.e., auditing capabilities of the systems development process?

3.1.1.3 Support for traceability across engineering domains

A major challenge in the engineering of distributed, flexible and complex IT systems is to extend the scope of QA from software artifacts to include software-relevant parts of artifacts in other engineering domains. Thus a key research issue is traceability, i.e., how to link the relevant elements of models for requirements, design, implementation, and testing across engineering disciplines as foundation for better integrated product assessment and improvement. As manual tracing has been found effort-consuming and error-prone, automated approaches for software engineering have been developed to capture dependencies based on syntactical identity (e.g., keyword-matching as in information retrieval approaches). A major limitation of these automated approaches is their inability to capture dependencies completely, because they cannot capture dependencies between semantically related artifacts without syntactic identity (semantic gap). Therefore, we derive the following research issue (refer also to research challenge 5 in Figure 15).

RI-1.6. Support for traceability across engineering domains. Investigate how the EKB framework can support traceability (e.g., of requirements to source code and further on to test cases) across engineering domain boundaries. This research issue primarily is addressed in section 7.1.3.

3.1.1.4 Support for End-to-End Testing

In a typical engineering environment, system testing requires end-to-end testing, but since the single tools are distributed and using heterogeneous data models, an end-to-end test is hard to perform. Although testing tools are available to perform testing at multiple levels, most testing tools are incapable of building composite interdependent tests across technology platforms, languages and systems. Therefore the challenges in testing are driven by the distributed, heterogeneous nature of the used tools and a growing market of third-party services implying that there is not a single owner of the complete system. Based on this, we derive the following research issue (refer also to research challenge 5 in Figure 15).

RI-1.7. Support for end-to-end testing. Investigate how the EKB framework can support end-to-end testing across engineering tool and /or engineering domain boundaries. This research issue primarily is addressed in section 7.1.4.

3.1.2 Comparison of the Proposed Approach to Other Solutions

In this thesis, we compare the EKB framework to two different kinds of alternative solutions, namely solutions that primarily rely on implicit knowledge, such as common repositories and data warehouses, as well as solutions that also rely on explicit knowledge, such as other ontology-based approaches. The second research issue category deals with this comparison.

3.1.2.1 Comparison with Common Repository-based approaches

In this thesis, we evaluate and discuss the benefits and limitations of the proposed EKB framework in comparison to a traditional solution using a common repository, in particular, more efficient support for data exchange between automation systems engineering tools. Key goal is to investigate to what extent the explicit and machine-understandable knowledge stored in the EKB helps support time-consuming engineering processes, such as data exchange between tools or model checking across tools. Relevant derived issues for investigation are the following research issues. These research issues primarily are addressed in sections 7.1 and 7.2.3.

RI-2.1. Comparison of the EKB framework and Common Repository-based approaches. What are the advantages of using the EKB framework for these processes, compared to using a common repository?

RI-2.2. More effective and efficient engineering using the EKB framework. Investigate whether the EKB framework provides an overall more efficient and effective engineering process regarding typical requirements for engineering process tasks such as low delay, low effort for achieving the information exchange between the engineering tools, and flexibility of the approach regarding the involved engineering tools and data definitions.

RI-2.3. EKB framework usage trade-off analysis. Analyze how the extra effort for involving the EKB framework is likely to pay off in a typical engineering context.

3.1.2.2 Comparison to Data Warehouse-based approaches

Important management decisions such as decisions regarding product quality (e.g., measured by defect density in artifacts or the average time needed to fix major defects) or decisions regarding the development team (e.g., identification and preservation of core developers) are typically based on data originating from a range of tools. Currently, data collection is based on queries from a wide range of sources such as mailing lists, version control systems, and issue trackers. This approach has become very time-consuming. In addition, this data has to be available with little delay to support quickly reacting to various internal such as changes in the development team as well as external condition such as new releases of new software libraries used in a project. The faster relevant data can be retrieved, the more agile project steering can become. As project circumstances can change quickly or releases if new software versions are performed periodically, data collection has to be as well repeated with the same frequency, which is infeasible without proper tool support. Furthermore, once the data is retrieved, the process of analyzing and evaluating this data is even more difficult, if the data is collected from inhomogeneous sources with a variety of different, often incompatible data formats. Another issue is the data quality. Invalid, malformed, and irrelevant data elements further complicate the evaluation process, often making the outcome unsuitable for decision support.
In this thesis, we propose the EKB framework which enables automated collection and integration of data originating from a set of heterogeneous tools used in software development. To be of use for decision support, data integration has to be carried out efficiently to provide quasi-instant availability of the data, despite the fact that these tasks are very complex. The integrated and validated data can then be used as basis for basic of project data analysis and data improvement such as aggregation of data. Based on the data, also more advanced project management methods such as quality prediction

(Wahyudin, Mustofa et al. 2007), in-time notification of relevant stakeholders (Wahyudin, Heindl et al. 2008), or decision support for project managers can be applied. From this approach we derive the following research issues. These research issues primarily are addressed in section 7.3.1.

RI-2.4. Comparison of a traditional Date Warehouse-based data collection process to a semantically-enabled data collection process. Currently, the collection and integration of data originating from a set of heterogeneous tools is a mainly manual task. There exists tool support for the loading processes of a Data Warehouse; however, these tools are often only useable for specific applications and therefore hard to use for more generic processes without major adaptations. The EKB framework supports the collection and integration process by providing automated process steps, such as time-triggered collection or automated checks of data consistency and integrity. While we expect the EKB framework to make the data collection and validation process steps significantly more efficient, we also see reasonable effort investment in setting up the framework in a given context. Thus empirical evaluation is necessary to assess by when a breakeven point is likely to be achieved.

RI-2.5: Integration of additional data sources. Current tools used in a distributed software engineering environment are not fixed, but frequently change over time or according to new project requirements. In order to support such changes of data sources, the proposed approach needs to provide extensibility regarding both the underlying process as well as the designed data model. To assess the cost and benefit of this extensibility, the effort needed for the inclusion of another data source needs to be measured as well as the relationship of the extension effort to the number and types of already integrated data sources.

3.1.2.3 Comparison to other approaches which use explicit knowledge

There exist related Semantic Integration approaches (refer to section 2.5) which rely on ontologies for providing their functionality. In order to classify the proposed EKB framework, these approaches need to be identified, analyzed and compared with the proposed EKB framework. Based on this, we derive the following research issue. This research issue primarily is addressed in section 4.1.3.3.

RI-2.6: Relation of the proposed engineering environment integration approach to other ontology-based Semantic Integration approaches. Identify similar Semantic Integration approaches which also use ontologies as their modeling method. Analyze and discuss similarities and differences of the identified approaches.

3.1.3 Specific Semantic Research Areas of the Proposed Approach

The process of applying the EKB framework introduced in 3.1.1.1 uses ontologies as modeling methods. Therefore, the third category of research issues deals with two semantic web specific research areas, namely the usage of Ontology Alignment methods for providing the required mappings, as well as a conceptual approach for structuring big ontologies in order to increase usability and maintainability.

3.1.3.1 Combination with Ontology Alignment methods

In general, ontology alignment is used to connect partially different ontologies of one certain domain and overcomes therewith the heterogeneity problem. Ontology alignment tries to find a corresponding entity in one ontology for an entity in another ontology, with the same or at least a similar meaning. To align two or more ontologies, connections between the entities (concepts, relations or instances) of the ontologies need to be discovered. These entities can be equal, similar or different. Alignments between ontologies can be detected by using information sources like a common reference ontology (upper ontologies, background knowledge), lexical information, ontology structure, user input, external resources (WordNet, synonym databases, dictionaries) or prior matches (Noy and Stuckenschmidt 2005).
From an integration point of view, a major goal was to improve the capability of assuring validity of an integration solution while facilitating team work and tool support. From this general goal, we derived the following research issues regarding the EKB framework. These research issues primarily are addressed in sections 5.5.1 and 7.4.3.

RI-3.1. Safety-Critical Ontology Alignment: Investigate to what extent the mainly manual ontology alignment tasks of the EKB framework approach could be supported by other more automated ontology alignment approaches without violating the requirements regarding safety-criticalness.

RI-3.2. Risks of applying state-of-the-art ontology alignment approaches: Investigate the risks of using standard Ontology Alignment approaches within the EKB framework. Analyze the requirements resulting from the use case which have to be fulfilled by the investigated Ontology Alignment approaches.

3.1.3.2 Derivation of design guidelines for ontologies

Ontologies can provide relevant advantages like explicitly specifying the application domain semantics in heterogeneous engineering contexts. However, ontologies can get complex themselves and may get hard to extend, understand and manage. Therefore, we propose approaches for ontology design that aim at making ontology parts easier to handle and recombine, the so-called Ontology Areas. The general idea of Ontology Areas is to structure a comprehensive ontology into smaller building blocks with the following benefits for the designer and user of the ontology:

- A *smaller ontology* based on Ontology Areas that contains the minimal necessary knowledge for a specific task can be selected from a comprehensive ontology to facilitate more efficient use and change.
- We expect a smaller ontology (consisting of selected Ontology Areas) to exhibit *lower cognitive complexity* for designers who work with ontologies to make tools that support the automation of stakeholder tasks.
- Specific Ontology Areas can contain the more volatile ontology elements and thus make the design of the overall ontology *more stable against changes*.

We used the following guidelines to design the Ontology Areas: a) concepts that a particular stakeholder needs to fulfill his typical tasks in order to achieve cohesiveness of the Ontology Areas; b) discern between common domain concepts and local add-ons of a stakeholder (such as terminology), which may change in different project contexts; c) keeping apart more stable design-time concepts from more volatile run-time concepts; and d) structuring volatile run-time data by manageable time intervals depending on the frequency of data elements' change.
We empirically evaluate these approaches to find out whether the added complexity from using an ontology outweighs their intended benefits. As measurement criteria for evaluation we use the size of an ontology (and an Ontology Area) by counting the number of facts and relationships. In our study context the comprehensive ontology consists of: a) the production automation domain concepts for design-time and run-time elements; and b) stakeholder extensions to the data model, such as local terminologies and mappings, for all stakeholders.
We derive the following research issue to investigate the benefits of an ontology structured with Ontology Areas compared to an ontology without Ontology Areas. This research issue primarily is addressed in sections 6.4.2 and 7.3.2.

RI-3.3. Supporting engineering roles by lowering the cognitive complexity of the used ontologies. Evaluate the capability of Ontology Areas for supporting each engineering role by allowing using their local terminology to communicate with other stakeholders. For this task sufficient Ontology Areas need to contain for the communicating stakeholders: the common domain concepts in their universe of discourse, local terminologies, mappings between local terminology elements and common domain concepts (on class level).

3.2 *Research Methods and Evaluation Concept*

This section describes the research methods used in this thesis as well as the evaluation concept and evaluation criteria.

3.2.1 Research Methods

For research and review of related literature, we performed a systematic literature review (Brereton, Kitchenham et al. 2007) on Semantic Integration (see section 2.5), Ontology Alignment (see section 2.4.4) and Semantic Web Services (see section 2.6). A systematic literature review is primarily concerned with the problem of aggregating empirical evidence which may have been obtained using a variety of techniques, and in (potentially) widely differing contexts. Performing a systematic review involves several discrete activities, which can be grouped into three main phases: planning; conducting the review; and reporting the review. Fig. 1 illustrates the overall 10-stage review process.

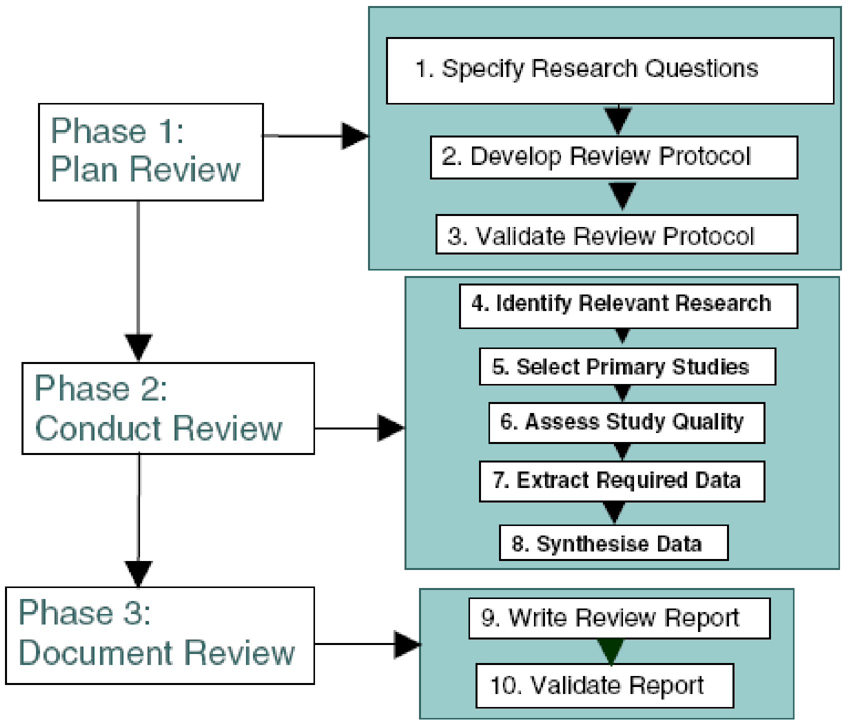


Figure 17: Systematic literature review process (Brereton, Kitchenham et al. 2007).

For modeling, we used three different modeling methods: for understandability reason, we try to use the standard Unified Modeling Language (UML) wherever it is possible. The UML has been the industry standard for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. As the de facto standard

modeling language, the UML facilitates communication and reduces confusion among project stakeholders (Booch, Rumbaugh et al. 2005). For semantic modeling (Hull and King 1987), we stick with the entity-relationship model (Chen 1976). This model incorporates some of the important semantic information about the real world. The entity-relationship model can be used as a basis for unification of different views of data: the network model, the relational model, and the entity set model. Semantic modeling provides richer data structuring capabilities, e.g., for database applications. Semantic modeling provides mechanisms for representing structurally complex interrelations among data typically arising in commercial applications. In general terms, semantic modeling complements work on knowledge representation (in artificial intelligence) and on database models based on the object-oriented paradigm of programming languages.
For feasibility evaluation, we realize prototypes as proof-of-concept of our conceptual approaches (Floyd 1984). The term prototype in connection with software development indicates a primary interest in a process rather than in the prototype as a product. The goal of the prototyping process is the identification of processes which involve an early practical demonstration of relevant parts of the desired software, and which are able to be combined with other processes in system development with a view to improving the quality of the target systems. Many software developers are motivated to employ prototyping by important conclusions drawn from their working experience.
For performance evaluation, we follow the guidelines for empirical research in software engineering (Kitchenham, Pfleeger et al. 2002). The guidelines are intended to assist researchers, reviewers, and meta-analysts in designing, conducting, and evaluating empirical studies. For statistical evaluation, we use descriptive statistics as well as statistical tests.

3.2.2 Evaluation Concept

For investigating these research issues requirements were gathered from two application scenarios from two different industrial application domains. As next step, the EKB framework was applied and adapted to the special requirements of the particular application domain. For each of the two application scenarios (please refer to chapter 5 and chapter 6), all required process steps needed for applying the EKB framework are described in detail. In addition, this thesis aims at identifying generic, domain-independent concepts of the EKB framework which are summarized in chapter 4.
For empirical evaluation (see chapter 7), the following evaluation criteria were established. We classified the evaluation criteria into three categories, namely general functionality, error recovery, and industrial application; with the main focus lying on the first category.

3.2.2.1 General functionality

For the general functionality and runtime of the EKB framework, effectiveness, efficiency and performance are the major evaluation criteria as described below:

- ***Effectiveness:*** the feasibility, validity and correctness of the proposed EKB framework regarding the original requirements, as well as the possibility to practically implement the EKB framework

- ***Efficiency:*** the effort needed for setting up the EKB framework, as well as the effort needed for typical engineering tasks when supported by the EKB framework

- ***Performance:*** the run-time performance of the EKB framework, i.e., the time needed by the EKB framework to perform certain transformations or analyses

3.2.2.2 Error recovery

The EKB framework should be able to recover itself in case of errors, therefore the evaluation criterion robustness is the major evaluation criterion for this category as described below:

- ***Robustness:*** the identification and handling of defects by the EKB framework, as well as the susceptibility of the EKB framework regarding typical failures in engineering processes

3.2.2.3 Industrial application

For industrial application and acceptance, the major evaluation criteria are scalability and usability of the proposed EKB framework, as described below:

- ***Scalability:*** the extendibility of the EKB frameworks architecture, i.e., how well the performance of the EKB framework can be increased by increasing e.g., computational power

- ***Usability:*** the usability of the EKB framework for typical non-IT personnel, e.g., from the production automation domain, as well as efforts needed for training

3.3 *Application Scenarios*

The EKB framework is applied to two different applications domain, namely the Air Traffic Management (ATM) domain and the Production Automation domain to show the frameworks ability to adapt to changed domain-specific requirements. The following two subsections shortly describe the two application scenarios and their special characteristics.

3.3.1 System Wide Information Sharing (SWIS)

SWIS is targeted as an information sharing network within the Air Traffic Management domain. This domain is characterized by very demanding safety and security requirements as well as the need for high availability, leading to conservative IT structures at present.

Today companies and organizations today operate in a highly complex environment requiring well-defined but flexible means for communication and cooperation that can be easily adapted to potentially frequently changing business processes. Traditionally, most organizations have developed IT infrastructures consisting of numerous stand-alone applications, which are connected via point-to-point links. Such infrastructures cannot always support increasing demands for more flexibility and, in particular, more interpretability. Over the last years several approaches have been taken to solve this architecture problem, like Enterprise Application Integration as a concept and the Service Oriented Architecture. These approaches provide mechanisms for a flexible interconnection of various business applications in one domain.

In the ATM case (see also Figure 18), many actors are involved (e.g. airports, airlines, military users, General Aviation, Air Traffic Service Providers, Air Traffic Flow Management instances, providers of meteorological and other data…). In the past, their actions and decisions were more or less de-coupled from each other; however, the expected growth of air traffic in the next decades is anticipated to force all ATM actors towards co-operative handling of virtually shared information during the entire life cycle of a flight. All ATM strategic documents regard the concepts of Collaborative Decision Making (CDM) and System-Wide Information Management (SWIM) as key enablers for sustained growth of air traffic. However, these high-level concepts imply an appropriate underlying technical solution for such co-operative handling of information is in place and that operational interoperability has been established between the involved actors. In other words, for the high-level concepts to work, a low-level mechanism for information sharing shall be established and the corresponding operational procedures and practices agreed and installed at all actors' premises. Currently, only relatively conservative communications capabilities and mechanisms can be found in some domains that cannot be really seen as a mean for information sharing.

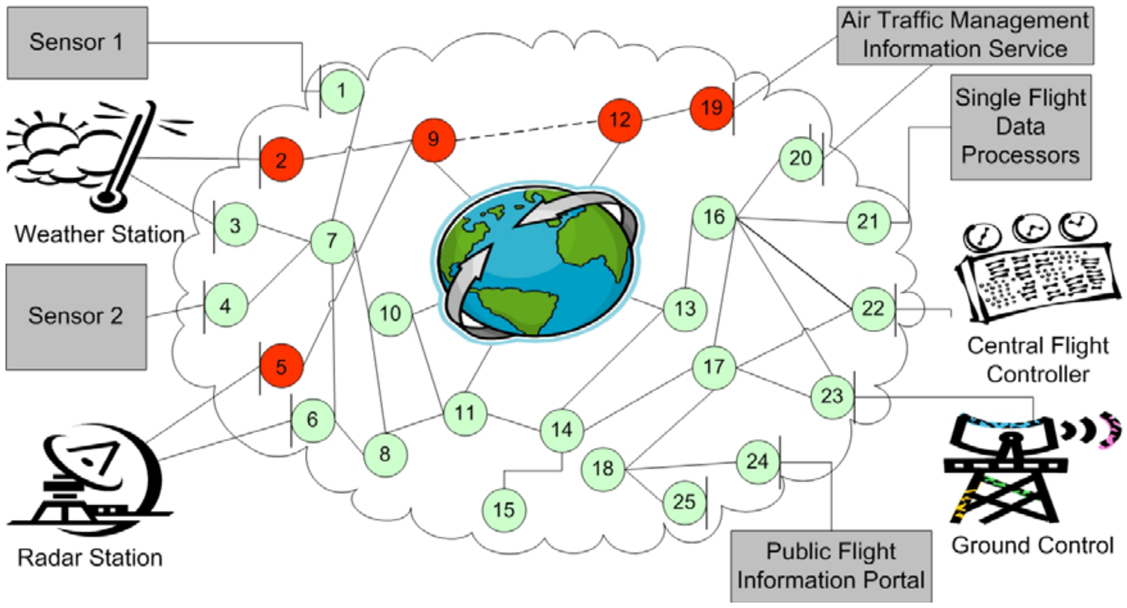

Sensor 1
Weather Station
Sensor 2
Radar Station
Air Traffic Management Information Service
Single Flight Data Processors
Central Flight Controller
Ground Control
Public Flight Information Portal
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

Figure 18: Overview of an ATM environment (Moser, Mordinyi et al. 2009).

In the ATM environment, the degree of heterogeneity of existing legacy systems, solutions, actors, their practices, and preferences may well preclude any "end-to-end" interoperable solution. It is essential to keep low-level information sharing mechanisms strictly de-coupled from high-level applications that rely upon these mechanisms. The demand for an improved solution in the ATM domain generates a need for the development of a "System-Wide Information Sharing Network" (SWIS) based upon adequate and sound concepts. SWIS has to enable sharing of information in a highly distributed environment, taking demanding requirements regarding performance, scalability, maintainability, safety and security into account. In a SWIS-based solution, SWIS should just provide basic harmonized mechanisms for information sharing that are required by all actors; each actor can then use these basic SWIS capabilities to make the best possible use of available information for his own local applications and operational purposes.

3.3.2 Simulation of Assembly Workshops (SAW)

The major goal of the Simulation of Assembly Workshops (SAW) project was to design and implement a simulator that uses a multi-agent system to represent an automated production system able to carry out specific production sequences. These assembly tasks are bedded into a production planning process which allows generating a complete production planning and control process cycle beginning from the request of goods by a customer towards the delivery at the end of the assembly line out of the inventory. The use of a simulator provides the advantage to find a nearly optimal solution to arrange entities like machines, transport systems and robots or to try out different production strategies to fulfil incoming orders to reach the production goals. The simulator usage allows an easy entrance to verify all possible production sequences by changing the different production parameters influencing the process. Basing on the results various decisions can be made to optimize the calculated production plan.

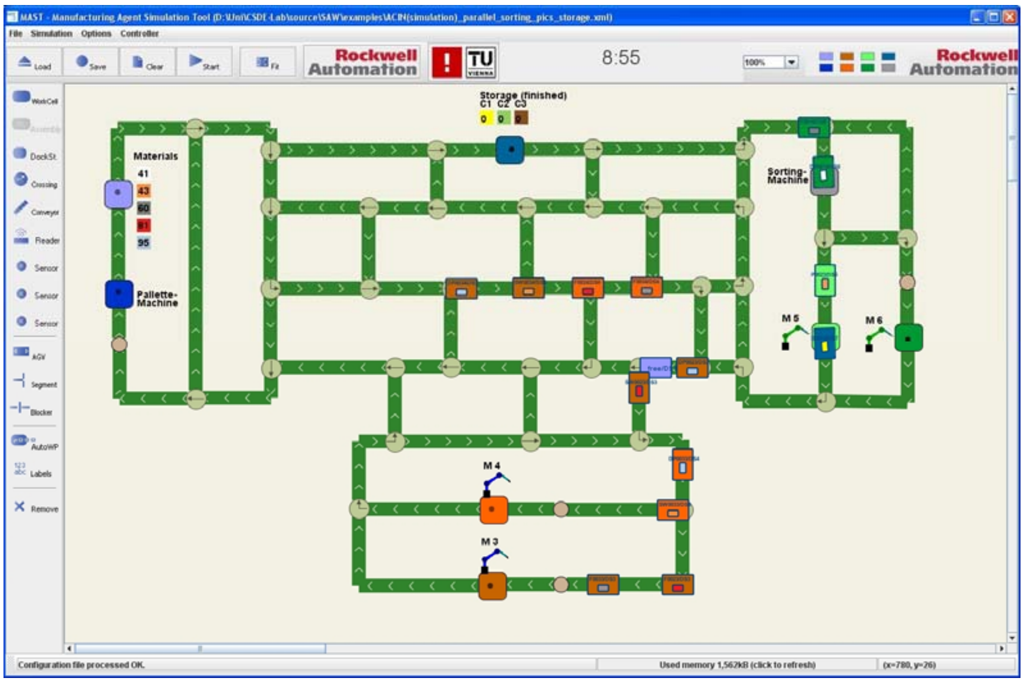


Figure 19: Screenshot of the SAW simulator (Merdan, Moser et al. 2008).

The simulator realized in the SAW project (see Figure 19) bases on Multi-Agent Systems (MAS) and has its roots in the Distributed Artificial Intelligence (DAI) domain and is implemented using the Java Agent Development Framework (JADE)[4]. The various agents of the system act as community to solve the production problem handed over to the production simulator. The simulated system representation using several agents facilitates an efficient evaluation and optimization of the production system performance. The agents act autonomously and heterogeneously, taking their own knowledge and the received knowledge out of the communication with other agents in the environment and manage his next actions due to this information. So all agents in the system try to solve their own local task but always keep their common goal to achieve the production process in focus. The information transition between these agents is essential for the correct function of the system. For example a machine has to inform its logical predecessor in the production sequence, for example the crossing redirecting the goods to the transport system leading to the machine, that it is not reachable because of a damage to prevent the whole system for overall breakdowns.

[4] http://jade.tilab.com/

This interaction is realized using the Agent Communication Language of the Foundation for Intelligent Physical Agents (FIPA-ACL)[5].

The chosen assembly workshop of the SAW project is based on a miniature model situated at the Odo Struger lab of the Automation Control Institute (ACIN)[6] at the Vienna University of Technology. The software simulator to build up the assemble line bases on the production system simulation kit origins from Rockwell Automation International Research situated in Prague (Vrba 2003). During the SAW project, this tool to create assembly lines out of agents like docking stations, machines, conveyor belts, crossings and sensors was enhanced with further intelligence to be able to simulate more complex behaviours, e.g. sorting machines and waiting loops, which are needed to simulate the production of more complicated products consisting of parts where the assembly sequence is important.

Furthermore the simulator of Rockwell is extended with coordination agents that represent the interface between the more business-oriented layers which are responsible for the order dispatching and the rather technical layer responsible for the workshop floor simulation. By feeding this simulation system with different parameter settings, it can be used as a test system for various possible scheduling strategies on an assembly line with redundant machines. This parallel machine scheduling problem is defined as a production system that has to fulfil the outlined tasks on the available machines with the constraints of a number of underlying conditions. The simulator tries to find out the obvious production sequence for the tasks. Taking all these facts into consideration the production system represented by the simulator can be defined as a closed-queuing transfer network with redundant paths through the different lines and nodes.

The focus of the SAW project lies on the design process of the described simulator. Therefore the whole production process beginning on the business layer down to the technical execution layer with the in between lying simulation possibility to optimize the production planning has to be analyzed and represented. A central coordination component needs a global view onto the system to make proper decision which management and production steps are useful to be done next. This coordination is done by a so called "dispatcher" who interprets all available information in the system to calculate an adequate solution or tries out different possibilities by using the possibilities of the simulation if there is enough time. This dispatcher acts upon coordination patterns which guarantee the hierarchical organisation of the agents within the system. It is a kind of decision hub coordinating the communication between the upper business layer (where the dispatcher is rather situated in and acting from) and the down lying technical/operational layer of the workshop (where the production entities represented by agents fulfil their assigned working tasks).

The production planning and control process allows a well arranged layer view onto the production process. The layer concept provides a view for all the different involved

[5] http://www.fipa.org/repository/aclspecs.html

[6] http://www.acin.tuwien.ac.at/

roles during the production process. Hence, the break-down process of incoming orders to single working steps for machines can be separated into layer which fit to their level of aggregation. Figure 20 shows the separation of the business process of the production in an organization into different layers basing on the various views of involved roles. Each role has different responsibilities and passes information to the other roles leading to a top-down information flow as well as to a bottom-up up flow of the same information.

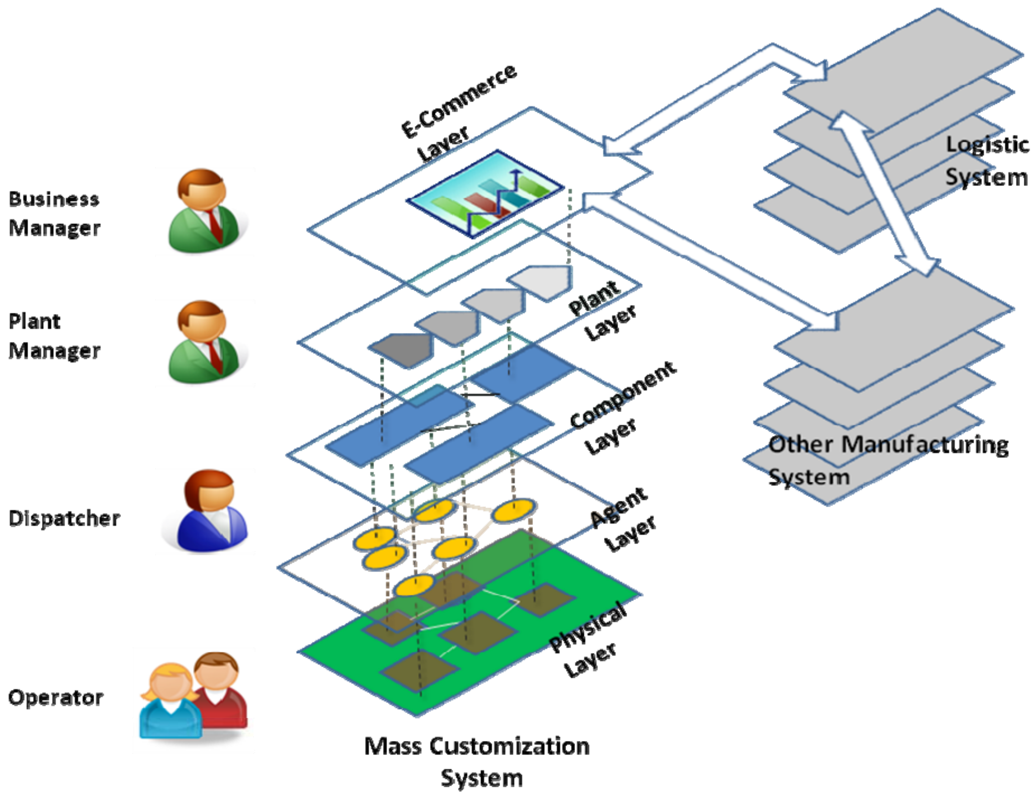


Figure 20: SAW layer model for production processes (Moser, Merdan et al. 2010).

Based on this layer concept, the business process cycle of the production planning and control together with a simulation of incoming order towards the real-life production, leads to the identification of various layers in the project in congruence to the introduced layer concept. Because of the large quantity of information processed during the production and the different roles which are only interested in relevant information a further consideration about the data management had to be done. The dependencies of role specific data, the resulting data structures and the congruency with the layer concept leaded to the approach to realize a knowledge base by using the technology of ontologies. Ontologies also provide the possibility to separate information on several self defined layers with using the area concept.

Chapter 4

4 Engineering Knowledge Base Framework

This chapter summarizes the proposed Engineering Knowledge Base (EKB) framework. In the first section, an overview of the framework is given, as well as an explanation of the used technologies and preconditions for the usage of the EKB framework. In addition, the first section summarizes challenges of the EKB framework and tries to classify the approach regarding related approaches. The second section presents usage scenarios that could benefit from using the EKB framework. Finally, the third section details the generic architecture of the EKB framework as well as the two major phases of the process of using the EKB framework.

4.1 *Overview*

Industrial automation systems depend on distributed software to control the system behavior. The behavior of automation systems must be testable and predictable to meet safety and quality standards. Modern automation systems have to be designed for better interoperability and flexibility to satisfy increasing customer needs for product variety, manufacturing agility, and low cost. In systems engineering, software engineering tasks depend on specification data and plans from a wide range of engineering expert domains in the overall engineering process, e.g., physical plant design, mechanical, and electrical engineering, and production process planning. This expert knowledge is embodied in domain-specific standards, terminologies, people, processes, methods, models, and software (Lüder, Peschke et al. 2004).
However, a major challenge in current industrial development and research approaches is insufficient semantic model integration between the expert disciplines (Schäfer and Wehrheim 2007; Biffl, Sunindyo et al. 2009). Different and partly overlapping terminologies are used in these expert disciplines, which hampers understanding. Consequently, the weak tool support for semantic integration of the expert knowledge across domain boundaries hinders flexible engineering process automation and quality management, leading to development delays and risks for system operation.
The strategic goal of making the systems engineering process more flexible without delivering significantly more risky end products translates into the capability to efficiently re-configure the engineering process and tool instances of a project environment. While there are approaches based on a common repository that holds all relevant project data (Schäfer and Wehrheim 2007), experience has shown that such a repository tends to get large, inflexible, and hard to maintain surprisingly fast, which

makes the knowledge in the repository hard to reuse in new projects. Further, if several organizational units are involved in a project, even agreeing on a common data model is difficult. Thus a key goal is to allow all participants to continue using their own data models and provide a mechanism for translation between these data models. In the past several approaches for providing engineering knowledge in machine-understandable syntax have been investigated (McGuire, Kuokka et al. 1993; Lovett, Ingram et al. 2000; Liao 2005). However, these approaches focus primarily on storing existing homogeneous knowledge rather than providing support for managing and accessing heterogeneous knowledge, which is the focus of this thesis.

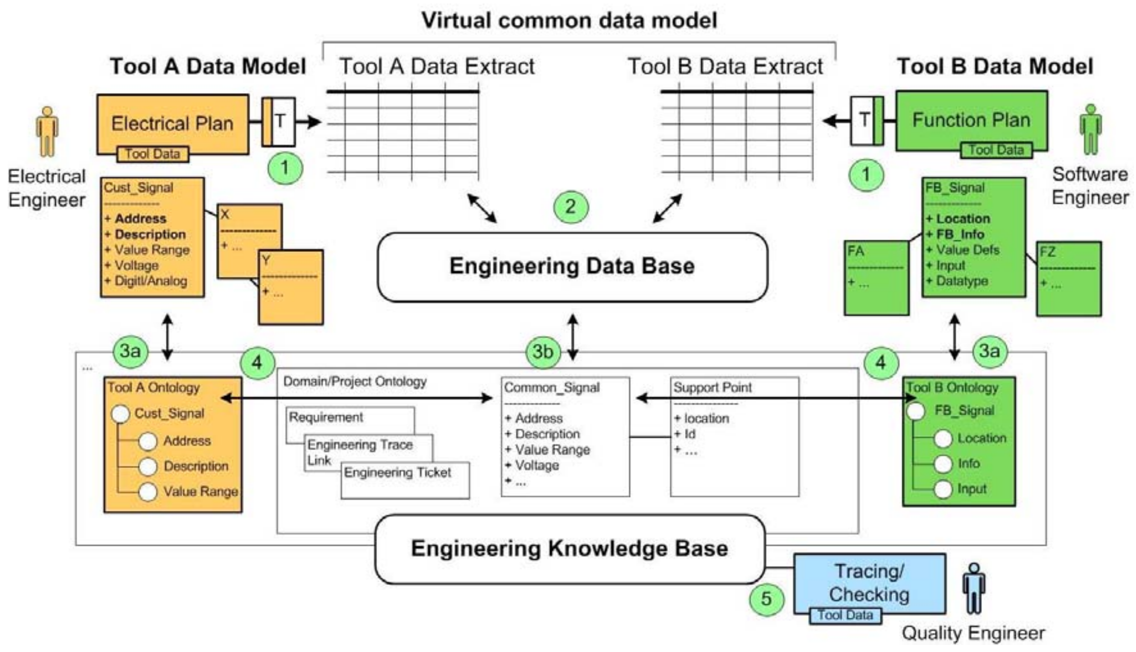


Figure 21: Overview Semantic Integration Approach (Moser, Biffl et al. 2010).

In this thesis, we introduce an generic approach for semantic integration in systems engineering (see Figure 21) with a focus on providing links between data structures of engineering tools and systems to support the exchange of information between these engineering tools and thus making systems engineering more efficient and flexible. Our approach is the so called Engineering Knowledge Base (EKB), an ontology-based data modeling approach which support explicit modeling of existing knowledge in machine-understandable syntax. Therefore, we can automate on project level processes that build on this machine understandable knowledge The EKB framework stores the engineering knowledge in ontologies and provides semantic mapping services to access design-time and run-time concepts and data. The EKB framework aims at making tasks, which depend on linking information across expert domain boundaries, more efficient. A fundamental example for such engineering tasks is checking the consistency and integrity of design-time and run-time models across tool boundaries (Moser, Biffl et al.

2010). The numbered tags in Figure 21 represent the process steps and major components of the EKB framework and are explained in detail in section 4.3.1.
The top part of Figure 21 illustrates the use case scenario with the EKB framework for engineering a production automation system. In this example, there are two types of engineers (electrical engineer, software engineer) who come from to different engineering domains respectively. These roles use specialized engineering tools for their tasks. These tools contain local data sources, which produce and/or consume data with heterogeneous data structures. The EKB is used to facilitate the efficient data exchange between these engineering tools and data sources by providing a so-called "virtual common data model". Based on this data exchange, more complex engineering process tasks like model checking across tools are supported. The bottom part of Figure 21 shows the internal architecture of the EKB, which is described in more details in section 4.3.1.

4.1.1 Used Technologies

This section shortly summarizes the major technologies used for the implementation and usage of the EKB framework.

4.1.1.1 Jena

Jena[7] is an open source java framework for building semantic web applications. Jena evolved during the work with the HP Labs Semantic Web Research. Jena provides API's for RDF, RDFS, OWL and SPARQL. Further a rule-based inference engine is offered by the framework. A benefit of Jena is that it is documented well and many helpful examples do exist. In Jena, a separate graph structure holds each imported ontology document. This fact is most important, otherwise it would be impossible to trace where a statement came from. Each arc in an RDF model is called a statement. Each statement asserts a fact about a resource. A statement is a triple consisting of a subject, predicate, and object. The subject is the resource from which the arc leaves. The predicate is the property that labels the arc and the object is the resource or literal pointed to by the arc.

4.1.1.2 Apache Lucene

Apache Lucene[8] is an API for a search engine allowing efficient text search of large archives. The basic unit of Lucene is a "document" which contains "fields". Every

[7] http://jena.sourceforge.net/
[8] http://lucene.apache.org/

document along with its fields is stored in a central index, where the data is rearranged in a way to allow fast lookup by any combination of fields. This is very similar to the main EKB concept, with the exception that in Lucene not all document fields (~key-value pairs of GenericContent) may be indexed and/or optimized for search. Large base64-strings, i.e. the binary content of any proprietary file format may not be accessible by search, only metadata such as name, date, author and so on may lie in the index; the remaining data will be persisted.

4.1.1.3 Java Content Repository (JCR)

Java Content Repository[9] (JCR) Specification states that its motivation is to provide a "common programmatic interface to (...) [content] repositories". JCR was designed to specify an API to mediate between different types of data storages to provide a generic and centralized view, thus providing full data integration. JCR defines three levels of compliance, "Level 1", basically defining read-only access, export and the data representation, "Level 2", which is mostly about write access and "Optional/Full compliance" which adds transactions and versioning. The basic representation of data is a "repository" that holds a tree of "nodes" with "properties", not to be confused with leaves, that contain the actual data. Nodes may also hold an ID for direct reference (i.e. if the data to integrate is based only on keys and not on hierarchical structures) and can be versionable. Further features of JCR include XPath-and SQL-like queries (at the same time), user access control and distributed repositories.

4.1.1.4 Protégé

Protégé[10] is an open source ontology editor and knowledge-base framework. Protégé was developed by Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine. With the framework it is possible to export the Protégé ontologies into a variety of formats namely RDF(S), OWL, and XML schema. At its core, Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Protégé can be customized to provide domain-friendly support for creating knowledge models and entering data. Further, Protégé can be extended by way of a plug-in architecture and a Java-based Application Programming Interface (API) for building knowledge-based tools and applications.
The Protégé platform supports two main ways of modeling ontologies, in this thesis we primarily focus on the second way:

[9] http://jcp.org/en/jsr/detail?id=283
[10] http://protege.stanford.edu/

- The Protégé-Frames editor enables users to build and populate ontologies that are frame-based, in accordance with the Open Knowledge Base Connectivity protocol (OKBC). In this model, an ontology consists of a set of classes organized in a subsumption hierarchy to represent a domain's salient concepts, a set of slots associated to classes to describe their properties and relationships, and a set of instances of those classes - individual exemplars of the concepts that hold specific values for their properties.

- The Protégé-OWL editor enables users to build ontologies for the Semantic Web, in particular in the W3C's Web Ontology Language (OWL). An OWL ontology may include descriptions of classes, properties and their instances. Given such an ontology, the OWL formal semantics specifies how to derive its logical consequences, i.e. facts not literally present in the ontology, but entailed by the semantics. These entailments may be based on a single document or multiple distributed documents that have been combined using defined OWL mechanisms.

4.1.1.5 SPARQL

SPARQL[11] is an RDF query language; its name is a recursive acronym that stands for SPARQL Protocol and RDF Query Language. It was standardized by the RDF Data Access Working Group (DAWG) of the World Wide Web Consortium, and is considered a key semantic web technology. On 15 January 2008, SPARQL became an official W3C Recommendation. SPARQL allows for a query to consist of triple patterns, conjunctions, disjunctions, and optional patterns. Furthermore, SPARQL queries hide the details of data management, which lowers costs and increases robustness of data integration on the Web.

4.1.2 Scope and Preconditions

The scope of the EKB framework is the support of multi-disciplinary engineering teams that produce software. Each discipline has specific engineering models and tools. These engineering models work well for the specific discipline or expert, but are not well designed for interdisciplinary cooperation. These engineering teams follow an (at least implicit) engineering process, e.g. the V-Model XT (activities, roles, tools) (Broy and Rausch 2005).
The target audience of the EKB framework are the following three stakeholder classes: engineering domain experts, knowledge beneficiaries and ontology experts. Engineering

[11] http://www.w3.org/TR/rdf-sparql-query/

domain experts, e.g., engineers, want to effectively and efficiently follow their engineering process. However, often problems like high effort to perform tasks like change impact analyses or model checks across domain boundaries or risk of defects hinder them in following their engineering process. Knowledge beneficiaries, e.g., project or quality managers, want to monitor, control and improve engineering processes. This intention is often complicated by the needed high effort for performing cross domain process analyses, as well as by the impossibility to easily re-use these analyses in other projects. Finally, ontology experts, e.g., semantic technology expert with general engineering know-how, want to design and validate semantic solutions in an engineering application context.
The major precondition for using the EKB framework is a working communication link between the engineering tools to be integrated. An existing approach to (re-)integrate the tools has, among other solutions, led to the concept of the Enterprise Service Bus (ESB) (Chappel 2004). Its idea is to provide a common infrastructure for tools to communicate with each other. However, current ESB implementations only provide limited possibilities for integration out of the box. The Open Engineering Service Bus (OpenEngSB) (Biffl, Schatten et al. 2009) aims at extending the capabilities of an ESB by introducing "tool domains". Any tool used is considered to have two properties; First, it is part of a workflow (else no integration would be required) and second, it can be replaced by another tool providing similar functionality (even if this may only be a newer or older version of the same tool). The first property implies that tool domains are in some relation to other tool domains which can be abstracted from the specific tools. The second property leads to the conclusion that every tool of a domain has a similar data model and provides certain services which are usually a single step of a workflow. Therefore tool domains can be used when modeling workflows with no need to explicitly address tools. This abstraction of course leads to the necessity of "tool connectors" used to connect a tool to its corresponding tool domain. With tool domains available, existing workflows can be improved, supervised and assisted more easily. This helps engineering groups to increase their efficiency by ensuring a correct and complete flow of information and a proper chain of actions. Creation and tracing of these workflows is required by engineering process engineers who describe workflows on a domain level (i.e., tool-independent) and by project and quality managers who need to trace and validate processes across tool boundaries.

