FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

An Emergent System for Multimedial Document Management

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medieninformatik

eingereicht von

Alexander Sinnl, BSc.
Matrikelnummer 0928723

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: Ao. Univ.-Prof. Mag. Dr. Horst Eidenberger

Wien, 25. Jänner 2017

Alexander Sinnl	Horst Eidenberger

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

An Emergent System for Multimedial Document Management

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Media Informatics

by

Alexander Sinnl, BSc.
Registration Number 0928723

to the Faculty of Informatics

at the TU Wien

Advisor: Ao. Univ.-Prof. Mag. Dr. Horst Eidenberger

Vienna, 25th January, 2017

Alexander Sinnl	Horst Eidenberger

Erklärung zur Verfassung der Arbeit

Alexander Sinnl, BSc.
Wollmannsberg 43, 2003 Leitzersdorf

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, 25. Jänner 2017

Alexander Sinnl

Kurzfassung

Die von der *Digitalen Revolution* ausgelösten Veränderungen unserer Welt bringen diverse Herausforderungen mit sich. So handelt eine der großen Fragen dieser Zeit davon, wie ein nachhaltiger Umgang mit (digitalen) Daten aussehen kann. Jeder Mensch produziert und nutzt tagtäglich eine Vielzahl an Daten, die abhängig vom jeweiligen Kontext sinnstiftend zu Informationen zusammengefasst werden. Eine dauerhafte Speicherung dieser Informationen lässt sich mithilfe multimedialer Dokumente realisieren. Gesamtheitlich betrachtet, entsteht aus diesen Einzelhandlungen ein natürlicher Kreislauf, in dem Menschen maschinenunterstützt und in Beziehung zueinander Informationen erzeugen wie abrufen.

Für den konkreten Anwendungsfall dieser Diplomarbeit, dem Schaffen eines Softwareprototyps für die Erfassung und Verwaltung von Studienbewerbungen an der Technischen Universität Wien, sind folgende Minimalanforderungen umzusetzen: Der entwickelte *Student Records Manager* speichert Informationen (Dokumente und Metadaten) in Studierendenakten und präsentiert diese zur Bearbeitung via Benutzeroberfläche in drei verschiedenen Sichten. *Studierende* haben die Möglichkeit, Dokumente für Studienbewerbungen hochzuladen und diese abzusenden. *Mitarbeiter* der Studien- und Prüfungsabteilung der TU Wien können diese Bewerbungen evaluieren. *Studiendekane* sind in der Lage, Stellungnahmen zu verfassen.

Um diesen Anforderungen gerecht zu werden, basiert der *Student Records Manager* auf dem frei zugänglichen elektronischen Dokumentenmanagementsystem *Nuxeo Platform*, das sich im Zuge einer Marktsondierung als geeignete Grundlage erwiesen hat. Der Prototyp setzt auf einen benutzerzentrierten Ansatz, bei dem der Anwender gezielt automationsunterstützt wird. Das Hauptaugenmerk liegt auf der annähernd optimalen Integration der Stärken und Schwächen von Mensch und Maschine, um einen möglichst effiziente Behandlung von Studienbewerbungen für alle beteiligten Akteure zu gewährleisten.

Die abschließend durchgeführte Evaluierung des Prototyps bestätigt dessen grundsätzliche Tauglichkeit für die Erfassung und Verwaltung von Studienbewerbungen an der Technischen Universität Wien. In einem möglichen nächsten Schritt könnte der *Student Records Manager* testweise in der geplanten Einsatzumgebung implementiert werden, um in der Folge weitere Maßnahmen abzuleiten.

Abstract

The *digital revolution* changes our world in unprecedented manner and generates manifold challenges for humankind thereby. One of the major questions in the present time is concerned with the sustainable handling of (digital) data. Each individual produces and uses a multitude of data on a daily basis, which are assembled to meaningful information in reference to the particular context. A permanent storage of this information can be realised by means of multimedial documents. Altogether, these single activities constitute a natural cycle of collaboration, where humans capitalise on automation to create and retrieve information in relation to each other.

The concrete use case to be realised within this diploma thesis includes the development of a software prototype for the acquisition and management of students' applications at the Vienna University of Technology. Several measures have to be taken into account to achieve this goal: the prototypical *Student Records Manager* allows to store information (documents and metadata) in student records on the one hand, and provides a user interface with three different views for the manipulation of information on the other hand. *Students* are enabled to upload documents and send their application in consequence. *Employees* of the Admission Office at the Vienna University of Technology are empowered to review these applications. *Deans* are set in a position to compose statements on the applications.

In order to be able to meet these requirements, the *Student Records Manager* is built upon the open source electronic document management system *Nuxeo Platform*, which qualified as an adequate foundation in the course of a market sounding. Overall, the prototype is premised on a user-centred approach with selective automation support. Therefore, the main focus is directed towards an optimised integration of the strengths and weaknesses of humans and machines, that should ensure the most efficient processing of university applications for all involved participants.

The concluding evaluation of the software prototype acknowledges its suitability for the acquisition and management of applications at the Vienna University of Technology. As a next step, the *Student Records Manager* could possibly be implemented in the designated operation environment and tested accordingly.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Scientia potentia est: with the development of speech and writing, humanity was able to submit the world to their rules. May this be fortunate or not, the ability to exchange information has been one of the milestones in the evolution of men. Still, the decisive step for progressing in art and skills up to the level we experience in these days is owed to another fact: the power to capture and store information in media. We can rely on the knowledge of thousands of years and do not have to reinvent the wheel once again.

With the *Digital Revolution*, the amount of available data was taken to another dimension. Providing loads of opportunities for improvement, this advancement also exceeds the natural capabilities of humans at the same time. One is not able to filter all the surrounding data and suffers from *information overload* in many domains of life. This is the point, were technology has to intervene.