4.1.3 Classification of the Engineering Knowledge Base Framework

This section tries to classify the novel EKB framework by providing on the one hand side differentiations to other technologies and approaches, and on the other hand side by mentioning limitations of related approach which hinder an efficient and effective realization.

4.1.3.1 *Usage of standards in development processes*

A possible solution approach is the usage of standards (e.g., RUP[12], SysML[13]) for platforms, data models, modeling languages and tools in the development process. This works well, if the standard is defined in an early phase of the project and if all project partners adhere to the standard, however, it is hard to define and maintain standards for cross-domain engineering, and even harder or nearly impossible for a larger number of project partners to agree on a standard, which usually takes longer than the time horizon of the project (Kruchten 2000; Weilkiens 2008).

The Systems Modeling Language (SysML) is a general-purpose modeling language for systems engineering applications. It supports the specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems. SysML was originally developed by an open source specification project, and includes an open source license for distribution and use. SysML is defined as an extension of a subset of the Unified Modeling Language[14] (UML) using UML's profile mechanism. SysML reuses seven of UML 2's thirteen diagrams, and adds two diagrams (requirements and parametric diagrams) for a total of nine diagram types.

The advantages of SysML over UML for systems engineering become obvious if you consider a concrete example, such as modeling an automotive system. With SysML you can use Requirement diagrams to efficiently capture functional, performance and interface requirements, whereas with UML you are subject to the limitations of Use Case diagrams to define high-level functional requirements. Likewise, with SysML you can use Parametric diagrams to precisely define performance and quantitative constraints. UML provides no straightforward mechanism to capture this sort of essential performance and quantitative information.

However, many projects do not yet use a common data schema which could be represented using SysML; therefore it is important to provide mechanisms allowing also participants with different heterogeneous data models to cooperate.

4.1.3.2 *Usage of common project repositories*

The usage of common project repositories ("data dumps") is a typical solution for modern data-driven tool integration which well solves the challenges of persistency and versioning of data, but poses new challenges since the stored data often is hard to access and query. Databases that are widely used in an engineering context do not allow the storage of heterogeneous concepts and also do not store meta-information.

Bernstein and Dayal (Bernstein and Dayal 1994) define a repository as "*shared database of information about engineered artifacts produced or used by an enterprise*".

[12] http://www-01.ibm.com/software/awdtools/rup/

[13] http://www.omgsysml.org/

[14] http://www.omg.org/spec/UML/2.2/

Examples of such artifacts include software, documents, maps, information systems, and discrete manufactured components and systems (e.g., electronic circuits, airplanes, automobiles, industrial plants). Storing this information in a common repository has several benefits. First, since the repository provides storage services, tool developers do not need to create tool-specific databases. Second, a common repository allows tools to share information so they can work together. Without a common repository, special protocols would be needed for exchanging information between tools. By conforming to a common data model (i.e., allowable data formats) and information model (i.e., schema expressed in the data model), tools can share data and metadata without being knowledgeable about the internals of other tools. Third, the information in the repository is subject to common control services, which makes sets of tools easier to use. Since a repository is a database, it is subject to database controls, such as integrity, concurrency, and access control. However, in addition, a repository system provides checkout/checkin, version and configuration control, notification, context management, and workflow (Heiler 1995).
However, Bernstein and Dayal (Bernstein and Dayal 1994) state that it is unavoidable that many tools will, for the foreseeable future, have replicated heterogeneous repositories, for the following reasons:

- Many existing tools are already committed to a private repository implementation, e.g., database systems. These repositories are already well-tuned to the tool's performance requirements.

- Many tools need to be portable across operating systems. Therefore, they can only depend on a repository manager that runs on those operating systems and there are few such products on the market.

- In an object-oriented world, some objects will be designed to maintain some state that describes the object. It will be some time before repository technology is so mature that all objects will entrust all their state to a shared repository manager.

Thus, the problem of maintaining consistent heterogeneous repositories must be faced. Or we will have to wait for a repository technology to dominate the product world and for tools to be written or re-written to use that technology.

4.1.3.3 Complete Transformation between project data models

The ultimate alternative solution is the complete transformation between data models of tools, i.e., the translation of engineering model parts from one tool for work in another tool. While the vision of this solution is the seamless cooperation between project partners using well-known and established tools and notations, the feasibility of this

approach is hard to verify and the effort required for establishing the needed transformations is considerable.
In the Modale[15] project, Assmann et al (Assmann, Dörr et al. 2005) developed an ontology-based data integration approach in order to realize a seamless integration between cooperating partners in the field of digital production engineering. The major obstacle was the syntactic, structural and semantic heterogeneity of the internally used tools in the digital production engineering domain. As proof of concept, the researchers have also provided a web-service based prototypic implementation of their approach.
However, many questions still remain open, requiring more research effort to be invested. The main directions that concern in the short to medium term revolve around the following:

- a methodology that allows an efficient (even automatic) construction of the necessary models and semantic bridges
- standardized, domain-specific extensions to which would allow a very short start-up time for projects
- issues concerning the integration with each partner's internal processes, in order to achieve minimal disturbances in their existing workflows

4.2 *Exemplary Usage Scenarios of the Engineering Knowledge Base Framework*

This section details four exemplary usage scenarios of the EKB framework in the context of production automation systems engineering, namely the most basic usage scenario, data-exchange between tools, as well as three more advanced usage scenarios, namely model consistency checking across tool boundaries, impact analysis of model value changes, and end-to-end analysis.

4.2.1 Data-Exchange Between Tools

To cooperate the engineers have to exchange relevant parts of the data structures (i.e., information required in another tool should become available as soon as it has been saved in the original tool) in their tools with each other with the goal of a consistent overall view on certain aspects in the project, e.g., when producing a specification for a subcontractor. Currently, every role uses organization-, domain-, and tool-specific data formats and terms, thus the data exchange takes considerable expert knowledge on the

[15] http://www.modale.de

receiving end to make sense of the incoming data, typically as large PDF document or tool-specific import file.
In the SAW context (refer to section 3.3.2), the different types of experts may have their own terminologies. However, at the interfaces between their processes these experts need to have common concepts to cooperate. For example, the business manager uses the term "client purchase", while the software engineer uses the term "business order" for the same concept. The ontology allows mapping between both terms to common concept called "customer order", so we can have a translation between business manager and software engineer.

4.2.2 Model Consistency Checking Across Tool Boundaries

Model checking, i.e., the validation of model data elements regarding their integrity and consistency, typically is performed at project milestones before the model elements can be used in the next stage of engineering. For a safety-critical domain such as the production automation domain, model checking is required for obtaining relevant system certifications. Currently, model checking is limited to single engineering tools or engineering domains. In addition to syntactical checks, plausibility checks of model elements regarding their usage in other engineering domains are needed.
In the SAW research system (refer to section 3.3.2) model checks are necessary after concurrent engineering model changes to ensure that models do not violate design constraints. Design-time model checks include checking the workshop layout validity and the availability of all machine functions needed for valid production orders. Run-time model checks include checking sufficient machine capacity for producing orders planned for a shift and checking the impact of relevant machine/conveyor failures on the overall production output.

4.2.3 Impact Analysis of Model Value Changes

In difference to single system models, where changes of model values have direct impacts on other model elements, model value changes in cross-domain modeling require additional transformation and checks before the actual impact of a value change can be estimated or measured. In the SAW context (refer to section 3.3.2), model checks are necessary after concurrent engineering model changes to ensure that models do not violate design constraints. Design-time model checks include checking the workshop layout validity and the availability of all machine functions needed for valid production orders. Run-time model checks include checking sufficient machine capacity for producing orders planned for a shift and checking the impact of relevant machine/conveyor failures on the overall production output.

4.2.4 End-to-End Analysis

In distributed engineering in heterogeneous environments, typically a set of different models is used along the engineering chain. In order to ensure validity and consistency of the overall engineering process, it is important to ensure that required data fields can be enforced during the whole lifecycle of the engineering chain (Moser, Winkler et al. 2010).

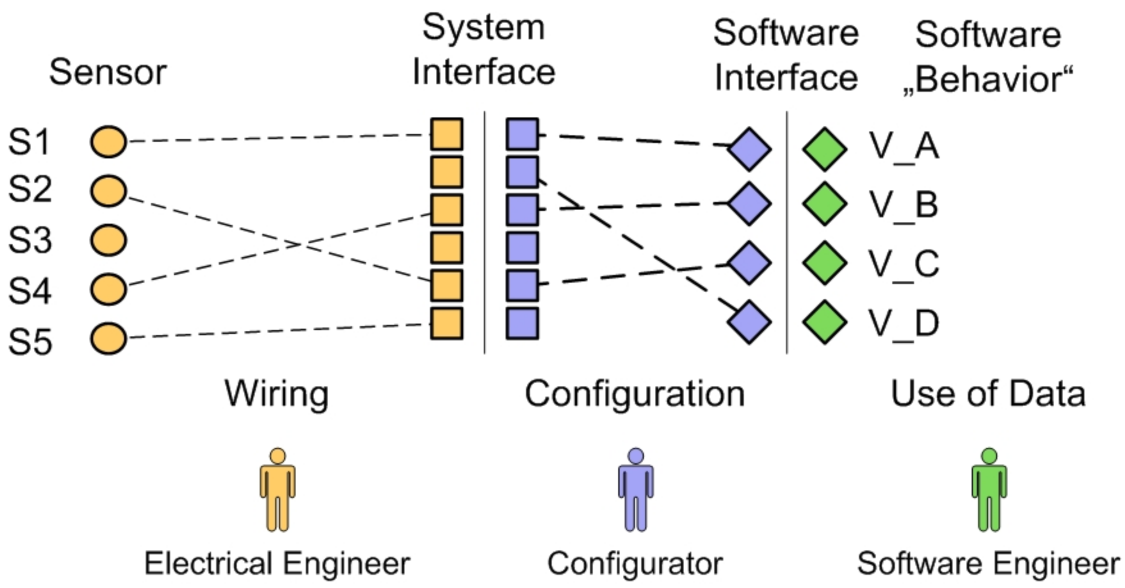


Figure 22: Overview End-to-End Analysis (Moser, Winkler et al. 2010).

In the SAW context (refer to section 3.3.2), this may be defined as a list of hardware sensors and software variables (as shown in Figure 22), which are connected to a system interface by virtual links in models or by wiring in the real-world. Internally, the signals are mapped from the system interface to a software interface, where these signals are represented as variables. A typical consistency and validity check may be used to check whether there exist any incomplete chains between variables and sensors.

4.3 *Architecture and Process of the Engineering Knowledge Base Framework*

This section introduces the Engineering Knowledge Base (EKB) framework, describes the EKB architecture and pictures the process for establishing and using the EKB framework. For understandability reasons, an example from the production automation engineering domain is chosen. The EKB framework consists of two major phases: the preparation of the engineering environment and the use of the project environment in the engineering process. This separation could also be seen as a separation into design time and runtime usage of the EKB. During design time, the data structures and model of the local engineering tools are described and mapped to more general common engineering concepts, while during runtime transformation instructions, which are derived from the design time knowledge, are executed in order to convert data to the required target format.

4.3.1 Generic Engineering Knowledge Base Architecture

This section describes the internal architecture of the EKB, as shown in the bottom part of Figure 21. The general mechanism of the EKB framework uses common engineering concepts identified beforehand as basis for mappings between proprietary tool-specific engineering knowledge and more generic domain-specific engineering knowledge to support transformation between these engineering tools (Moser, Biffl et al. 2010). In the following, the internal architecture of the EKB is described in detail; the numbers directly refer to the numbered tags in Figure 21.

4.3.1.1 Extraction of Tool Data (1)

As first step, the data elements contained in a particular tool need to be extracted in order to be available to the EKB framework. Since by now only a few engineering tools provide APIs for directly accessing the contained data, the export functionality of the tools is used. The exported data then is parsed and transformed into an internal format consisting of key-value pairs for each data attribute, which is easier to handle in the later steps.

4.3.1.2 Storage of Extracted Tool Data (2)

The extracted and transformed key-value pairs are stored using a Java Content Repository (JCR) implementation, the so-called Engineering Data Base (EDB). For data storage, a tree structure is used, and additional functionality like versioning or roll-back is provided. The EDB is indexed and can be queries using Apache Lucene.

4.3.1.3 Description of Tool Knowledge (3a)

The tool ontologies define the engineering-tool-specific, proprietary view on the information exchanged (e.g., a list of signals) in an integration scenario. This includes the view on the format of the information, but can also describe the meaning or the use of the specific view on the existing information, since there can exist multiple views for the same information. The most important part of this description is the definition of the exchanged information, i.e., the definition of the data structures either provided or consumed by a tool.

4.3.1.4 Description of Domain Knowledge (3b)

The domain ontology contains the relevant shared knowledge between stakeholders in the particular application domain (in our case the Production Automation domain) and hence represents the collaborative view on the information exchanged in an integration scenario. In addition, the domain ontology is the place to model standardized domain-specific information (e.g., the description of concepts used throughout an application scenario such as the application domain independent description of business orders or machines in the context of production automation systems). The proprietary information of the engineering tools, which is defined in the tool ontologies, is mapped to the more general information of the domain ontology in order to allow the interoperability with other engineering tools. In contrast to a common data schema, the knowledge stored in the domain ontology is defined on a more general domain level compared to the knowledge stored in the tool ontologies.
This particular domain-specific knowledge described in the domain ontology can easily be updated or transferred to other EKB-based integration scenarios residing in the same domain. This approach allows a broad spectrum of new applications in a particular domain to benefit from the described domain knowledge.

4.3.1.5 Mapping of Tool Knowledge to Domain Knowledge (4)

Each data structure segment described in the tool ontology is mapped to either exactly one particular corresponding domain concept or domain concept attribute described in the domain ontology, or to e.g., all inherited sub-concepts of a target concept. In addition, the granularity of the mapped elements does not need to be the same, so that e.g., a concept can be mapped to the attribute of another concept, or vice versa. This defines the semantic context of the information contained in the segment and allows the detection of semantically similar information consumed and produced by other engineering tools. In addition, the format of the information is described, enabling an automated transformation from source to target format.

4.3.1.6 Usage of the EKB (5)

The mapping of concepts described in the tool ontologies to common concepts described in the domain ontology allows the creation of transformation instructions. These transformation instructions are the foundation to transform data structures

between two engineering tools, because the engineering tools may label or format their data structures in different ways.
Due to the mappings between tool ontologies and domain ontology data structures that are semantically equal can be identified, because they are either aligned to the same domain concept or belong to the same tree segment in the concept tree described in the domain ontology. The transformation instructions can be defined in XML syntax and consist of at least one input and output data structure segment. The segments contain a unique ID and instructions, how the input segment is transformed to an output segment. There is a set of basic transformations that can be combined to more complex transformations, like changing the name of a segment, converting the format using converters, merging or splitting a set of input segments or querying external services for transformation (Moser, Schimper et al. 2009). Based on these transformations, more complex applications can be implemented which use the integrated data of the virtual common data model to perform advanced tasks like tracing of artifacts, consistency checking across tool boundaries, change impact analyses or notification of stakeholders in case of changes.
Now that we have described the general mechanism and the internal architecture of the EKB framework, the next step is the setup and configuration of the EKB framework. As described in (Biffl, Schatten et al. 2009), we suggest to use an enterprise service bus-based approach to integrate engineering tools by describing the data structures they produce and consume as services. The EKB framework acts as a component in the proposed technical integration solution, which performs its transformation service on the message transmitted using the enterprise service bus. After the EKB is set up and configured properly, we show how the EKB framework supports typical engineering tasks, illustrated in the context of the SAW production automation research system.

4.3.2 Preparation of the Engineering Environment

This section describes the process for the preparation of the engineering environment using the EKB framework in more details. Figure 23 gives an overview of the process for the preparation of engineering environments. The following sections describe each process step in details.
As already mentioned, the EKB framework supports engineering process which are not yet fully automated and include media breaks requiring tedious and error-prone manual human work tasks, e.g., re-entering information in other tools because there is no connection between these tools possible yet. Another prerequisite is the solved technical integration, which means that the communication between the engineering tools is established and working correctly, e.g., by using an Engineering Service Bus (Biffl, Schatten et al. 2009). After all these technical heterogeneities have been addressed and solved, there still exist semantic heterogeneities between the involved tools respectively between their underlying data models. That means that either two or more models use

different terminologies for the same concepts or that the concepts are defined and used using different levels of granularity. Goh (Goh 1996) and (Halevy 2005) classified semantic heterogeneities into three main categories: confounding conflicts (e.g., equating concepts are actually different), scaling conflicts (e.g., using different units for the same concept), and naming conflicts (e.g., synonyms and homonyms). The EKB's focus of addressed semantic heterogeneities are confounding conflicts and naming conflicts.

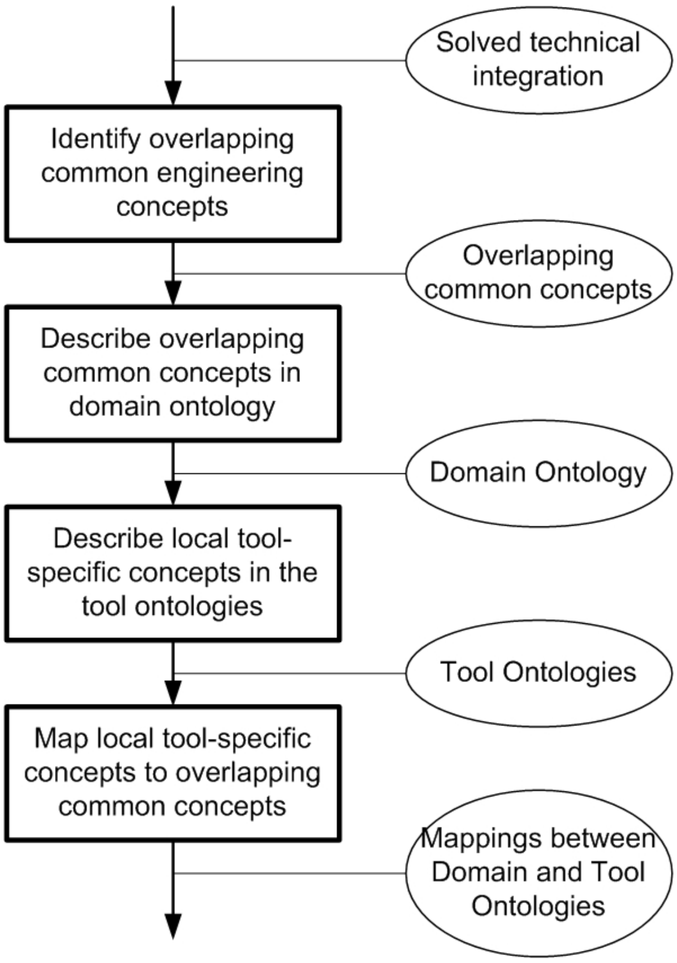

Solved technical integration
Identify overlapping common engineering concepts
Overlapping common concepts
Describe overlapping common concepts in domain ontology
Domain Ontology
Describe local tool-specific concepts in the tool ontologies
Tool Ontologies
Map local tool-specific concepts to overlapping common concepts
Mappings between Domain and Tool Ontologies

Figure 23: Preparation of the Engineering Environment.

4.3.2.1 Identify overlapping engineering concepts

As a first step for the preparation of the engineering environment, the involved models are analyzed to identify overlapping concepts between pairs of engineers and their tools. The top of Figure 24 shows the engineering tool data models of three different engineering roles: namely process engineer (blue), electrical engineer (orange), and software engineer (green). Each of these data models consists of a set of local concepts, as well as a set of common concepts display in the intersection of the three models (white). The goal of this process step is to identify the used common concepts of all participating engineering roles respectively of their used tools and the underlying data models. A good starting point is both the analysis of the interfaces or export artifacts of the involved engineering tools, as well as the identification and analysis of available standards in the problem domain (e.g., AutomationML for the automation engineering domain).

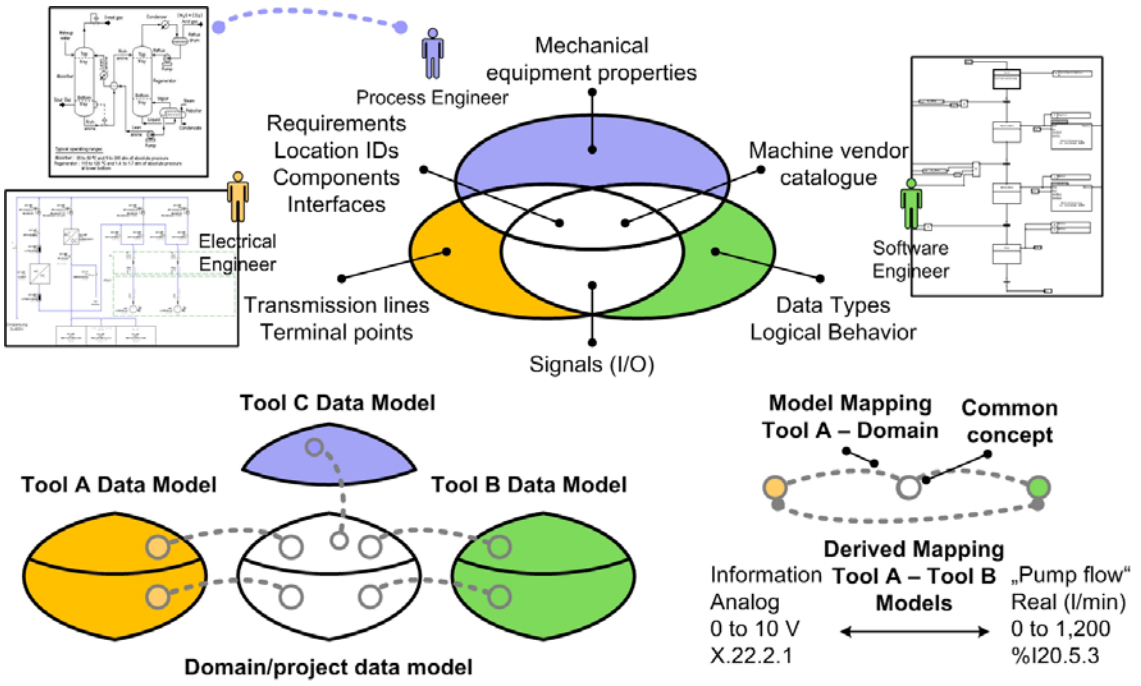


Figure 24: Identification of common concepts across disciplines (Biffl 2009).

AutomationML[16] (Automation Markup Language) is a neutral data format based on XML for the storage and exchange of plant engineering information, which is provided as open standard. Goal of AutomationML is to interconnect the heterogeneous tool landscape of modern engineering tools in their different disciplines, e.g. mechanical plant engineering, electrical design, etc. AutomationML describes real plant components as objects encapsulating different aspects. An object can consist out of other sub-objects, and can itself be part of a bigger composition. It can describe a screw, a claw, a robot or a complete manufacturing cell in different levels of detail. The result of this analysis typically is a set of overlapping engineering concepts used in the engineering process. While this has proven to be true for well-established projects, the overlapping engineering concepts may not yet be available. In order to support future engineering projects, the knowledge regarding existing engineering projects should be used to derive guidelines to support similar future engineering projects.

[16] http://www.automationml.org

4.3.2.2 Describe overlapping common concepts in domain ontology

Once the overlapping engineering concepts are identified, these concepts need to be described or modeled in the domain ontology. In order to support this process step, we assume that the overlapping engineering concepts as well as their relationships will be modeled using well-established modeling standards, e.g., UML or EER diagrams. The models can then easily be transformed into valid OWL ontologies, e.g., by using UML2OWL[17].
With the UML2OWL tool (Leinhos 2006) it is possible to transform an existing UML class diagram into a valid OWL DL document. Thereby, all UML concepts are maintained and transformed. Furthermore, no adjustments or additional enrichments (e.g. through stereotypes) are necessary. Every modeled data or information is transformed and maintained and the resulting OWL DL document can be used e.g. as common basis for application integration.

4.3.2.3 Describe local tool-specific concepts in the tool ontologies

Once the domain ontology has been derived from the models and revised, the local tool-specific concepts need to be described or modeled in the particular tool ontologies. Again, we assume that for each of the used engineering tools at least some rudimentary data model exists, which again can be transformed into a valid OWL ontology using the UML2OWL tool. One important aspect is which non-overlapping concepts should be made available in the tool ontologies, because it is important to only store really essential non-overlapping information in the models to avoid big and unhandy tool ontologies.

4.3.2.4 Map local tool-specific concepts to overlapping common concepts

After the tool ontologies are derived and revised, the final step of mapping local tool-specific concepts to overlapping common concepts needs to be performed. This is shown in the left bottom of Figure 24. Here, we differentiate between simple mappings and more complex mappings. Simple mappings are those which are obvious due to their identification and usage during the analysis of overlapping concepts, i.e., concepts which were “elevated” from the tool level to the domain level to allow data exchange. The identification of complex mappings is directly related to the research area of Ontology Alignment (refer to section 2.4.4).
For performing the Ontology Alignment, we use the Ontology Alignment tool Lily (Wang and Xu 2007; Wang and Xu 2008). Lily is an ontology mapping system, and it has four main features: generic ontology matching, large scale ontology matching,

[17] http://diplom.ooyoo.de

semantic ontology matching and mapping debugging. To accurately describe what the real meaning of an entity in the original ontology is, Lily extracts a semantic sub-graph for each entity. Then it exploits both linguistic and structural information in semantic sub-graphs to generate initial alignments. If necessary, using these initial results as input, a subsequent similarity propagation strategy could produce more alignments, which often cannot be obtained by the previous process. The outcome of the Ontology Alignment process is used to suggest possible mappings to engineering experts, the actual mapping then is performed manually, e.g., by accepting suggested mappings of the Ontology Alignment process.
The mappings can then consecutively be used to enable transformation between different local tool concepts as shown in the right bottom of Figure 24 and as described in section 4.3.1.6.

4.3.3 Use of the Project Environment in the Engineering Process

This section describes the process for the use of the project environment in the engineering process using the EKB framework in more details. Figure 25 gives an overview of the process for the use of the project environment in the engineering process. In the following each process step is described in details.

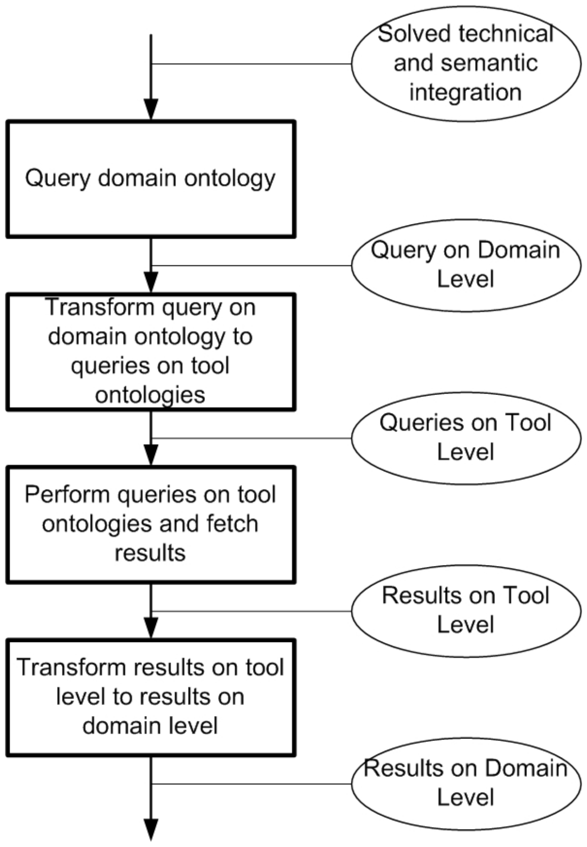

Solved technical and semantic integration
Query domain ontology
Query on Domain Level
Transform query on domain ontology to queries on tool ontologies
Queries on Tool Level
Perform queries on tool ontologies and fetch results
Results on Tool Level
Transform results on tool level to results on domain level
Results on Domain Level

Figure 25: Use of the project environment in the engineering process.

As a first step for the use of the engineering environment, the overlapping engineering concepts described in the domain ontology are queried, using e.g., SPARQL (see section 4.1.1.5). This query is then automatically transformed into a set of queries on the tool ontologies, which include concepts that are mapped to the concepts of the domain ontology that were included in the original query. These tool ontology specific queries are then executed using the Jena framework (see section 4.1.1.1) and the results are fetched. These results then again are transformed into their representation in the domain ontology by exploiting the mappings between tool ontologies and the domain ontology. Finally, the combined results are returned using the representation described in the domain ontology.
For an example of the use of the engineering environment for performing end-to-end analyses refer to section 7.1.4.2.

Chapter 5

5 Semantic Modeling of Requirements and Capabilities for Configuration Derivation

This chapter summarizes the results of applying the EKB framework to an application scenario from the Air Traffic Management domain, as described in section 3.3.1. The three major process parts, namely the modeling of the problem space, the modeling of the solution space and the matching of the problem and the solution space and the generation of system configurations, directly relate to the general three steps of the EKB framework. The modeling of the problem and solution space can be seen as a combination of the **data integration and transformation** step, while the matching of the problem and the solution space and the generation of system configurations can be seen as an **advanced application** provided by the EKB framework.
In the first section, an overview of the application scenario and domain is given, as well as a detailed description of the specific requirements of this application scenario regarding the EKB framework. In the second section, the overall process used for the application scenario is introduced and explained superficially, while the following three sections describe the three major process parts, namely the modeling of the problem space, the modeling of the solution space and the matching of the problem and the solution space and the generation of system configurations, in more details. Finally, the sixth section concludes the chapter and summarizes limitations, extensions and specifics of applying the EKB framework to this specific application scenario.

5.1 *Overview*

In the Air Traffic Management domain complex information systems need to cooperate to provide data analysis and planning services, which consist in the core of safety-critical Air Traffic Management services and also added-value services for related businesses. Air Traffic Management is a relevant and dynamic business segment with changing business processes that need to be reflected in the integration of the underlying information and technical systems.
A major integration challenge is to explicitly model the knowledge embedded in systems and Air Traffic Management experts to provide a machine-understandable knowledge model for integration requirements between a set of complex information systems. Complex information systems consist of a large number of heterogeneous subsystems. Each of these subsystems may have different data types as well as heterogeneous system architectures. In addition, complex information systems typically

have significant quality-of-service demands, e.g., regarding security, reliability, timing, and availability. Many of today's Air Traffic Management complex information systems were developed independently for targeted business needs, but when the business needs changed, these systems needed to be integrated into other parts of the organization (Halevy 2005). Most of the system knowledge is still represented implicitly, either known by experts or described in human-only-readable sources, resulting in very limited tool support for systems integration. The process of adapting the cooperation the business system is traditionally a human-intensive approach of experts from the Air Traffic Management and technology domains (Moser, Mordinyi et al. 2009).

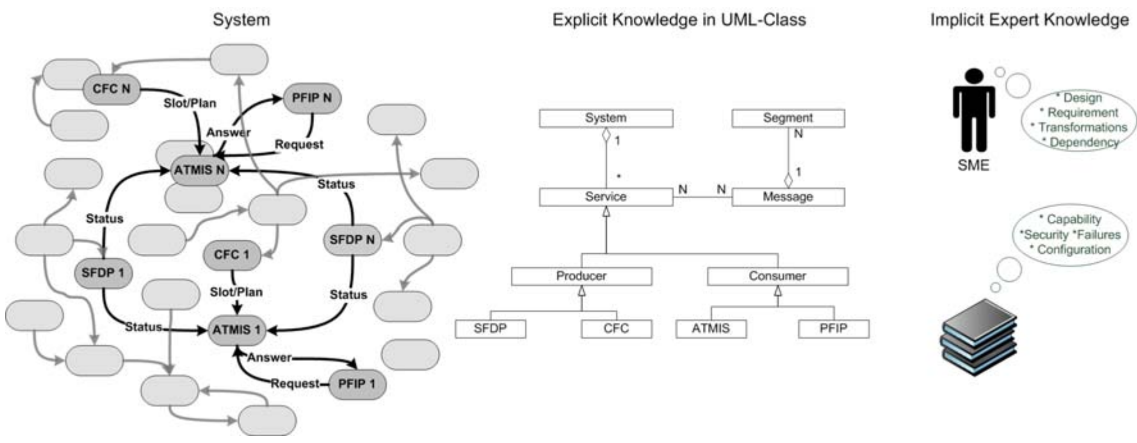


Figure 26: Explicit and Implicit ATM Expert Knowledge (Moser, Mordinyi et al. 2009).

Making the implicit expert knowledge explicit (see Figure 26) and understandable for machines can greatly facilitate tool support for systems integrators and engineers by providing automation for technical integration steps and automatic validation of integration solution candidates. Consequently, we employ the EKB framework as a data-driven approach that explicitly models the semantics of the problem space, i.e., integration requirements and capabilities (Moser, Mordinyi et al. 2009); the solution space, i.e., the connectors, and data transformations between heterogeneous legacy systems (Mordinyi, Moser et al. 2009); and finally provide a process to bridge problem and solution spaces, i.e., find out whether there are feasible solutions and minimize the cost of integration (Moser, Mordinyi et al. 2009).

5.2 *Process Description*

This section describes a traditional UML-based integration process approach, and an EKB-based semantically-enabled integration approach that makes expert knowledge explicit to facilitate tool support. Both process variants are based on a generic integration process described in section 5.2.1 (Moser, Mordinyi et al. 2009).

5.2.1 Generic Systems Integration Process

The generic systems integration process (see Figure 27) consists of 3 major steps: 1. modeling system requirements and capabilities, 2. derivation and optimization of an integration system configuration; and 3. lab/field testing and performance measurement. Between these major steps, Quality Assurance steps are needed for assuring both a correct working system model and a valid integration system configuration (Moser, Mordinyi et al. 2009).

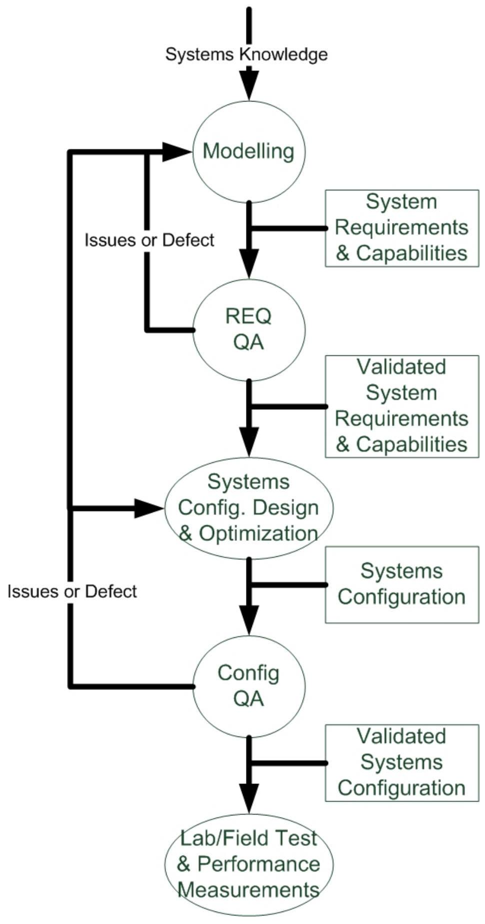

Systems Knowledge
Modelling
System Requirements & Capabilities
Issues or Defect
REQ QA
Validated System Requirements & Capabilities
Systems Config. Design & Optimization
Issues or Defect
Systems Configuration
Config QA
Validated Systems Configuration
Lab/Field Test & Performance Measurements

Figure 27: Generic System Integration Process Steps (Moser, Mordinyi et al. 2009).

5.2.1.1 Modeling of Systems Requirements & Capabilities

Subject Matter Experts provide systems knowledge to describe the data exchange requirements and capabilities of the participating legacy systems. This includes the descriptions of the interfaces to be shared, a detailed description of the exchanged messages types and a description of the global and/or local additional (non-functional) requirements of the systems (e.g., the maximal time allowed for message delivery). Output of this process step is a model representing the requirements and capabilities of the systems to be integrated. Typical requirement and capability models include a) communication contracts for defining the communication capabilities and requirements of business systems; b) policies for reflecting interests of the organizations contributing to systems; and c) infrastructure capabilities for describing the topology and characteristics of the underlying network (Moser, Mordinyi et al. 2009).

5.2.1.2 Requirements Quality Assurance

QA personnel validate and check the model created in the previous step for defects and issues by comparing the knowledge captured in the model with the knowledge given as input to the modeling process step. In case of issues raised, these issues are reported back to the modeling step for resolution (Moser, Mordinyi et al. 2009).

5.2.1.3 Systems Configuration Design & Optimization

The Integration Expert (IE) uses the validated and checked model created in the first process step to derive as output a technical system configuration representing the integration solution for the participating legacy information systems (Moser, Mordinyi et al. 2009).

5.2.1.4 Configuration Quality Assurance

Quality Assurance personnel validate and check the system configuration created in the previous process step for defects and issues (e.g., unsuitable integration partners). This is achieved by comparing the knowledge captured in the systems configuration with both the knowledge captured in the system requirements and capabilities model as well as the knowledge given as input to the modeling process step. In case of issues raised, these issues are reported back to either the systems configuration creation step or the modeling process step for resolution (Moser, Mordinyi et al. 2009).

5.2.1.5 Lab/Field Test and Performance Measurement

The integration tester tests the validated and checked technical system integration configuration in lab and field tests to measure system performance characteristics (Moser, Mordinyi et al. 2009).

5.2.2 Traditional (UML-based) Systems Integration Approach

This section describes a traditional (i.e., UML-based) integration approach (see top process in Figure 28) (Moser, Mordinyi et al. 2009).

5.2.2.1 System Description

For each legacy information system to be integrated, the Subject Matter Expert (SME) responsible for the particular system describes the requirements and capabilities of the system using human-readable language. The outcome of this process step is a set of legacy systems interface description documents (Moser, Mordinyi et al. 2009).

5.2.2.2 Integration Partner Derivation

In order to identify possible and select suitable integration partner legacy systems, the Subject Matter Experts of all participating systems, a domain expert (DE) who is capable of managing the knowledge involved in the problem domain and an integration expert (IE) who is responsible for the actual integration need to cooperate. The integration partner candidates are identified by the Subject Matter Experts by comparing the legacy systems interface description documents created in the previous step and by the domain expert by identifying similar knowledge represented in the participating systems. The integration expert then selects the best fitting integration partners from the pool of possible integration partners. The outcome of this process step is a set of accepted integration partners (Moser, Mordinyi et al. 2009).

5.2.2.3 Transformation Instruction Generation

In order to allow the interoperability between proprietary and heterogeneous legacy information systems, semantic transformation is needed at run time. Instructions are needed to perform these transformations. In this process step, the domain expert and the Subject Matter Experts of the particular affected system cooperate in order to derive these transformation instructions. The outcome of this process step is a document representing the transformation instructions needed for the integration solution (Moser, Mordinyi et al. 2009).

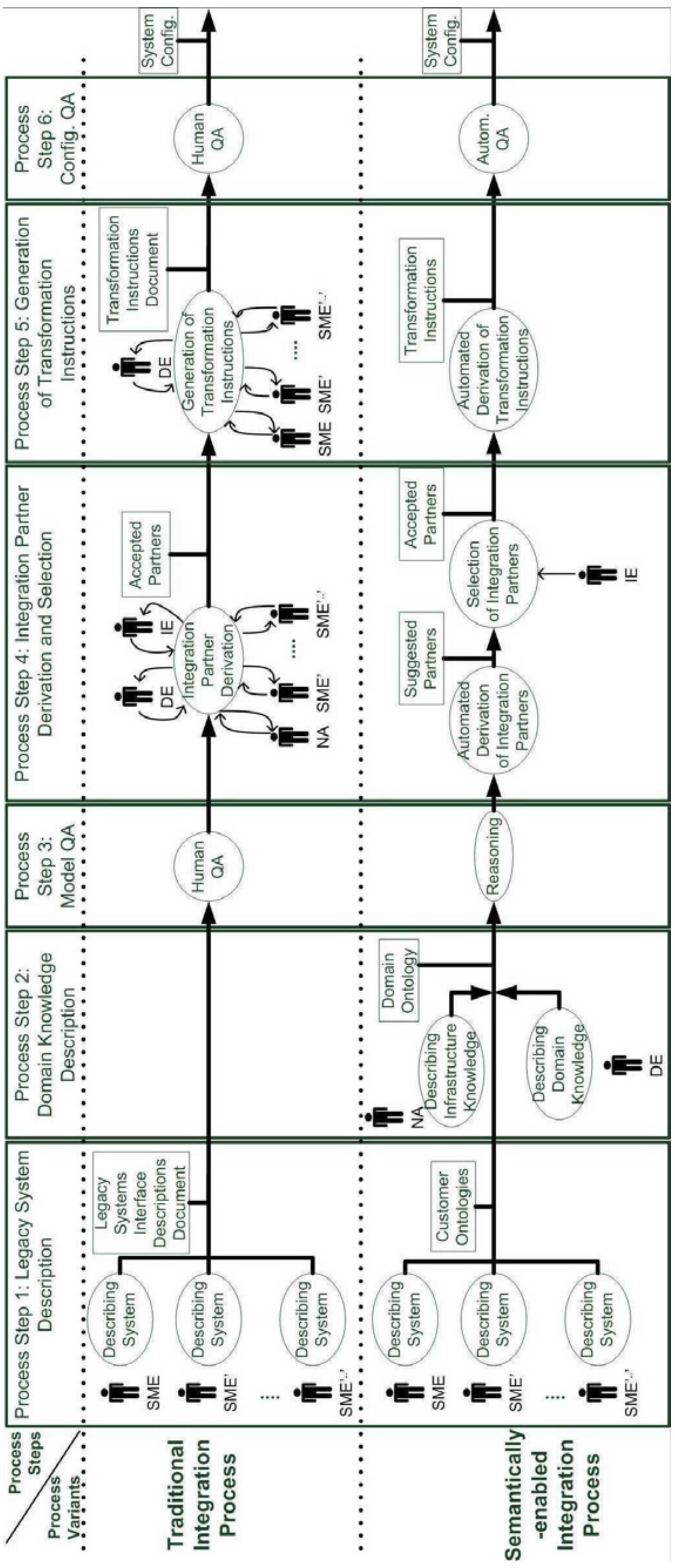

Process Steps
Process Variants
Traditional Integration Process
Semantically -enabled Integration Process
Process Step 1: Legacy System Description
Process Step 2: Domain Knowledge Description
Process Step 3: Model QA
Process Step 4: Integration Partner Derivation and Selection
Process Step 5: Generation of Transformation Instructions
Process Step 6: Config. QA
SME
SME'
SME''...'
Describing System
Legacy Systems Interface Descriptions Document
Human QA
Integration Partner Derivation
NA
DE
IE
Accepted Partners
Generation of Transformation Instructions
Transformation Instructions Document
System Config.
Customer Ontologies
Describing Infrastructure Knowledge
Describing Domain Knowledge
Domain Ontology
Reasoning
Automated Derivation of Integration Partners
Suggested Partners
Selection of Integration Partners
Automated Derivation of Transformation Instructions
Transformation Instructions
Autom. QA

Figure 28: UML-based and EKB-based approaches (Moser, Mordinyi et al. 2009).

5.2.3 Engineering Knowledge Base-based Integration Approach

This section describes the EKB-based semantically-enabled system integration approach (see bottom process in Figure 28). The following subsections summarize the process steps, with special regard to a continuous example from the ATM domain presented in Figure 29 (Moser, Mordinyi et al. 2009).

5.2.3.1 Legacy System Description

For each legacy information system to be integrated, the Subject Matter Expert (SME) responsible for the particular system describes the requirements and capabilities of the system using machine-understandable notations. In comparison to the traditional integration process, the outcome of this process step is a set of ontologies describing the requirements and capabilities of the legacy information system to be integrated, as well as the mapping of this information to general domain knowledge (Moser, Mordinyi et al. 2009).
In the continuous example, there are 4 business systems on the left hand side which provide a total of 5 services that send messages, and 2 business systems on the right hand side which provide a total of 3 services that receive messages. The content of these messages is represented using a tuple-based notation. Additionally, services can define extra requirements, like secure transmission (Moser, Mordinyi et al. 2009).

5.2.3.2 Domain Knowledge Description

In addition to the description of the requirements and capabilities of the participating systems, the domain expert describes the common knowledge of the problem domain used in the integration scenario. This externalized domain knowledge is used by the Subject Matter Experts (SMEs) while describing the particular legacy systems, who map proprietary system information to more general knowledge represented in the domain ontology in order to overcome semantic gaps between legacy systems. On infrastructure level the network administrator (NA) describes the architecture and capabilities of the underlying network. The outcome of this process step is an ontology describing the shared problem domain knowledge as well as the integration network infrastructure. This domain ontology can be reused for several integration scenarios in this domain (Moser, Mordinyi et al. 2009).

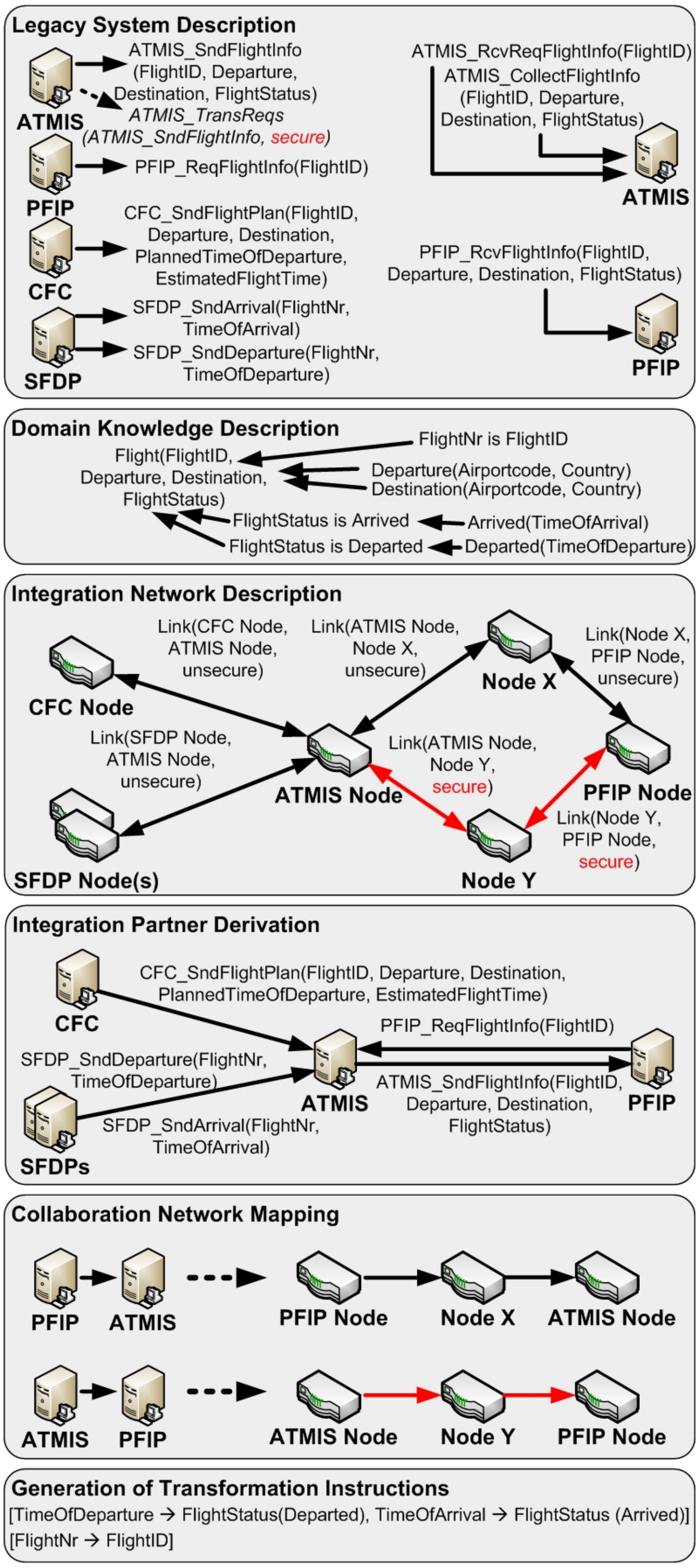

Legacy System Description
ATMIS_SndFlightInfo (FlightID, Departure, Destination, FlightStatus)
ATMIS_TransReqs (ATMIS_SndFlightInfo, secure)
ATMIS
PFIP_ReqFlightInfo(FlightID)
PFIP
CFC_SndFlightPlan(FlightID, Departure, Destination, PlannedTimeOfDeparture, EstimatedFlightTime)
CFC
SFDP_SndArrival(FlightNr, TimeOfArrival)
SFDP_SndDeparture(FlightNr, TimeOfDeparture)
SFDP
ATMIS_RcvReqFlightInfo(FlightID)
ATMIS_CollectFlightInfo (FlightID, Departure, Destination, FlightStatus)
ATMIS
PFIP_RcvFlightInfo(FlightID, Departure, Destination, FlightStatus)
PFIP
Domain Knowledge Description
FlightNr is FlightID
Flight(FlightID, Departure, Destination, FlightStatus)
Departure(Airportcode, Country)
Destination(Airportcode, Country)
FlightStatus is Arrived
Arrived(TimeOfArrival)
FlightStatus is Departed
Departed(TimeOfDeparture)
Integration Network Description
Link(CFC Node, ATMIS Node, unsecure)
Link(ATMIS Node, Node X, unsecure)
Link(Node X, PFIP Node, unsecure)
CFC Node
Node X
Link(SFDP Node, ATMIS Node, unsecure)
Link(ATMIS Node, Node Y, secure)
ATMIS Node
PFIP Node
Link(Node Y, PFIP Node, secure)
SFDP Node(s)
Node Y
Integration Partner Derivation
CFC_SndFlightPlan(FlightID, Departure, Destination, PlannedTimeOfDeparture, EstimatedFlightTime)
CFC
PFIP_ReqFlightInfo(FlightID)
SFDP_SndDeparture(FlightNr, TimeOfDeparture)
ATMIS_SndFlightInfo(FlightID, Departure, Destination, FlightStatus)
ATMIS
PFIP
SFDP_SndArrival(FlightNr, TimeOfArrival)
SFDPs
Collaboration Network Mapping
PFIP
ATMIS
PFIP Node
Node X
ATMIS Node
ATMIS
PFIP
ATMIS Node
Node Y
PFIP Node
Generation of Transformation Instructions
[TimeOfDeparture → FlightStatus(Departed), TimeOfArrival → FlightStatus (Arrived)]
[FlightNr → FlightID]

Figure 29: Continuous EKB-based example (Moser, Mordinyi et al. 2009).

The first part of the continuous example shows the description of the domain knowledge. The domain knowledge is exemplarily represented using a tuple-based notation plus a set of arrows to indicate relationships between domain knowledge elements, e.g., the element “FlightStatus” could either be defined using the element “Arrived” or the element “Departed”, or the elements “FlightNr” and “FlightID” can be treated equally. The second part shows the description of the integration network infrastructure. On the one hand, the architecture of the network is represented by a set of nodes and links which connect these nodes, on the other hands additional capabilities of nodes (e.g., secure transmission) are described (Moser, Mordinyi et al. 2009).

5.2.3.3 Automated Integration Partners Derivation and Selection

The externalized knowledge of the Subject Matter Experts (SMEs), the domain expert (DE), and the network administrator (NA) which was captured in the ontologies in the previous steps is used to automatically derive the set of possible Integration Partner candidates with ontology-based reasoning, allowing an easier and less error-prone identification of possible Integration Partners compared to the traditional integration process. The integration expert (IE) is responsible for choosing suitable Integration Partners from the set of possible Integration Partners derived in the previous step. The outcome of this process step is a set of accepted Integration Partners (Moser, Mordinyi et al. 2009).
The first part of the continuous example shows the derivation of the possible Integration Partners. Based on the legacy system descriptions, the description and mapping of the domain knowledge and the description of the architecture and capabilities of the integration network, the possible sending and receiving service partners are derived using heuristics and ontology-based reasoning (Moser, Schimper et al. 2009). In the example, this is represented as a graph consisting of the possible collaborations (i.e., the services which are able to communicate) and the exchanged messages. The second part shows the mapping of these derived collaborations to the underlying network infrastructure. The example focuses on the collaboration between “PFIP” and “ATMIS”, showing that the request collaboration initiated by “PFIP” used the unsecure route via “Node X”, while the reply collaboration initiated by “ATMIS” used the secure (“red”) route via “Node Y”, as defined in the additional service requirements of the “ATMIS” business system (Moser, Mordinyi et al. 2009).

5.2.3.4 Automated Derivation of Transformation Instructions

In this process step, instructions for the transformations between the participating heterogeneous legacy systems selected in the previous step are automatically derived from the ontologies created in the first 2 process steps. The outcome of this process step

is a set of transformation instructions needed for the integration solution (Moser, Mordinyi et al. 2009).
In the continuous example, 3 exemplary transformation instructions are generated, e.g., the transformation of the element "FlightNr" to the element "FlightID", or the transformation of the element "TimeOfDeparture" to the element "FlightStatus(Departed)" (Moser, Mordinyi et al. 2009).

5.2.3.5 Quality Assurance Steps

There are 2 Quality Assurance steps in the EKB-based integration process, which can be very well supported with tools based on ontology-based reasoning. This allows a much faster and more reliable Quality Assurance compared to the traditional integration process and relieves scarce experts from tedious work (Moser, Mordinyi et al. 2009).

5.3 *Modeling of the Problem Space*

This section pictures the semantic modeling of heterogeneous knowledge using a set of ontologies as model. The ontology architecture (Moser and Anjomshoaa 2007) is described in detail as well as the distribution of the modeled information among the layers (Moser, Mordinyi et al. 2009).
The ontologies used as input models for the derivation of the system configuration are organized using a subdivided architecture, consisting of three different types of ontologies. The ontology types building the semantic model for a specific scenario are the abstract integration scenario ontology (AIS), the domain-specific ontologies, and the integration system ontologies (see Figure 30). The domain ontologies extend the abstract integration scenario ontology by adding concepts describing the common domain knowledge used. In addition, the integration system ontology uses the other two ontologies for aligning its concepts with the more general concepts defined in either the AIS or domain ontology (Moser, Mordinyi et al. 2009).

5.3.1 Abstract Integration Scenario Ontology

The abstract integration scenario (AIS) ontology is defined in an application-domain-independent manner, allowing its use across different domains. This domain independent definition is a powerful mechanism to provide a flexible base for information sharing scenarios, completely independent of a particular domain. The terms in the AIS ontology are defined in an abstract way to simplify the conceivability of the use in different domains (Moser, Mordinyi et al. 2009).

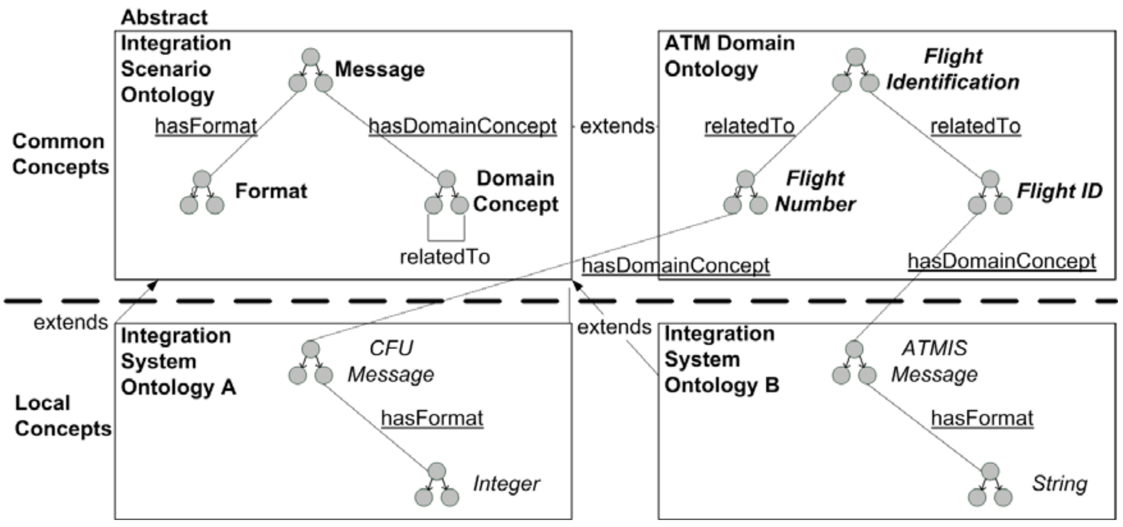


Figure 30: Simplified Ontology Architecture Example (Moser, Schimper et al. 2009).

5.3.2 Domain Ontology

The domain ontology includes the main shared knowledge between stakeholders of the particular domain (e.g., ATM domain) and hence represents the collaborative view on the information exchanged in an integration scenario. In addition, the domain ontology is the place to model standardized domain-specific information. The customers map their proprietary in-formation, which is defined in the integration system ontologies, to the standardized information in order to allow the interoperability with other participants (Moser, Mordinyi et al. 2009).
This domain-specific information is used for the detection of semantically identical information provided or consumed by participating applications or organizations, independent of the format or identifiers used for the information, and therefore improves or enables the communication between these organizations. The identification of possible integration partners is simplified and the tool-supported transformation of semantically identical information existing in different formats allows further communication between new partners (Moser, Mordinyi et al. 2009).
This particular domain-specific knowledge described in the domain ontology can easily be updated or transferred to other EKB-based integration scenarios residing in the same domain. This allows a broad spectrum of new applications in a particular domain to benefit from the described domain knowledge. Instead of modeling the domain know-ledge from scratch it is also possible to use as starting point a description of the problem domain, a so-called "world model". The advantage of this approach is the reduced effort for modeling the domain knowledge; however a tradeoff exists in the complexity of typical "world model" ontologies, resulting in a longer waiting time when searching for concrete domain knowledge (Moser, Mordinyi et al. 2009).

5.3.3 Integration System Ontology

The integration system ontology (ISO) defines the customer-specific, proprietary view on the information exchanged in an integration scenario. This includes the view on the format of the information (as required by the legacy application), but can also describe the meaning or the use of the specific view on the existing information, since there can exist multiple views for the same information. The ISO defines the structure of the legacy applications, services and messages, i.e., the services provided by a legacy application, the messages provided or consumed by a service and the message segments a message consists of, by adding instances of the concepts defined in either the AIS or domain ontology (Moser, Mordinyi et al. 2009).
The most important part of this description is the definition of the exchanged information, i.e., the definition of the messages either provided or consumed by the legacy applications. The ISO describes the semantic context and the format of each message segment, supported by the domain expert. Each message segment is mapped to exactly one particular domain concept. This defines the semantic context of the information contained in the segment and allows the detection of possible collaborations for an integration scenario. In addition, the format of the information is described, enabling automated transformation between formats (Moser, Mordinyi et al. 2009).

5.4 *Modeling of the Solution Space*

This section describes the modeling of the solution space, separated into two subsections, namely the model-driven systems configuration (MDSC) and the integration platform (Mordinyi, Moser et al. 2009).

5.4.1 Model-driven System Configuration

This section describes the model-driven system configuration (MDSC) process (Moser, Mordinyi et al. 2009). As shown in Figure 31, the MSDC process consists of 5 major process steps. In the following subsections, these steps are explained in detail, with regard to the example data presented in Figure 31 (Mordinyi, Moser et al. 2009).

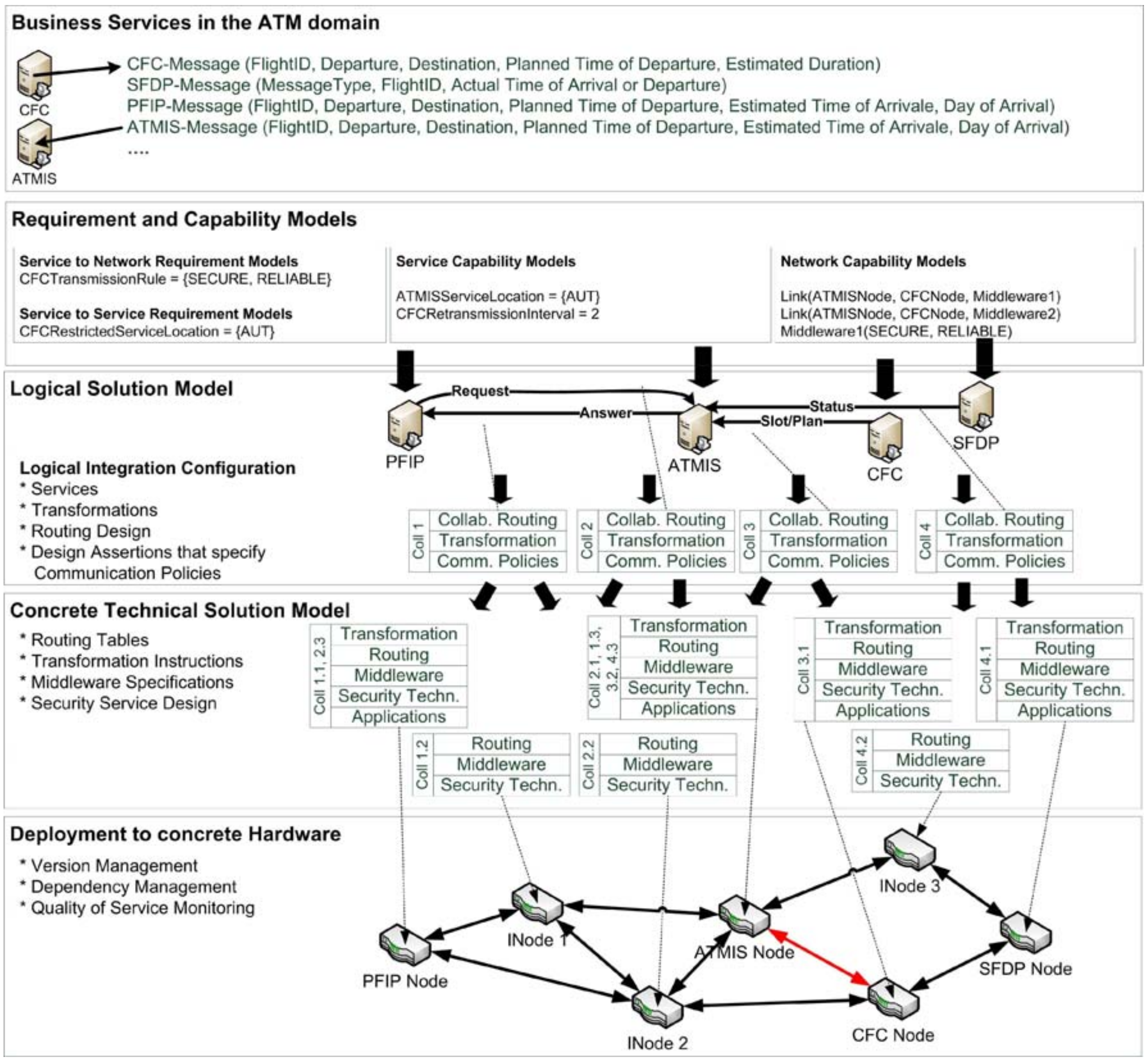
5.4.1.1 Business Services in the ATM domain

For each legacy information system to be integrated, the Subject Matter Expert (SME) responsible for the particular system describes the messages which are either provided or consumed by the business services provided. Both the structure (i.e., data types and

semantic meanings) and the format of the exchanged messages are described (Moser, Mordinyi et al. 2009).
In the example, there are 2 business services shown on the on the left hand side, “CFC” and “ATMIS”. Additionally, 4 message types are presented using a tuple-based notation. The “CFC” service provides the “CFC-Message” consisting of information about a certain flight, i.e., flight number, departure and destination airport, planned time of departure and estimated duration of the flight. In contrast, the “ATMIS” service consumes the “ATMIS Message” consisting of additional flight information compared to the “CFC-Message”, namely the estimated day and time of arrival of a certain flight (Mordinyi, Moser et al. 2009).

5.4.1.2 Requirement and Capability Models

In addition to the description of the provided or consumed message of the participating business services, the business services define extra requirements regarding either possible integration partner candidates (i.e., other business services) or the underlying heterogeneous network infrastructure (Moser, Mordinyi et al. 2009).
The example shows three different kinds of requirement and capability models. On the left hand side, the requirements of the “CFC” service regarding both the transmission using the underlying heterogeneous network infrastructure (e.g., the transmission needs to be both secure and reliable) as well as the required capabilities of possible integration partners (e.g., an integration partner services has to be an Austrian governmental service) are presented exemplary. In the middle, the capability models of business services are shown exemplary. E.g., the “ATMIS” service has a defined service location of “Austria”, and the “CFC” service has a defined retransmission interval of 2 seconds. On the right hand side, the capability models of the underlying integration network infrastructure are presented. The integration network consists of nodes and links. Each link presents a specific middleware technology and may define additional capabilities, e.g., the capability of performing secure and reliable transmissions (Mordinyi, Moser et al. 2009).
In comparison to a traditional integration process, the outcome of the so far described process steps is a set of machine-understandable knowledge models describing both the message structures as well as the requirements and capabilities of the legacy information system to be integrated and the capabilities of the underlying integration network infrastructure (Mordinyi, Moser et al. 2009).



Business Services in the ATM domain
CFC
ATMIS
CFC-Message (FlightID, Departure, Destination, Planned Time of Departure, Estimated Duration)
SFDP-Message (MessageType, FlightID, Actual Time of Arrival or Departure)
PFIP-Message (FlightID, Departure, Destination, Planned Time of Departure, Estimated Time of Arrivale, Day of Arrival)
ATMIS-Message (FlightID, Departure, Destination, Planned Time of Departure, Estimated Time of Arrivale, Day of Arrival)
....
Requirement and Capability Models
Service to Network Requirement Models
CFCTransmissionRule = {SECURE, RELIABLE}
Service to Service Requirement Models
CFCRestrictedServiceLocation = {AUT}
Service Capability Models
ATMISServiceLocation = {AUT}
CFCRetransmissionInterval = 2
Network Capability Models
Link(ATMISNode, CFCNode, Middleware1)
Link(ATMISNode, CFCNode, Middleware2)
Middleware1(SECURE, RELIABLE)
Logical Solution Model
Request
Answer
Status
Slot/Plan
PFIP
ATMIS
CFC
SFDP
Logical Integration Configuration
* Services
* Transformations
* Routing Design
* Design Assertions that specify Communication Policies
Coll 1
Coll 2
Coll 3
Coll 4
Collab. Routing
Transformation
Comm. Policies
Concrete Technical Solution Model
* Routing Tables
* Transformation Instructions
* Middleware Specifications
* Security Service Design
Coll 1.1, 2.3
Coll 2.1, 1.3, 3.2, 4.3
Coll 3.1
Coll 4.1
Transformation
Routing
Middleware
Security Techn.
Applications
Coll 1.2
Coll 2.2
Coll 4.2
Routing
Middleware
Security Techn.
Deployment to concrete Hardware
* Version Management
* Dependency Management
* Quality of Service Monitoring
PFIP Node
INode 1
ATMIS Node
INode 2
INode 3
CFC Node
SFDP Node

Figure 31: Model-driven System Configuration process (Mordinyi, Moser et al. 2009).

5.4.1.3 Logical Solution Model

The externalized knowledge which is captured in the knowledge models created in the previous steps is used to automatically derive the set of possible integration partners using ontology-based reasoning, allowing an easier and less error prone identification of possible integration partners compared to the traditional integration process (Moser, Mordinyi et al. 2009; Moser, Schimper et al. 2009).

Based on the legacy system descriptions, the description and mapping of the domain knowledge and the description of the architecture and capabilities of the integration network, the possible sending and receiving service partners are derived using heuristics and ontology-based reasoning .In the example, this is represented as a graph consisting of the possible collaborations (i.e., the services which are able to communicate). As shown in Figure 31, there are 4 automatically derived collaborations: collaboration 1 between “ATMIS” and “PFIP”, collaboration 2 between “PFIP” and “ATMIS”, collaboration 3 between “CFC” and “ATMIS”, and collaboration 4 between “SFDP” and “ATMIS” (Mordinyi, Moser et al. 2009).

5.4.1.4 Concrete Technical Solution Model

Based on the logical solution model derived in the previous process step, the technical solution model for each integration node is generated automatically. This technical solution model is an XML configuration which is interpreted by the integration platform introduced in section 5.4.2. The major components of the technical solution model are a) routing tables that specify where certain received messages belonging to a specific collaboration should be forwarded to – there are more than one routing targets for a specific collaboration (so called “backup routes”), which are automatically used in case of unavailability of the original target integration node; b) transformation instructions that define how messages originating from business services should be transformed before sending them to other business services via the integration nodes; c) middleware specifications that define the configuration parameters and access methods for each connected specific middleware technology of a particular integration node; d) application specifications that define the configuration parameters and access methods for each connected business service of a specific integration node; and finally e) security specifications of the particular integration node (i.e., encryption protocols or certificates to use for the transmissions) (Mordinyi, Moser et al. 2009).
As shown in Figure 31, the concrete technical solution model for each single integration node contains information about all collaborations which use this particular node. Additionally, there is a difference between integration nodes that are connected to business services and integration nodes without connected business services (so called “intermediate nodes”). While the technical solution model of intermediate nodes only contains routing tables, middleware specifications and security information, the technical solution model of integration nodes connected to business services additionally contains transformation instructions and application specifications (Mordinyi, Moser et al. 2009).

5.4.1.5 Deployment to concrete Hardware

Finally, the concrete technical solution model for each single integration node is deployed to the particular integration platform (see section 5.4.2) (Mordinyi, Moser et al. 2009).

5.4.2 Integration Platform

The model-driven system configuration (MDSC) process results in a solution model that needs to be deployed. Additionally, the process is capable of improving the system's configuration by means of monitoring data collected during execution. In the following the integration platform for the MDSC approach is described (Mordinyi, Moser et al. 2009).
The main task of the integration platform (see Figure 32) is to interconnect business services a) by binding business services and middleware technologies; b) by routing messages in a fault-tolerant manner with respect to virtual sender groups over heterogeneous middleware technologies; and c) by transforming messages to overcome the semantic gaps between business services. The integration platform is installed on every node described in the network capability model and uses the derived solution model to configure its components (Mordinyi, Moser et al. 2009).

5.4.2.1 Application Adapter

Based on the automatically derived configuration, the adapter loads the so called Application Gateways. An Application Gateway represents the connection to the business service and is implemented by the developers of the service. Similar to JBI (Ten-Hover and Walker 2005), by means of the Application Gateway the service is capable of sending and receiving messages asynchronously. The interfaces of the Application Adapter and of the Application Gateway do not need to be described by means of WSDL since their capabilities have already been defined in the capability models. Messages sent by the Application Gateway additionally receive a so called CollaborationID representing the collaboration that has been calculated in the Logical Solution Model. This ID helps the Routing Component to route messages (i.e., the Routing Component looks up the specific network route for a particular CollaborationID) (Mordinyi, Moser et al. 2009).

5.4.2.2 Middleware Adapter

Based on the automatically derived configuration, the adapter loads the so called Middleware Gateways. A Middleware Gateway represents a communication link with a specific middleware technology between two nodes only. If there are several different communication links between two nodes, then there has to be one Middleware Gateway available for each possibility. The Middleware Gateway functions as a wrapper and knows how to interact with the real middleware that is actually forwarding the message to the destination specified by the routing component. The responsibility of the Middleware Gateway is to operate and to optimize the middleware technology according to the capabilities which have been specified in the capability models and selected during the derivation of the logical solution model. Similar to the Application Gateway, the interface of the Middleware Gateway consists of methods for sending and receiving messages asynchronously (Mordinyi, Moser et al. 2009).

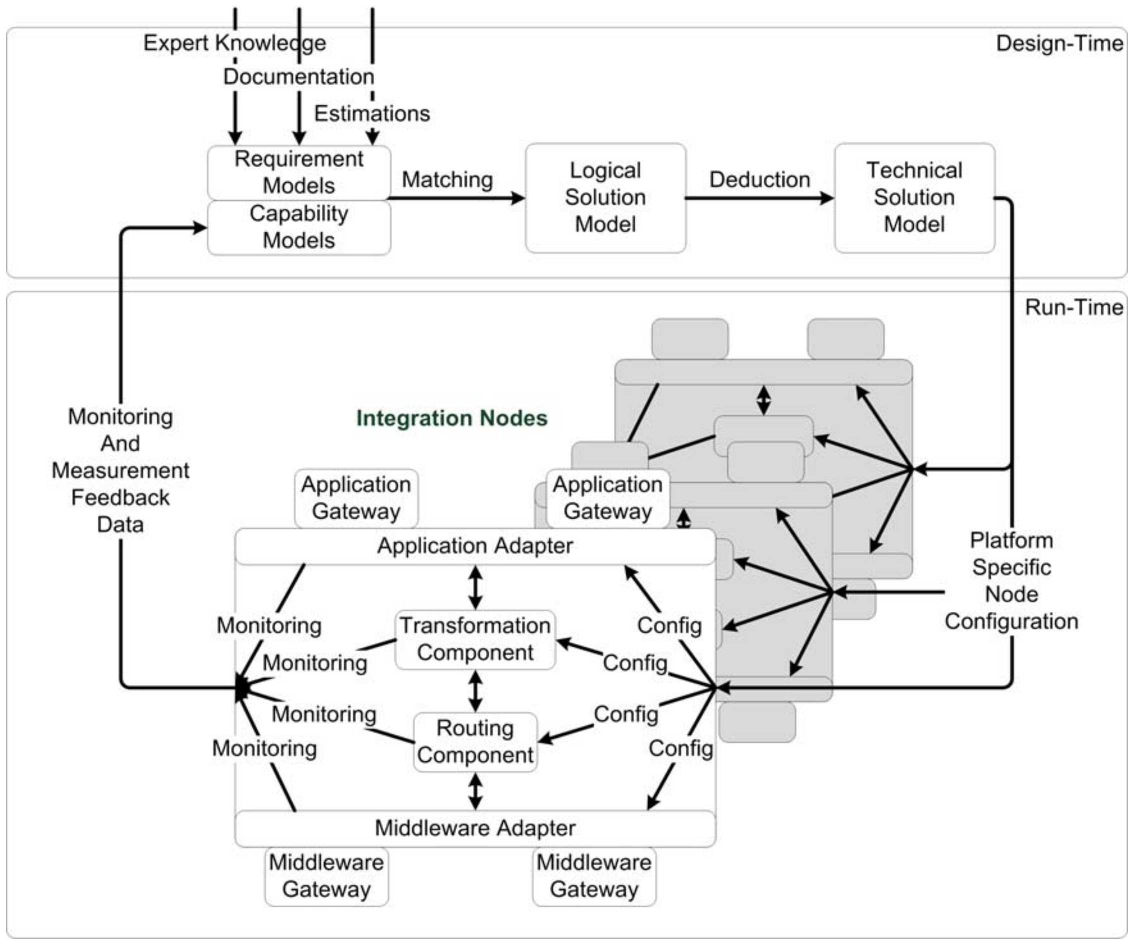


Figure 32: Integration Platform overview (Mordinyi, Moser et al. 2009).

5.4.2.3 Transformation Component

The solution model provides transformation instructions specifying how to handle certain message types. The instructions have been derived from the requirement and capability models describing the services and thus the Transformation Component is able to manipulate message structure and content accordingly. The component can change message data types, split messages into several segments, merge different segments into a message, replace or enrich certain information of a message, or perform any combination of the described possibilities (Mordinyi, Moser et al. 2009).

5.4.2.4 Routing Component

The automatically derived configuration of the routing component contains a routing logic specifying where to forward a message. A message can be either forwarded to a local service via the Transformation Component or to one of the installed the Middleware Gateways. Consequently, the routing logic specifies either the target Application Gateway or the target the Middleware Gateway which is used to send the message to the next hop along the route to the final target service. The chosen the Middleware Gateway has a priori been selected during the derivation of the concrete technical solution model. Additionally, the solution model contains several other routes, so called "backup routes", which have been calculated during the derivation of the concrete technical solution model. These backup routes are used by the routing logic of the integration platform if the originally targeted next hop is not available any more. This allows the integration platform to react on changing network conditions quickly (Mordinyi, Moser et al. 2009).

To keep the abstraction interface reduced to the methods send and receive and the implementation of the integration platform less complex, it has been avoided to add a component responsible for group communication only. Additionally, traditional group communication mechanisms are not capable of coordination over multiple heterogeneous middleware technologies. Therefore, the interface is configurable as well, and as such the send method can be intercepted by aspects representing the appropriate strategy for coordination (David 1996). The need for a virtual sender groups is derived from the data in the service capability and requirement models. Additional collaborations with unique collaboration-IDs are set up between the members of a virtual sender group, and in the concrete technical solution model an appropriate route between those nodes is calculated. This means that the aspect is configured with information about the virtual sender group and which collaboration-ID it has to use to reach other virtual sender group members. When the aspect receives a message from a business service that is a member of the virtual sender group, it withholds the message until the group has reached a decision. Either the message is then sent via the original the Middleware Gateway or discarded (Mordinyi, Moser et al. 2009).

The task of the Monitoring Component of the integration platform is to collect information that may help to improve the capability models reflecting a more realistic description of the network infrastructure and the business services. This results in a configuration that is adapted to the real circumstances and environment. The Monitoring Component collects data on e.g. transmission speed and maximum bandwidth between two nodes, the number and size of exchanged messages, the time needed to reach an agreement in a virtual sender group, or the number of node failures resulting in accurate failure probability values (Mordinyi, Moser et al. 2009).

5.5 *Matching of Problem and Solution Space*

Safety-critical systems and business processes, e.g., in the Air Traffic Management (ATM) domain, have to become more flexible to implement changes due to new business environments (e.g., mergers and acquisitions), new standards and regulations. A promising approach follows the service-oriented architecture (SOA) paradigm that builds flexible new systems for business processes based on a set of software services provided by system nodes in a network. A key design challenge is the matchmaking of business processes and software services, i.e., finding the software services that a) best meet the requirements of the business processes under consideration and b) can be implemented with the available network capabilities. The solution space is typically large even for small problems and a general semantic solution to enable comprehensive tool support seems infeasible (Moser, Mordinyi et al. 2009).
Figure 33 provides an overview on the integration layers, data flows between the integration layers, and the steps of the semantic service matchmaking process (Moser, Mordinyi et al. 2009).

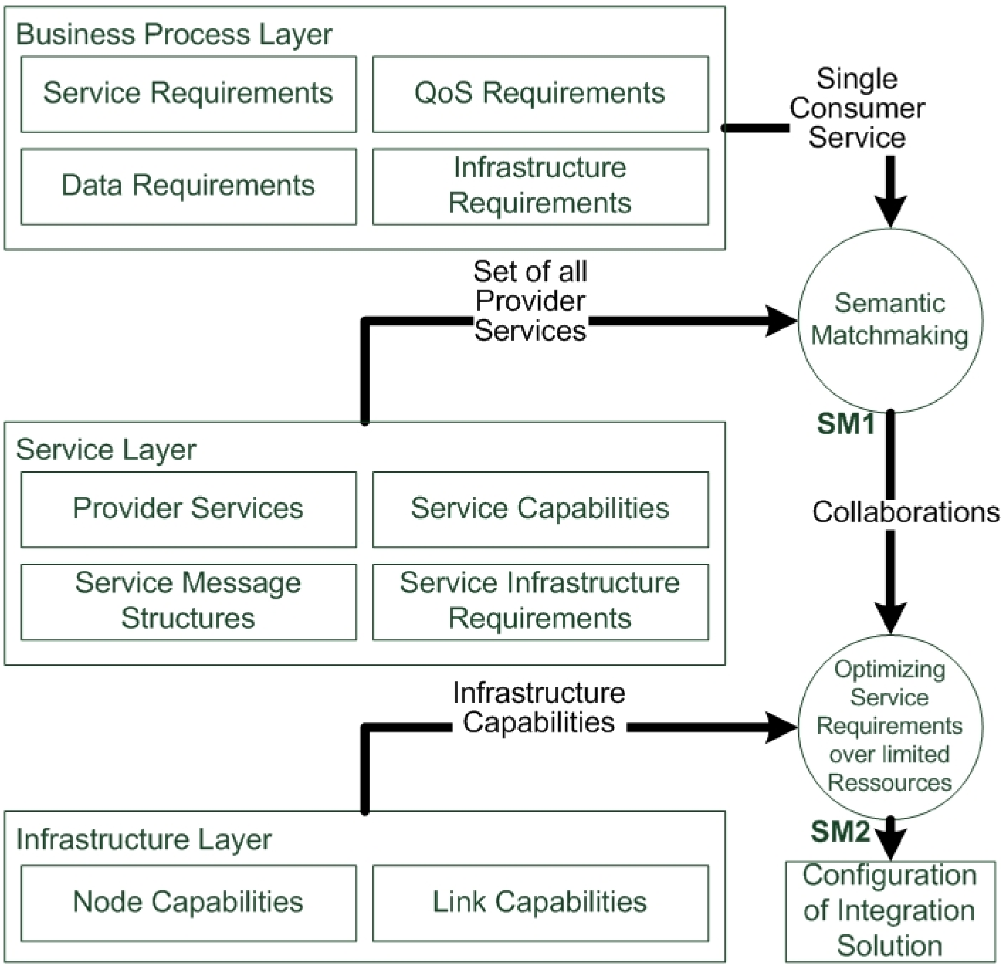


Figure 33: Semantic Service Matchmaking Process (Moser, Mordinyi et al. 2009).

SM1: For each business processes, identify the suitable software services sets, which fulfill all business processes service and data requirements. From these possible business process and software services sets, the system integrators choose the most promising sets, the so-called collaboration sets (Moser, Mordinyi et al. 2009).
SM2: The selected collaboration sets are then optimized regarding the original infrastructure requirements of both the business processes and the software services, as well as the available limited capabilities of the infrastructure's nodes and links. The outcome of SM2 is an optimized configuration of the integration solution, consisting of the selected collaboration sets as well as their grounding to the underlying integration network infrastructure (Moser, Mordinyi et al. 2009).

The following subsections describe the semantic service matchmaking approach as well as the multi-objective optimization of the chosen integration services candidates (Moser, Mordinyi et al. 2009).

5.5.1 Identification of Possible Collaboration Candidate Sets

The identification of possible collaboration candidate sets is implemented as a heuristic algorithm. Step by step, the possible collaboration candidate sets are reduced by applying the rules described to the possible collaboration candidate sets. The heuristic rules that are applied during the source/sink matching are described in the following subsections (Moser, Mordinyi et al. 2009).

5.5.1.1 Message mapping

During the description of the software service messages, each software service message segment was mapped to a domain concept, which has been specified in the common domain ontology. Therefore, for all segments of the message required by a certain business process, it is searched for messages of the software services that contain segments, which are mapped to the same domain concept, and if possible, to the same message format (Moser, Mordinyi et al. 2009).

5.5.1.2 Service Policies

In addition, software services can define requirements (policies) regarding preferred or unwanted software service partners, as well as other non-functional requirements, e.g., QoS requirements regarding the underlying integration network. A policy is a restriction or a condition for a single collaboration or a set of collaborations, in order to allow the communication via the underlying integration network. In the application scenario, there are two kinds of policies. On the one hand, there are policies which are valid for all collaborations. They specify global conditions that need to be fulfilled by all collaborations, e.g., a maximum time for the delivery of messages. On the other hand, there are policies which are required only for a specific subset of collaborations. These policies specify conditions that need to be fulfilled by the collaborations containing particular software services, e.g., the communication has to use only secure links, or only a specified set of other software services is allowed to participate in the collaboration. The software service policies that regard other software services are evaluated by checking whether the attributes and tags of every software service of the particular collaboration candidate meet the service policies defined by the business process (Moser, Mordinyi et al. 2009).

5.5.1.3 Format Translation

If a message segment is mapped to the same domain concept as the required message segment, but the formats of the two segments differ, check whether there is a converter defined for the two formats. A converter is used to convert the format of message segments from one basic data type to a different one. An explicit identifier is defined to allow the search for the converter at runtime (e.g., by using Java Reflection) (Moser, Mordinyi et al. 2009).

5.5.1.4 External Service Transformation

If the message segments differ in the domain concept they are mapped to, check if a service exists that consumes a segment mapped to the same domain concept as the segment of the message of the software service and provides a message with a segment mapped to the same domain concept of the segment of the message of the business process (Moser, Mordinyi et al. 2009).

5.5.1.5 Route Deduction

As last rule it is checked whether there is an existing route between the nodes connecting the software services and the node connecting the business process.
If all the rules mentioned above are successfully applied to a set of one or more software services and a business process, then the particular set is accepted as collaboration candidate. If any of the rules cannot be met, the particular set is discarded as collaboration candidate (Moser, Mordinyi et al. 2009).

5.5.2 Validity-Check and Optimization of Collaborations

Once all collaborations have been specified a Scenario is derived. A Scenario contains beside all collaborations a specification detailing how to configure the network infrastructure, so that the integration solution is optimized according to the given objectives. In the following the process steps needed to optimize the scenario is explained (Moser, Mordinyi et al. 2009).

5.5.2.1 Preliminary Checks

The process step checks whether there is at least one single network route for each collaboration satisfying all global and collaboration specific policies. If this step cannot be completely satisfied the process raises an exception. The system integrator either updates or removes the collaborations which cannot be mapped to a network route, and restart the process step, or adapts the semantic infrastructure model, by adding additional nodes and links (Moser, Mordinyi et al. 2009).

5.5.2.2 Route Derivation

Once it has been verified that each collaboration can be mapped to at least one route in the network, the process step derives every possible route for each collaboration. The only restrictions are that no node is allowed to appear twice within the same route and all policies have to be satisfied. The valid ones are retained; the ones violating the restrictions are removed. At the end of this process step, each collaboration will have either a single route or a set of valid routes to choose from (Moser, Mordinyi et al. 2009).

5.5.2.3 Creating Scenarios

The processing step combines each route of each collaboration with each other. This means that a scenario consists of a set of collaborations where each collaboration represents exactly one route. The more scenarios are created, the higher the probability to find a scenario that is well suited for achieving the stated optimization objectives (Moser, Mordinyi et al. 2009).

5.5.2.4 Evaluation

The process iterates through all scenarios and calculates their fitness according to the optimization objectives. The fitness of a scenario is the fitness of all its containing collaborations, and represents the real values (e.g. the time a message needs and the costs along the chosen route) of the objectives. The fitness represents the trade-off of the configuration, the routes of each collaboration predetermine. The set of fitness values is then analyzed according to the Pareto Front approach (Ehrgott 2005). The Pareto Front contains either a single Scenario or a set of Scenarios. In the latter case there may be several "nearly equivalent" configurations as integration solutions. Thus, the system integrator has to decide which one to pick for practical deployment (Moser, Mordinyi et al. 2009).

5.5.2.5 Multi-Objective Optimization

We have accomplished the process of optimizing collaborations by implementing a Java version of the mPOEMS approach into the SWIS framework. mPOEMS is an evolutionary algorithm using the concept of dominance for multi-objective optimization (Moser, Mordinyi et al. 2009). The results and explanations of the approach can be found at (Kubalík, Mordinyi et al. 2008).

5.6 *Summary*

In this chapter we proposed and evaluated the EKB-framework to integrate heterogeneous legacy systems in the ATM domain to provide integration services with little extra integration effort, short time to market, and explicit and easy-to-understand integration knowledge to simplify the overall system evolution. In contrast to integration technologies like web services or the enterprise service bus, the EKB-based approach externalizes explicit integration requirements and capabilities in machine-understandable formats, making them easier to change and maintain.
The following sub-sections describe the findings and results for the EKB-based process description, for the modeling of the problem space, for the modeling of the solution space, and for the matching of the problem and the solution space and the generation of system configurations.

5.6.1 Process Description

Based on use cases from a research project in the ATM domain with two industry partners, we evaluated the EKB-based approach in comparison to an UML-based modeling approach. Major results of the evaluation are: a) the semantically enabled approach was found to be more efficient to retain expert knowledge and make this knowledge available to experts from different domains; b) the EKB-based approach took considerably shorter for the modeling phase and lowered the risk of errors in the system configuration. While the integration analysis with explicit knowledge modeling takes slightly more effort than the traditional approach, the more efficient QA and configuration generation can be expected to return this investment after two iterations of systems integration (based on conservative estimates). In many projects experiences have been that a high modeling effort which has to be invested before any benefit can be shown is not accepted. Therefore an approach such as the presented can only succeed if convincing ways exist to minimize modeling efforts. As the approach also introduced new sources of complexity by more fully modeling the integration knowledge, empirical evaluation of larger cases are necessary to validate the benefits and limitations of the approach (Moser, Mordinyi et al. 2009)

5.6.2 Modeling of the Problem Space

In this chapter, we introduced and evaluated a domain-specific approach for ATM to make expert knowledge on heterogeneous systems and system integration requirements explicit to facilitate tool-support for design and QA. An important contribution of this chapter is to enable new research and application areas for semantic techniques that help control complex information system. Major results of our research evaluation of the EKB-based approach in an industrial case study were: a) the explicit and machine-understandable knowledge in the EKB-based approach helps to automate time-consuming systems integration steps like consistency and completeness checks. Furthermore, it allows automating later integration processing steps, like deriving integration partner candidates or automatically generating transformation instructions for message exchange between the integrated systems; b) the evaluation showed that the integration effort needed with the EKB-based approach is slightly higher in case of integration from the scratch, but comparatively a lot smaller when adaptations due to changing business needs have to be performed. In addition, the advantage of centrally storing the domain ontology together with the mappings of individual system knowledge lies in the possibility of an automated QA and automation of further integration steps resulting in less integration efforts and less failures (Moser, Mordinyi et al. 2009)

5.6.3 Modeling of the Solution Space

The Model-Driven System Configuration (MDSC) approach has proven to be especially suitable for integration scenarios with frequent reconfiguration due to changing business requirements or network infrastructure. This allows manipulating capability and requirement models in order to simulate integration scenarios for fine-tuning of business interactions. The benefit arises from the option to cheaply generate system versions that can be analyzed to better understand the trade-offs of different capabilities in the case

study context, e.g., the valuation of different middleware technologies on the distribution of traffic in the system. Additional advantage of the approach is that the complexity of manipulating models and as consequence the solution model for the integration platform is focused at a central point that can be managed by a few experts only. In the traditional integration process, administrators have just a partial view of the entire system and may try to optimize their business interactions locally. Compared to traditional high-level MDA based approaches, the MDSC approach is adapted to a specific domain (like ATM), resulting in a skipped CIM, and directly in a PIM that is not very high-level. The collection of monitoring data from integration platform allows a) comparing the described capability models with the real behavior of the system and b) updating the existing values of the capability models automatically based on the measured real life data (Mordinyi, Moser et al. 2009)
The automatically derived configuration predetermines the overall behavior of the integration scenario. The more specifications the configuration contains the better the integration platform can react on changing circumstances. This allows the integration platform to work in a predefined deterministic way without "surprises" during execution regarding e.g. network failures, service failures, or network bottlenecks. The complexity of integrating business services is shifted away from the integration platform (run-time) to the MDSC approach at design time, minimizing the time criticality of the integration solution. This allows keeping the implementation of the integration platform itself as simple as possible since it is entirely dependent on configuration instructions only. Compared to traditional integration solution the middleware adapter abstracts any kind of middleware technologies. While in traditional solutions connectors between each used combination of different middleware technologies need to be implemented, the integration platform requires only the binding to the interface of the middleware adapter only. Although the approach of a common interface is not sophisticated, the benefit of it is a common interface with different transmission semantics. The semantic of the method, e.g. reliable or secure communication, depends on the capability of the middleware that is represented by that interface (Mordinyi, Moser et al. 2009; Moser, Mordinyi et al. 2009)

5.6.4 Matching of the Problem and the Solution Space

The example shows that even for small problems the solution space is typically large. However, large Business Process and Software Service integration networks consist of hundreds of integration nodes; and changes of Software Service properties and network capabilities make the correct and efficient identification of feasible Business Process and Software Service pairs a recurring complex and error-prone task. By providing only sets of feasible/promising service provider and consumer candidates, semantic matchmaking supports designers and system integrators by providing sets of possible integration partners regarding both structural and semantic attributes. However, the

relevant semantic concepts are hard to define unambiguously for general domains, thus the focus on a well-defined domain like ATM provides semantic clarity (Moser, Mordinyi et al. 2009).
We used the concept of describing Service policies using a knowledge representation language like OWL, but defined our own extendable policy representation language which is better suitable for the ATM domain. We do not use standardized Web Service description frameworks because, since the strengths of Web Service description frameworks lies in the generality of the approach, however their weakness is that it may become complicated to describe domain-specific issues. For specific domains, it may be useful to use the principles of web service descriptions but tailor them to the domain. Additionally, we defined our own ontology-based architecture for describing the properties and features of the ATM services (Moser, Mordinyi et al. 2009).
Current service matchmaking approaches focus on either technical or semantic integration issues (Verma, Akkiraju et al. 2005), while business process support is, to our knowledge, missing. In the EKB framework, we presented a combined service matchmaking approach that performs matching based on the data of the services and available service policies regarding other services. The EKB framework's semantic service matchmaking enables an effective search space reduction and poses lower risk and effort compared to the current human-based approaches (Moser, Mordinyi et al. 2009; Moser, Schimper et al. 2009).
The optimization process steps allow using existing resources efficiently. Out of all possible collaborations for a single business process which are creatable by means of the proposed semantic matchmaking approach, only those are desirable to be deployed in the integration solution which fulfills certain criteria. Those criteria are set up by the integration expert so that existing collaborations use the underlying integration network infrastructure with its limited resources as efficient as possible (Mordinyi, Moser et al. 2009; Moser, Mordinyi et al. 2009).

Chapter 6

6 Semantic Integration of Production Automation Engineering Environments

This chapter summarizes the results of applying the EKB framework to an application scenario from the Production Automation domain, as described in section 3.3.2. Again, the process directly relates to the three general steps of the EKB framework. While the description of the underlying SAW ontology data model displays the **data integration and transformation** step, the Quality Assurance support, the support for runtime decisions and the semantic event correlation can be seen as **advanced applications** provided by the EKB framework.
In the first section, an overview of the application scenario and domain is given, as well as a detailed description of the specific requirements of this application scenario regarding the EKB framework. In the second section, the overall process used for the application scenario is introduced and explained superficially. The third section gives an overview of the system architecture of the Simulation of Assembly Workshops (SAW) simulator. The fourth section introduces the ontology used as underlying data model for the SAW simulator and additionally introduces and details the Ontology Area concept. The fifth section summarizes research results of applying the ontology-based architecture for supporting typical Quality Assurance steps. The sixth section reports on research regarding the support of runtime decisions using design time information. The seventh section presents results of research regarding semantic event correlation. Finally, the eighth section concludes the chapter and summarizes limitations, extensions and specifics of applying the EKB framework to this specific application scenario.

6.1 *Overview*

Industrial production automation systems depend on distributed software to control the system behavior. The behavior of automation systems must be testable and predictable to meet safety and quality standards. Modern automation systems have to be designed for better interoperability and flexibility to satisfy increasing customer needs for product variety, manufacturing agility, and low cost. In Automation Systems Engineering (ASE) software engineering tasks depend on specification data and plans from a wide range of engineering expert domains in the overall engineering process, e.g., physical plant design, mechanical, and electrical engineering, and production process planning. This expert knowledge is embodied in domain-specific standards, terminologies, people, processes, methods, models, and software (Lüder 2000).

However, a major challenge in current industrial development and research approaches is insufficient semantic model integration between the expert disciplines (Schäfer and Wehrheim 2007; Biffl, Sunindyo et al. 2009). Different and partly overlapping terminologies are used in these expert disciplines, which hampers understanding. Consequently, the weak tool support for semantic integration of the expert knowledge across domain boundaries hinders flexible engineering process automation and quality management, leading to development delays and risks for system operation.
The strategic goal of making the ASE process more flexible without delivering significantly more risky end products translates into the capability to efficiently re-configure the engineering process and tool instances of a project environment. While there are approaches based on a common repository that holds all relevant project data (Schäfer and Wehrheim 2007), experience has shown that such a repository tends to get large, inflexible, and hard to maintain surprisingly fast, which makes the knowledge in the repository hard to reuse in new projects. Further, if several organizational units are involved in a project, even agreeing on a common data model is difficult. Thus a key goal is to allow all participants to continue using their own data models and provide a mechanism for translation between these data models. In the past several approaches for providing engineering knowledge in machine-understandable syntax have been investigated (McGuire, Kuokka et al. 1993; Lovett, Ingram et al. 2000; Liao 2005). However, these approaches focus primarily on storing existing homogeneous knowledge rather than providing support for managing and accessing heterogeneous knowledge, which is the focus of this chapter.

6.2 *Process Description*

Current Software Engineering approaches like Component Based Software Engineering (CBSE) reuse configurable domain assets for the efficient derivation of new product variants. A main challenge is to manage concurrent development of software artifacts due to changing requirements resulting in inconsistent or error-prone artifacts, partly because the models for the variability of system variants seem not to cover the entire downstream development and Quality Assurance (QA) processes. Ontologies can support the requirements engineering by providing a continuous model for software development processes supporting elicitation, representation, and analysis of the interdependencies among artifacts. Ontology-based reasoning can facilitate analyzing the impact of requirement changes, supporting a more consistent handling of changing requirements (Biffl, Mordinyi et al. 2007).
There are reports on using ontologies for software engineering: a) for describing the problem domain; b) for the semantic description of transformations between models in Model-Driven Development (MDD); and c) for QA reasoning on semantic inconsistencies between models (Baclawski, Kokar et al. 2002). However, we found

very little work on ontologies to provide a continuous model for linking different stages of the engineering process of software-intensive systems (Bosch, Florijn et al. 2002).
In this chapter we introduce a) an SE process for CBSE using a continuous engineering ontology for efficient system development and for improving variability management; b) semantic support for requirements modeling, requirements transformation, simulation and testing; c) support for role-oriented views on the engineering ontology assisting domain experts and roles that do not easily accept abstract data models.
The semantic model is the foundation for appropriate tool support along the CBSE process, which allows bridging specific models of roles with different viewpoints such as sales people, architects, developers, and QA personnel. Ontologies seem well suited to provide their reasoning capabilities for component selection and valid parameterization, transforming models for downstream activities (e.g., generating test cases for system performance evaluation), and checking models for consistency both at design time and at run time. "Ontology areas" for a concrete domain can help keep the model grounded and prevent unnecessary complexity (Biffl, Sunindyo et al. 2009).

6.2.1 Ontology-Supported Variability Management

Figure 34 shows the process for development and generation of new system versions for production automation that supports variability modeling and traces design decisions by means of ontology-supported continuous modeling (Biffl, Mordinyi et al. 2008).

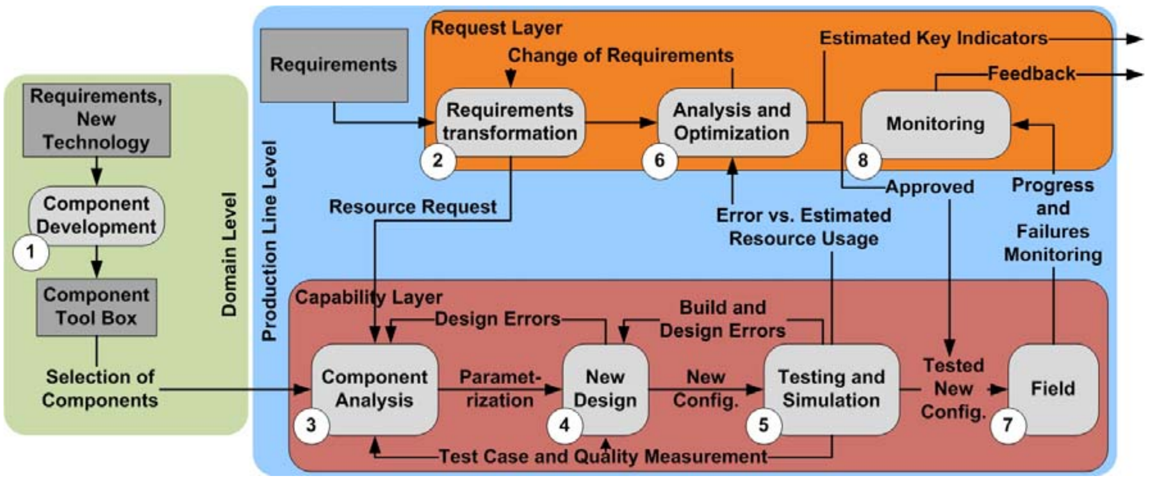


Figure 34: Feedback-driven and ontology-based approach (Biffl, Mordinyi et al. 2008).

The ontology-supported software engineering process is divided into the "domain level" and "production line level" development processes. On each level requirements and capabilities are described semantically. The "domain level" represents the development activities for a reusable set of software components, the "component tool box". The "production line level" outlines the activities of the actual system configuration in order

to build a particular product. This process is divided into two more layers, “capability layer” and “request layer”. It demonstrates the coordination steps needed for optimization over all sets of selected component capabilities in order to achieve correct component parameterization (Biffl, Mordinyi et al. 2008).
The “capability layer” is in charge of fulfilling the transformed requirements with the resources available. For estimating the potential available for the currently requested requirements the layer has to take into account already used resources reserved for previously requested requirements. The “request layer” optimizes the number and type of requirements based on the estimated potential of the “capability layer” in order to achieve an effective and efficient usage of available resources with respect to the number and type of requirements (Biffl, Mordinyi et al. 2008).

6.2.1.1 Step 1: Component Development

Based on requirements or triggered by new technologies or roles, components are developed on domain level for the production automation Component Tool Box.

6.2.1.2 Step 2: Requirements Transformation

The input for the “request layer” is a set of functional (e.g., customer orders) and non-functional requirements (e.g., production time) that is transformed into requests for resources needed to fulfill the incoming requirements. The transformed set is forwarded to the capability layer.

6.2.1.3 Step 3: Component Analysis

The system reconfiguration cycle at the production line level is triggered either by new or changed requirements or components. Additional input to component analysis are selected components from the Component Tool Box. Further input is the current combination of components representing the current production system. The analysis step creates all valid combinations of the input with respect to compatibility of the components with each other. The set of components is then parameterized according to the analysis of historical test cases measurements as explained in the next section.

6.2.1.4 Step 4: New Design

During the design phase complex requirements have to be fulfilled focusing on choosing the right combination of components. The step selects the combination that fulfills non-functional requirements like production time, cost or machine utilization.

The selected combination is then transformed into a configuration view that can be interpreted by the production system.

6.2.1.5 Step 5: Testing and Simulation

The operation of tools to measure system quality and performance of new configurations is mandatory to assure that the configurations meet overall requirements including system safety before deployment to real-world environments. The introduced development cycle represents another source of error leaving the possibility of remaining unresolved (uncritical) defects. One solution is to execute simulations of relevant properties of the target system to capture performance measurements from monitoring for evaluation and use in step 3 for component selection and in step 4 for analyzing component combinations. In comparison to traditional approaches with implicit feedback by manually analyzing the results of test case runs, this approach explicitly provides measurement feedback integrated into the ontology for step 3.

6.2.1.6 Step 6: Analysis and Optimization

Based on simulation and test data representing the available resource capacity the layer has to optimize selected performance criteria, e.g. maximize the number of accomplished requirements.

6.2.1.7 Step 7: Field Environment

If the reported data is promising, the created configuration is approved in order to be deployed. Reports of progress or potential errors are forwarded to the monitoring step.

6.2.1.8 Step 8: Monitoring

Monitoring supports feedback on the current system states by providing measurement data to be used for comparison with the estimated key indicators created by step 6.

6.2.2 System Measurement Specification

Figure 35 shows the System Measurement Specification (SMS) approach for the automated derivation of new product versions and variants using the measurements and results of historical test cases as input. We use an engineering ontology consisting of different areas in order to model and store the data of the SMS. It is possible to combine a collection of areas to provide a complete ontology model for certain tasks, e.g., testing, statistical data analysis, business strategy planning, requirements tracing to test results. Historical test cases and their results are stored in the so called System Measurement Specification Database (SMSDB), a derivation of the EKB, in order to support the selection of parameters for future test cases (Biffl, Mordinyi et al. 2008).

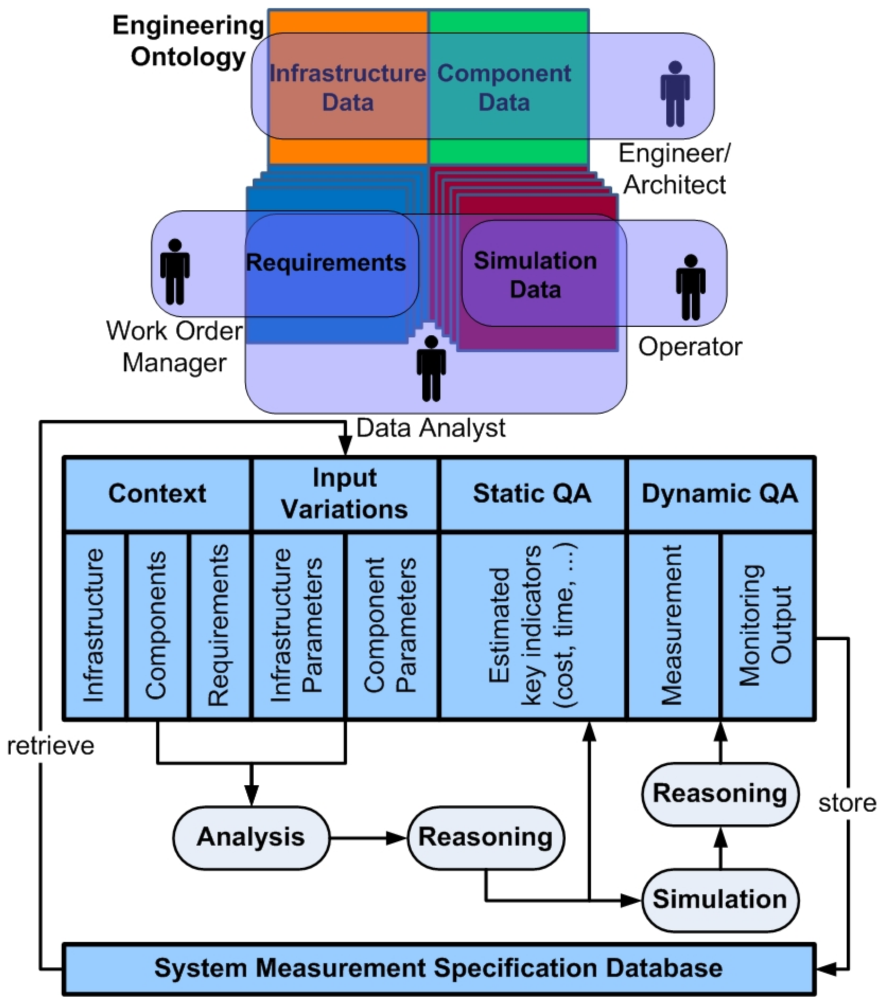


Figure 35: SMS & Role-oriented EKB views (Biffl, Mordinyi et al. 2008).

In our approach, a test case consists of static data (context) and dynamic data (input variations). The context consists of definitions of the infrastructure (e.g., layout of an assembly workshop), descriptions of the components used in the test case (e.g., workshop machines) and the requirements of the test case (e.g., work orders to fulfill). The input variations contain parameters for the infrastructure (e.g., overall size of the workshop) and parameters for components (e.g., speed of a conveyor belt) (Biffl, Mordinyi et al. 2008).

From Figure 35, the following 4 steps can be derived.

6.2.2.1 Step 1: Requirements Transformation

The given requirements (both internal and external) can be transformed into a set of test cases, specified by a static context and dynamic input variations. The selection of the input parameters is done by querying the SMSDB for matching historical test cases with the context of the current test case. This allows systematic variation on the input parameters enabling repeatable testing and statistical data analysis on system performance. The variation of context parameters allows not only to test a particular system version, but to build and test a range of system versions and consequently investigate the impact of system variation factors on the quality and performance of system versions derived, e.g., in a product family.

6.2.2.2 Step 2: Static QA

The context and the input variation are then checked according a given rule set using the ontology-based reasoning support. This allows a check of the validity of the input parameters with regard to the used context and the original requirements.

6.2.2.3 Step 3: Dynamic QA

After the static QA steps are finished, the simulation is started and the output of the simulation can be checked for validity with regard to the requirements. In this step, mainly runtime requirements like performance or failure rates are checked.

6.2.2.4 Step 4: Storage in SMSDB

As a last step, the test case and the results of the static and dynamic QA steps are stored in the SMSDB. The SMSDB actually is a specific view on the engineering ontology, representing the set of conducted, historical test cases, which may be queried using ontology-based reasoning.

6.2.3 Role-specific Views on the Engineering Knowledge Base

Figure 35 shows a number of role-specific views on the EKB. This allows more effective management of ontology areas a certain role is interested in, since the data can be presented in a well-accepted format/tool for this role, e.g., work order manager, operator, architect, or data analyst personnel (Biffl, Mordinyi et al. 2008).
Figure 35 illustrates a number of roles supported by the EKB. The work order manager is responsible for fulfilling the incoming work orders represented as requirements,

which are shown by his ontology view. The operator supervises the simulation and reacts in case of errors accordingly. His ontology view therefore includes information about the operation and measurement data of the simulation. The architect is in charge of component development and monitoring the creation of the configuration during the design step. The dedicated ontology view for this role includes both infrastructure and component data. The data analyst performs statistical analysis of historical test case data like a complete data analysis of the whole set resulting in more significant assertions of the result data. His ontology view therefore includes both information about the test case requirements and the simulation output data. This ontology view has already been introduced as SMSDB, which is represented at the bottom of Figure 35 (Biffl, Mordinyi et al. 2008).
The usage of the ontology-supported and test case-based SMS approach entails a number of advantages: 1) Based on the assumption that systems can be defined as combinations of selected components and their parameters, at design time a data vector can capture the fixed context information and input variations for a test case. 2) Static QA based on ontology reasoning can provide initial feedback on the test case data vector, e.g., point out errors in the parameters. 3) Dynamic QA based on simulation or testing provides the actual result of the test case and system performance measurements (Biffl, Mordinyi et al. 2008).

6.3 *Simulation of Assembly Workshops (SAW) System Architecture*

Modern production automation systems need to become more flexible to support the timely reaction to changing business and market needs. However, the overall behavior of the many elements in a production automation system with distributed control can get hard to predict as these heterogeneous elements may interact in complex ways (e.g., timing of redundant fault-tolerant transport system and machine groups) (Lüder, Peschke et al. 2004).
Software agents seem particularly well suited to model and design flexible, modular, and self-organizing systems that are robust to changes in their environment. Multi-Agent Systems (MAS) can help simulate in a distributed control system the effects of production strategies that coordinate the behavior of the entities in the production automation system (Jennings and Wooldridge 1998). However, designing coordination for several levels of agents in a flexible control structure is a major challenge and can benefit from design patterns that can be systematically validated and reused in a range of contexts. In the production automation domain there are the levels of a) business processes; b) redundant robust transport system and machine functions; and c) control components in the machines of the production automation system (see Figure 36).

In order to allow the interaction between these levels, coordination between agents on each layer is required. Coordination patterns are design/implementation paradigms suitable for solving certain problem scenarios. In our context, coordination patterns are used to enhance the design and implementation of a) domain expert knowledge in their particular layer (model of world, constraints, strategies, goals, etc.); b) the allocation of available resources (e.g., production strategies, auctions, or work load balancing); and c) information model and control (e.g., master/slave (following the layer concept), message exchange, or blackboard pattern) (Moser, Merdan et al. 2010).
In this section, we describe the pattern-based extension of a MAS-based production automation simulation tool, Manufacturing Agents Simulation Tool (MAST) (Vrba 2003). The architecture of our extension, the SAW (Simulation of Assembly Workshops) framework consists of three major parts: a) the original MAST simulator; b) the work order scheduling system; and c) the performance test management system (MAST-TMS) (Merdan, Moser et al. 2008; Merdan, Moser et al. 2008). The work order scheduling system transforms incoming business orders into feasible working tasks and then coordinates these tasks with the original MAST simulator. MAST-TMS supports the creation and execution of test cases to measure system performance for exploring the systematic effects of a range of strategies and system configurations and to enable the quantitative evaluation of a large number of simulation runs. The simulation system has been validated with the hardware of this research lab to ensure the external validity of simulation measurement and analysis (Moser, Merdan et al. 2010).

6.3.1 Framework Architecture

The coordination of the agent-based production planning system is the main goal of the framework presented in this section. The production planning system is based on a simulator developed by Rockwell Automation, which already includes a set of FIPA[18]-compliant simulation agents. This agent system is modified and extended by a coordination component that should monitor the agents' tasks and activities. Figure 36 provides an overview on the layers of the agent system: on the business layer the dispatcher (order agent) converts customer orders into work orders that are sent to the workshop layer (product agent) (Moser, Merdan et al. 2010). Figure 36 shows the layer model of the production automation system used in the SAW framework.

[18] http://www.fipa.org/

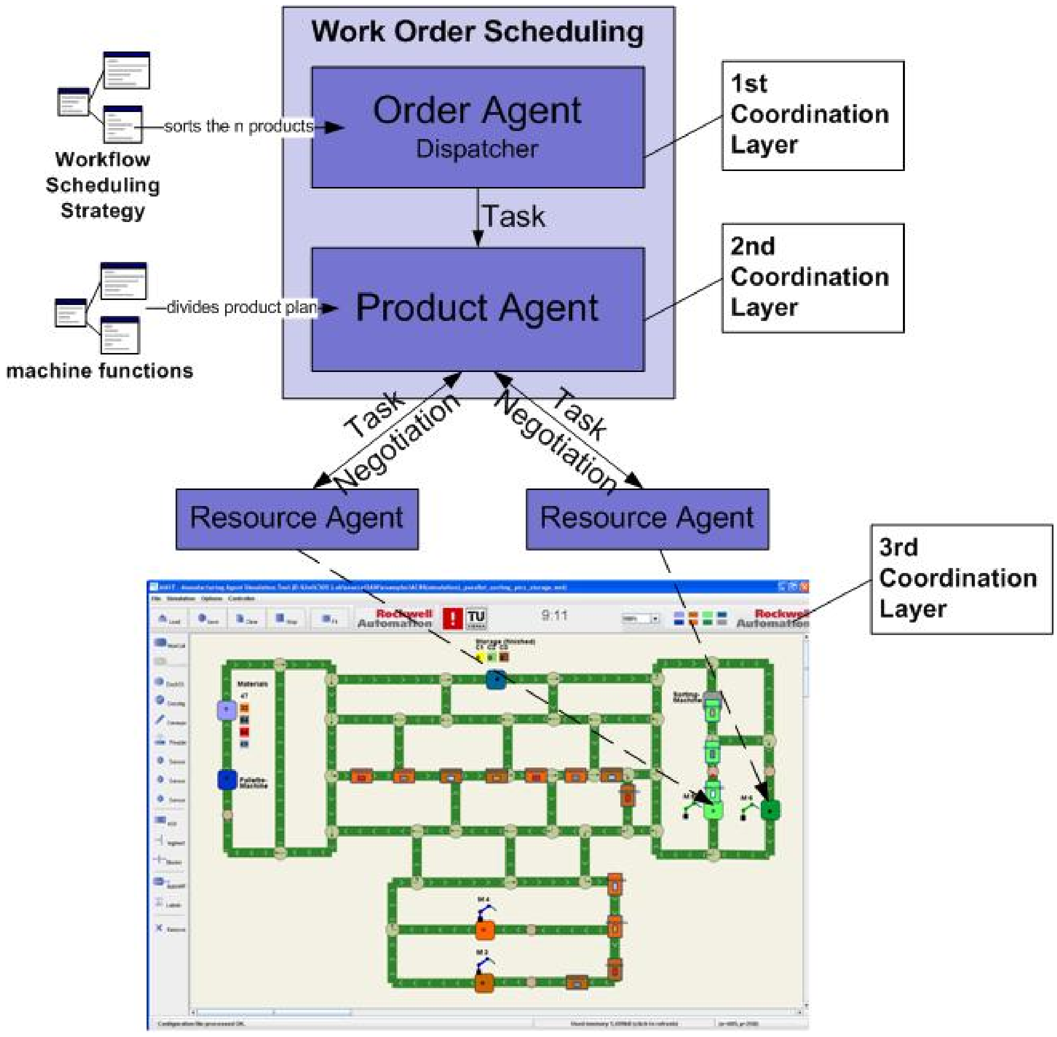


Figure 36: Coordination layer model of SAW (Moser, Merdan et al. 2010).

6.3.1.1 *Business process*

The order agent in the first layer is responsible for the incoming business orders from the costumer, monitoring and guiding a single product through the simulator. The business orders represent guidelines for the arrangement of product sequences depending on the selected workshop scheduling strategy. After the sorting mechanism the dispatcher registers n product agents (PA) for each product and forwards the product, as a defined so-called product plan, to these agents.

6.3.1.2 *Workshop Scheduling*

These n PAs operate in the second layer of the coordination model and represent the interface between the dispatcher and the simulation agents. They analyze the product and divide the product plan in more detailed working steps, i.e. transport and assembly steps. These working steps are delegated to Resource Agents (RAs). This decision making process includes a very important negotiation and allocation sequence between the PA and the RAs. The choice which RA will get the working steps follows an auction pattern. The Auction pattern provides the possibility of offering a good or a service to other participators. In this case the PA sends an announcement message, including an identifier and the machine function, to the RAs. The RAs offering the required function send back a message containing the estimated processing time of the machine function plus the estimated time needed for the transportation to the machine. The PA picks the RA with the lowest overall machine function time and delegates the task to it. During the simulation processes the product agents of the second layer can influence the simulator and monitor the events and states within the simulator.

6.3.1.3 MAST

The third layer is the simulator, where the simulation agents communicate with each other as well as with the agent of the layer above. The agents can be classified into Resource Agents, which work on tasks delegated by the PAs, and Transport Agents, which have to fulfill various transport steps (e.g. sending palettes). The simulation agents implemented by Rockwell Automation already include some self-coordinated behavior such as flexible routing. This coordination functions have been modified and adjusted to the needed requirements. The agents have to fulfill the various tasks getting from the agents in the second layer and have to report about their status, capacities, failures and the measurements.

6.3.2 Performance Test Management System

Figure 37 illustrates the design of the performance test management system (MAST-TMS) that allows automatically running a large number of systematic variations of test scenarios to evaluate the effects of these variations on the overall system performance. The test system is a harness that starts up the SAW with the current test scenario parameters and customer order events. The SAW in turn runs the software agents, the coordination patterns, and logs the results of the test run. The test system collects the measurement results of each test run in a database for further statistical analysis (Moser, Merdan et al. 2010).
The Performance Test Management System (MAST-TMS) is a part of the SAW framework and uses the MAST system for automatically running pre-defined sets of test cases described in XML files. These test cases describe goals, strategies, and constraints: the products to assemble, assembly steps, the workflow scheduling strategy to apply, the number of pallets to use, and the duration of a production shift. For running a test suite, the MAST system is reset to a starting state, the XML file is parsed

and the test cases are consecutively injected into the MAST system, which acts on the input parameters of the test cases to run the agent-based simulation and control system. Relevant events and result data (e.g., number of finished products, machine utilization rates) are measured using pluggable measuring methods and algorithms to the agents and then logged to an XML output file. This approach allows scheduling automated runs of a large number of systematic test case variations, resulting in comprehensive output data for statistical data analysis. In addition to the automated runs of test cases, the MAST-TMS provides a generator for preparing a given number of systematically derived test cases, e.g., for evaluating the impact of the variation of one or more parameters in the simulation (e.g., with exhaustive enumeration of the parameter range or with statistical sampling) (Moser, Merdan et al. 2010).

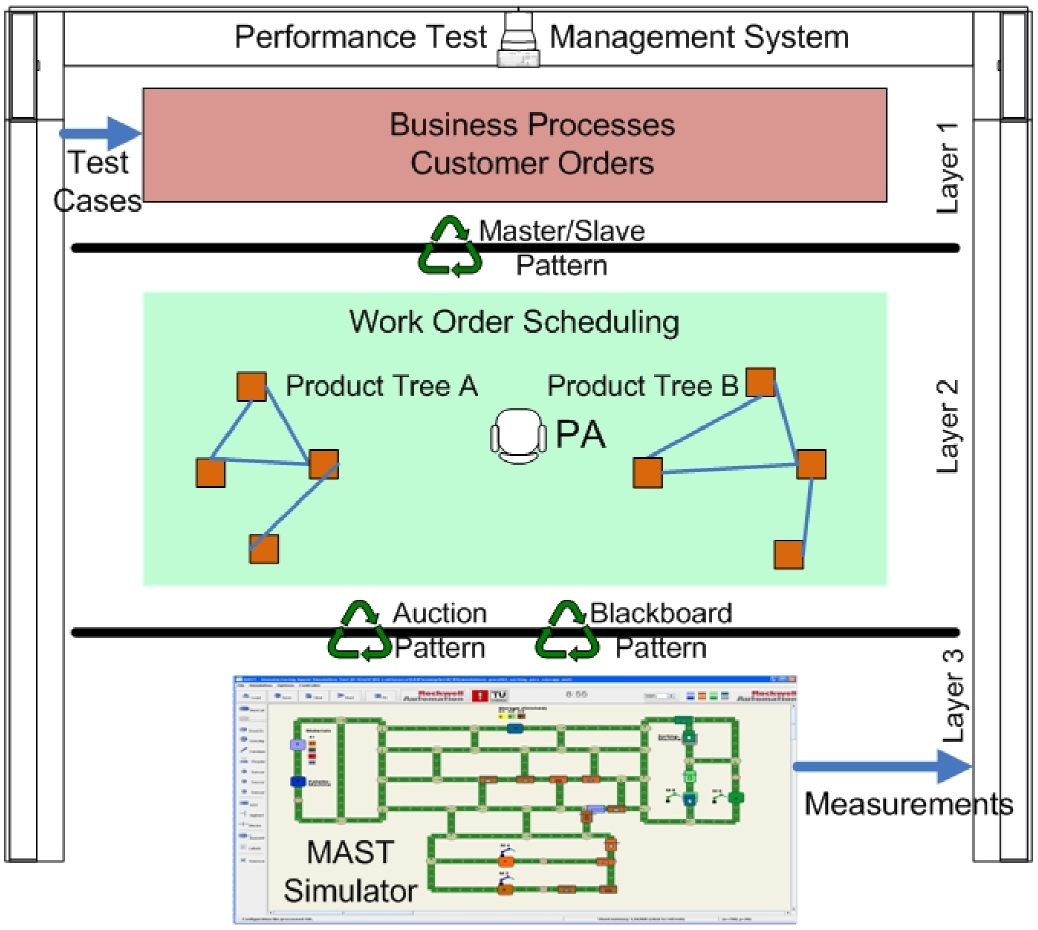


Figure 37: SAW Performance test management system (Moser, Merdan et al. 2010).

Results of the performance evaluation can be found in (Merdan, Moser et al. 2008; Merdan, Moser et al. 2008).

6.4 *Simulation of Assembly Workshops (SAW) Ontology Architecture*

The integration of business processes and IT systems in homogeneous environments (i.e., consistent data formats and terminology) is supported by well-established approaches like data integration using Scheer's ARIS for CIM (Scheer 1989). However, in more heterogeneous environments with a range of data formats and local terminologies like the production automation domain, typically stakeholders from several areas (e.g., business experts, software engineers and electrical engineers) work together to develop and operate software-intensive systems. A homogenization of these environments is often not achievable, if the stakeholders come from different organizational backgrounds or organizations change over time due to mergers and acquisitions. The precondition for successful semantic integration is a common understanding on the relevant concepts in the problem domain of the project.

An example for a collection of common problem domain concepts is the *Enterprise-Control System Integration*[19] (ECSI) standard (American National Standard 2000) for developing automated interfaces between enterprise and control systems. The objectives of ECSI are to provide a) a consistent terminology as foundation for supplier and manufacturer communications, b) consistent information models, and c) consistent operations (process) models, which are the basis for clarifying application functionality and how information shall be used.

However, a standard like ECSI can only cover parts of the problem domain without getting too complex and hard to use. Further, many key players in the production automation domain currently do not follow this standard, which often hinders the cooperation of stakeholders in projects, since transformations between stakeholder terminologies to overcome semantic gaps between the stakeholders need to be conducted by scarce experts or carefully hand-crafted.

Ontologies are flexible open-world data models for knowledge representation, which store information in machine-understandable notation (Gruber 1993). Therefore, ontologies can help to bridge semantic gaps between partial data models by providing mappings between them via common domain concepts. Ontologies usually capture problem-domain-specific information which can be reused later. Due to their concurrent development ontologies need to be checked for inconsistencies to stay useful. However, ontologies in practice usually have to combine several view points and thus get large and complex, particularly, if the ontology contains volatile domain elements, such as run-time data (Biffl, Mordinyi et al. 2008).

In this section, we propose a data modelling approach that helps structure ontologies with ontology building blocks, so-called *"Ontology Areas"*. An Ontology Area is a meaningful part of an ontology for a stakeholder, which helps ontology users managing a complex ontology. The combination of all needed Ontology Areas represents the

[19] http://www.isa-95.com

overall ontology for supporting the original engineering process (Biffl, Sunindyo et al. 2009; Moser, Biffl et al. 2010).

6.4.1 Semantic Gaps Between Stakeholders

Figure 38 illustrates sources of semantic gaps between stakeholders: stakeholder domain layers with different local terminologies; and design-/run-time views which are semantically not well connected. The data model, in our case an ontology model (the Engineering Knowledge Base), contains common domain concepts to bridge the semantic gaps between stakeholder terminologies and design-/run-time views (Biffl, Sunindyo et al. 2009).

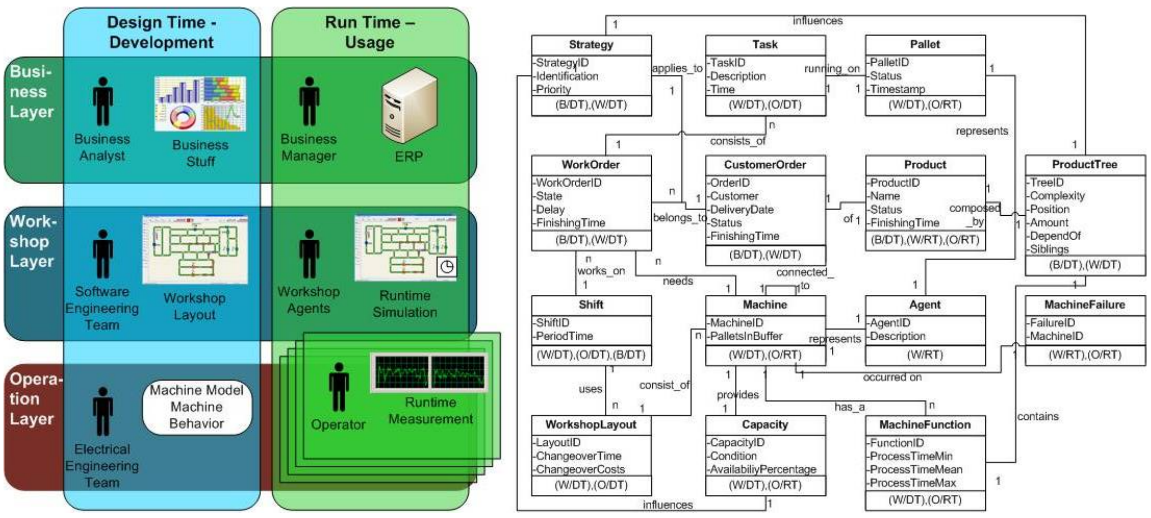


Figure 38: Semantic gaps between stakeholders (Biffl, Sunindyo et al. 2009).

The three stakeholder layers in Figure 38 are: a) the *business layer (B)* for production planning to fulfill customer orders by assigning optimal work orders to the workshop; b) the *workshop layer (W)* for coordinating the complex system of transport elements and machines to assemble smaller basic products into larger more comprehensive products according to the work orders; and c) the *operation layer (O)* for monitoring the individual transport system elements and machines to ensure their contributions to the workshop tasks. Those three layers are divided into two parts based on the time those layers worked on, namely design time (development) and run time (usage) (Biffl, Sunindyo et al. 2009).

Figure 38 (right hand side) illustrates part of the data model that represents common domain concepts for the uses cases in UML-class-diagram style notation. The bottom box of each data element shows which stakeholder layer (B, W, and O) needs this data element to conduct their tasks and when: at Design Time (DT) or Run Time (RT) (Biffl, Sunindyo et al. 2009; Moser, Biffl et al. 2010).

6.4.2 Ontology Areas for Bridging Semantic Gaps

An ontology area is a subset of ontology as a building block that can solve a certain task. The ontology can be broken into ontology areas based on several aspects, for example by the time, volatility, layer and roles. Figure 38 shows the breakdown of ontology into several ontology areas based on the stakeholder layers (business, workshop, operation) and time when models are mostly used (design time and run time). Some parts of the data mode are much more volatile than others, e.g., run-time process measurements compared to design-time workshop layout. For example, each data point measured once a second in a shift that takes 8 hours produces around 30,000 data point instances, which need to be reduced by statistical methods or will take considerably storage space (Biffl, Sunindyo et al. 2009).
To make an OA from the whole ontology, we can follow this basic algorithm. First, define a task that is needed to be solved by the stakeholder. Second, find related classes for doing the task. Third, find classes that linked to the classes in step two. Fourth, drop other classes that are not needed and save as a new ontology. Also, we can reconstruct the whole ontology from the ontology areas, by merging them together into a single ontology by using ontology tool like Protégé (Biffl, Sunindyo et al. 2009).

6.4.2.1 Translation between local stakeholder terminologies

The business manager on the business layer receives customer orders and schedules work tasks to the coordinator in the workshop layer. While they have a defined interface for exchanging work task information, they use local terminologies for concepts that are only occasionally needed to resolve scheduling issues, e.g., reference to specific customer orders if limited workshop capacity does not allow to fulfill all work tasks in a shift and negotiation on which tasks have higher priority are necessary to determine which customer orders will be fulfilled. Because the stakeholders use different terminologies, translations are necessary to automate references to customer orders between stakeholders in business and workshop layers. The stakeholders of the production automation systems need to work together to achieve their goal. A common data schema is not possible because the stakeholders usually use different data formats, local terminologies and tools to access the data from the system. The ontology (Engineering Knowledge Base) plays a role as a common domain concept, where the local terminologies from the stakeholders will be mapped to. By mapping each local terminology to the ontology, the system can translate the local terminologies from one stakeholder to the other stakeholders. The translation could be the name of function, some names in the argument of the function, different data format, or the meaning of some parameters. However, the complexity of the ontology may increase when the number of the terminologies and the stakeholders is also increases, since the ontology should store all terminologies, the mappings and the common concepts.

By using the ontology areas, the stakeholder can take a small part of the ontology that he really cares and solving his task with the same results but less complexity than by using the full ontology. The example is illustrated in Figure 39 (Biffl, Sunindyo et al. 2009). For a more complex transformation example, please refer to Figure 48.

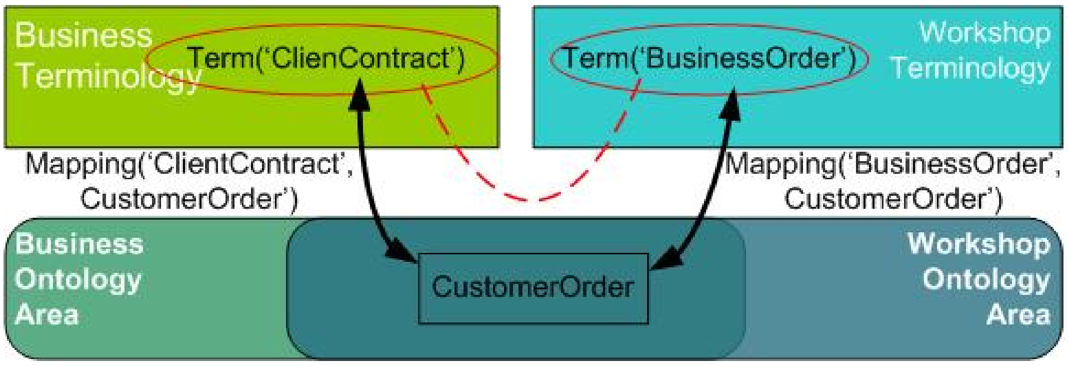


Figure 39: Transformation between Terminologies (Biffl, Sunindyo et al. 2009).

The business stakeholder has the local terminology *ClientContract*, while the workshop stakeholder has the local terminology *BusinessOrder*. Both share the common concept *CustomerOrder* in the Ontology Areas. Then, both terminologies will be mapped to the class *CustomerOrder* as mentioned in Listing 1.

Listing 1: Mapping terminologies to common concepts (Biffl, Sunindyo et al. 2009).

```
mapping('ClientContract','CustomerOrder').
mapping('BusinessOrder','CustomerOrder').
```

From the mappings above, we can have a translation between two local terminologies by using a rule, e.g., the rule described in Listing 2. The query and result can be seen in Listing 3.

Listing 2: Simple translation rules (Biffl, Sunindyo et al. 2009).

```
translate(Term1,Term2) :-
 mapping(Term1,CommonConcept),
 mapping(Term2,CommonConcept),
 not(Term1 = Term2).
```

Listing 3: Translation result (Biffl, Sunindyo et al. 2009).

```
translate(X,Y).
X = 'ClientContract'
Y = 'BusinessOrder'
```

The translation is just one example for translations in general. Ontology Areas for this use case would just consider the parts of the ontologies for the stakeholders involved (see Figure 39): stakeholder concepts, their local terminologies and mappings, which can more easily be added to and removed from an ontology as stakeholders change in a particular context (Biffl, Sunindyo et al. 2009).

6.4.2.2 Run-time measurement data representation and analysis for design model improvements

If an engineering knowledge base is available to support run-time decisions with design knowledge, it is easy to also provide all kinds of run-time measurements linked to design elements, e.g., actual capacity of infrastructure, to iteratively improve the accuracy of design estimates with feedback from run time.

Run-time measurement information can be used to make design time information more accurate. Volatile information like run-time measurement can produce large amounts of data which would make a single ontology unnecessary large and slow down the performance of the ontology. The need for storing a high volume of run-time measurement data in the ontology occurs if the concrete future statistical analysis procedures are not known at the time of measurement (Biffl, Sunindyo et al. 2009).

Partitioning of the ontology in areas of similar volatility allows building partial ontologies for the task or query at hand. Run-time measurement at the frequency of 1 data point per second provides 30,000 data points of shift of 8 hours. If this is too much information for the ontology to hold, it is possible to define Ontology Areas for smaller time windows, which allow including the data for a certain time frame to be loaded into the ontology for data analysis as needed without exceeding the capacity of the ontology.

Listing 4: Run-time data with semantic annotation (Biffl, Sunindyo et al. 2009).

```
% process (machine function id, batch number, status, timestamp)
process(‘MF1’,’B-100’,’start’,2009-02-03 T 10:01:06.01)
process(‘MF1’,’B-100’,’stop’,2009-02-03 T 10:01:06.11)
process(‘MF2’,’A-200’,’start’,2009-02-03 T 10:01:06.12)
process(‘MF1’,’B-101’,’start’,2009-02-03 T 10:01:06.13)
process(‘MF1’,’B-101’,’stop’,2009-02-03 T 10:01:06.21)
process(‘MF2’,’A-200’,’stop’,2009-02-03 T 10:01:06.24)
```

Semantic gaps between run-time measurement and design-time information occur when we have data elements from the interface of the machine at run time, but there is no machine-understandable documentation for the design of the interface. To solve this problem, we first give meaning to run-time data that are needed to be stored in the ontology and then provide a link from run-time to design-time semantics.

For example, to find out the maximum process time of certain machine functions, we can measure the process duration of that machine function in one shift, so we collect sufficient and still manageable data. The measurement result is an event named “process” that consists of the id, the batch number, status and timestamp of machine

function. Listing 4 shows several measurement results that can be obtained by filtering run time data. The real data themselves is a very long list.
To calculate the maximum process time of certain machine function, first we should calculate each process time by using predicate *process_time* to find the difference between the timestamp of *stop* status and the related timestamp of *start* status from the same machine function and batch number, and keep it in the list using *list_of_process_time* predicate. Then with using the predicate *maxprocess* we will find the maximum value of process time of certain machine function (*MFun*) from the list of process time. Listing 5 pictures this example using Prolog-style notation.

Listing 5: Example analysis rule of runtime data (Biffl, Sunindyo et al. 2009).

```
max(X,Y,X) :- X >= Y.
max(X,Y,Y) :- X < Y.
maxlist([X],X).
maxlist([X,Y|Tail],Max) :-
maxlist([Y|Tail],MaxTail),max(X,MaxTail,Max).
process_time(MF,SN,T) :-
  process(MF,SN,start,X),
  process(MF,SN,stop,Y),
  T is Y - X.
list_of_process_time(List,MFun) :-
findall(T,(process_time(MF,SN,T),MF = MFun),List).
maxprocess(MFun,T) :-
  list_of_process_time(List,MFun),
  maxlist(List,T).
```

For query, for example we want to know the maximum process time of *MF1*. The result of the query *maxprocess('MF1',T)* would be 0.1.

The machine function entity in design time consists of the id and process time attributes. Usually the values of process time attributes come from estimation, but by using run-time measurement on process time, we can compare the previous design-time estimates to actual run-time data analysis for research on design improvements. The illustrating example above is simple enough to conduct statistical analysis at run time, but for more complex statistical analyses, a solution for storing large amounts of data in an ontology may be necessary, which would inflate ontology size and decrease the ontology reasoning performance. Ontology Areas allow to manage stacks of run-time data elements and keep the size of ontology within well-performing capacity ranges (Biffl, Sunindyo et al. 2009).

6.5 *Ontology-Supported Quality Assurance*

A focus of requirements engineering is identifying and aligning the value propositions of project stakeholders towards explicit requirements (Biffl, Aurum et al. 2006). Based on these requirements, Quality Assurance (QA) and Project Management (PM) can measure both internal and external quality to guide software development. Substantial research has been reported on views of internal quality, while the external (customer) views of quality seem harder to measure. In order to meet the customer quality requirements, they need to be properly transformed and implemented, often concurrently, by many contributors to a software-intensive system.
Traditional software development approaches (e.g., RUP) are based on linking the requirements to project artifacts and responsibilities of software development roles. Since requirements typically change during the software lifecycle, the artifacts need to be adapted to stay consistent to the current requirements. Therefore, a major challenge of Quality Assurance is continually representing the stakeholders' value propositions and checking their consistency with artifacts of the software development process (Biffl, Aurum et al. 2006). Currently, domain experts and engineers use a multitude of notations and tools to represent their views on a software system and the evolution process; however, the views represented in these notations and tools are often fragmented, inconsistent, and challenging to reconcile and check for Quality Assurance.
Ontologies can support the requirements engineering and Quality Assurance processes by providing a continuous model for software-intensive systems, their environments, and processes supporting elicitation, representation, and analysis of the interdependencies among artifacts of software-intensive systems on engineering and domain levels. Another application of ontologies to systems engineering is modeling the system requirements together and their connections to development artifacts (Lee and Gandhi 2005). Ontology-based reasoning can facilitate analyzing the impact of requirement changes, supporting a more consistent handling of changing requirements and a more continuous representation of the stakeholders' value propositions.
In this section we introduce a continuous engineering ontology (the EKB) for Quality Assurance of software and system development. We report from an industry case study that a) introduces an ontology approach for iteratively designing component-based dependable systems in production automation, and b) discusses the expected benefits and risks for building and assuring stakeholder-related quality compared to a traditional development approach (Biffl, Mordinyi et al. 2008).

6.5.1 Ontology-Supported Life Cycle Quality Assurance

Figure 40 shows the process for development and generation of new system versions for production automation that implements stakeholder quality requirements and traces design decisions by means of ontology-supported continuous modeling.

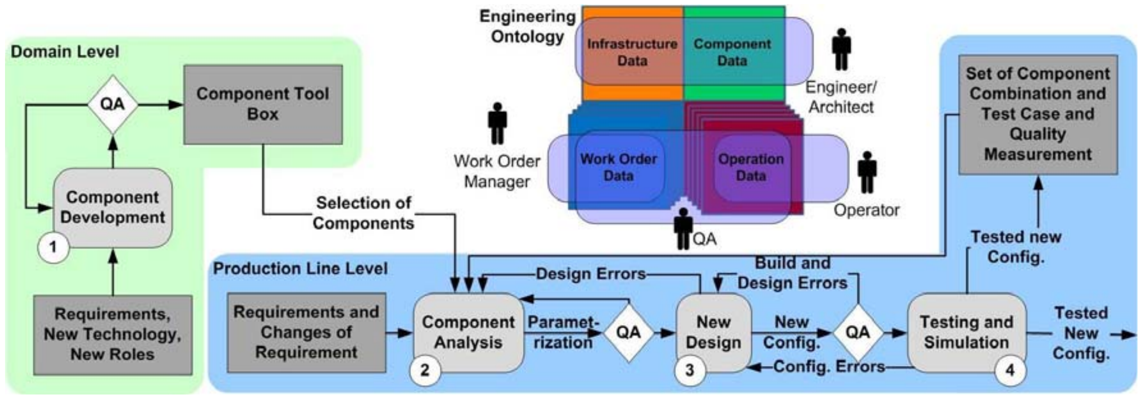


Figure 40: EKB-based Engineering Approach (Biffl, Mordinyi et al. 2008).

The ontology-supported software engineering processing is divided into the domain level and production-line level development process. On each level requirements and capabilities are described semantically. The domain level represents the development activities for a reusable set of software components, the “component tool box”. The production line level outlines the activities of the actual system configuration in order to build a particular product. This process consists of component analysis, design, testing and simulation of new configuration versions. New production line system versions are defined from components in the component tool box and the configuration of the production system. Since quality measurement, Quality Assurance, and auditing are major issues in safety-critical systems, we describe the key steps in the cycle that deal with stakeholder-relevant Quality Assurance (Biffl, Mordinyi et al. 2008).

6.5.1.1 Step 1: Component Development

Based on requirements or triggered by new technologies or roles, components are developed which are used in the production automation system. The developed component runs through the first static Quality Assurance test using ontology support. Based on the requirement descriptions of the component, tests instances are generated; e.g. unit tests for specific functionality. In addition it can be checked whether all component dependencies and security aspects are fulfilled. If the tests are successful the component is added to the Component Tool Box; errors are reported to the developer of the component (Biffl, Mordinyi et al. 2008).

6.5.1.2 Step 2: Component Analysis

The system reconfiguration cycle at the production line level is triggered either by new or changed requirements or components. The reason could be the selection of a new production strategy due to changed working capacities or altered customer requirements. The input to the component analysis step is a set of components from the

Component Tool Box that fulfill the specified requirements. Additionally, the current combination of components representing the current production system is taken as input as well. The analysis step creates all possible combinations of the input with respect to compatibility of the components with each other. The set of components is then parameterized according to the analysis of historical test cases measurements, which are returned to the Component Analyses step by means of a feedback cycle (Biffl, Mordinyi et al. 2007). The next Quality Assurance check point has to ensure that each parameterized combination still fulfills customer requirements. Issues and defects are reported back to the component analysis step (Biffl, Mordinyi et al. 2008).

6.5.1.3 Step 3: New Design

During the design phase complex requirements have to be fulfilled focusing on choosing the right combination of components. The step selects the combination that fulfills non-functional requirements like production time, cost or machine utilization. The selected combination is then transformed into a configuration view that can be interpreted by the production system. The third Quality Assurance checkpoint focuses on the new configuration that has to pass tests which e.g. check its completeness and syntax. Issues and design defects are reported to step 2 (Biffl, Mordinyi et al. 2008).

6.5.1.4 Step 4: Testing and Simulation

In general it is necessary for safety-critical systems, such as for production automation, to assure that the configurations meet overall requirements including system safety before deployment to real-world environments. Therefore, the operation of tools to measure system quality and performance of the new configuration is mandatory. The introduced cycle represents another source of error, so there is the possibility of remaining unresolved (uncritical) defects, wrong responses to failure scenarios. One solution is to execute simulations representing relevant properties of the target system so that the built in monitoring functionality is able to produce monitoring data which can be further evaluated and used by step 2 for component selection. This means that in comparison to traditional approaches with implicit feedback by manually analyzing the results of test case runs, this approach explicitly provides measurement feedback integrated into the ontology for step 2. The successfully tested and simulated configuration can be deployed and used as new current system component for step 2. Defects found during simulation and testing are reported (Biffl, Mordinyi et al. 2008).

6.5.2 Using the Engineering Knowledge Base for Quality Assurance

The continuous model used during the engineering approach is the Engineering Knowledge Base. This ontology consists of several ontology areas containing the concepts and individuals of a certain category of the production automation environment. As sketched in Figure 40, the EKB consists of areas describing the infrastructure and layout of the assembly workshop, the building plans and properties of the components, the data of the concrete work orders derived from the business orders and the measured data of the operation/simulation (Biffl, Mordinyi et al. 2008).
Multiple versions of ontology areas may be used sequentially, e.g., for analysis. This means that certain ontology areas can be populated using either time ("time slices") or version constraints. Using this approach it is possible to define a number of test cases, which should be executed consecutively (Biffl, Mordinyi et al. 2008).
Figure 40 shows a number of role-specific to access the engineering ontology. This allows more effective management of ontology areas a certain role is interested in, since the data can be presented in a well-accepted format/tool for this role, e.g., work order manager, operator, architect, or Quality Assurance personnel (Biffl, Sunindyo et al. 2009).

6.6 *Supporting Runtime Decisions using Design Time Information*

Engineers, who want to adapt the system at runtime, need information from software models that reflect dependencies between components at design and run time, e.g., the workshop layout, customer orders and assembly procedures that translate into needs for machine function capacities over time; and the coordination of tasks for redundant machines in case of a failure. During development design-time software models like data-oriented models (e.g., class or EER diagrams) or workflow-oriented models (e.g., sequence diagrams or state charts) are the basis to derive run-time models but are often not provided in machine-understandable format to reflect on changes at runtime, i.e., the knowledge is kept in an explicit human-understandable way but cannot be accessed by components automatically. Domain and software experts are needed to integrate the fragmented views (e.g., propagating model changes into other models, cross-model consistency checks) from these models, which often is an expensive and error-prone task due to undetected model inconsistencies or lost experience from personnel turnover.
Practitioners, especially designers and quality assurance (QA) personnel, want to make reconfigurable software-intensive systems (which like SAW consist of components defined by general design-time behavior, derived run-time configuration, and run-time specific behavior enactment) more robust against important classes of failures: machine

failures, misuse from invalid supply, and failure-related changes in machine capacities at runtime. QA people could benefit from more effective and efficient tool support to check system correctness, by improving the visibility of the system defect symptoms (e.g., exceptions raised from assertions).
Challenges to detect and locate defects at run-time come from the different focus points of models: e.g., components and their behavior are defined at design time, while configurations may change at run time and violate tacit engineering assumptions in the design-time models. Without an integrated view on relevant parts of both design-time and run-time models inconsistencies from changes and their impact are harder to evaluate and resolve between design and run time.
Better integrated engineering knowledge can improve the quality of decisions for run-time changes to the system, e.g., better handling severe failures with predictable recovery procedures, lower level of avoidable downtime, and better visibility of risks before damage occurs.

In this section we present an approach to improve support for run-time decision making with an ontology: a domain-specific engineering knowledge base (EKB) that provides a better integrated view on relevant engineering knowledge in typical design-time and run-time models, which were originally not designed for machine-understandable integration. The EKB can contain schemas on all levels and instances, data, and allows reasoning to evaluate rules that involve information from several models that would be fragmented without machine-understandable integration (Moser, Schatten et al. 2009).
Using the EKB provides the following benefits: a) Uniform and efficient access to related data in design-time, deployment, and run-time models on the levels of schemata, instances, and configurations. This feature allows reasoning to effectively evaluate specific decision alternatives, e.g., the importance of defect messages. Further the feature supports checking the impact of changes in one model on the consistency of other models; and b) derivation of assertions that should hold at runtime to establish and maintain causal links between design/runtime models and the running software. These assertions can support systematic exception handling and escalation procedures; e.g., check correct sequences of messages; valid behavior of component groups; or check actual run-time performance with design estimates/limits (Moser, Schatten et al. 2009).

6.6.1 Engineering Knowledge Base Architecture

In this section, we introduce the Engineering Knowledge Base (EKB) used for this scenario, a set of relevant information elements about components in machine-understandable format using ontology syntax. Components can query the EKB at run time to retrieve information for decision making, e.g., enriching and filtering failure information or run-time coordination of machine workloads due to changes in the available machine capacity (Moser, Schatten et al. 2009).

Figure 41 illustrates 3 major phases in the life cycle of software-intensive systems in the production automation domain: 1. *Design time*: Models that describe the workshop layouts, the building plans of manufactured products, etc. are transformed into executable program code and design-time configuration instructions. 2. In the *Deployment* phase, the executable program code is deployed into installable packages and the run-time configuration is derived from the design-time configuration. 3. At *Run Time* the deployed program code for system operation gets installed to a set of components and the run-time configuration gets injected into these components. This architecture has proven effective for systems whose properties change seldom, since the effort needed for transformation, deployment, and injection is considerable (Moser, Schatten et al. 2009).

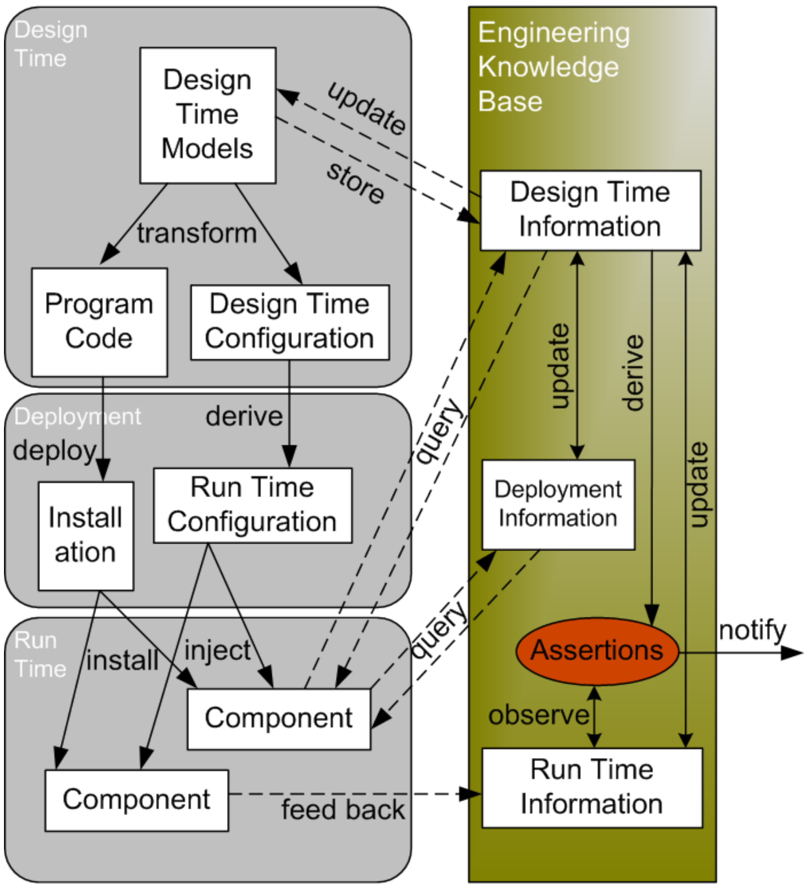


Figure 41: An Engineering Knowledge Base in context (Moser, Schatten et al. 2009).

However, typical software-intensive systems also undergo reconfiguration phases, e.g., if some components fail or become unavailable. To support reconfiguration, the components need to be able to perform decisions at run time, since a complete new iteration of model transformation, program code deployment, and configuration injection would take too long. A major challenge of run-time decisions is to provide access to relevant design-time information that is usually stripped away during transformation for efficiency reasons (Moser, Schatten et al. 2009).

The Engineering Knowledge Base (EKB) provides a place for storing design-time information that seems valuable for supporting run-time decisions of components, especially in the case of handling failures or unplanned situations (but not transformed into run-time code or configuration to limit their complexity) (Moser, Schatten et al. 2009).
Components can query the EKB at run time with query languages like SPARQL[20] (SPARQL Protocol and RDF Query Language) or SWRL[21] (Semantic Web Rule Language), which provide to the components the full expressive power of ontologies, including the ability to derive new facts by reasoning. In addition, components can feed back interesting observations into the run-time information collection of the EKB and therefore help to improve the design-time models (e.g., by improving estimated process properties with analysis of actual run-time data) and/or check the information based on a certain set of assertions. Furthermore, valuable deployment information can also be stored in the EKB in order to support and enhance for further deployments (Moser, Schatten et al. 2009).
Based on the design-time information, it is possible to define a set of run-time assertions in the EKB. These run-time assertions observe the run-time information fed back into the EKB and can notify a specific role or system if the violation of an assertion has been detected (Moser, Schatten et al. 2009).

6.6.2 Examples for Supported Run-Time Decisions

In this section, we describe a real-world use case on system adaptation to accommodate runtime failures. The use case is based on the SAW simulation. In the simulation context we collect evidence to which extent a richer and better integrated semantic knowledge base can translate into more accurate faster and cheaper making. Based on the SAW data model (see also section 6.4), we derived two use case scenarios which show the run-time decision (RTD) support and derived assertions based on the capabilities of the Engineering Knowledge Base (Moser, Schatten et al. 2009).

6.6.2.1 RTD-1: Error message filtering and sequencing

Discussions with industry partners showed that there exists need for a more effective detection and handling of failures at runtime. Currently, an operator conducts front-line failure handling and may receive a (potentially very large) set of error messages from distributed components. These components and their errors may be interrelated in several ways (logical, process, physical, etc.). As example for a wide range of failures

[20] www.w3.org/TR/rdf-sparql-query/
[21] www.w3.org/Submission/SWRL/

of components that are interdependent (not always in obvious ways): an initial failure of a non-redundant conveyor may cause follow-up failures at (distant) machines that depend on transport from the failing conveyor. Typically, the operator will be notified about a set of failures and may not know which failure is of first importance; in most cases the operator tends to handle failures of expensive machines first before attending to less expensive components, such as the conveyor belt, which may lead to an ineffective sequence of recovery activities.

Using the EKB approach, it is possible a) to model the component interdependencies and their relationships to errors/failures and b) derive the original source of failure using ontology-based reasoning as well as present additional important failure information like instructions and responsibilities to the operator. Important run-time decisions in this scenario are which messages should be provided to the operator and which messages should be filtered out (e.g., less important defects) as well as the sequence in which the messages should be presented to the operator (e.g., messages from most important components first – most important does not necessarily mean largest or most expensive component).

An assertion for this run-time decision can be to flag not only direct failures of components but also non-reachable components or parts of the overall workshop. If relevant parts of the workshop are not reachable for an extended period, both the operator and some higher-level roles like a dispatcher or a ERP system need to be notified about the (temporary) change of the workshop system capacity so they can react to the foreseeable capacity change in time.

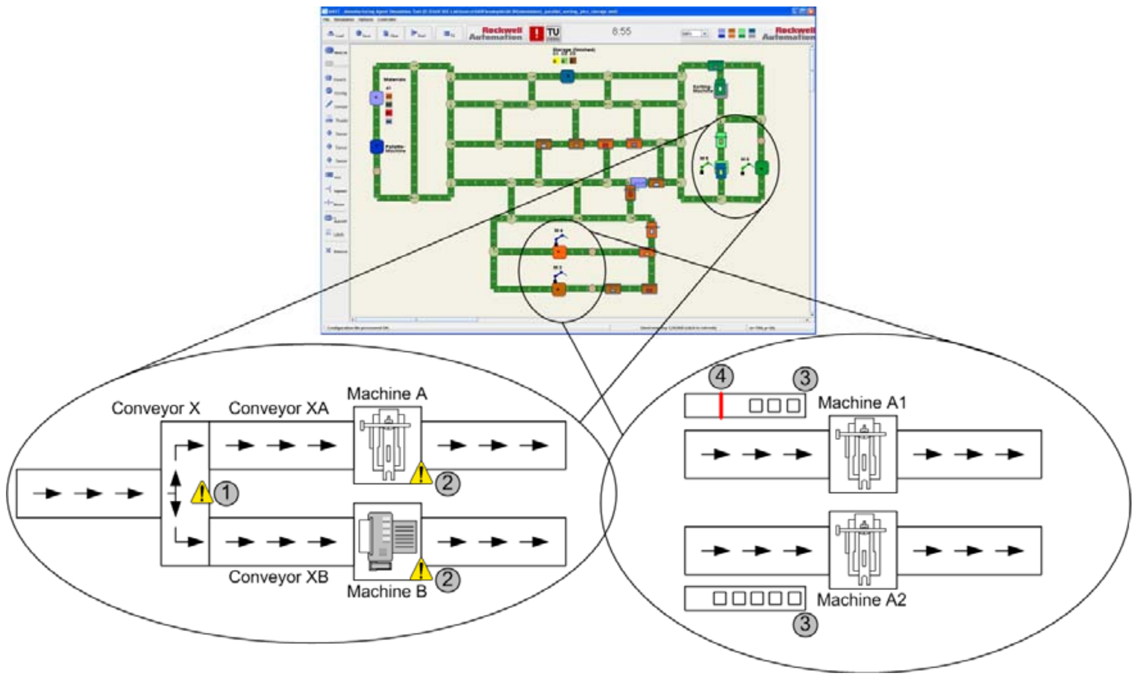


Figure 42: EKB-supported runtime decisions (Moser, Schatten et al. 2009).

Figure 42 (bottom left) shows how a failure in Conveyor X (1) will result in unreachable Machines A and B (2) and consequent follow-up failure messages. However, the operator will receive in general several failure warnings, and has to reason on their overall meaning. If he chooses to handle the failure of Machines A or B first, valuable time will be lost. The EKB can deduce that Conveyor X is a predecessor of both Machines A and B using the *connectedTo* relation between Conveyor and Component.
Thus it is possible to define a query whether the failure of a component should be reported to an operator or should be filtered out; see Listing 6 for a simple query definition which implies a failure of a component C should be filtered out if a failure of any predecessing component P occurs at the same time and if the Failure ID is not severe, e.g., of failure class "3"; in Listing 1 *reportFailure*(C) will return false, since the failure code is "5" and P is a predecessor of C. This query can be used as soon as a number of failures occur in order to identify a possible source of failure and to suppress confusing follow-up failures.

Listing 6: Rules for Reporting Failures (Moser, Schatten et al. 2009).

```
Failure(C,5,"noPalletInput").
Failure(P,4,"motorBroken").
predecessor(X,Y) :- connectedTo(X,Y).
predecessor(X,Y) :- connectedTo(X,Z),
                          predecessor(Z,Y).

reportFailure(C) :- not predecessor (C,P).
reportFailure(C) :- Failure(C,3).
```

6.6.2.2 RTD-2: Individual Preparation of Machine Maintenance Tasks

A second relevant run-time decision in the production automation scenario is when to perform machine maintenance tasks in order to keep a certain minimum level of production output.
This decision could also be taken during design time, resulting in a decreased ability to react to new or changing environment conditions (e.g., failures, reconfiguration). Since system flexibility supports operational efficiency in production automation, the decision when to perform machine maintenance tasks (e.g., cleaning, refurbishment) and the preparations for these tasks (i.e., emptying the machine buffers) could be taken by the machines themselves taking into account the state of other machines and workshop environment conditions. The idea is to coordinate the maintenance tasks of a set of related machines to minimize the impact on the overall production process. This planned maintenance should also be reported to a controlling system (e.g., an ERP system) in order to allow in-time reaction to the future capacity changes.

An assertion for this run-time decision can be to detect a situation in which too many machines plan to go into maintenance mode at the same time and react by preventing some machines from entering maintenance mode and notifying an operator to investigate the actual situation.

Figure 42 (bottom right) illustrates two redundant machines (machines that provide at least in part similar functionality): Machines *A1* and *A2*, have independent virtual input buffers (3) that contain all work orders which are scheduled to be processed at the particular machine. In addition, every machine has a counter counting the number of performed operations, as well as a threshold defining the approximate number of operations after which machine maintenance tasks should be performed. Based on the average time a machine function takes, a machine can derive the optimal moment for starting maintenance tasks. As preparation of these tasks, the virtual machine buffer needs to be closed (4) and all remaining work orders need to be processed. To avoid that all redundant machines switch to maintenance mode at the same time, the planned maintenance time needs to be stored in the EKB, so other machines can adjust their maintenance plans.

Listing 7: Reporting capacity changes to ERP (Moser, Schatten et al. 2009).

```
Task(T1,F1).
Task(T2,F1).
Task(T3,F1).
Machine(A1,F1,1,0).
Machine(A2,F1,2,15).

ReportCapacityChange(M1) :-
   Machine(M1,X,_,_),
   count(Task(_,X)) > Machine(M2,X,Y,_).
```

Listing 7 shows a simple query to check whether a capacity change caused by a machine going into maintenance mode can be handled by the system itself or should be reported to a dispatcher and/or ERP system. There are three tasks which need the same machine function F1 and two machines *A1* and *A2*, which offer machine function *F1*. *A1* has a capacity of 1 task but needs to go into maintenance mode now since its number of allowed operations before a planned maintenance is 0. Machine *A2* has a capacity of 2 tasks and 15 more allowed operations before its planned maintenance phase. The query now checks whether the number of tasks needing a certain machine function (*F1* in our example) is greater than the available capacity of the other machines which offer this machine function (in our example machine *A2*) and returns true if the number of tasks is greater than the available capacity. In our example the query would return true, since there are three tasks needing the machine function F1, but the available capacity of machine *A2* is only 2 tasks.

6.7 *Semantic Event Correlation*

Complex event processing (CEP) is a modern software engineering paradigm that aims at integrating heterogeneous software systems based on the events they produce and consume. Event correlation is the act of connecting related events gathered from various sources to scan for patterns and detect situations of interest. So far, event correlation has been limited to match only equal values of event attributes to decide if two events are related. However, this approach is adequate only if the data quality of the attribute values on which the correlations are based is guaranteed. While this is even hard to ensure for a single organization, it is even more difficult for various, different organizations. Semantic correlation provides an explicit way to model these differences in a decoupled layer providing the means for better adaptability and reusability. Furthermore, it facilitates building correlations based on inherited meanings of terms as well as on relationships between them.

In this section, we examine how ontologies can be used in software engineering to complement the current approach and add semantics to the evaluation of correlations sets. We identified 3 application scenarios for event correlation: basic, inherited and relation-based semantic correlation. Basic correlation does not require the events to have exact equal values but matches equivalent terms. Inherited semantic correlation goes one step further through the use of a taxonomy consisting of a set of sub-concepts defined using ontologies. Relation-based semantic correlation uses relations defined in ontologies to correlate events if their attribute values are in such a relation. For example, the ontology defines that department X is responsible for shipping product Y in country Z. This can be used to correlate events from this very department and the related shipping events for this product in that country (Moser, Roth et al. 2009).

6.7.1 Complex Event Processing

The term of Complex Event Processing (CEP) was first introduced by David Luckham (Luckham 2002) and defines a set of software engineering technologies to process large amounts of events, utilizing these events to monitor, steer and optimize businesses in real time. The main application field for CEP generally is in areas where low latency times in decision cycles are needed, combined with a high throughput for observed relevant business events of predefined or exceptional situations, indicating opportunities and problems.

Typically, these are areas like financial market analysis, trading, security, fraud detection, logistics, compliance checks, customer care and relationship management; and more generally speaking, the monitoring of business processes and the reaction to these processes with short delay. A CEP system continuously processes and integrates the data included in events without the need for batch processes to extract and load data from different sources and store this data in a data warehouse for further processing or analysis. CEP solutions capture events from various sources and establish a relational connection between them. Schiefer et al. introduced *SARI* (Sense and Respond Infrastructure) (Schiefer and Seufert 2005) and included the original event correlation approach (Schiefer and McGregor 2004) which we extend in this section to semantic relationships between events.

6.7.1.1 Correlation Meta Model

The correlation meta-model, illustrated in Figure 43, is a straight-forward, nested structure for defining correlations for event processing model components. A correlation set contains at least one correlation tuple. A correlation tuple is identified by a unique id and can contain more than one Event Type Selector. An Event Type Selector can only contain one Attribute Selector which selects the value used to check on quality with the event attribute value of another event type. The Event Type Selectors defines which attributes of the corresponding types make up the relationship. The Attribute Selector itself can be defined using EAExpressions or XPath, whereas EAExpressions are a domain specific language that allows to access event type details and perform various calculations, evaluations or apply mathematical functions (Moser, Roth et al. 2009).

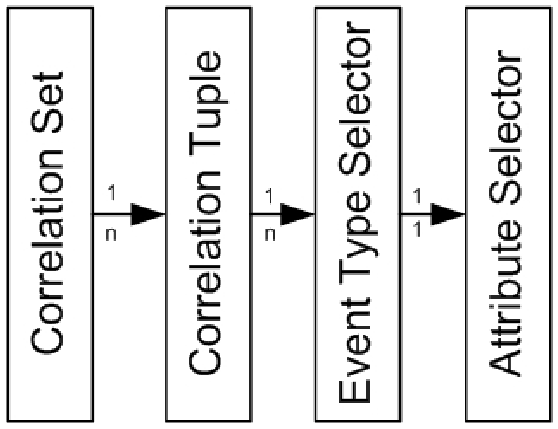


Figure 43: Correlation meta-model (Moser, Roth et al. 2009).

6.7.2 Semantic Correlation of Events from Heterogeneous Systems

Semantic correlation complements the correlation approach by integrating ontologies in the correlation meta-model. Our approach facilitates 3 different possibilities to semantically correlate events which are: 1) building correlations because of equal meaning, not just because of the exact equality of event attributes, 2) resembling terminology hierarchies to correlate events with differing event attribute values derived from the actual meaning and 3) defining relationships between terms. This gives us the

powerful means to define correlations which depend on an event attribute of one event type and several attributes of another. In addition, ontologies can be used to integrate events coming from different organizations or using different terminologies, by providing a kind of domain knowledge base containing the knowledge representing the events to be integrated. The events are then mapped to the knowledge terms they represent, allowing a transformation between different event types. If needed, this mapping can be broken down to the attribute level, allowing an even more fine-grained transformation of the events (Moser, Roth et al. 2009).
The alternative approach to solve the problem at hand would be to translate event attribute values to generally accepted terms before applying the traditional correlation approach. Apart from the very likely usage of ontologies in both cases, in both cases, this is not an ideal approach in our opinion since several problems prevail. First of all, it only helps to solve the basic semantic correlation problem but it does not provide the opportunity to define inherited semantic correlations and relation-based semantic correlations. Secondly, a single event attribute may be used multiple times in a correlation definition. Furthermore, the decision on how to map these values has to be made at design time whereas our semantic correlation approach can be applied ex-post, e.g. for analytical purposes. Finally, incorporating this knowledge explicitly using ontologies in a decoupled way is favorably in our opinion as far as maintainability and reusability are concerned (Moser, Roth et al. 2009).
In the following, we use an example from a production automation environment where different products are produced by a set of machines providing different machine functions to illustrate the three identified possibilities. Product orders arrive continuously and are assigned automatically to suitable machines. Products are made of different materials and need a certain set of machine functions to process these materials. Upon the arrival of a new product order, all the machines publish an offer which consists of their available set of functions, their utilization rate, the potential cost as well as other information. Semantic correlation is used to match suitable responses to the order event. Using this approach, the assignment of orders to machines can be extracted into a higher level definition allowing a more flexible integration of new machines and products. In the following, special cases of this example will be used to three different degrees of usages of ontologies (Moser, Roth et al. 2009).

6.7.2.1 Basic Semantic Correlation

In production environments many heterogeneous systems communicate with each other, each using its own terminology. Ontologies support the transformation between events from these systems and therefore shorten the development cycle.
A correlation set based on the correlation meta-model shown in Figure 43 puts two events into the same correlation if the selected data elements of each one of the correlation tuples are equal. A minimalist approach using an ontology-based semantic

correlation meta-model allows us to be more flexible and to match on equal meaning rather than on equal value. With these correlated events it is possible to measure the total amount of orders in a specific time or the average available delivery time. A certain product may be known under different names, depending on the context. Until now, every order was either required to use the same product name, or it had to be mapped somewhere. The use of ontologies makes this mapping explicit, reusable and easily adaptable allowing the participants to use their own terminologies.
Figure 44 illustrates this example. For brevity, only one concept of the ontology and two of its individuals, *Björn* and *Bjørn*, are shown and should indicate that different product names exist for the same product. The self-correlating correlation set definition is linked to this concept and therefore, event attributes containing any of these individuals are semantically equivalent and the events become correlated.

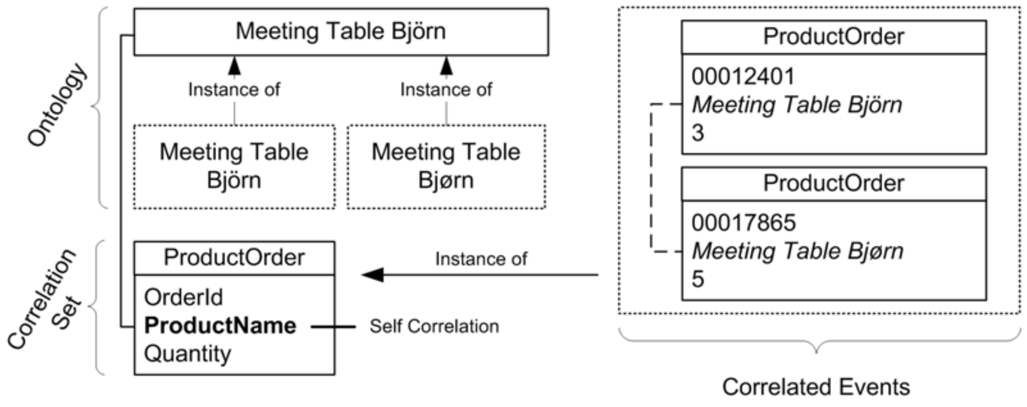


Figure 44: Basic semantic correlation (Moser, Roth et al. 2009).

6.7.2.2 Inherited Semantic Correlation

Semantic correlation based on derived terms that share the same, inherited meaning as the one being matched loosens the concepts of correlations even more. Using ontologies to define inheritance hierarchies of the domain terminology isolates this aspect and makes it easier to define correlation sets if the values of event attributes can be more fine-grained but when this level of detail is of no importance.
For example as shown in Figure 45, products could be grouped in product categories which can be further grouped in product lines. Inherited semantic correlation can now be used to define semantic correlations on all orders for products from a certain product group as well as from the same product line. The information from this correlation can then be used to calculate metrics on the product orders on the level of product groups or product lines.

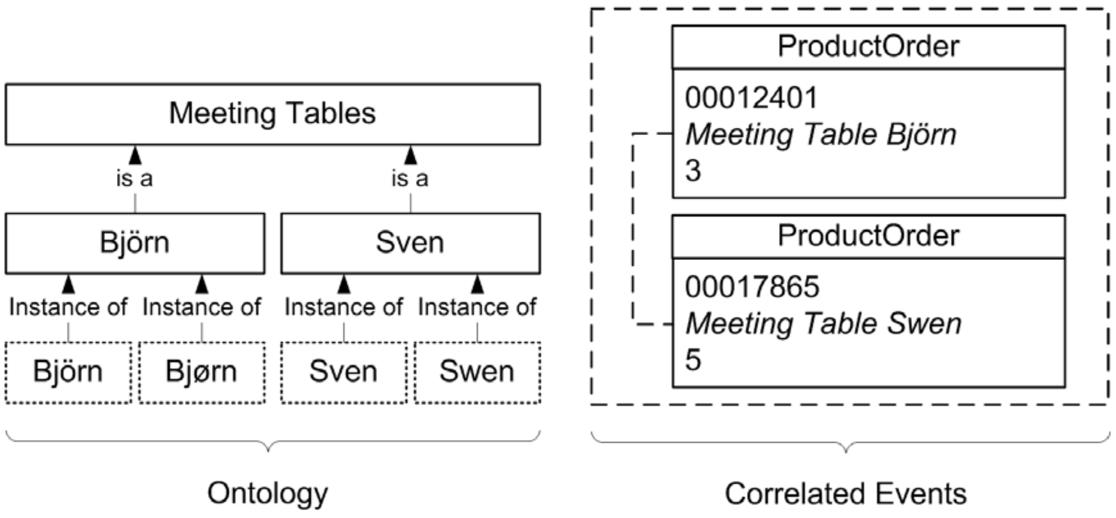


Figure 45: Inherited semantic correlation (Moser, Roth et al. 2009).

6.7.2.3 Relation-Based Semantic Correlation

Finally, ontologies allow defining relations between terms. The correlation meta-model defined in section 6.7.1.1 uses correlation tuples which exactly match one event attribute of each event type. Relation-based semantic correlations on the other side allow matching multiple event attributes of each event type which define the semantic meaning of this tuple. In other words relation-based semantic correlations allow the correlation of different events, using their semantic relations.

In addition to the characteristics introduced in the previous examples, products consist of a set of one or more different materials and are assembled/produced using at least one specific machine function. Machine functions are offered by different machines, each machine offers at least one machine function.

The example shown in Figure 46 takes place in the following way: As a first step, a certain product is ordered. This order consists of the product ID and the amount of the product. Using the semantic description of the product defined in the ontology, it is possible to determine which materials and machine functions are needed in order to assemble the target product. All available machines periodically broadcast their available machine functions, their costs and their utilization rates. As next step, these broadcast events are correlated with the order event, using the machine functions needed for the assembly of the product, which are retrieved from the ontology, and the available machine functions of the periodical broadcast events of all machines, which are also retrieved from the ontology. This correlation is used to identify all machines providing the needed machine functions for the production of a certain product. In addition, the production costs and the utilization rates can be used to identify the machine representing the best choice for the production process.

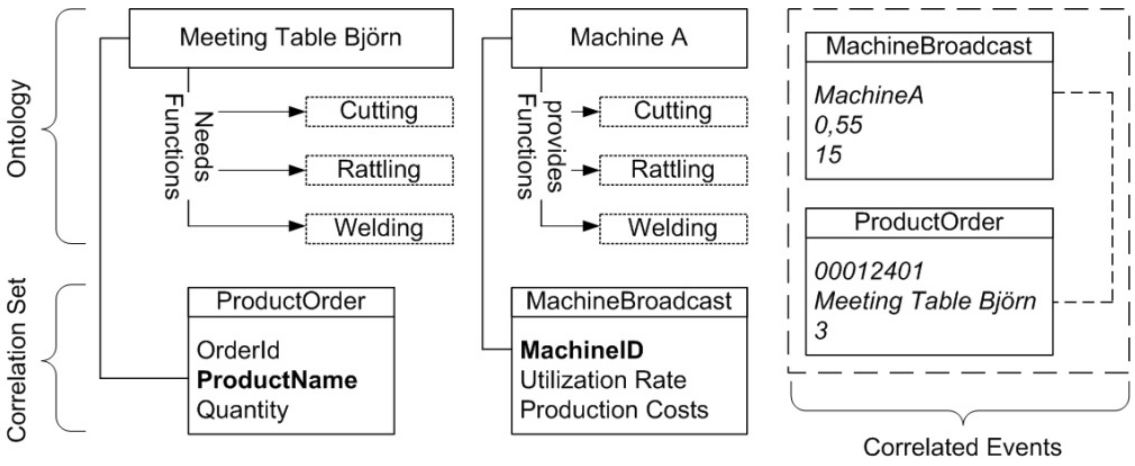


Figure 46: Relation-based semantic correlation (Moser, Roth et al. 2009).

6.7.2.4 Semantic Correlation Meta-Model

After presenting 3 different possibilities of semantic correlations we now extend the correlation meta-model presented in section 6.7.1.1 to include ontologies. Since correlation tuples define the different event types as well as one event attribute for each one of these event types, an ontology concept is assigned to each semantic correlation tuple. An ontology consists of several concepts which themselves can contain sub-concepts and/or individuals (Moser, Roth et al. 2009).

For example, a concept *product name* containing sub-concepts for each available product name with *individuals* different product names for the same product can be used to correlate events on different product names for the same product. As we have seen in section 6.7.2, semantic correlations furthermore allow having more than one event attribute to serve as the matching criteria. Therefore, an event type selector can now have multiple attribute selectors as opposed to the correlation meta-model, i.e. the attributes can be selected not only based on EAExpressions or XPath described in section 6.7.1.1., but also based on semantic criteria like e.g. semantic equality (Moser, Roth et al. 2009).

Figure 47 shows the extended correlation meta-model. The integration of SPARQL (Prud'hommeaux and Seaborne 2007) allows for even more flexibility. SPARQL queries can be used to uncover knowledge not explicitly known by just analyzing the ontology. In addition, SPARQL queries may be used in order to check either the consistency of ontologies or some user defined constraints (Moser, Roth et al. 2009).

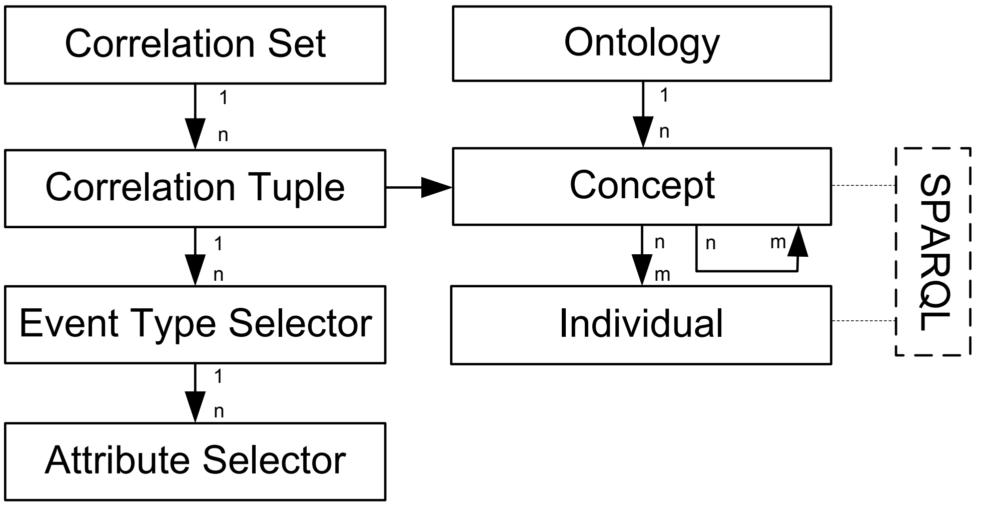


Figure 47: Semantic correlation meta-model (Moser, Roth et al. 2009).

Basic semantic correlation is used to identify semantically identical events, even with different terminologies. This is achieved by tracing an individual defined in the ontology back to its concept. The concept of the events to be correlated is defined in the definition of the particular correlation tuple. During runtime evaluation, an event is looked up in the ontology, and if this individual is derived from the concept defined in the correlation tuple, this specific event is added to the correlation set (Moser, Roth et al. 2009).

Inherited semantic correlations use a similar mechanism as basic semantic correlation. In addition to checking the concept of certain individuals, this type of semantic correlation is used to identify individuals either of the specified concept or of any sub-concept of the specified concept, without any restriction on the hierarchical depth. As before, this concept is defined in the definition of the particular correlation tuple. During runtime evaluation, again an individual is looked up in the ontology. This time, the specific event is added to the correlation set, if either it is an individual of the defined concept or of any sub-concepts of the defined concept (Moser, Roth et al. 2009).

Relation-based semantic correlation takes into account the semantic relations between autonomous concepts. Two event types are correlated based on their semantic concept and object properties. The concepts and the object properties to be correlated are stated in the definition of the particular correlation tuple. During runtime evaluation, for two events that are individuals of the defined concepts it is checked whether the relation between the two concepts defined by their object properties is identical with the semantic meaning of the attributes of the particular events. In case of a positive match, these two events are then added to the correlation set (Moser, Roth et al. 2009).

6.8 *Summary*

In this chapter we proposed and evaluated the EKB framework for semantic mapping in Automation Systems Engineering based on the real-world use case SAW, with a focus on providing links between data structures of engineering tools and systems to support the exchange of information between these tools and thus making Automation Systems Engineering more efficient and flexible.
The major difference compared to traditional approaches, i.e., using a common repository, is the lack of the need for a common data schema when using the EKB framework, which can be seen as the main advantage of the EKB framework compared to a common repository as the common data schema is a source of extra maintenance effort and tool evolution risk. Additionally, the number of needed converters is $O(n^2)$ when using a common repository, compared to $O(n)$ converters needed when using the EKB framework. Further, using the EKB framework allows a more generic definition and execution of model checks on an application domain level, and additionally enables more advanced checks regarding the plausibility and semantic correctness of data structures by exploiting the querying capabilities of ontologies.
The following sub-sections describe the findings and results for the EKB-based process description, for the Ontology Areas concept, for Ontology-Supported Quality Assurance, for supporting runtime decisions using design time information, and for semantic event correlation.

6.8.1 Process Description

In this chapter we introduced ontology support for development and generation of new system versions for production automation that supports variability modeling and traces design decisions by means of ontology-supported continuous modeling. Based on an industry case study, we described the engineering process, the coordination concept, and how software variant performance can be measured and improved (Biffl, Mordinyi et al. 2008).

6.8.1.1 Variability modeling using an iterative feedback driven process

The ontology and CBSE paradigms reinforce each others' advantages: the ontology-supported CBSE approach seems to be more effective and efficient due to reasoning support for selecting and parameterizing the most suitable components out of a component tool box with respect to a certain set of requirements or dependencies between components automatically and therefore without significant sources of defects like manual interaction. The EKB framework supports both static and dynamic QA during the production engineering process. Test result measurements are stored in the engineering ontology and can be used for component analysis in next iteration of the production engineering process in order to create new system versions, and so completing the feedback cycle. Furthermore, the results from running test case documented in simulations in a way that allows efficient quality analysis and comparison of the results with the original assertions (Biffl, Mordinyi et al. 2008).

6.8.1.2 Test case-based engineering approach

The use the EKB framework during the engineering approach provides a continuously available and evolving representation of the stakeholder requirements. Compared to traditional methods, the use of ontologies entails a number of advantages. The output of the simulation is automatically fed back in the ontology, resulting in a combination of a test case and its outcome. The context-based matching of historical test case data allows the effective reuse of knowledge achieved in previous process iterations. These have proven to be useful for performing more advanced statistical analysis on the data, leading to more significant assertions (Biffl, Mordinyi et al. 2008).

6.8.1.3 Optimization of role-oriented views on EKB Ontology Areas

Role-oriented views on the Engineering Knowledge Base assist domain experts and roles that do not easily accept abstract models by presenting data in a well-accepted format/tool. This allows more effective management of the engineering ontology by different roles (Biffl, Mordinyi et al. 2008).

6.8.2 Ontology Areas Concept

Ontologies support the translation between stakeholder local terminologies via common domain concepts, in our case production automation concepts. Typically, the ontology models become very large and complex compared to the basic data model (such as used in a data base to automate run-time processes) if they include several aspects on a domain and some parts of the data model are volatile. In this chapter, we proposed a data modeling approach based on ontology building blocks, so-called “Ontology Areas”, which allow solving tasks with smaller parts of the overall ontology. We evaluated the proposed approach with use cases from the production automation domain. Major result in the study context is that Ontology Areas improved the efficiency of data collection task for decision making by lowering the cognitive complexity for designers and users of the ontology (Biffl, Sunindyo et al. 2009).

6.8.2.1 Lesson learned

From the experiences with the use cases (see section 6.4.2), we can learn the following lessons (Biffl, Sunindyo et al. 2009).

Building a smaller ontology for a task. As Ontology Areas allow focusing on the content of interest for a stakeholder task, we could show that the resulting ontology is considerable smaller. A smaller ontology is often also more efficient to handle and allows tackling tasks that use a particularly large number of data elements (e.g., run-time measurements).

Focus stakeholders on relevant data elements. The combination of Ontology Areas, design-time, and run-time data elements allowed filtering relevant data elements for stakeholders, which would not be possible without the combination. Thus the Ontology Area approach helped lower the cognitive complexity for stakeholders by providing just the relevant subset of the comprehensive ontology.

Version management for ontology areas. With the Ontology Area concept we can flexibly build task-oriented ontologies based on different criteria (like volatileness, layers, roles, etc.). It is even possible to compare different versions of the same Ontology Area (e.g., production automation system designed with different parameter settings) to compare the run-time reactions to from changing design parameters. However, this ability also raises the need for better version management for Ontology Areas to ensure the building of consistent ontologies for specific tasks.

6.8.3 Ontology-Supported Quality Assurance

In this chapter we introduced ontology support for systems engineering that explicitly describes stakeholder quality requirements and traces design decisions to generate new system versions that implement these requirements. Based on an industry case study, we described the ontology concept of the system, the development process, and how software quality can be measured and improved (Biffl, Mordinyi et al. 2008).

6.8.3.1 Explicit and continuous modeling

The use of the EKB framework during the engineering approach provides a continuously available and evolving representation of the stakeholder requirements. Compared to traditional methods, the use of ontologies entails a number of advantages. As shown in the case study, this allows a more automated QA support. In addition the usage of the ontology area concept creates a personalized view on the data model for each role. The output of the simulation is automatically fed back in the ontology, resulting in a combination of a test case and its outcome. This automated feedback cycle has proven useful for performing more advanced statistical analysis on the data, leading to more significant assertions for the generation of new system versions (Biffl, Mordinyi et al. 2008).

6.8.3.2 *Tool support for transformation of explicit requirements and for Quality Assurance*

The ontology and CBSE paradigms reinforce each others' advantages: the ontology-supported CBSE approach seems to be more effective and efficient due to reasoning support for selecting and parameterization of the most suitable components out of a component tool box. This is automatically achieved with respect to a certain set of stakeholder requirements or dependencies between components and therefore without significant sources of defects like manual interaction. The EKB framework supports both static and dynamic QA during the production engineering process. Test result measurements are stored in the engineering ontology and can be used for component analysis in next iteration of the production engineering process in order to create new system versions, and so completing the feedback cycle. Furthermore, the results from running test cases are documented in simulations in a way that allows efficient quality analysis and comparison of the results with the original assertions (Biffl, Mordinyi et al. 2008).

6.8.3.3 *Measurement of stakeholder-level quality of the product and development process*

Stakeholder-level quality is assured by means of ontology-based reasoning, allowing tracing customer-specific requirements continuously throughout the entire production engineering process. The Ontology Area approach and the role-specific views of selected data effectively and efficiently allow the involved roles to check the mapping and tracing of their value proposition and requirements at all times. Conflicts during dynamic QA can directly refer to the quality requirements of a certain configuration (Biffl, Mordinyi et al. 2008).

6.8.4 Supporting Runtime Decisions using Design Time Information

In this paper we described an ontology-based approach to provide relevant design-time and run-time engineering knowledge stored in a so called Engineering Knowledge Base (EKB). The EKB provides a better integrated view on relevant engineering knowledge

contained in typical design-time and run-time models in machine-understandable form to support runtime decisions. This approach is useful in the automation domain, and can more generally be used for other (distributed) engineering systems. We illustrated our approach with two types of run-time decisions from a real-world case study in the area of software-intensive production automation systems (Moser, Schatten et al. 2009).
Major results of the evaluation of the proposed EKB approach were: Due to separation of automation code, diagnosis and decision support the complexity of single components can be reduced by approximately 20-30%, since these components now can rely on external information. Another benefit is the possibility to define assertions in the EKB which are checked based on the run-time information input of the running components. This can be seen as external Quality Assurance without interfering with the original production system and therefore it has proven to be easier to enrich existing applications without the need to make changes to legacy systems (smoother migration path). Further, the quality of information presented to an operator is improved since all information both from design-time as well as from run-time is available, leading to more intelligent run-time analysis and decision support (Moser, Schatten et al. 2009).

6.8.5 Semantic Event Correlation

In this paper we described the role of event correlation in Complex Event Processing and a meta-model for traditional syntactic correlation sets which define how events are related to each other. We extended this meta-model to include semantic correlation sets by incorporating ontologies. We identified three application scenarios that show how ontologies and correlation sets can be combined to semantically correlate events based on meaning, inheritance and relations. Using the proposed semantic correlation approach allows to use and correlate events which by now could not be correlated effectively because of semantically heterogeneous terminologies of participating systems/organizations. The possibility to perform these correlations without the need to change existing events and therefore no need to change running systems strongly increases the flexibility of Complex Event Processing (Moser, Roth et al. 2009).
Using the three identified use cases for semantic correlation, the possibilities for the identification and processing of events are broadened, allowing further usages of events. Events that by now could not have been correlated directly using traditional syntactic correlation methods can now be described and processed using semantic techniques. Compared to the alternative approaches which required changes of the original events and therefore of the running systems, the proposed semantic correlation approach provides a much higher flexibility. In addition, the semantic definition of events and their properties and relations, contributes to the overall understanding of the systems to be integrated and their produced events (Moser, Roth et al. 2009).

Chapter 7

7 Evaluation and Discussion

This chapter presents the results of the evaluation of the EKB framework and discusses these results with regard to the specified research issues (see section 3.1). In the first section, the prototypic realization of the four usage scenarios, namely data exchange between tools, model consistency checking across tool boundaries, impact analysis of model value changes, and end-to-end analysis, are described in detail. The second section describes the evaluation of the SWIS application scenario, while the third section describes the evaluation of the SAW application scenario. In the fourth section, the benefits and limitations of the EKB framework are summarized, and the specifics of applying the EKB framework to the two application scenarios, SWIS and SAW, are described.

7.1 *Prototypic Realization of the Usage Scenarios*

This section describes the prototypic realization of the four usage scenarios of the Engineering Knowledge Base framework identified in section 4.2.

7.1.1 Data Exchange Between Tools

To cooperate the engineers have to exchange relevant parts of the data structures (i.e., information required in another tool should become available as soon as it has been saved in the original tool) in their tools with each other with the goal of a consistent overall view on certain aspects in the project, e.g., when producing a specification for a subcontractor. Currently, every role uses organization-, domain-, and tool-specific data formats and terms, thus the data exchange takes considerable expert knowledge on the receiving end to make sense of the incoming data, typically as large PDF document or tool-specific import file (Moser, Biffl et al. 2010).

7.1.1.1 Common Repository Approach

The exchange of data structures originating from different engineering tools using a common repository requires a set of prerequisites. Either, all participating tools need to agree on a common data schema used for the data structure exchange. All exchanged information is then structured according to this schema. While this is even hard for tools originating from the same engineering domain, it becomes nearly impossible for tools originating from different and typically heterogeneous engineering domains. In addition, changes to one or more of the engineering tools regularly require an update of the common schema, which then needs to be forwarded to the other engineering tools which use this schema. So the major functionality of the common repository is to store all information, while at the same time providing point-to-point integration between all participating tools using converters for each possible combination of the tools (Moser, Biffl et al. 2010).

Once set up and configured properly, this data exchange method has a low delay, i.e., information made available by an engineering tool is available for all other engineering tools that need these information. However, the configuration of this approach requires high effort, since converters need to be written for all needed pairs of *n* engineering tools, with $O(n^2)$ required converters. In addition, the common repository is inflexible and fragile in case of changes of single engineering tools, since converters need to be adapted or complete rewritten in this case (Moser, Biffl et al. 2010).

7.1.1.2 Engineering Knowledge Base (EKB) Approach

A first step in using the EKB framework is the identification of common concepts used in the participating engineering tools. These common concepts are then described in the Domain Knowledge Base (DKB, see Figure 48). As a next step, the proprietary tool-specific knowledge is mapped to the more general knowledge stored in the DKB. Based on these mappings, the EKB framework semi-automatically generates transformation instructions for transforming data structures between tool-specific formats. This semi-automated generation exploits the mappings stored in the EKB and makes suggestions for possible transformations to be reviewed by a human expert. The human expert then can revise the suggested transformation, change them or add new or more complex transformation rules manually. Since for each of the *n* participating engineering tools a single transformation instruction is required, the number of overall needed transformation instructions is $O(n)$ (Moser, Biffl et al. 2010).

While the EKB framework requires similar or at most slightly higher effort for setup and configuration compared to the common repository approach, new benefits come from using ontologies for storing the engineering knowledge. The ontologies enable the semi-automated generation of the required converters, both initially and when engineering tools evolve. The number of required converters is also smaller with $O(n)$ converters for *n* engineering tools. Further, once set up, the delay of the data exchange method is similar to the delay using the traditional common repository based approach (Moser, Biffl et al. 2010).

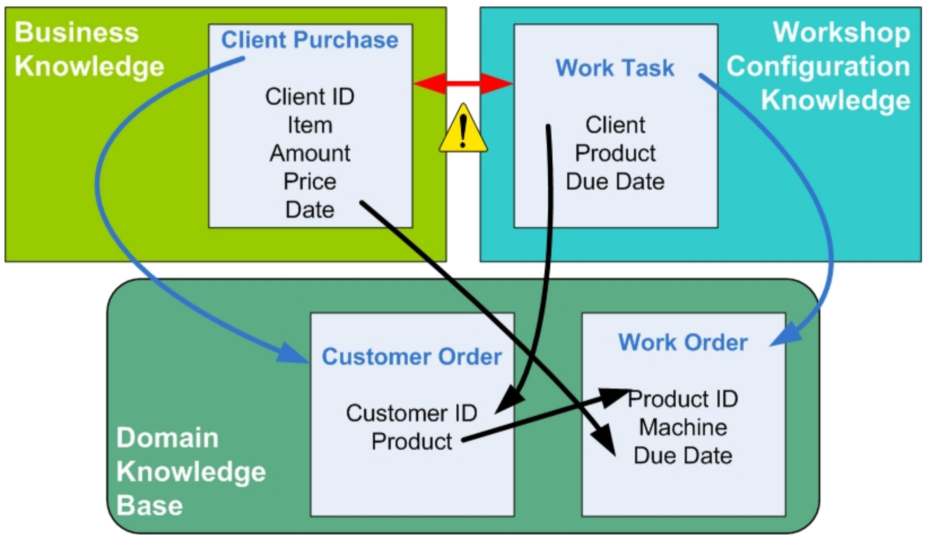


Figure 48: Translation between Business and Workshop Configuration Knowledge.

Figure 48 illustrates two different engineering-tool-specific terminologies of the SAW production automation system. The business knowledge uses the concept *ClientPurchase* as a local terminology, while in the workshop configuration knowledge the concept *WorkTask* is used as a local terminology. As shown in the figure, both concepts are mapped to the corresponding domain concepts, *CustomerOrder* and *WorkOrder* respectively. In addition, the attribute *Date* of *ClientPurchase* is mapped to the attribute *DueDate* of *WorkOrder*, while the attribute *Client* of *WorkTask* is mapped to the attribute *CustomerID* of *CustomerOrder*. Further, in the Domain Knowledge Base the concepts *CustomerOrder* and *WorkOrder* are linked by their common attribute *Product* and *ProductID* respectively. Using these mappings, we can identify work tasks which belong to a specific client purchase or vice-versa identify the corresponding client purchase for a specific work task, without the need to establish a direct link or mapping between the two local terminologies (Moser, Biffl et al. 2010).

7.1.2 Model Consistency Checking Across Tool Boundaries

Model checking refers in the evaluation study context to the inspection of model data elements regarding their consistency and integrity. As a first step towards comprehensive model checking, checks of local data structures belonging to a specific engineering tool can be performed. For these checks, no access to data structures of other engineering tools is needed. However, since the data structures of the single models are viewed independent of the data structures of other engineering tools, more advanced checks regarding a combination of data structure of multiple engineering models. For this use case, we focus on two types of models checks, namely a) consistency and integrity checks of model changes; and b) the derivation of runtime functionality for automated testing and monitoring (Moser, Biffl et al. 2010).

An example for consistency and integrity checks of model changes is a hardware pump which supports a certain number of input/output signals (I/Os), and which is controlled by a pump control software using either analog or digital signal processors. Analog signal processors can handle 8 I/Os, digital 64 I/Os. If the signal processor type is changed in the pump control software model, it needs to be validated whether the new signal processor type can handle all available I/Os of the hardware pump. Respectively, if the I/Os are changed (e.g., new I/Os added) it has to be checked whether they all can be controlled using the chosen signal processor type of the pump control software. Another example for the derivation of runtime functionality for automated testing and monitoring is again a hardware pump which can handle approximately 1000 liters per hour. A time-based analysis of the events originating from the reservoir located behind the hardware pump could show impossible conditions or sensor states, e.g., if the reservoir capacity of 10000 liters is reached within 5 hours starting from an empty condition (Moser, Biffl et al. 2010).

7.1.2.1 Common Repository Approach

Using a common repository enables to perform advanced checks regarding the data structures of more than one engineering tool, such as checking the consistency of single data structure elements across tool boundaries or analyzing the possible impact of changes to data structures belonging to a specific engineering tool on the data structures of other engineering tools. The major drawback of this approach of performing model checks is the need for manual involvement of human experts. The experts need to explicitly define the checks and select the data they want to include in these checks. This definition needs to be updated after every change to the involved data elements. Additionally, the nature of the common repository allows only for syntactical checks (e.g., the availability of all obligatory data fields or the validity of data types regarding a certain data schema) of the data, but not for other checks such as regarding the semantic correctness or plausibility of data structures. Other checks, such as checks regarding logical connections of data elements, are not supported out of the box using a common repository, since the data elements in the repository are stored unaltered and without meta-information. However external analysis tools can use the data elements stored in the common repository for performing such model checks (Moser, Biffl et al. 2010).

7.1.2.2 Engineering Knowledge Base (EKB) Approach

The EKB framework enables automated checks regarding both syntactical issues as well as plausibility checks regarding semantic correctness of data structures. The EKB framework exploits the querying capabilities of ontologies to allow even more advanced checks, such as checks regarding completeness of available information. Human experts define checks regarding specific domain or tool-specific concepts, which are then on-the-fly transformed into checks regarding tool-specific data structures accordingly. The results are then collected and again transformed into the domain concept level, allowing experts both to define checks as well as to view the results of the defined checks in their well-known syntax, terminologies and notations (Moser, Biffl et al. 2010).

7.1.3 Impact Analysis of Model Value Changes

The problem from the SAW context (see 3.3.2) described here corresponds to the sales manager’s task of identifying the maximal amount of products that can be produced during a shift. To do this task, the sales manager needs to collect information from other stakeholders and make calculations based on the information collected before getting to the final result (Moser, Biffl et al. 2010).

7.1.3.1 Common Repository Approach

All information of the process production is placed in the common repository. The sales manager retrieves the needed information out of the common repository, while the other stakeholders put the information in the common repository. The drawbacks of this approach are updates submitted by different stakeholders that are hard to handle concurrently, and different formats and syntax originating from different stakeholders. The sales manager has to deal with other data not necessarily needed for his tasks and manual and therefore error-prone steps are required to get the needed data from other stakeholders (Moser, Biffl et al. 2010).

7.1.3.2 EKB Approach

By using the EKB framework, the automated analyses are supported using the following these steps (refer also to Listing 8 for a detailed description of these steps using simplified OWL syntax): The sales manager queries the global view to find out the current shift time. The shift time is identified in the global view and the mapping of the shift time to the workshop manager’s local view is exploited, and subsequently the shift time is queries in the workshop manager’s local view and represented in the global view. As next step, information about the finishing time of the product type of product *prod6* is queried in the global view, resulting again in an exploiting of the mapping to the business manager’s local view, a query of this local view and a representation of the product type and finishing time of the product *prod6* in the global view. Finally, in the global view the maximum amount of products that can be produced in the current shift is calculated using the previously queried information and the result is presented to the sales manager (Moser, Biffl et al. 2010).

```
workshop:shift1 workshop:lasts workshop:14400
workshop:shift1 workshop:order workshop:prod6
business:prod6 business:finishingTime business:50

SELECT(?x) WHERE {sales:shift1 sales:lasts ?x}

SELECT(?x) WHERE {global:shift1 global:lasts ?x}

SELECT(?x) WHERE {manager:shift1 manager:lasts ?x}
Result: x = manager:14400

manager:14400 owl:equalTo global:14400

SELECT(?y) WHERE {global:prod6 global:finishingTime ?y}

SELECT(?y) WHERE {business:prod6 business:finishingTime ?y}
Result: y = business:50

business:50 owl:equalTo global:50

SELECT(?z) WHERE {?x owl:equalTo global:shiftTime,
                 ?y owl:equalTo global:finishingTime,
                 global:prod6 global:has ?z: x/y)
Result: z = global:288

global:288 owl:equalTo sales:288
```

Listing 8: EKB Impact Analysis Example (Moser, Biffl et al. 2010).

7.1.4 End-to-End Analysis

In distributed engineering in heterogeneous environments, typically a set of different models is used along the engineering chain (see also Figure 22). In order to ensure validity and consistency of the overall engineering process, it is important to ensure that required data fields can be enforced during the whole lifecycle of the engineering chain (Moser, Winkler et al. 2010). In the following subsections, we present a database approach for end-to-end analyses of semantically homogeneous data, as well as an EKB framework approach for end-to-end analyses of semantically heterogeneous data.

7.1.4.1 A database approach for end-to-end analyses of semantically homogeneous data

Figure 49 shows the scenario used for the end-to-end analysis example using a homogeneous data set. There are three different engineering roles (electrical engineer, configurator, software engineer) from different engineering disciplines that use different engineering tools.

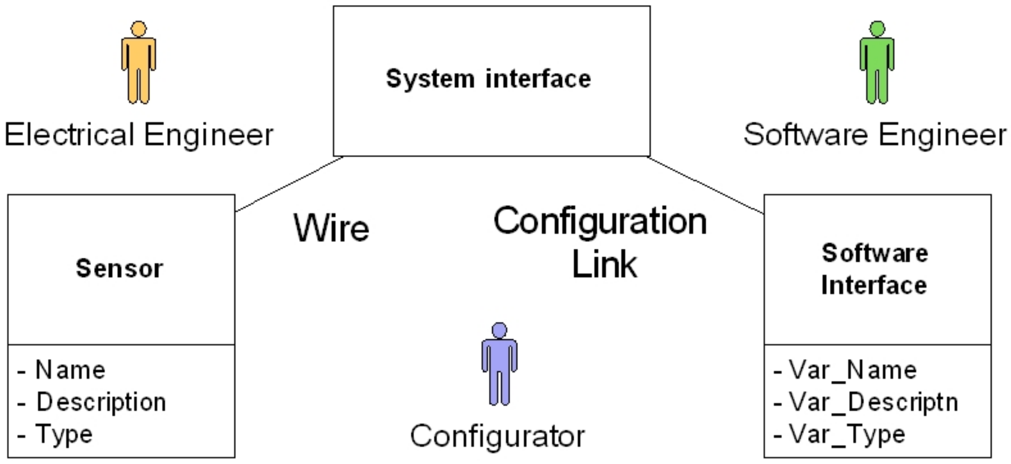
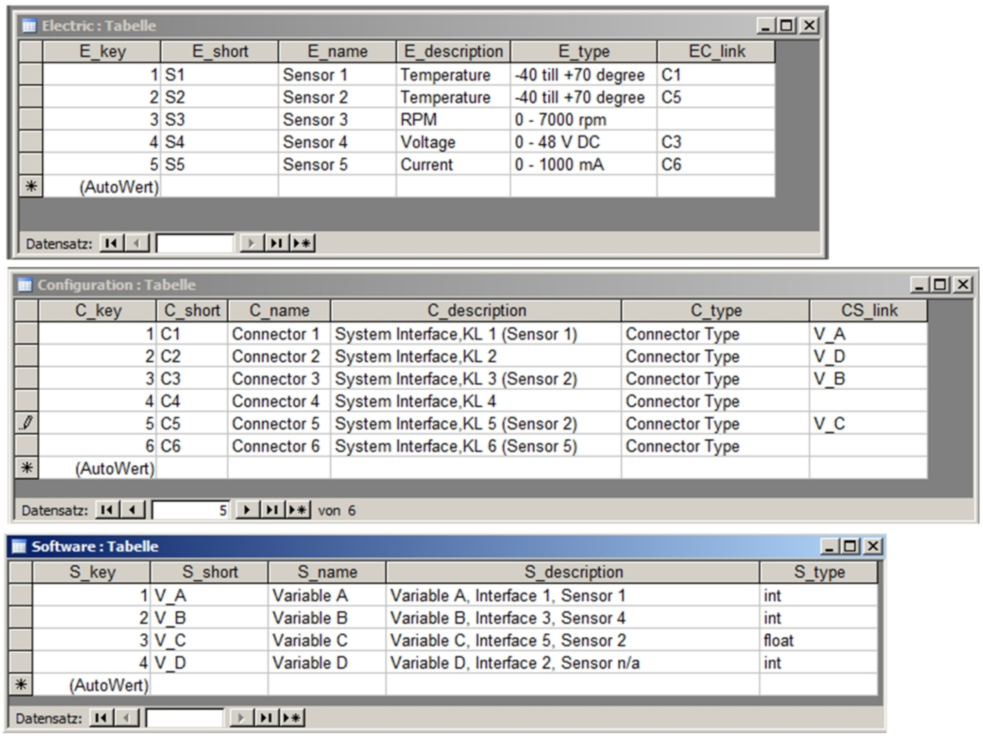


Figure 49: End-To-End Analysis of homogeneous data (Moser, Winkler et al. 2010).

In the following, we show an exemplary query using the data provided in Figure 50.



Electric : Tabelle

E_key	E_short	E_name	E_description	E_type	EC_link
1	S1	Sensor 1	Temperature	-40 till +70 degree	C1
2	S2	Sensor 2	Temperature	-40 till +70 degree	C5
3	S3	Sensor 3	RPM	0 - 7000 rpm	
4	S4	Sensor 4	Voltage	0 - 48 V DC	C3
5	S5	Sensor 5	Current	0 - 1000 mA	C6
(AutoWert)					

Datensatz:

Configuration : Tabelle

C_key	C_short	C_name	C_description	C_type	CS_link
1	C1	Connector 1	System Interface,KL 1 (Sensor 1)	Connector Type	V_A
2	C2	Connector 2	System Interface,KL 2	Connector Type	V_D
3	C3	Connector 3	System Interface,KL 3 (Sensor 2)	Connector Type	V_B
4	C4	Connector 4	System Interface,KL 4	Connector Type	
5	C5	Connector 5	System Interface,KL 5 (Sensor 2)	Connector Type	V_C
6	C6	Connector 6	System Interface,KL 6 (Sensor 5)	Connector Type	
(AutoWert)					

Datensatz: 5 von 6

Software : Tabelle

S_key	S_short	S_name	S_description	S_type
1	V_A	Variable A	Variable A, Interface 1, Sensor 1	int
2	V_B	Variable B	Variable B, Interface 3, Sensor 4	int
3	V_C	Variable C	Variable C, Interface 5, Sensor 2	float
4	V_D	Variable D	Variable D, Interface 2, Sensor n/a	int
(AutoWert)				

Datensatz:

Figure 50: End-to-End analysis - Homogeneous data (Moser, Winkler et al. 2010).

In the query, we want to identify all sensors, connectors and variables that are used end-to-end. Listing 9 shows the query in SQL syntax, as well as the results of the query using relational notation.

```
SELECT Electric.E_short, Electric.E_name, Configuration.C_short,
       Configuration.C_name, Software.S_short, Software.S_name

FROM
      (Electric INNER JOIN Configuration ON
            Electric.EC_link = Configuration.C_short)
      INNER JOIN Software ON Configuration.CS_link = Software.S_short;


(S1, Sensor 1, C1, Connector 1, V_A, Variable A)
(S4, Sensor 4, C3, Connector 3, V_B, Variable B)
(S2, Sensor 2, C5, Connector 5, V_C, Variable C)
```

Listing 9: Homogeneous End-to-End Query 1 (Moser, Winkler et al. 2010).

7.1.4.2 *EKB framework approach for end-to-end analyses of semantically heterogeneous data*

If we now assume that the data originating from the three different engineering disciplines is not homogeneous, but rather is only available in heterogeneous form (as shown in Figure 51), the simple SQL-based approach presented in section 7.1.4.1 is not working any more.

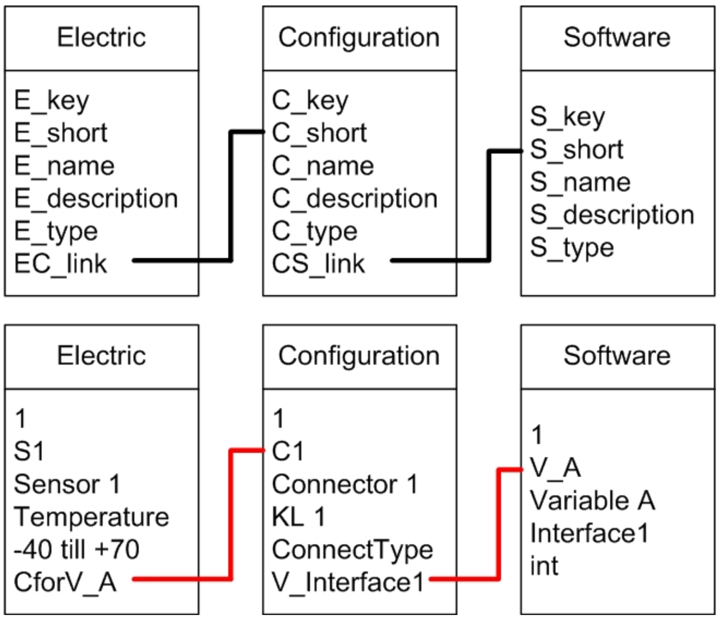


Figure 51: End-to-End analysis – Heterogeneous data (Moser, Winkler et al. 2010).

As shown in the top of Figure 51, the links between the electric and the configuration entity or the configuration and software entity requires the *EC_link* and the *C_short* attributes or the *CS_link* and *S_short* attributes to be the same for the queries to work. If however these attributes are not equal as shown in the bottom of Figure 51 (*CforV_A* and *C1*, respective *V_Interface1* and *V_A*), queries like the ones presented in section 7.1.4.1 do not work.

A solution approach would be to provide mapping tables in the database, which store the mappings between the attributes defining the links between entities. However, this results in both an increasing complexity of the SQL statements, as well as in the need to adapt these mapping tables each time attribute values are changed.

We now model this scenario using the EKB framework, as shown in Figure 52. There are three different layers, namely the domain ontology, the tool ontologies and the instance data layer. In the domain ontology, the general attributes (the so-called overlapping engineering concepts) and the relations (dotted lines in the figure) of the concepts electric, configuration and software are modeled. In the tool ontologies, all attributes of the tool-specific concepts electric, configuration, and software are modeled. In addition, the mappings (dashed lines in the figure) between the tool ontologies and the attributes of the generic concepts in the modeled in the domain ontology are described. Finally, the third layer shows an example for concrete instances (individuals) of the modeled concepts.

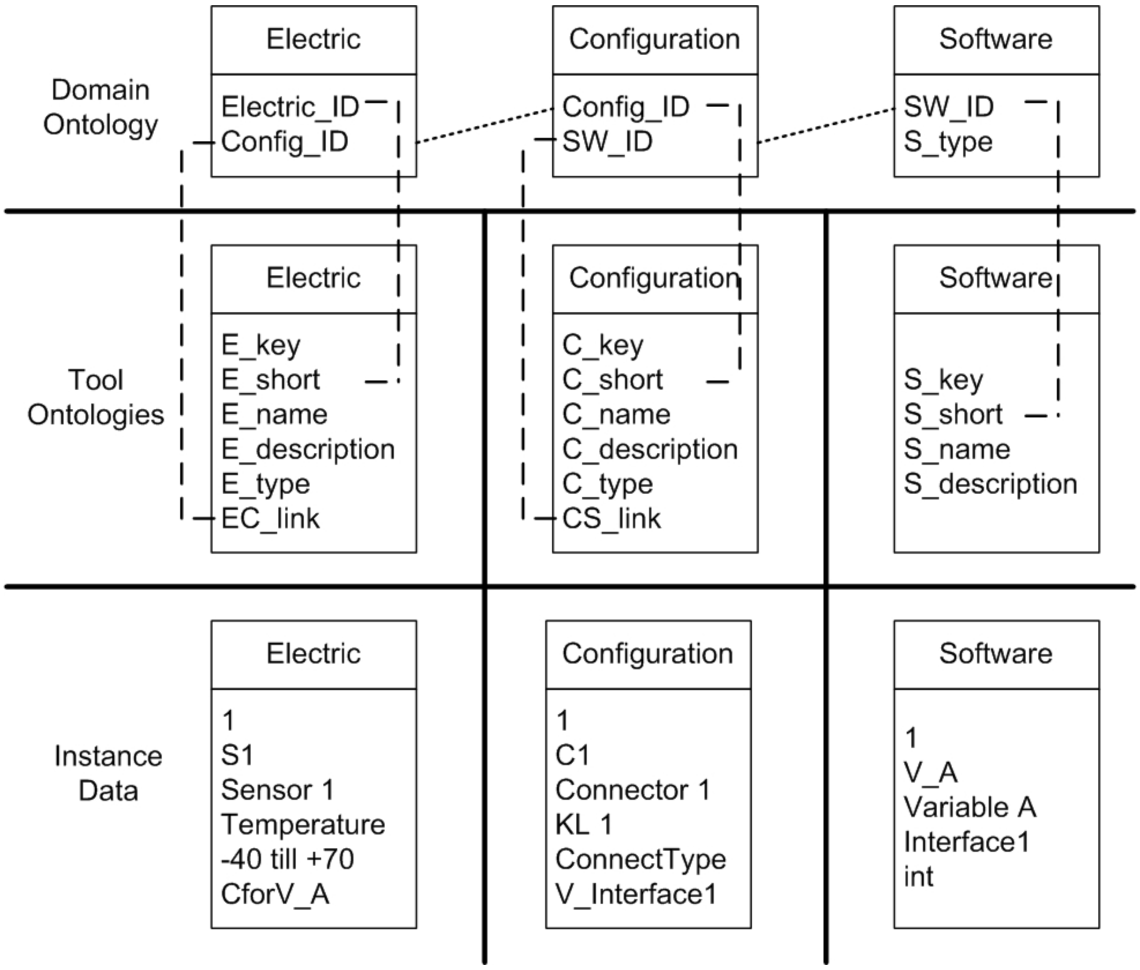


Figure 52: EKB Framework - End-to-End analysis (Moser, Winkler et al. 2010).

Using this model allows us to define queries on the domain ontology layer. For example, we again want to identify all sensors, connectors and variables that are used end-to-end. Listing 10 shows the SPARQL query and the result of the query.

```
SELECT ?Electric_ID, ?Config_ID, ?SW_ID

WHERE { el:E_short ekb:mapsTo ?Electric_ID.
        ?dom:Electric dom:Electric_ID ?Electric_ID.
        ?dom:Electric dom:Config_ID ?Config_ID.
        cfg:C_short ekb:mapsTO ?Config_ID.
        ?dom:Configuration dom:Config_ID ?Config_ID.
        ?dom:Configuration dom:SW_ID ?SW_ID.
        sw:S_short ekb:mapsTo ?SW_ID.
        ?dom:Software dom:SW_ID ?SW_ID.
      }

(S1, C1, V_A)
(S4, C3, V_B)
(S2, C5, V_C)
```

Listing 10: Heterogeneous End-to-End Query 1 (Moser, Winkler et al. 2010).

As shown in Figure 53, we now assume that there exist two different types of variables, namely normal variables (V_B and V_C) and safety-critical variables (V_A and V_D). Safety-critical variables need to be connected by a minimum of two sensors to allow continuous functionality in case of the failure of a sensor.

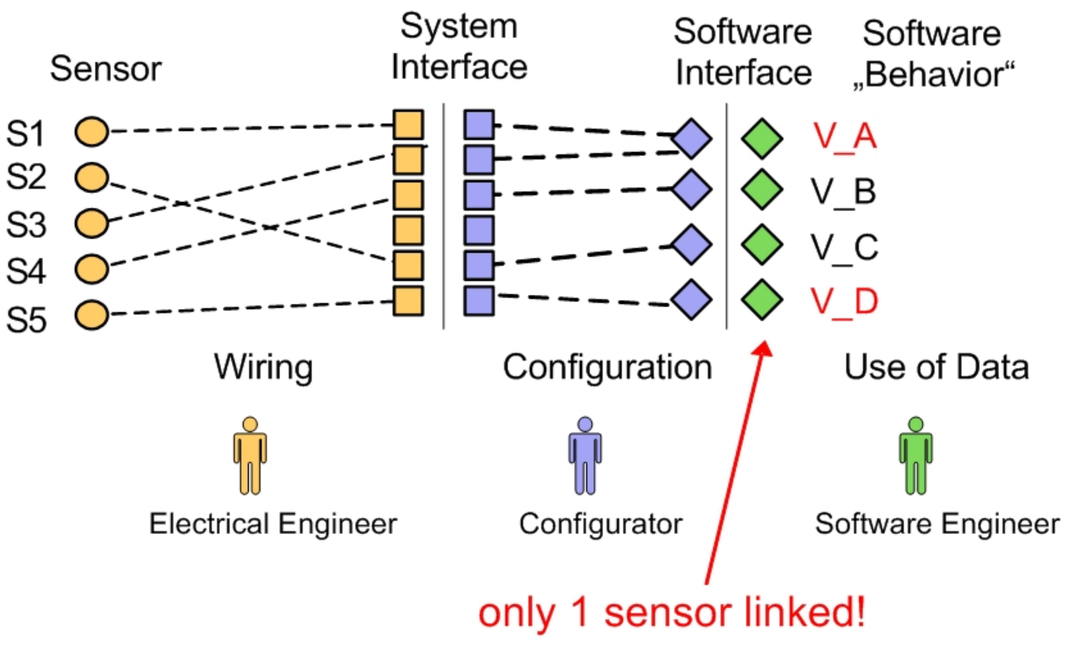


Figure 53: End-to-End analysis Safe Variables (Moser, Winkler et al. 2010).

Using this model allows us to define queries regarding the correct connectivity of all safety-critical variables. For example, we again want to identify all safety-critical variables that are not connected to 2 sensors. Listing 11 shows the SPARQL query and the result of the query.

```
SELECT ?SW_ID, count ?Electric_ID

WHERE { ?dom:Software dom:SW_ID ?SW_ID.
        ?dom:Software dom:S_type ‘safe’.
        ?dom:Configuration dom:SW_ID ?SW_ID.
        ?dom:Configuration dom:Config_ID ?Config_ID.
        cfg:C_short ekb:mapsTO ?Config_ID.
        ?dom:Electric dom:Config_ID ?Config_ID.
        ?dom:Electric dom:Electric_ID ?Electric_ID.
        el:E_short ekb:mapsTo ?Electric_ID.
      }

(V_A, 2)
(V_D, 1)
```

Listing 11: Heterogeneous End-to-End Query 2 (Moser, Winkler et al. 2010).

7.2 *Evaluation of the SWIS Application Scenario*

This section describes the evaluation of the SWIS application scenario in detail. First, the design of the evaluation and the evaluation criteria are summarized, then a step-by-step evaluation of the SWIS process is performed and described.

7.2.1 Evaluation Design for the SWIS Application Scenario

In order to assess the benefits and limitations of the EKB-based approach, we performed an evaluation by means of applying the proposed entire EKB-based approach. Therefore, we derived four parameters to compare the EKB-based approach with the traditional one. Table 1 summarizes the effort and duration needed for integration, the quality assurance efficiency, the complexity of the models, and finally the level of automation support both approaches provide (Moser, Mordinyi et al. 2009).
The evaluation is based on two scenarios within the ATM domain application scenario. The first scenario (Scenario 1) determines the results based on an integration project from the scratch. The second scenario (Scenario 2) assumes that an initial integration project has been accomplished providing a first integration solution, but due to changing business requirements some system adaptations have to be performed, like the need to update the domain model. Scenario 1within the ATM domain application scenario has the following characteristics: 5 systems (applications) with 30 integration points

(services) and 100 data structures (logical entities). In case of Scenario 2, 10 integration points of 3 different systems have been updated resulting in 2 new data structures and 10 updated ones. The overall integration effort for Scenario 1 using the traditional approach was 415 PDs[22] and for scenario 2 76 PDs. When using the EKB-based approach, the overall integration effort for scenario 1 was 435 PDs, compared to 32 PDs for scenario 2 (Moser, Mordinyi et al. 2009).

Table 1: UML- and EKB-based approaches (SWIS) (Moser, Mordinyi et al. 2009).

Evaluation parameters	Traditional approach	EKB-based approach
Integration Effort	System knowledge is described in human-readable documents by Subject Matter Experts No explicit domain knowledge used	System knowledge is externalized in a machine-readable ontology by Subject Matter Experts Domain knowledge is incrementally externalized in a machine-readable ontology by the Domain Expert
QA Efficiency	Low Manual checks of documents and models needed	High Automated ontology reasoning allows quickly locating inconsistent knowledge in the model
Model Complexity	High and distributed	High and centralized
Level of Automation Support	Low Exhaustive communication of Subject Matter Experts, Domain Expert, and Integration Expert needed to clarify integration partners Domain Expert coordinates the generation of transformation instructions with the affected Subject Matter Experts Manual checks of documents and system configuration needed	High Automated derivation of possible integration partners by means of ontology based reasoning Automated derivation of transformation instructions by means of ontology based reasoning Automated ontology reasoning allows quickly locating invalid system configurations

[22] PD: Person Day (Full Time Equivalent).

7.2.2 Evaluation Criteria for the SWIS Application Scenario

This section describes the criteria used for the evaluation, as well as the results obtained for each particular criterion (Moser, Mordinyi et al. 2009).

7.2.2.1 Integration effort

The results of the evaluation show that the overall integration effort is similar for both approaches in case of small number of systems to be integrated and slightly higher for the EKB-based approach in case of larger systems. The higher effort comes from the need to manage the domain model, since additional mappings between the integration system ontology and the domain model are needed. The effort to create the integration system ontology or the interface description is similar since in both approaches the conducted Subject Matter Experts has to cope with the same problem of finding the right information describing the system interfaces with its semantics. The EKB-based has the advantage that in case of adaptation the knowledge already gathered is explicitly given and can be reused in further discussions compared to the traditional approach where this knowledge exists implicitly only. In case of reconfiguration issues the EKB-based process has proven to be more efficient than the traditional approach since once the knowledge has been externalized, it can be reused with little extra effort. Furthermore, in case of the traditional approach each system expert has to be contacted for any kind of changes resulting in discussions (Moser, Mordinyi et al. 2009).
In case of the EKB-based approach the domain expert is needed in major changes only where the mapping of the integration system ontology to the domain ontology has to be altered as well. In case of minor changes, affecting the characteristics of the system only, the Subject Matter Experts are needed. Additionally, performing changes, like structure modifications, based on documents is more difficult and time consuming than compared with ontologies where you deal with classes. Changes can be performed much faster and can be done during the discussion concerning the integration project as well. The duration of the traditional approach tends to be higher due to error-prone mainly manual process steps resulting in additional efforts to discuss error sources and possible solutions. The proposed EKB-based approach reports errors or missing information immediately due to in-time consistency and completeness checks based on ontology reasoning. In case of describing systems, parallel processing is possible in both approaches. However, the following EKB-based processing steps are running mainly automated from the third processing step on, while the traditional approach is still human-driven resulting in time consuming and error-prone processing steps. Therefore, the duration depends strongly on of automation support (Moser, Mordinyi et al. 2009).

7.2.2.2 Quality Assurance efficiency

Since the traditional approach focuses on manual validity checks, it is therefore more time consuming and error-prone. This also results in the fact that missing information is often detected in a later integration step. The quality assurance efficiency is measured by the number of failures detected in each system description weighted by the time of detection. The later the failure detected the higher the weighting rate. The EKB-based approach uses ontology-based reasoning. This allows performing consistency and completeness checks in-time automatically, resulting in a lower failure rate and in-time notification of the Subject Matter Expert about missing/incorrect information. Additionally, since the EKB-based approach is mainly automated, it allows returning to any processing state in order to e.g., reproduce errors or revise decisions taken (Moser, Mordinyi et al. 2009).

7.2.2.3 Model complexity

The model used in the traditional approach is smaller and therefore less complex compared to the model used in the EKB-based approach, since a considerable part of the integration knowledge is not described explicitly. In the EKB-based approach, the number of relations, i.e., the number of mappings from the integration system ontology to the domain ontology introduces a higher structural complexity. The benefit of a more complex ontology model lies in the way how later integration steps can be supported by a higher level of automation. From the Subject Matter Expert's point of view the complexity remains the same in both approaches. For the domain expert the EKB-based approach reduces his efforts to the task of managing the structural complexities of the ontologies and to support the Subject Matter Experts in mapping. In the traditional way the domain experts need to cope with the major part of the complexity, since he is responsible for ensuring the consistency and completeness as well as managing the integration of the Subject Matter Experts' legacy system descriptions (Moser, Mordinyi et al. 2009).

7.2.2.4 Level of automation support

The EKB-based approach supports the user while entering the data with consistency and completeness checks. Additionally, it influences the integration process in later steps by automatically deriving integration partner candidates and automatically generating transformation instructions for message exchange between the integrated systems (Moser, Mordinyi et al. 2009).

7.2.3 Step-by-Step Evaluation of the SWIS Process

Within a research project with two industry partners, the approach has been evaluated by means of several different scenarios from the ATM domain. We determine the effort for both process step variants and compare the overall outcome. The following subsections summarize the effort needed to perform the particular process steps. The effort estimations are based on the expertises of the integration experts from both companies (Moser, Mordinyi et al. 2009).

7.2.3.1 Step 1: Legacy System Description

The externalization of legacy system knowledge using ontologies needs slightly more effort than the traditional approach using only human-readable artifacts like documents because the knowledge needs to be transformed from implicit expert or system knowledge into machine-readable ontology models (Moser, Mordinyi et al. 2009).

7.2.3.2 Step 2: Domain Knowledge Description

In the traditional integration process the domain knowledge is not made explicit but implicitly captured by domain experts and documents in a non-machine-readable way requiring no additional effort. Additionally, the integration network knowledge (i.e., the architecture and capabilities of the underlying network infrastructure) are described, which again represents an additional effort compared to the implicit knowledge of the traditional integration process. Using the EKB-based approach the domain and integration network knowledge has to be incrementally externalized by the domain expert and the network administrator resulting in medium effort in the first instance. This effort is reduced due to reuse within similar integration scenarios or additional process iterations triggered by reconfiguration issues (Moser, Mordinyi et al. 2009).

7.2.3.3 Step 3: Model Quality Assurance

The traditional approach requires a lot of effort to check the consistency and completeness of the documents since it has to be done manually. The EKB-based approach uses automated ontology based reasoning techniques to assure consistent models leading to comparatively low model Quality Assurance effort (Moser, Mordinyi et al. 2009).

7.2.3.4 Step 4: Derivation and Selection of Integration Partners

This traditional integration process step demands exhaustive communication between the involved roles (Subject Matter Expert, Domain Expert, Integration Expert, Network Administrator) in order to derive possible integration partners and clarify considerable dependencies between legacy systems. This results in very high integration effort for the traditional integration process while the EKB-based approach provides automated derivation of possible integration partners by means of ontology-based reasoning. The step involves the Integration Expert only who is responsible for selecting the most suitable set of integration partners from the provided suggestions; the mapping of the selected integrations partners to the underlying integration network is done fully automated using the externalized integration network knowledge described in step 2 (Moser, Mordinyi et al. 2009).

7.2.3.5 Step 5: Generation of Transformation Instructions

In case of the traditional approach the effort for generating transformation instructions is higher than with the EKB-based approach because the derivation of those instructions has to be done manually, but still lower than in the previous step because the number of involved roles is lower. The EKB-based process step is performed automatically using ontology based reasoning for deriving transformation instructions based on the explicitly captured knowledge (Moser, Mordinyi et al. 2009).

7.2.3.6 Step 6: System Configuration Quality Assurance

Consistency and completeness checks in the traditional approach are time-consuming and error-prone, leading to a high level of manual human effort. On the other hand, the EKB-based approach again uses automated ontology-based reasoning techniques to quickly locate invalid system configurations, resulting in much lower effort for this process step (Moser, Mordinyi et al. 2009).

7.3 *Evaluation of the SAW Application Scenario*

This section describes the evaluation of the SAW application scenario in detail. First, UML- and Ontology-based approaches for process improvement in developing agile Multi-Agent Systems are investigated and then the Ontology Area concept use case is evaluated.

7.3.1 Investigating UML- and Ontology-Based Approaches for SAW

This section reports on an evaluation of the process variants described in section 6.2 based on a scenario taken from industry and discusses the results. We focus on the reconfiguration level process and investigate the enactment of the general process from the roles MAS developer and QA (a) using a traditional UML-based modeling approach and (b) using an EKB-based approach that promises to model important reconfiguration aspects in a continuous model. We conduct a feasibility study to investigate actual strengths and limitations of both approaches. In the study the roles were taken by 2 persons with good technical experience in the application area and UML/ontologies fitting to the process variant they were to enact (Moser, Kunz et al. 2008).
In this section we also identify typical defects that may occur in UML-based approach that should be addressed by the ontology-based approach.
The subsections describe the scenarios and the results of modeling new configurations in the context of the SAW system. The research application context involved 7 agent classes with the following number of agent instances 25 conveyors, 6 junctions; 6 diverters, 6 index stations; 4 RFID readers, 3 robots, and one storage rack. We analyzed both approaches regarding model complexity; modeling effort and quality risk.

Scenario 1 – adding a conveyor: In order to increase the throughput of the system a new connection conveyor should be added, leading from the storage area to the assembly area.
Scenario 2 – conveyor removal: One connection conveyor should be removed from the system.
Scenario 3 – change of conveyor direction: The direction of a connection conveyor should be changed.

Precondition for each of the reconfiguration scenarios is to conduct a dependency analysis to identify relevant dependencies between agent instances. In this context, dependencies exist with the direct neighborhood of an agent (e.g., in- and out-node of a conveyor), but also in design patterns where a group of agents cooperates (e.g., several conveyors can provide a loop to temporarily store pallets). Dependencies impact design changes and QA in both approaches similarly.

7.3.1.1 Scenario - Change of a Resource Agent - UML

This subsection describes the results of reconfiguration using the UML-based approach. In order to identify, if adding, removing, and change direction of the conveyor has an impact on the system behavior, we conduct a dependency analysis to find out, which behavior or which design patterns may be affected. This dependency analysis has to be conducted manually by analyzing existing design patterns.

Just to give an example, adding a conveyor to the system results in a small number of changes. The first step is to analyze the requirements for the new conveyor. If the conveyor does not need new functionality beyond the capabilities available from the agent tool box, we can take the agent as it is. In this case the only change is in the XML configuration file, which has to be extended by the new conveyor.
However, if the conveyor is added as an additional rerouting possibility because of higher work load in the system, it is necessary to model the rerouting design pattern with additional sequence charts. It is also important to model the failure handing behavior for the new conveyor with additional sequence diagrams. Also a new design pattern may be necessary. The changes made to the different models and the newly created models result in at least one QA cycle (manual review) for each of the models.

7.3.1.2 Scenario - Change of a Resource Agent - EKB

This subsection describes the results of reconfiguration using the EKB-based approach (Moser, Kunz et al. 2008).

Scenario 1 – *adding a conveyor:* The requirement to include a new conveyor in the system can be satisfied by creating a conveyor-supervising software agent, which is responsible for the proper behavior and for scheduling of the underlying hardware conveyor. Further we add a new instance of *Conveyor* concept in the ontology and fill in its properties. Defects introduced during adding activities, e.g., a non-connected conveyor, can be prevented or found early by running the ontology-based reasoner with logical queries based on the system design and coordination patterns that are applicable to a conveyor.

Scenario 2 – *conveyor removal*: In order to remove a conveyor from the system, the responsible software agent has to be destroyed, the instance of Conveyor concept, as well as the supervising agent instance, have to be removed from the ontology and all the machine agents supervising connected resources (e.g., intersections) have to delete the link to conveyor instances in the relevant ontology property values. Defects introduced during the activities, e.g., a non-connected intersection or separated line clusters (partially connected network) can be again prevented by running logical queries. Some of these situations can be resolved by changing the direction of a remaining conveyor.

Scenario 3 – *change of the conveyor direction*: Changing the conveyor direction requires few actions to be performed: The values of the conveyor properties inNode and outNode have to be exchanged and connected intersection agents have to change the intersection instance class from Diverter to Junction and vice-versa and shift the changed conveyor instance between its inConveyor and outConveyor properties. Changing the conveyor direction can however cause defect situations when, e.g., the

junction (i.e., new in-node) does not have the switching capability, so it cannot act as a diverter, or the direction change could cause the creation of a logistically unreachable subsystem. After changing the hardware conveyor direction, all the software agents controlling diverter intersections perform recalculation of the shortest paths to the line nodes by running corresponding general logical rules. These rules are able to discover any unreachable nodes and issue warnings.
Logical reasoning, on the other hand, can effectively prevent the logical changing of the junctions to diverters as their switching behavior is required and has to be denoted in the corresponding property value. Some of such defect situations might have a solution in changing the direction of another conveyor in the system.

7.3.1.3 Comparison of UML- & EKB-based Processes

Table 2 provides a side-by-side view on complexities, effort, and risks encountered in the study scenario. Process effort measurements depend on several factors in addition to process and tool support (e.g., task selection and skill factors, maturation of subjects), thus we present the effort ranges without consideration of statistical significance. We are well aware of the external validity limitations of such an initial study; however, such studies are important to motivate the investment into more comprehensive empirical studies in practitioner environments (Moser, Kunz et al. 2008).
Overall both approaches were well suited to deal with the limited complexity of the initial evaluation scenarios. The main difference in evaluation between the two approaches is in the modeling effort, which is considerably higher for the UML-based approach than for the ontology-based approach, which can be supported by automated reasoning. Manual dependency checks and QA need considerably more time and are also more error prone. A similarity in the approaches is in the interdependency of effort for conducting the actual model changes and the number of models to be changed. However, in the UML-based approach we have to change more artifacts than in the ontology-based approach (Moser, Kunz et al. 2008).
The quality risks of the UML-based approach depend mainly on the accuracy of the system designer. Especially the dependency analysis and QA tasks have to be conducted carefully and completely to avoid mistakes, which may lead to system failures, which are only detected during system operation. Thus, the quality risk is rated medium. The quality risk for the ontology-based approach is rated low, because QA can be well supported with automated reasoning, which avoids the human factor and thus has a very low probability of missing inconsistencies (Moser, Kunz et al. 2008).
Perceived strengths of the UML-based approach for reconfiguration were to help designers get an overview on a domain and identify components to work with: a) to provide well readable and understandable visualization for the designer and developer, especially for the software development process of reusable assets (e.g., an agent tool box); and b) to provide a well understandable high-level overview on agent classes,

their properties and of agents states, so behavior modeling can be focused to most relevant agents (Moser, Kunz et al. 2008).

Table 2: UML- and EKB-based approaches (SAW) (Moser, Kunz et al. 2008).

	UML-Based Approach	*EKB-based Approach*
Scenario 1 – adding a conveyor		
Model complexity	*Number of affected artifacts:* 6	*Number of affected artifacts:* 4
Modeling effort	*Model changes:* 50-60 min. *Dependency analysis:* 80-90 min. *Quality assurance:* 160-180 min.	*Model changes:* 35-45 minutes *Dependency analysis:* 50-60 min. *Quality assurance:* 25- 35 min.
Quality risk	*medium*	*low*
Scenario 2 –conveyor removal		
Model complexity	*Number of affected artifacts:* 7	*Number of affected artifacts:* 4
Modeling effort	*Model changes:* 60-70 min. *Dependency analysis:* 80-90 min. *Quality assurance:* 160-180 min.	*Model changes:* 25-35 min. *Dependency analysis:* 50-60 min. *Quality assurance:* 25-35 min.
Quality risk	*medium*	*low*
Scenario 3 –change of conveyor direction		
Model complexity	*Number of affected artifacts:* 10	*Number of affected artifacts:* 3
Modeling effort	*Model changes:* 90-100 min. *Dependency analysis:* 80-90 min. *Quality assurance:* 160-180 min.	*Model changes:* 25-35 min. *Dependency analysis:* 50-60 min. *Quality assurance:* 25-35 min.
Quality risk	*medium*	*low*

Perceived weaknesses of UML-based approach for reconfiguration were a) to needs model extensions to the reconfiguration information, which leads to a fractured view (that needs to be compensated with appropriate QA methods, e.g., inspection of consistency between several models that capture partly overlapping aspects); b) could not well capture the agent instance and configuration information; c) had a higher error proneness due to the need for manual model reviews (Moser, Kunz et al. 2008).

The EKB-based approach seems to be stronger to keep an overview on all the detailed information and dependencies on agents and instances over several iterations of reconfigurations in the context of a well-defined big picture (Moser, Kunz et al. 2008).
Perceived strengths of EKB-based approach for reconfiguration were a) to capture system configuration description (schema and data) in an integrated model; b) the integrated model allows tool support for QA checks (reasoning to conduct consistency and plausibility checks among the model aspects); c) to provide the source to explore the system structure using an ontology editor; and d) to provide a model that is directly usable both at design time and run time, enabling dynamic reconfiguration checks even at run-time (e.g., when handling of hardware failure scenarios) (Moser, Kunz et al. 2008).
Perceived weaknesses of EKB-based approach for reconfiguration were a) higher complexity of the ontology model; b) to provide human readable visualization of relationships among entities and a general overview on a domain with generic standard tools; c) the challenge to understand the contribution of an ontology element without good understanding of the domain, the design task, and the overall ontology design. In addition it has to be stated that compared to traditional SE qualifications like data modeling using UML, ontologies are a fairly new topic. Therefore, additional training or qualification of the involved software engineers is needed in order to allow the EKB-based approach to be used in an efficient way (Moser, Kunz et al. 2008).

7.3.2 SAW Ontology Area Concept Use Case Evaluation

We have implemented the Ontology Areas from the SAW ontology (see section 6.4.2) using Protégé 3.3.1. The SAW ontology consists of 24 classes and 3,000 instances from the simulation of production automation system. The evaluation compares the measurement of the whole ontology and the ontology areas for the two different use cases explained in section 6.4.2 (Biffl, Sunindyo et al. 2009).

7.3.2.1 UC-1: Translation between local stakeholder terminologies

We compare the complexity (size) of the minimal ontology with Ontology Areas to the complexity of the overall ontology in the study context. For the minimal ontology with Ontology Areas, the business and workshop stakeholders have local terminologies of 300 and 400 words, respectively. Both need 100 words to communicate with each other. There are 200 to 700 data elements representing common knowledge, and 200 words for mapping from both local terminologies to the common concepts. Totally 1,100 to 1,600 entities are needed for the Ontology Areas.
Meanwhile, the comprehensive ontology for 6 stakeholders consists of around 1,800 words for local terminologies and around 300 words to communicate with each other.

There are 1,600 words of common knowledge, and 600 to 1,800 words for mapping of all local terminologies to common concepts. In total, the comprehensive ontology consists of 4,200 to 5,400 words. In this case, Ontology Areas can reduce the ontology size to 20 to 30 % of the comprehensive ontology.
We can compare the efficiency of the minimal ontology with Ontology Areas to the efficiency of the whole ontology in conducting the translation task as follows. To produce 100 words of translation results from 200 words of mapping, the Ontology Areas needs 3 operators of query applying to those mapping.
The comprehensive ontology can produce more translations (300 words) with 3 operators of query as well. But the query should be applied to more mapping (600 to 1,800 words). With Ontology Areas we can reduce the size of mapping and make the operation faster (Biffl, Sunindyo et al. 2009).

7.3.2.2 *UC-2: Run-time measurement and analysis for design improvement*

For evaluation we determined the minimal complexity of Ontology Areas to support a specific data analysis task more efficiently, such as calculating process characteristics. Then we will compare the result with Ontology Areas to the (cognitive) complexity using a comprehensive complexity.
In the Ontology Areas of the specific task, for 1 volatile entity the run-time measurement consists of 30,000 data points per shift. In the overall ontology, there may be many more, e.g., 300,000, data points in one shift. By using the Ontology Areas, the user can focus only on entity that he needs, and thus reduce the complexity of data handling considerably.
The efficiency of the minimal ontology with Ontology Areas is compared to the efficiency of the overall ontology in the case to conduct the data analysis task as follows. In the Ontology Area, to obtain 5 data points analysis, it needed to run 3 operators of query over 30,000 data points at one shift. Hence 18,000 operations on data points are needed to obtain one of the measurements.
In the whole ontology, to obtain 20 data points analysis, it needed to run 3 operators of query over 300,000 data points at one shift. Hence 45,000 operations on data points are needed to obtain one of the measurements. Ontology Areas are notably more efficient than a single ontology (Biffl, Sunindyo et al. 2009).

7.4 *Discussion*

This section discusses the results of the evaluation of the EKB framework with regard to the research issues identified in section 3.1.

7.4.1 Functionality and Feasibility of the Proposed Approach

In this thesis, we applied the EKB framework to two application scenarios from two different application domains. The first research issue category deals with the general functionality and feasibility of the EKB architecture and processes. In the following, the specific research issues are answered.

RI-1.1. Feasibility of the proposed engineering environment integration approach. In this thesis, we introduced the novel Engineering Knowledge Base (EKB) framework for supporting engineering environment integration in multi-disciplinary engineering projects with a focus on providing links between data structures of engineering tools and systems to support the exchange of information between these tools and thus making software and systems engineering more efficient and flexible. Based on two real-world application scenarios, we applied and implemented the EKB framework in a prototypic way. Since standards are hard to apply in projects where experts from different organizations participate, who have invested into different kinds of local standards or approaches, these experts may use their well known local tools and data model, and additionally can access data from other tools in their local syntax. The EKB is located on top of a common repository and stores explicit engineering knowledge to support access and management of engineering models across tools and disciplines by providing a) data integration by exploiting mappings between local and common engineering concepts; b) transformations between local engineering concepts by following these mappings; and c) advanced applications using these foundations, e.g., end-to-end analyses. Only a selection of the most relevant data elements to achieve interaction between engineering tools and experts is stored in the EKB in order to avoid time-consuming and effort-intensive transformations of full engineering models. As a result experts from different organizations may use their well known local tools and data model, and additionally can access data from other tools in their local syntax. Since the engineering project participants by now already work together, they already use common knowledge for their project tasks. By using the EKB framework we make this existing knowledge explicit and machine-understandable, and therefore can automate on project level tasks that build on this explicit and machine-understandable knowledge. Furthermore, using the EKB framework allows a more generic definition and execution of model checks on an application domain level, and additionally enables more advanced checks regarding the plausibility and semantic correctness of data structures by exploiting the querying capabilities of ontologies.

RI-1.2. Foundations for tool support for automation of engineering process steps. The explicit and machine-understandable knowledge in the EKB framework helps to automate time-consuming engineering process steps like consistency and completeness checks. Furthermore, e.g., in the SWIS context, it allows automating later integration processing steps, like deriving integration partner candidates or automatically generating transformation instructions for message exchange between the integrated systems.

RI-1.3. Explicit modeling of stakeholder requirements. The use of an “engineering” ontology during the engineering approach provides a continuously available and evolving representation of the stakeholder requirements. Compared to traditional methods, the use of ontologies entails a number of advantages. As shown in the case study, this allows a more automated QA support. In addition the usage of the ontology area concept creates a personalized view on the data model for each role. The output of the simulation is automatically fed back in the ontology, resulting in a combination of a test case and its outcome. This automated feedback cycle has proven useful for performing more advanced statistical analysis on the data, leading to more significant assertions for the generation of new system versions.

RI-1.4. Tool support and QA for requirements transformation. The ontology and CBSE paradigms reinforce each others’ advantages: the ontology-supported CBSE approach seems to be more effective and efficient due to reasoning support for selecting and parameterization of the most suitable components out of a component tool box. This is automatically achieved with respect to a certain set of stakeholder requirements or dependencies between components and therefore without significant sources of defects like manual interaction. The engineering ontology supports both static and dynamic QA during the production engineering process. Test result measurements are stored in the engineering ontology and can be used for component analysis in next iteration of the production engineering process in order to create new system versions, and so completing the feedback cycle. Furthermore, the results from running test cases are documented in simulations in a way that allows efficient quality analysis and comparison of the results with the original assertions.

RI-1.5. Stakeholder-level quality measurement. Stakeholder-level quality is assured by means of ontology-based reasoning, allowing tracing customer-specific requirements continuously throughout the entire production engineering process. The ontology area approach and the role-specific views of selected data effectively and efficiently allow the involved roles to check the mapping and tracing of their value proposition and requirements at all times. Conflicts during dynamic QA can directly refer to the quality requirements of a certain configuration.

RI-1.6. Support for traceability across engineering domains. A major challenge in the engineering of distributed, flexible and complex IT systems is to extend the scope of QA from software artifacts to include software-relevant parts of artifacts in other engineering domains. Thus a key research issue is traceability, i.e., how to link the relevant elements of models for requirements, design, implementation, and testing across engineering disciplines as foundation for better integrated product assessment and improvement. As manual tracing has been found effort-consuming and error-prone, automated approaches for software engineering have been developed to capture dependencies based on syntactical identity (e.g., keyword-matching as in information retrieval approaches). A major limitation of these automated approaches is their inability to capture dependencies completely, because they cannot capture dependencies between semantically related artifacts without syntactic identity (semantic gap). Using the EKB framework allows to define and query traces on domain level, without the need stick with tool-specific terms and notations. The queries on domain level are automatically transformed into the tool-specific format and vice-versa.

RI-1.7. Support for end-to-end testing. In a typical engineering environment, system testing requires end-to-end testing, but since the single tools are distributed and using heterogeneous data models, an end-to-end test is hard to perform. Although testing tools are available to perform testing at multiple levels, most testing tools are incapable of building composite interdependent tests across technology platforms, languages and systems. Therefore the challenges in testing are driven by the distributed, heterogeneous nature of the used tools and a growing market of third-party services implying that there is not a single owner of the complete system. As shown in section 7.1.4.2, using the EKB framework enables engineers to perform end-to-end analyses also on heterogeneous data sets, since the queries are defined on domain level, and then automatically transformed to the tool-specific levels by exploiting the mappings between tool ontologies and domain ontology.

7.4.2 Comparison of the Proposed Approach to Other Solutions

In this thesis, we compared the EKB framework to two different kinds of alternative solutions, namely solutions that primarily rely on implicit knowledge, such as common repositories and data warehouses, as well as solutions that also rely on explicit knowledge, such as other ontology-based approaches. The second research issue category deals with this comparison. In the following, the specific research issues are answered.

RI-2.1. Comparison of the EKB framework and Common Repository-based approaches. The explicit and machine-understandable knowledge in the EKB framework helps to automate time-consuming automation systems engineering process steps like the exchange of data structures between heterogeneous engineering tools or consistency and completeness checks of data structures. Further, the EKB allows automating later integration processing steps, like automatically generating transformation instructions for data structure exchange between the integrated engineering tools. The major difference between the two evaluated approaches is the lack of the need for a common data schema when using the EKB framework, which can be seen as the main advantage of the EKB framework compared to a common repository. However, additional expert skills are necessary when using a fairly new technology in a quite traditional application context.

RI-2.2. More effective and efficient engineering using the EKB framework. The advantage of centrally storing the domain knowledge together with the mappings of individual tool knowledge lies in the possibility of an automated QA and automation of further engineering process steps. As described in (Biffl, Mordinyi et al. 2008), using ontologies for storing the knowledge in the EKB framework, enables more efficient and effective QA for component-based systems such as production automation systems. The major differences between the two evaluated approaches are the amount of needed human involvement to define and perform model checks, and the types of model checks which are supported by the approaches.

RI-2.3. EKB framework usage trade-off analysis. The evaluation showed that the effort needed for certain automation systems engineering process steps with the EKB framework is slightly higher in case of performing it from the scratch, but comparatively a lot smaller when adaptations due to changing business needs have to be performed since new converters only need to be generated semi-automatically for each new or changed engineering tool in comparison to the need of creating or adapting a vast number of converters for each affected combination of the new or changed engineering tool manually.

RI-2.4. Comparison of a traditional Date Warehouse-based data collection process to a semantically-enabled data collection process. To succeed in building a tool that is capable of mastering all described requirements, one has to carefully choose the right technology to handle the described required tasks. The approach using an ontology is obvious, since, in contrast to a database, an ontology is capable of a proper knowledge representation based on well-defined semantics. While a database only supports integrity checks on a structural level, conducting integrity checks on a semantic level is an intrinsic part of an ontology. Furthermore an ontology provides extensive reasoning capabilities, which means the possibility to use a priori hidden knowledge by deducting new facts out of known ones. By providing an explicit specification of the stored data's intended meaning, instead of the sole data itself, an ontology allows sophisticated querying. This is important to address the issue of being able to provide project managers with a proper tool for decision support.

RI-2.5: Integration of additional data sources. During the design process special care was taken to retain the possibility of integrating support for additional data sources. Despite the fact that implementing support for additional data sources (as mentioned earlier) is time-consuming, providing this possibility is important, since it allows the integration of the proposed tool into already existing environments. During the integration process, when merging data retrieved from the various data sources into the ontology to successfully carry out the combination, the routines have to be able to recognize relations between the various entries. A topic of discussion is the implementation of the possibility to integrate data from two or more different projects into the same ontology. This could enable for the analysis of possible synergy effects between different projects as well as combined statistics. Of course, the corresponding project leaders would have to evaluate, whether this step makes sense for their particular projects.

RI-2.6: Relation of the proposed engineering environment integration approach to other ontology-based Semantic Integration approaches. The ultimate alternative semantic integration solution is the complete transformation between data models of tools, i.e., the translation of engineering model parts of one tool for work in another tool. While the advantage of this solution is the seamless cooperation between project partners using well-known and established tools and notations, the feasibility of this approach is hard to verify and the effort required for establishing the needed transformations is considerably high.
In the Modale[23] project, Assmann et al (Assmann, Dörr et al. 2005) developed an ontology-based data integration approach in order to realize a seamless integration between cooperating partners in the field of digital production engineering. The major obstacle was the syntactic, structural and semantic heterogeneity of the internally used tools in the digital production engineering domain. As proof of concept they have also provided a web-service based prototypic implementation of their approach. However, many questions still remain open, requiring more research effort to be invested.

7.4.3 Specific Semantic Research Areas of the Proposed Approach

The process of applying the EKB framework uses ontologies as modeling methods. Therefore, the third category of research issues deals with two semantic web specific research areas, namely the usage of Ontology Alignment methods for providing the required mappings, as well as a conceptual approach for structuring big ontologies in order to increase usability and maintainability. In the following, the specific research issues are answered.

RI-3.1. Safety-Critical Ontology Alignment. Since the alignment should not be performed fully automated because of the safety-criticalness of the application domains, we propose a semi-automated approach that provides suggestions to the user which can be accepted or declined. The advantages of using such a semi-automated approach are a significant reduction of time and effort needed for the mapping, the reproducibility of

[23] http://www.modale.de

the given suggestions (and mappings), and the detection of consistency failures in the domain ontology.

RI-3.2. Risks of applying state-of-the-art ontology alignment approaches. While presenting a set of advantages, the adaptation of the state-of-the-art ontology alignment approaches for the EKB ontology alignment also bears some risks. The quality of the alignment suggestions heavily and primarily depends on the implemented ontology alignment method and may not always suggest the optimal alignment candidate or sometimes – even worse – result in no or false suggestions. Therefore it is necessary to use an appropriate algorithm or a combination of algorithms to gain the best possible result.

RI-3.3. Supporting engineering roles by lowering the cognitive complexity of the used ontologies. Ontologies are flexible open-world data models for knowledge representation, which store information in machine-understandable notation (Gruber 1995). Therefore, ontologies can help to bridge semantic gaps between partial data models by providing mappings between them via common domain concepts. Ontologies usually capture problem-domain-specific information which can be reused later. Due to their concurrent development ontologies need to be checked for inconsistencies to stay useful. However, ontologies in practice usually have to combine several view points and thus get large and complex, particularly, if the ontology contains volatile domain elements, such as run-time data.
In this thesis, we propose a data modelling approach that helps structure the ontologies using in the EKB framework with ontology building blocks, so-called *"Ontology Areas"*. An Ontology Area is a meaningful part of an ontology for a stakeholder, which helps ontology users managing a complex ontology. The combination of all needed Ontology Areas represents the overall ontology for supporting the original engineering process. An ontology area is a subset of ontology as a building block that can solve a certain task. The ontology can be broken into ontology areas based on several aspects, for example by the time, volatility, layer and roles. The Ontology Area Concept provides the following benefits:

- As Ontology Areas allow focusing on the content of interest for a stakeholder task, we could show that the resulting ontology is considerable smaller. A smaller ontology is often also more efficient to handle and allows tackling tasks that use a particularly large number of data elements.

- The combination of Ontology Areas, design-time, and run-time data elements allowed filtering relevant data elements for stakeholders, which would not be possible without the combination. Thus the Ontology Area approach helped lower the cognitive complexity for stakeholders by providing just the relevant subset of the comprehensive ontology.

- With the Ontology Area concept we can flexibly build task-oriented ontologies based on different criteria (like volatileness, layers, roles). It is even possible to compare different versions of the same Ontology Area (e.g., production automation system designed with different parameter settings) to compare the run-time reactions to from changing design parameters. However, this ability also raises the need for better version management for Ontology Areas to ensure the building of consistent ontologies for specific tasks.

Chapter 8

8 Conclusion and Perspectives

Software-intensive systems in business IT and industrial automation and software engineering projects bring together experts from several engineering domains and organizations, who work in a heterogeneous engineering environment with a wide range of models, processes, and tools that were originally not designed to cooperate seamlessly (Schäfer and Wehrheim 2007). A core question is how to integrate data models across tools and domain boundaries. Current semantic engineering environment integration is often ad hoc and fragile, making the evolution of tools and re-use of integration solutions across projects risky (Halevy 2005; Noy, Doan et al. 2005).
In order to reach the common goal of developing software products in the engineering team, it is important to share the necessary knowledge for common work processes between engineering domain experts (Schäfer and Wehrheim 2007). However, this knowledge is often only implicitly available and therefore inefficient to share, resulting in time-consuming repetitive tasks; often it is hard or even impossible to create and maintain common shared knowledge repositories. A method and platform for making expert knowledge explicit and efficiently shareable is needed in order to support quality and project managers in their data analyses based on engineering knowledge and concrete data in the engineering tool models, which is currently achieved using inefficient or fragile approaches.
This work proposes the Engineering Knowledge Base (EKB) framework for supporting engineering environment integration in multi-disciplinary engineering projects. Since standards are hard to apply in projects where experts from different organizations participate, who have invested into different kinds of local standards or approaches, these experts may use their well known local tools and data model, and additionally can access data from other tools in their local syntax. The EKB is located on top of a common repository and stores explicit engineering knowledge to support access and management of engineering models across tools and disciplines by providing a) data integration by exploiting mappings between local and common engineering concepts; b) transformations between local engineering concepts by following these mappings; and c) advanced applications using these foundations, e.g., end-to-end analyses. Only a selection of the most relevant data elements to achieve interaction between engineering tools and experts is stored in the EKB in order to avoid time-consuming and effort-intensive transformations of full engineering models. As a result experts from different organizations may use their well known local tools and data model, and additionally can access data from other tools in their local syntax.
By now, Semantic Integration research has focused on finding general approaches for schema integration which can be used in many contexts. However, these general

approaches do not take into account the specifics of a domain and therefore tend to be inefficient and often fail to solve specific problems that are hard to solve in general. In this work, we build on domain-specific knowledge of engineering processes, models and analyses to enable designing semantic integration methods and tools. Since the engineering project participants by now already work together, they already use common knowledge for their project tasks. By using the EKB framework we make this existing knowledge explicit and machine-understandable, and therefore can automate on project level tasks that build on this explicit and machine-understandable knowledge. Key contributions of this work are the industrial application and proof-of-concept of the proposed semantic integration approach, as well as design guidelines for Semantic Integration in the engineering domain.
The research results were evaluated in two industrial application domains, distributed business systems and services and software-intensive production automation systems, regarding effort, feasibility, performance, scalability, robustness and usability. The evaluation is based on prototypes for a set of specific use cases of the two industrial application domains, as well as on empirical studies of beneficiary roles as proof-of-concept. Major results of this work are the feasibility of the EKB framework, i.e., the process, method and tool support is usable and useful across engineering domains, as well as better accuracy, effectiveness and efficiency. In addition, defects are found earlier in the engineering process, resulting in risks like errors or inconsistent entries in data models being mitigated earlier and more efficiently. Initial evaluation results indicate an effort reduction of more than 20% for re-use in new engineering projects and finding defects earlier in the engineering process. In addition, the engineers found the method useable and useful; furthermore, new kinds of analysis could be performed easily.

8.1 *Highlights and Lessons Learned*

In this section, we summarize the main results of the work done for researchers and practitioners.

- ***Systematic Literature Review on Semantic Integration:*** In section 2.5, the results of a systematic literature review on the Semantic Integration are presented. The section introduces the research field of Semantic Integration, classifies and explains the different available approaches, and in addition gives exemplary application scenarios for the usage of ontologies for Semantic Integration. Based on the results of this systematic literature review, the Engineering Knowledge Base (EKB) framework was positioned regarding alternate semantic integration approaches.

- ***Challenges and Solution Approach for Semantic Integration in the Engineering Domain:*** In chapter 3, we identify the challenges of efficient data integration and transformation between heterogeneous engineering experts' data models, and present a generic solution approach, the so-called Engineering Knowledge Base (EKB) framework, in section 4.3.

- ***Semantic Modeling of Requirements and Capabilities for Configuration Derivation in the ATM domain:*** In chapter 5, we apply the EKB framework to the Air Traffic Management (ATM) domain. We use the EKB framework to semantically model business service requirements and IT infrastructure capabilities, and then match these requirements and capabilities in order to identify suitable communication partners and generate integration system configurations.

- ***Ontology Area Concept:*** In section 6.4.2, we introduce the Ontology Area concept. Ontology Areas are a data modeling approach based on ontology building blocks, which allow solving tasks with smaller parts of the overall ontology. Ontology Areas improve the efficiency of data collection task for decision making by lowering the cognitive complexity for designers and users of the ontology.

- ***Supporting Runtime Decisions using Design Time Information:*** In section 6.6, we present an approach to improve support for run-time decision using the EKB framework. The EKB framework provides a better integrated view on relevant engineering knowledge in typical design-time and run-time models, which were originally not designed for machine-understandable integration. This allows uniform and efficient access to related data in design-time, deployment, and run-time models, as well as the derivation of assertions that should hold at runtime.

- ***End-to-End Analyses across Domain Boundaries:*** In distributed engineering in heterogeneous environments, typically a set of different models is used along the engineering chain. In order to ensure validity and consistency of the overall engineering process, it is important to ensure that required data fields can be enforced during the whole lifecycle of the engineering chain. In section 7.1.4, we introduce an EKB framework-based approach for end-to-end analyses of semantically heterogeneous engineering data.

8.2 *Research Challenges and Solution Approach*

As shown in Figure 54, each engineering role (e.g., electrical engineer or software engineer) has a tailored tool set that works on data relevant to the engineer's tasks. In a typical process step in the engineering process an engineer exports data from his tool to a transfer document and integrates this document in a common repository accessible by a set of partner engineering tools. The major challenges here are identification and description of tool data, the data integration itself, and the re-use of at least parts of integration solutions for other projects. In order to support data exchange between these sets of partner engineering tools, transformations between the different notations and format of the particular partner engineering tools is needed. The major challenges of the transformation process are the adaptation of transformation instructions to new or changed tool data structures and their runtime performance. Using these foundations, i.e., export, integration and transformation, additional methods like Quality Assurance (QA) support or other advanced methods like model consistency checking or end-to-end analyses are allowed. Currently there is high effort needed to perform typical engineering project tasks, which may also lead to risks of defect.

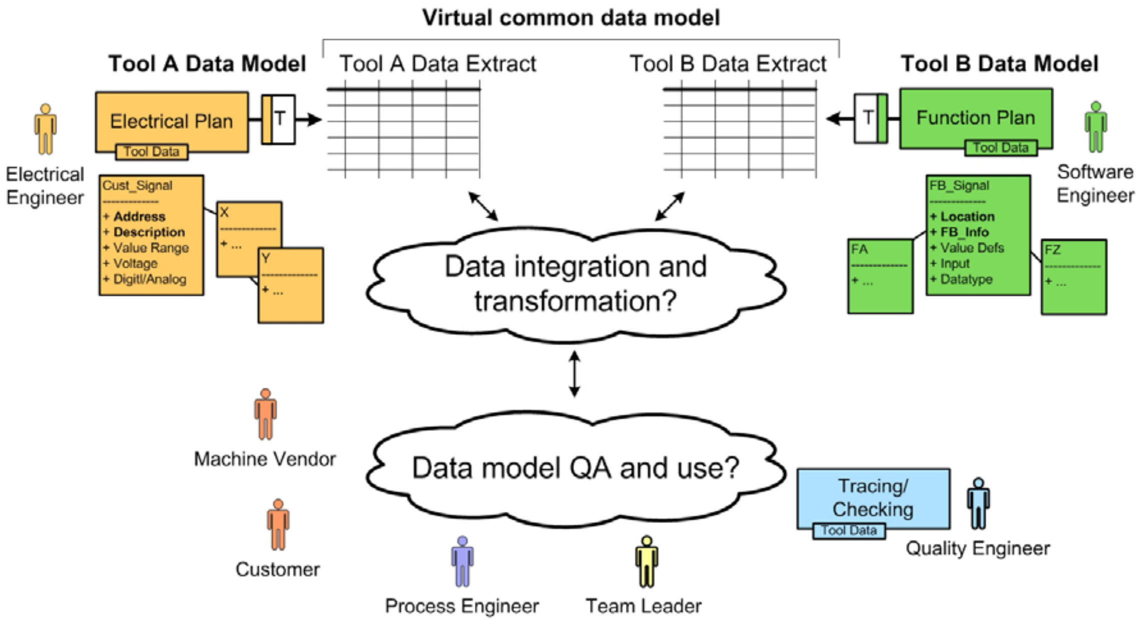


Figure 54: Overview of the research challenges.

For these tasks, we propose to use the novel Engineering Knowledge Base (EKB) framework (see Figure 55) with a focus on providing links between data structures of engineering tools and systems to support the exchange of information between these engineering tools and thus making systems engineering more efficient and flexible. The EKB framework is an ontology-based data modeling approach which supports explicit modeling of existing knowledge in machine-understandable syntax. Therefore, we can automate on project level processes that build on this machine understandable knowledge. The EKB framework stores the engineering knowledge in ontologies and provides semantic mapping services to access design-time and run-time concepts and data. The EKB framework aims at making tasks, which depend on linking information across expert domain boundaries, more efficient.
In comparison to a simple data storage such as a common repository, a knowledge base stores information (i.e., the original data plus meta-data describing links between data

elements or annotations of data elements using machine-understandable syntax which can be used to automate time-consuming tasks and support human experts in doing their work. The EKB stores explicit engineering knowledge to support access to and management of engineering models across tools and disciplines by providing a) data integration by exploiting mappings between local and common engineering concepts; b) transformations between local engineering concepts; and c) advanced applications using these foundations, e.g., end-to-end analyses.

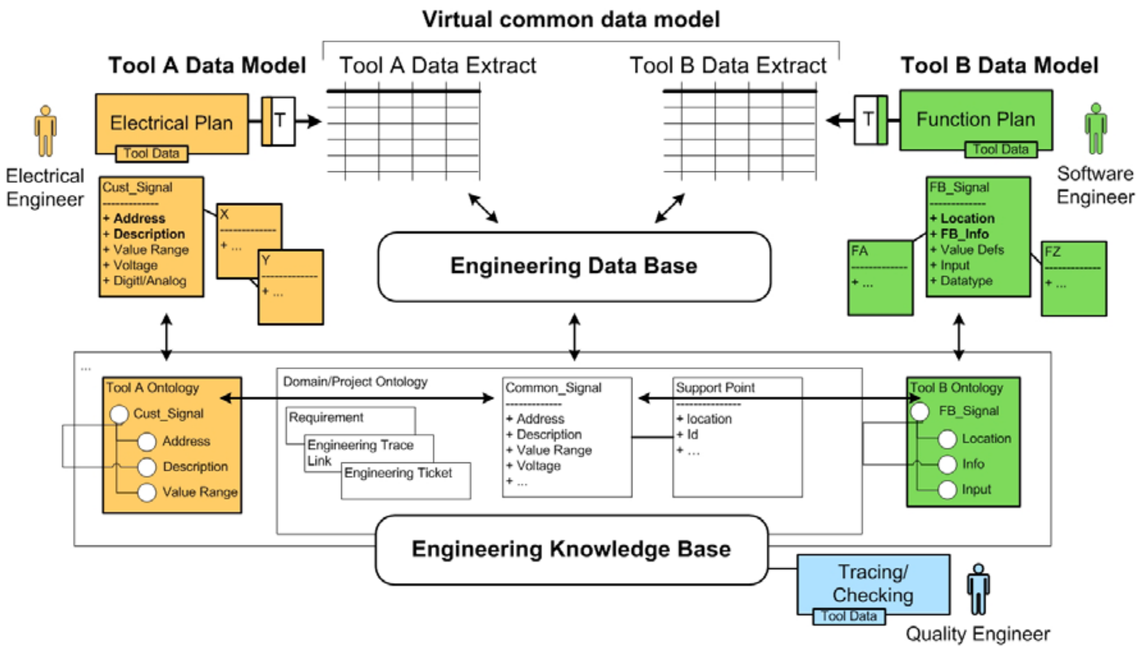


Figure 55: Overview of the Solution Approach (Moser, Biffl et al. 2010).

8.3 *Future Research*

This section identifies future research areas opened up by the research on the EKB framework. The following subsections describe research topics such as ontology-supported generation of test cases, semantic integration of heterogeneous data sources for monitoring frequent-release software projects, and Ontology Alignment in a safety-critical domain.

8.3.1 Ontology-Supported Generation of Test Cases

Production automation systems are often complex systems as the behavior of the overall system cannot easily be predicted from the behavior of the subsystems. Thus simulation is used to study the behavior of complex production automation systems. In addition to the accuracy of the simulation the system performance is an important issue,

particularly if many parameter variants for system behavior are to be tested. Parameters for assembly lines are, for instance, scheduling strategy, failure handling strategy and the number of products.
Software testing investigates the quality of the product or service under test. In order to fully test all requirements of an application, there must be at least one test case for each requirement. The goal is to generate test cases in a fully automated and systematic way to find suitable scenarios for most of the requirements in a short period of time.
Future research will investigate two different approaches for providing test cases. One approach is the use of a static specific generator script where it is difficult to add new parameters. In addition, the users need programming skills for both setting and modifying parameters. The second approach uses a dynamic generic generator script together with an ontology as data model. Test cases are generated with respect to the chosen parameters by the user. Important advantages of using an ontology are efficient tool support for modifying ontologies and the fact that the generator script is not affected by the modification. Thus users do not need programming skills to add new parameters.
The focus of the practical part of this future research lies on enhancing the existing ontology of the simulation tool to include the test case generator domain. A dynamic generic generator script will be worked out to generate test cases from the ontology and export them. The implemented generator script and the ontology are coupled loosely. Therefore changes to the ontology do not necessarily lead to changes of the dynamic generic script. This fact enables a flexible and high-level test description. Furthermore, the results of executed simulations can be integrated into the ontology as feedback. As a result, an optimal set of parameters can be achieved. The evaluation of this future research will investigate whether reduced costs for test description, increased flexibility, and definable test coverage can be achieved with the ontology-based approach.

8.3.2 Semantic Integration for Monitoring Software Projects

Open source software (OSS) projects rely on experts from various backgrounds and have gained an impressive level of stability and performance, in some areas even outperforming comparable commercial tools (e.g., tool sets of the Apache Software Foundation[24]). OSS teams routinely develop complex software products in distributed settings with rather lightweight processes and project documentation. However, there are issues that slow down the proliferation of OSS for complex projects such as insufficient awareness of changes in a project (e.g., due to time zone differences) or misunderstandings (e.g., due to cultural differences or incompatible development style). Therefore project managers and task leaders need effective and efficient data collection services as foundation for the timely overview on progress, cost, and quality of the

[24] Apache Software Foundation – http://www.apache.org

project activities, similar to a data warehouse in long-running business processes, for exploring and analyzing large quantities of data in order to discover meaningful patterns (Berry and Linoff 1997).
Unfortunately, the broad range of means for communication (e.g., e-mail, personal instant messaging, communication forums, and blogs) and coordination (e.g., version control systems, requirement management tools, and issue trackers) used in distributed development project settings has made managing such projects an increasingly difficult task. A good project manager needs to get an overview on all relevant tools used in his project, as well as of the relevant data on the status of the project work contained within these tools. The ability to correlate and assess project data in distributed project tools is vital both for estimating the current project status and also for predicting future project risks and opportunities (Thai, Pekilis et al. 2001; Mockus, Fielding et al. 2002).
A major challenge of data collection is how to extract the relevant project management knowledge effectively and efficiently from the wide range of available software project data sources, such as artifact versions, bug reports, and discussion forums. Project participants communicate through a wide range of tools that contain knowledge on the status of tasks, artifacts, and processes. Unfortunately, these data sources exhibit semantically heterogeneous data formats and terminologies, which take significant effort to reconcile with a data warehouse approach. Further, to keep the overview, monitoring and evaluation processes have to be repeated regularly because of the more frequent system releases which are performed in line with user expectations for greater responsiveness and shorter cycle times (Brown and Booch 2002). Thus a manual approach seems infeasible due to the immense amount of data. While the data warehouse approach has been optimized (Nguyen, Tjoa et al. 2004) significantly for data from a stable type of tools, data from heterogeneous sources still poses a major challenge.
Future research will focus on applying the EKB framework in order to support semantic integration of data coming from a variety of data sources and tool support to enable efficient data collection, especially in projects with frequent iterations like OSS. Major challenges for the application of the EKB framework are the management of incomplete and/or inconsistent data. The retrieved data should be integrated into a suitable well-defined format to ease processing and analyzing, e.g., within a data warehouse. Based on data from real-world use cases in OSS projects we will compare the effort using the EKB framework and a traditional data warehouse approach in test scenarios for integrating data from OSS projects.

8.3.3 Ontology Alignment in a Safety-Critical Domain

Ontology Alignment is an automated process that tries to identify similarities between two or more heterogeneous ontologies based on a set of metrics, like string similarity or structural similarity measurements. This works well for big taxonomies that can tolerate a moderate failure rate regarding wrong mappings. However, using Ontology Alignment approaches in safety-critical domains needs further investigations to minimize the risk of misalignment (Moser, Mordinyi et al. 2009; Moser, Schimper et al. 2009). For the EKB correct mappings, e.g., between a large number of engineering model elements in different system, are a crucial foundation for applications that use the EKB services. Thus we will investigate the trade-off between the better precision of the mainly manual Ontology Alignment tasks of the EKB approach and the better efficiency of more automated Ontology Alignment approaches while satisfying the required safety levels in automation systems engineering.

Appendix

References

Aaen, I. (2003). "Software process improvement: Blueprints versus recipes." IEEE Software **20**(5): 86-93.

Abran, A., J. W. Moore, P. Bourque, R. Dupuis and L. L. Tripp (2004). Guide to the software engineering body of knowledge: 2004 version, IEEE Computer Society, Los Alamitos, CA; Tokyo.

Akkiraju, R., J. Farrell, J. Miller, M. Nagarajan, M. T. Schmidt, A. Sheth and K. Verma (2005). "Web Service Semantics-WSDL-S." W3C Member Submission **7**.

American National Standard (2000). Enterprise-Control System Integration. Part 1: Models and Terminology. North Carolina, USA, ISA (the Instrumentation, Systems, and Automation Society). **ANSI/ISA-95.00.01-2000:** 142.

Andrieux, A., K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, J. Pruyne, J. Rofrano, S. Tuecke and M. Xu (2004). Web Services Agreement Specification (WS-Agreement).

Assmann, D., J. Dörr, M. Eisenbarth, M. Hefke, M. Soto, P. Szulman and A. Trifu (2005). Using Ontology-Based Reference Models in Digital Production Engineering Integration. 16th IFAC World Congress. Prague, Czech Republic.

Aßmann, U., S. Zschaler and G. Wagner (2006). Ontologies, meta-models, and the model-driven paradigm. Ontologies for software engineering and software technology. C. Calero, F. Ruiz and M. Piattini, Springer.

Baclawski, K., M. K. Kokar, P. A. Kogut, L. Hart, J. Smith, J. Letkowski and P. Emery (2002). "Extending the Unified Modeling Language for ontology development." Software and Systems Modeling **1**(2): 142-156.

Bajaj, S., D. Box, D. Chappell, F. Curbera, G. Daniels, P. Hallam-Baker, M. Hondo, C. Kaler, D. Langworthy and A. Malhotra. (2006). "Web Services Policy Framework (WS-Policy)." Version Retrieved 2, 1.

Balasubramanian, K., A. Gokhale, G. Karsai, J. Sztipanovits and S. Neema (2006). "Developing Applications Using Model-Driven Design Environments." COMPUTER: 33-40.

Bechhofer, S., F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider and L. A. Stein (2004). "OWL Web Ontology Language Reference." W3C Recommendation **10**.

Bellwood, T., L. Clement, D. Ehnebuske, A. Hately, M. Hondo, Y. L. Husband, K. Januszewski, S. Lee, B. McKee and J. Munter (2002). "UDDI Version 3.0." Published specification, Oasis.

Bergamaschi, S., S. Castano and M. Vincini (1999). "Semantic integration of semistructured and structured data sources." SIGMOD Rec. **28**(1): 54-59.

Bernstein, P. A. and U. Dayal (1994). An Overview of Repository Technology. 20th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc.**:** 705-713.

Berry, M. J. A. and G. Linoff (1997). Data Mining Techniques For Marketing, Sales, and Customer Support, John Wiley & Sons.

Biffl, S. (2009). "Software Engineering Integration for Flexible Automation Systems." Presentation for the proposed Christian Doppler Laboratory.

Biffl, S., A. Aurum, B. Boehm, H. Erdogmus and P. Grunbacher (2006). Value-based software engineering, Springer-Verlag New York Inc.

Biffl, S., C. Ferstl, C. Höllwieser and T. Moser (2009). Evaluation of Case Tool Methods and Processes - An Analysis of Eight Open-source CASE Tools. 11th International Conference on Enterprise Information Systems (ICEIS 2009). Milan, Italy. **3:** 41-48.

Biffl, S., R. Mordinyi and T. Moser (2008). "Continuous Software Life Cycle Modeling with "Engineering" Ontologies." Technical Report (available online at: http://www.ifs.tuwien.ac.at/files/Continuous Software Life Cycle Modeling with Engineering Ontologies - Technical Report.pdf).

Biffl, S., R. Mordinyi, T. Moser and D. Wahyudin (2008). Ontology-supported quality assurance for component-based systems configuration. 6th International Workshop on Software Quality (WoSQ '08). Leipzig, Germany**:** 59-64.

Biffl, S., R. Mordinyi and A. Schatten (2007). A Model-Driven Architecture Approach Using Explicit Stakeholder Quality Requirement Models for Building Dependable Information Systems. Fifth International Workshop on Software Quality (WoSQ'07)**:** 1-6.

Biffl, S., A. Schatten and A. Zoitl (2009). Integration of Heterogeneous Engineering Environments for the Automation Systems Lifecycle. IEEE Industrial Informatics (IndIn) Conf., 2009.

Biffl, S., A. Schatten and A. Zoitl (2009). Integration of Heterogeneous Engineering Environments for the Automation Systems Lifecycle. IEEE Industrial Informatics (IndIn) Conf., 2009**:** 576-581.

Biffl, S., W. D. Sunindyo and T. Moser (2009). Bridging Semantic Gaps Between Stakeholders in the Production Automation Domain with Ontology Areas. 21st International Conference on Software Engineering and Knowledge Engineering (SEKE 2009). Boston, USA**:** 233-239.

Booch, G. (2006). "The accidental architecture." IEEE Software **23**(3): 9-11.

Booch, G., J. Rumbaugh and I. Jacobson (2005). Unified Modeling Language User Guide, The (2nd Edition) (Addison-Wesley Object Technology Series), Addison-Wesley Professional.

Bosch, J., G. Florijn, D. Greefhorst, J. Kuusela, J. H. Obbink and K. Pohl (2002). Variability Issues in Software Product Lines. Revised Papers from the 4th International Workshop on Software Product-Family Engineering, Springer**:** 13-21.

Brereton, P., B. A. Kitchenham, D. Budgen, M. Turner and M. Khalil (2007). "Lessons from applying the systematic literature review process within the software engineering domain." The Journal of Systems & Software **80**(4): 571-583.

Brown, A. W. and G. Booch (2002). Reusing Open-Source Software and Practices: The Impact of Open-Source on Commercial Vendors. 7th International Conference on Software Reuse: Methods, Techniques, and Tools, Springer**:** 123-136.

Broy, M. and A. Rausch (2005). "Das neue V-Modell® XT." Informatik-Spektrum **28**(3): 220-229.

Calero, C., F. Ruiz and M. Piattini (2006). Ontologies for Software Engineering and Software Technology, Springer-Verlag New York Inc.

Chappel, D. A. (2004). Enterprise Service Bus. Sebastopol, CA, O'Reilly Media.

Chen, P. P.-S. (1976). "The entity-relationship model - toward a unified view of data." ACM Trans. Database Syst. **1**(1): 9-36.

Cho, I.-H., J. D. McGregor and L. Krause (1998). A protocol based approach to specifying interoperability between objects. 26th International Conference on Technology of Object-Oriented Languages (TOOLS 26).

Christensen, E., F. Curbera, G. Meredith and S. Weerawarana. (2001). "Web Services Description Language (WSDL) 1.1." from http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

Cranefield, S. (2002). UML and the Semantic Web, IOS Press Inc.

Cruz, I. R., X. Huiyong and H. Feihong (2004). An ontology-based framework for XML semantic integration. International Database Engineering and Applications Symposium (IDEAS '04), IEEE**:** 217-226.

David, P. (1996). "Group communication." Commun. ACM **39**(4): 50-53.

Decker, B., E. Ras, J. Rech, B. Klein and C. Hoecht (2005). Self-organized reuse of software engineering knowledge supported by semantic wikis. Workshop on Semantic Web Enabled Software Engineering.

Djurić, D., D. Gašević and V. Devedžić (2005). "Ontology modeling and MDA." Journal on Object Technology **4**(1): 109–128.

Do, H.-H. and E. Rahm (2002). COMA: a system for flexible combination of schema matching approaches. 28th international conference on Very Large Data Bases. Hong Kong, China, VLDB Endowment.

Doan, A. and A. Halevy (2005). "Semantic integration research in the database community: A brief survey." AI Magazine **26**(1): 83-94.

Doan, A., J. Madhavan, P. Domingos and A. Halevy (2004). Ontology matching: A machine learning approach. Handbook on Ontologies. S. Staab and R. Studer, Springer**:** 385–516.

Doan, A., N. F. Noy and A. Y. Halevy (2004). "Introduction to the special issue on semantic integration." SIGMOD Rec. **33**(4): 11-13.

Ehrgott, M. (2005). Multicriteria Optimization, Springer.

Ehrig, M. (2007). Ontology Alignment: Bridging the Semantic Gap, Springer-Verlag New York Inc.

Ehrig, M. and S. Staab (2004). Efficiency of ontology mapping approaches. International Workshop on Semantic Intelligent Middleware for the Web and the Grid at ECAI 04. Valencia, Spain.

Ehrig, M. and S. Staab (2004). "QOM-quick ontology mapping." Lecture Notes in Computer Science: 683-697.

Ehrig, M., S. Staab and Y. Sure (2005). Bootstrapping ontology alignment methods with APFEL. International Semantic Web Conference (ISWC 2005), Springer**:** 186-200.

Ehrig, M. and Y. Sure (2005). FOAM–Framework for Ontology Alignment and Mapping Results of the Ontology Alignment Evaluation Initiative. Workshop on Integrating Ontologies**:** 72.

Feier, C., D. Roman, A. Polleres, J. Domingue, M. Stollberg and D. Fensel (2005). Towards intelligent web services: The web service modeling ontology (WSMO). International Conference on Intelligent Computing (ICIC).

Fensel, D. (2003). Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, Springer.

Fensel, D. and C. Bussler (2002). "The Web Service Modeling Framework WSMF." Electronic Commerce Research and Applications **1**(2): 113-137.

Fensel, D., F. Van Harmelen, I. Horrocks, D. L. McGuinness and P. F. Patel-Schneider (2001). "OIL: An ontology infrastructure for the semantic web." IEEE intelligent systems **16**(2): 38-45.

Floyd, C. (1984). "A systematic look at prototyping." Approaches to prototyping: 1-18.

Frakes, W. B. and R. Baeza-Yates (1992). Information retrieval: data structures and algorithms, Prentice-Hall, Inc. Upper Saddle River, NJ, USA.

Frankel, D. S. (2003). Model Driven Architecture, Applying MDA to Enterprise Computing, Wiley.

Gail, E. H., L. David, C. Jeromy, re, N. Fred, C. John and N. Martin (2003). Application servers: one size fits all ... not? Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications. Anaheim, CA, USA, ACM.

Gangemi, A., N. Guarino, C. Masolo and A. Oltramari (2003). "Sweetening WordNet with DOLCE." AI Magazine **24**(4): 13-24.

Gaševic, D., D. Djuric, V. Devedzic and V. Damjanovic (2004). Approaching OWL and MDA through technological spaces. Workshop WS5 at the 7th International Conference on the UML. Lisbon, Portugal.

Gašević, D., D. Djurić, V. Devedžić and B. Selić (2006). Model driven architecture and ontology development, Springer-Verlag New York, Inc. Secaucus, NJ, USA.

Goh, C. H. (1996). Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems, MIT. **PhD**.

Goméz-Pérez, A., M. Fernandez-Lopez and O. Corcho (2003). Ontological Engineering, Springer.

Gorton, I. and A. Liu (2004). Architectures and Technologies for Enterprise Application Integration. 26th International Conference on Software Engineering, IEEE Computer Society**:** 726-727.

Gorton, I., D. Thurman and J. Thomson (2003). Next generation application integration: challenges and new approaches. Computer Software and Applications Conference, 2003. COMPSAC 2003. Proceedings. 27th Annual International**:** 576-581.

Gruber, T. R. (1993). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Formal Ontology in Conceptual Analysis and Knowledge Representation. N. Guarino and R. Poli, Kluwer Academic Publishers.

Gruber, T. R. (1993). "A translation approach to portable ontology specifications." Knowledge acquisition **5**(2): 199-220.

Gruber, T. R. (1995). "Toward principles for the design of ontologies used for knowledge sharing." International Journal of Human Computer Studies **43**(5): 907-928.

Halevy, A. (2005). "Why your data won't mix." Queue **3**(8): 50-58.

Haller, A., E. Cimpian, A. Mocan, E. Oren and C. Bussler (2005). WSMX-a semantic service-oriented architecture. International Conference on Web Services (ICWS 2005), IEEE**:** 321-328.

Happel, H. J. and S. Seedorf (2006). Applications of ontologies in software engineering. Workshop on Sematic Web Enabled Software Engineering**:** 5-9.

Heiler, S. (1995). "Semantic interoperability." ACM Comput. Surv. **27**(2): 271-273.

Hohpe, G. and B. Woolf (2004). Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, Addison-Wesley Professional.

Horrocks, I. (2002). "DAML+ OIL: a description logic for the semantic web." IEEE Data Engineering Bulletin **25**: 4-9.

Hull, R. and R. King (1987). "Semantic database modeling: survey, applications, and research issues." ACM Comput. Surv. **19**(3): 201-260.

Jennings, N. and M. J. Wooldridge (1998). Agent technology: foundations, applications, and markets, Springer Verlag.

Juric, M. B. (2006). Business Process Execution Language for Web Services BPEL and BPEL4WS 2nd Edition, Packt Publishing.

Kiko, K. and C. Atkinson (2005). Integrating enterprise information representation languages. International Workshop on Vocabularies, Ontologies and Rules for The Enterprise. Enschede, The Netherlands.

Kitchenham, B. A., S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. El Emam and J. Rosenberg (2002). "Preliminary guidelines for empirical research in software engineering." IEEE Transactions on Software Engineering **28**(8): 721-734.

Kleppe, A. G., J. B. Warmer and W. Bast (2003). MDA Explained: The Model Driven Architecture: Practice and Promise, Addison-Wesley.

Knublauch, H., D. Oberle, P. Tetlow and E. Wallace (2006). "A semantic web primer for object-oriented software developers." W3C Working Group Note http://www.w3.org/TR/sw-oosdprimer.

Kolovski, V., B. Parsia, Y. Katz and J. Hendler (2005). Representing Web Service Policies in OWL-DL. 4th International Semantic Web Conference (ISWC 2005), Springer**:** 461-475.

Kramer, J. and J. Magee (1985). "Dynamic configuration for distributed systems." IEEE Transactions on Software Engineering **11**(4): 424-436.

Kruchten, P. (2000). The rational unified process: an introduction, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Kubalík, J., R. Mordinyi and S. Biffl (2008). Multiobjective Prototype Optimization with Evolved Improvement Steps. Evolutionary Computation in Combinatorial Optimization.

Lausen, H., A. Polleres and D. Roman (2005). "Web Service Modeling Ontology (WSMO)." W3C Member Submission **3**.

Lee, S. W. and R. A. Gandhi (2005). Ontology-based active requirements engineering framework. 12th Asia-Pacific Software Engineering Conference (APSEC '05) 481-490.

Leinhos, S. (2006). OWL ontology extraction and modelling from and with UML class diagrams - a practical approach. Munich, University of the Federal Armed Forces of Germany. **MSc**.

Levy, A. Y. (2000). Logic-based techniques in data integration. Logic-based artificial intelligence, Kluwer Academic Publishers**:** 575-595.

Li, L. and I. Horrocks (2004). "A Software Framework for Matchmaking Based on Semantic Web Technology." International Journal of Electronic Commerce **8**(4): 39-60.

Liao, S. (2005). "Technology management methodologies and applications: A literature review from 1995 to 2003." Technovation **25**(4): 381-393.

Lin, J., M. S. Fox and T. Bilgic (1996). "A requirement ontology for engineering design." Concurrent Engineering **4**(3): 279.

Lovett, P. J., A. Ingram and C. N. Bancroft (2000). "Knowledge-based engineering for SMEs - a methodology." Journal of Materials Processing Technology **107**(1-3): 384-389.

Luckham, D. (2002). The power of events: an introduction to complex event processing in distributed enterprise systems, Addison-Wesley Professional.

Lüder, A. (2000). Formaler Steuerungsentwurf mit modularen diskreten Verhaltensmodellen. Halle-Wittenberg, Martin-Luther-Universität. **PhD:** 156.

Lüder, A., J. Peschke, T. Sauter, S. Deter and D. Diep (2004). "Distributed intelligence for plant automation based on multi-agent systems: the PABADIS approach." Production Planning and Control **15**(2): 201-212.

Malveau, R. C. (2000). Software Architect Bootcamp: A Programmer's Field Manual, Prentice Hall PTR Upper Saddle River, NJ, USA.

Martin, D., A. Ankolekar, M. Burstein, G. Denker, D. Elenius, J. Hobb, L. Kagal, O. Lassila, D. McDermott, D. McGuinness, S. McIlraith, M. Paolucci, B. Parsia, T. Payne, M. Sabou, C. Schlenoff, E. Sirin, M. Solanki, N. Srinivasan, K. Sycara and R. Washington. (2004). "OWL-S 1.1 Release." from http://www.daml.org/services/owl-s/1.1.

Martin, D., M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou and M. Solanki (2005). Bringing Semantics to Web Services: The OWL-S Approach. First International Workshop on Semantic Web Services and Web Process Composition, Springer**:** 26-42.

Mayank, V., N. Kositsyna and M. Austin (2004). "Requirements Engineering and the Semantic Web, Part II. Representaion, Management, and Validation of Requirements and System-Level Architectures." Technical Report TR 2004-14. University of Maryland.

McGuire, J. G., D. R. Kuokka, J. C. Weber, Tenenbaum, J. M., T. R. Gruber and G. R. Olsen (1993). "SHADE:Technology for Knowledge-based Collaborative Engineering." Concurrent Engineering **1993**(1): 137-146.

McIlraith, S. A., T. C. Son and H. Zeng (2001). "Semantic Web Services." IEEE Intelligent Systems **16**(2): 46-53.

Mellor, S. J. and M. J. Balcer (2002). Executable UML: A Foundation for Model-Driven Architecture, Addison-Wesley.

Mellor, S. J., A. N. Clark and T. Futagami (2003). "Guest editors' introduction: Model-driven development." IEEE Software **20**(5): 14-18.

Mellor, S. J., S. Kendall, A. Uhl and D. Weise (2004). MDA distilled, Addison-Wesley.

Merdan, M., T. Moser, D. Wahyudin and S. Biffl (2008). Performance Evaluation of Workflow Scheduling Strategies Considering Transportation Times and Conveyor Failures. International Conference on Industrial Engineering and Engineering Management (IEEM). Singapore**:** 389-394.

Merdan, M., T. Moser, D. Wahyudin and S. Biffl (2008). Simulation of Workflow Scheduling Strategies Using the MAST Test Management System. 10th International Conference on Control, Automation, Robotics and Vision (ICARCV 2008). Hanoi, Vietnam**:** 1172-1177.

Mike, P. P. and H. Willem-Jan (2007). "Service oriented architectures: approaches, technologies and research issues." The VLDB Journal **16**(3): 389-415.

Miller, J. and J. Mukerji (2001). "Model Driven Architecture (MDA)." Object Management Group, Draft Specification ormsc/2001-07-01, July **9**.

Miller, J., K. Verma, P. Rajasekaran, A. Sheth, R. Aggarwal and K. Sivashanmugam (2004). "WSDL-S: Adding Semantics to WSDL-White Paper."

Mockus, A., R. T. Fielding and J. D. Herbsleb (2002). "Two case studies of open source software development: Apache and Mozilla." ACM Trans. Softw. Eng. Methodol. **11**(3): 309-346.

Mordinyi, R., T. Moser, A. Mikula and S. Biffl (2009). Foundations for a Model-Driven Integration of Business Services in a Safety-critical Application Domain. Track on Software Process and Product Improvements (SPPI) at the 35th Euromicro Conference Software Engineering and Advanced Applications (SEAA 2009).

Moser, T. and A. Anjomshoaa (2007). "FISN Semantic Architecture Document." Frequentis AG.

Moser, T., S. Biffl, W. D. Sunindyo and D. Winkler (2010). Integrating Production Automation Expert Knowledge Across Engineering Stakeholder Domains. International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2010). Krakow, Poland.

Moser, T., K. Kunz, K. Matousek and D. Wahyudin (2008). Investigating UML- and Ontology- Based Approaches for Process Improvement in Developing Agile Multi-Agent Systems. 34th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2008)**:** 224-231.

Moser, T., M. Merdan and S. Biffl (2010). A Pattern-Based Coordination and Test Framework for Multi-Agent Simulation of Production Automation Systems. Fourth Workshop on Engineering Complex Distributed Systems (ECDS-2010) co-located with the International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2010). Krakow, Poland.

Moser, T., R. Mordinyi, A. Mikula and S. Biffl (2009). Efficient System Integration Using Semantic Requirements and Capability Models: An approach for integrating heterogeneous Business Services. 11th International Conference on Enterprise Information Systems (ICEIS 2009). Milan, Italy. **1:** 56-63.

Moser, T., R. Mordinyi, A. Mikula and S. Biffl (2009). Making Expert Knowledge Explicit to Facilitate Tool Support for Integrating Complex Information Systems in the ATM Domain. International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2009). Fukuoka, Japan**:** 90-97.

Moser, T., R. Mordinyi, W. D. Sunindyo and S. Biffl (2009). Semantic Service Matchmaking in the ATM Domain Considering Infrastructure Capability Constraints. 21st International Conference on Software Engineering and Knowledge Engineering (SEKE 2009). Boston, USA**:** 222-227.

Moser, T., H. Roth, S. Rozsnyai, R. Mordinyi and S. Biffl (2009). Semantic Event Correlation Using Ontologies. 8th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2009). Vilamoura, Algarve-Portugal.

Moser, T., A. Schatten, W. D. Sunindyo and S. Biffl (2009). "A Run-Time Engineering Knowledge Base for Reconfigurable Systems." Technical Report (available online at: http://www.ifs.tuwien.ac.at/files/A Run-Time Engineering Knowledge Base for Reconfigurable Systems - Technical Report.pdf).

Moser, T., K. Schimper, R. Mordinyi and A. Anjomshoaa (2009). SAMOA - A Semi-automated Ontology Alignment Method for Systems Integration in Safety-critical Environments. 2nd IEEE International Workshop on Ontology Alignment and Visualization (OnAV'09). Fukuoka, Japan: 724-729.

Moser, T., D. Winkler and S. Biffl (2010). "Engineering Analyses across Domain Boundaries using the Engineering Knowledge Base Framework." Technical Report (available online at: http://www.ifs.tuwien.ac.at/files/Engineering Analyses across Domain Boundaries using the Engineering Knowledge Base Framework - Technical Report.pdf).

Nguyen, C. D., A. Perini and P. Tonella (2008). Ontology-based test generation for multiagent systems. 7th international joint conference on Autonomous agents and multiagent systems. Estoril, Portugal, International Foundation for Autonomous Agents and Multiagent Systems: 1315-1320.

Nguyen, T. M., A. M. Tjoa, G. Kickinger and P. Brezany (2004). Towards service collaboration model in grid-based zero latency data stream warehouse (GZLDSWH). IEEE International Conference on Services Computing (SCC 2004): 357-365.

Niles, I. and A. Pease (2001). Towards a standard upper ontology. 2nd International Conference on Formal Ontology in Information Systems, ACM: 2-9.

Noy, N. (2005). Ontology Mapping and Alignment KnowledgeWeb Summer School 2005. Video.

Noy, N. and H. Stuckenschmidt (2005). "Ontology alignment: An annotated bibliography." Dagstuhl Workshop Report: Semantic Interoperability and Integration.

Noy, N. F. (2004). "Semantic integration: a survey of ontology-based approaches." SIGMOD Rec. **33**(4): 65-70.

Noy, N. F., A. H. Doan and A. Y. Halevy (2005). "Semantic Integration." AI Magazine **26**(1): 7-10.

Noy, N. F. and M. A. Musen (2000). PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, AAAI Press / The MIT Press: 450-455.

Noy, N. F. and M. A. Musen (2001). Anchor-PROMPT: Using non-local context for semantic matching. Workshop on Ontologies and Information Sharing at the Seventeenth International Joint Conference on Artificial Intelligence: 63-70.

Noy, N. F. and M. A. Musen (2003). "The PROMPT suite: interactive tools for ontology merging and mapping." International Journal of Human-Computer Studies **59**(6): 983-1024.

Oldham, N., K. Verma, A. Sheth and F. Hakimpour (2006). Semantic WS-agreement partner selection. 15th International World Wide Web Conference. Edinburgh, Scotland, ACM: 697-706.

OMG (2006). "Ontology Definition Metamodel RFP." 6th Revised Submission **http://www.omg.org/dontology**.

Paolucci, M., T. Kawamura, T. R. Payne and K. Sycara (2002). Semantic Matching of Web Services Capabilities. First International Semantic Web Conference, Springer: 333-347.

Parreiras, F. S., S. Staab and A. Winter (2007). On marrying ontological and metamodeling technical spaces. 6th Joint Meeting on European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering: companion papers. Dubrovnik, Croatia, ACM**:** 439-448.

Pinelle, D. and C. Gutwin (2005). A groupware design framework for loosely coupled workgroups. Ninth conference on European Conference on Computer Supported Cooperative Work. Paris, France, Springer-Verlag New York, Inc.**:** 65-82.

Piyush, M. and P. Michael (2005). "Benchmarking message-oriented middleware: TIBCO versus SonicMQ: Research Articles." Concurr. Comput. : Pract. Exper. **17**(12): 1507-1526.

Pollock, J. (2002). "Integration's Dirty Little Secret: It's a Matter of Semantics." Whitepaper, Modulant: The Interoperability Company.

Powers, S. (2003). Practical RDF, O'Reilly & Associates, Inc. Sebastopol, CA, USA.

Prud'hommeaux, E. and A. Seaborne (2007). SPARQL Query Language for RDF W3C Candidate Recommendation 14 June 2007, Technical report, W3C, 2007.

Purtilo, J. M. and J. M. Atlee (1991). "Module Reuse by Interface Adaptation." Software - Practice and Experience **21**(6): 539-556.

Rahm, E. and P. A. Bernstein (2001). "A survey of approaches to automatic schema matching." VLDB Journal **10**(4): 334-350.

Rebstock, M. and H. Paulheim (2008). Ontologies-based Business Integration, Springer Verlag.

Rosenthal, A., L. Seligman and S. Renner (2004). "From semantic integration to semantics management: case studies and a way forward." SIGMOD Rec. **33**(4): 44-50.

Satoh, F., Y. Nakamura, N. K. Mukhi, M. Tatsubori and K. Ono (2008). Methodology and Tools for End-to-End SOA Security Configurations. Services - Part I, 2008. IEEE Congress on**:** 307-314.

Schäfer, W. and H. Wehrheim (2007). The Challenges of Building Advanced Mechatronic Systems. 2007 Future of Software Engineering - International Conference on Software Engineering. Washington, DC, IEEE Computer Society**:** 72-84.

Schäfer, W. and H. Wehrheim (2007). The Challenges of Building Advanced Mechatronic Systems. 2007 Future of Software Engineering - International Conference on Software Engineering, Washington, DC, IEEE Computer Society.

Scheer, A. W. (1989). Computer-Integrated Manufacturing, Springer.

Schiefer, J. and C. McGregor (2004). "Correlating Events for Monitoring Business Processes." Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS), Porto.

Schiefer, J. and A. Seufert (2005). "Management and Controlling of Time-Sensitive Business Processes with Sense & Respond." International Conference on Computational Intelligence for Modelling Control and Automation (CIMCA), Vienna.

Seidewitz, E. (2003). "What models mean." IEEE Software **20**(5): 26-32.

Selic, B. (2003). "The pragmatics of model-driven development." IEEE Software **20**(5): 19-25.

Sivashanmugam, K., K. Verma, A. Sheth and J. Miller (2003). Adding Semantics to Web Services Standards. International Conference on Web Services**:** 395–401.

Stachowiak, H. (1973). Allgemeine Modelltheorie, Springer-Verlag.

Studer, R., V. R. Benjamins and D. Fensel (1998). "Knowledge engineering: principles and methods." Data & Knowledge Engineering **25**(1-2): 161-197.

Sure, Y., M. Ehrig and R. Studer (2006). Automatische Wissensintegration mit Ontologien. Workshop "Modellierung für Wissensmanagement" im Rahmen der Tagung "Modellierung 2006".

Ten-Hover, R. and P. Walker (2005). Java™ Business Integration (JBI) 1.0, Sun Microsystems, Inc.

Tetlow, P., J. Z. Pan, D. Oberle, E. Wallace, M. Uschold and E. Kendall (2005). "Ontology driven architectures and potential uses of the semantic web in systems and software engineering." W3C Working Draft.

Thai, J., B. Pekilis, A. Lau and R. Seviora (2001). Aspect-oriented implementation of software health indicators. Software Engineering Conference, 2001. APSEC 2001. Eighth Asia-Pacific**:** 97-104.

Trastour, D., C. Bartolini and J. Gonzalez-Castillo (2001). "A Semantic Web Approach to Service Description for Matchmaking of Services." HP LABORATORIES TECHNICAL REPORT.

Trowbridge, D., U. Roxburgh, G. Hohpe, D. Manolescu and E. Nadhan (2004). Integration Patterns. Patterns & Practices, Microsoft Press.

Uschold, M. and M. Gruninger (2004). "Ontologies and semantics for seamless connectivity." SIGMOD Rec. **33**(4): 58-64.

Verma, K., R. Akkiraju and R. Goodwin (2005). Semantic Matching of Web Service Policies. 2nd International Workshop on Semantic and Dynamic Web Process (SDWP 2005).

Vrba, P. (2003). MAST: manufacturing agent simulation tool. Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference. **1:** 282-287 vol.1.

Wache, H., T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann and S. Hübner (2001). Ontology-based integration of information-a survey of existing approaches. Workshop on Ontologies and Information Sharing (IJCAI-01). Seattle, USA**:** 108-117.

Wahyudin, D., M. Heindl, B. Eckhard, A. Schatten and S. Biffl (2008). "In-time role-specific notification as formal means to balance agile practices in global software development settings." Lecture Notes in Computer Science **5082**: 208-222.

Wahyudin, D., K. Mustofa, A. Schatten, S. Biffl and A. M. Tjoa (2007). "Monitoring the" health" status of open source web-engineering projects." International Journal of Web Information Systems **3**(1-2): 116-139.

Wang, P. and B. Xu (2007). LILY: The Results for the Ontology Alignment Contest OAEI 2007. 2nd International Ontology Matching Workshop (OM-2007) co-located with the 6th International Semantic Web Conference (ISWC-2007). Busan, Korea**:** 179-187.

Wang, P. and B. Xu (2008). Lily: Ontology alignment results for OAEI 2008. Third International Workshop on Ontology Matching (OM-2008) co-located with the 7th International Semantic Web Conference (ISWC-2008). Karlsruhe, Germany**:** 167–175.

Wang, Y. and E. Stroulia (2003). Flexible interface matching for Web-service discovery. Fourth International Conference on Web Information Systems Engineering, (WISE 2003).

Weilkiens, T. (2008). Systems engineering with SysML/UML: modeling, analysis, design, Morgan Kaufmann.

Wouters, B., D. Deridder and E. Van Paesschen (2000). The use of ontologies as a backbone for use case management. European Conference on Object-Oriented Programming (ECOOP 2000), Workshop : Objects and Classifications, a natural convergence.

Xiaoying, B., X. Jihui, C. Bin and X. Sinan (2007). DRESR: Dynamic Routing in Enterprise Service Bus. e-Business Engineering, 2007. ICEBE 2007. IEEE International Conference on**:** 528-531.

Zaremski, A. M. and J. M. Wing (1995). "Signature Matching: A Tool for Using Software Libraries." ACM Transactions on Software Engineering and Methodology(4): 146--170.

Zaremski, A. M. and J. M. Wing (1997). "Specification matching of software components." ACM Trans. Softw. Eng. Methodology (TOSEM) **6**(4): 333-369.

Glossary

ACIN	Automation Control Institute
ACL	Agent Communication Language
API	Application Programming Interface
ASE	Automation Systems Engineering
ATM	Air Traffic Management
AutomationML	Automation Markup Language
BPEL	Business Process Execution Language
BPEL4WS	Business Process Execution Language for Web Services
CASE	Computer-Aided Software Engineering
CBSE	Component Based Software Engineering
CDATA	Character Data
CDM	Collaborative Decision Making
CEP	Complex Event Processing
CIM	Computation Independent Model
DAI	Distributed Artificial Intelligence
DAML	DARPA Agent Markup Language
DAWG	RDF Data Access Working Group
ebXML	Electronic Business using XML
ECSI	Enterprise-Control System Integration
EDB	Engineering Data Base
EER	Enhanced Entity-Relationship Model
EKB	Engineering Knowledge Base
ERM	Entity Relationship Model
ERP	Enterprise Resource Planning
ESB	Engineering Service Bus
FIPA	Foundation for Intelligent Physical Agents
FOAM	Framework for Ontology Alignment and Mapping
HTML	HyperText Markup Language
IDL	Interface Definition Language
IR	Information Retrieval
I/O	Input/Output
IT	Information technology
JADE	Java Agent Development Framework
JCR	Java Content Repository
MAS	Multi-Agent Systems
MAST	Manufacturing Agents Simulation Tool
MDA	Model-Driven Architecture
MDD	Model-Driven Development
MOF	Meta Object Facility
NOM	Naïve Ontology Alignment

OBA	Ontology-based architectures
OCL	Object Constraint Language
ODA	Ontology-Driven Architecture
ODD	Ontology-Driven development
ODM	Ontology Definition Metamodel
OEA	Ontology-Enabled architectures
OED	Ontology-Enabled development
OIL	Ontology Inference Layer
OKBC	Open Knowledge Base Connectivity
OMG	Object Management Group
OpenEngSB	Open Engineering Service Bus
OSS	Open Source Software
OWL	Web Ontology Language
OWL-DL	OWL Description Logic
OWL-S	Ontology Web Language for Services
PDF	Portable Document Format
PIM	Platform-Independent Model
PM	Project Management
PSM	Platform-Specific Model
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RI	Research Issue
RUP	Rational Unified Process
QA	Quality Assurance
QOM	Quick Ontology Mapping
SME	Subject Matter Expert
SPARQL	SPARQL Protocol and RDF Query Language
SAW	Simulation of Assembly Workshops
SHOE	Simple HTML Ontology Extension
SLO	Service Level Objectives
SOA	Service-Oriented Architecture
SQL	Structured Query Language
SUMO	Suggested Upper Merged Ontology
SWIM	System-Wide Information Management
SWIS	System-Wide Information Sharing
SWRL	Semantic Web Rule Language
sysML	Systems Modeling Language
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
WSMF	Web Service Modeling Framework
WSML	Web Service Modeling Language
WSMO	Web Service Modeling Ontology
XMI	XML Metadata Interchange
XML	Extensible Markup Language

XPath	XML Path Language
XOL	XML-based Ontology Language
XQuery	XML Query Language
XSL	Extensible Stylesheet Language
XSLT	XSL Transformations