Especially in working environments, where the paradigms typically state to be as time- and cost-effective as possible, processes need a certain degree of automation. Hence, it is necessary to understand, how the strengths and weaknesses of human beings and machines have to be combined for approximating to an optimal solution for the given tasks. In general, computers are able to provide processing power and storage space. That can be exploited for scaling down the quantities of data, presenting people only the information required for their actual activity. Whereas humans are talented in accomplishing these single activities and reacting to small changes in their daily routines.

Based on the above explanations, this thesis is concerned with the development of a software prototype for the management of student records at the Vienna University of Technology. In particular, the implemented system has to incorporate the tasks of sending, revising, evaluating, storing and searching university applications by humans in their different roles.

1.2 Problem definition

From an abstract point of view, the Vienna University of Technology can be described as a system comprised of subsystems. These organisational units are more or less connected to each other and engage human employees for fulfilling certain data-based tasks. Thus, information storage and retrieval is a central mechanism as well as benchmarking factor for the performance of these processes.

Currently, the procedure for gathering and handling student records by the *Admission Office* can be seen as a *workaround* based on existing electronical systems. As these systems have not been designed especially for the mentioned purpose, they share several limitations in terms of features and usability: in general, the Vienna University of Technology does not have an evolved Document Management System (DMS) for the whole entity, and relies on individual solutions in the different organisational units.

Focusing on the student records, essential parts of the working-process are only done paper-based and not stored for future utilisation. This leads to extra effort and expense in many cases. For instance, it is difficult to integrate the student records in the existing environment at the Vienna University of Technology overall, complicating the internal and external access. Employees need additional operations to locate the required information or spend time for double-checking already revised documents. Furthermore, redundant communication patterns and grown dependencies burden the general workflow.

As the amount of data to store is rising from semester to semester with the application of new students, there could be a natural and technical breaking point for the existing system. Accordingly, it is sensible to solve the emerging problems by the installation of a new system, adapted to the special requirements. Moreover, the novel software prototype can be generically enough, to deliver a possible foundation for a holistic Document Management System at the Vienna University of Technology in future.

1.3 Thesis outline

The overall ambition of this thesis is to develop a software prototype for the management of student records at the Vienna University of Technology. To achieve this goal, Chapter 2 provides the *Theoretical background* by facilitating the fundamental knowledge about Document Management Systems and the general setting. A literature review deepens the understanding by highlighting prime issues of DMS like efficient information storage and retrieval techniques or critical design factors for performant usability. Leading to the discussion, what the term *Document Management System* signifies for, and which areas of application do exist in consequence. In Chapter 3, all the previously acquired findings are utilised for derivating the requirements and planning of the implementation, establishing the *Project description*. The current environment at the Vienna University of Technology, in which the prototypical DMS for the student records will be embedded, is investigated. Considering this information, a market sounding of open-source DMS platforms leads to the selection of one particular framework, that the prototype will be

built upon. Eventually, a draft for the to-be implemented DMS is prepared. Chapter 4 then yields the resulting realisation of the prototype and takes a look on its components, accompanied by an evaluation of the newly created system in Chapter 5. Ultimately, the thesis concludes with a final reflection of the whole work and its process in Chapter 6.

CHAPTER 2

Theoretical background

This chapter is intended to give an introduction on basic concepts in the field of Document Management Systems. For this purpose, some surrounding topics are expounded to build the general context at first. Then, it is possible to narrow down the scope of DMS, identifying their essential features as well as critical factors for the development of such frameworks.

2.1 Information-based working

The general goal of information-based working can be summarised in one sentence: delivering the right information at the right time in the right format to the right person(s). As each part of this process implies its own requirements and challenges, multidisciplinary approaches have to be taken into account. It is important to unterstand, how human interaction with computers and amongst themselves operates at the core. For this reason, the topics of *Computer Supported Cooperative Work*, *Interaction Desgin* and *Information Storage and Retrieval* are examined in the following subchapters.

2.1.1 Computer Supported Cooperative Work

Recapitulating the words of Schmidt and Bannon [1], the domain of Computer Supported Cooperative Work (CSCW) unfolds in a wide range of applications. From a technical standpoint, all available computing technologies for the support of cooperative work (of humans) have to be considered as relevant building blocks. Since the development of these technologies is an ongoing, hardly predictable process, the shape of CSCW is constantly altering. As a result, CSCW is an interdisciplinary subject, that touches and connects to different fields of problems with their own disciplinary demands. On a general level, Greif [2, p. 6] summarises the term as a *computer-assisted coordinated activity such as communication and problem solving carried out by a group of collaborating individuals.*

Related to the practice, one key-insight is found in the following observation [2, p. 7]: *transaction-oriented database systems depend on "coordination technologies" for concurrency and access control and coordination.* Certainly, the relevant tools are managed by database administrators, instead being directly available to end-users. As a result, users are protected from accidentally corrupting data in the first place, rather than having the opportunity to build something together in a workgroup.

Thus, coordination capabilities have to be an integral part of working tools within. Because these tools are utilised in a situational context then again, ethnographical aspects should be mind as well. Consequently, Schmidt and Bannon describe the challenge of designing computing systems that support the coordination of cooperative work activities as follows [1, p. 5]: *the system not only has to support the execution of 'the theory' built into the model, but also 'the practice'. That is, whatever needs to be done, under current conditions, to transform some normative construct ('plan', 'procedure', etc.) into contingent action.*

According to these findings, the perspective of CSCW suggests in-depth workplace studies as an essential and proactive part in the development of technology. When the organisational and social environment of cooperative work is analysed at first, it is much more likely to carve out an adequate solution, because everything is set in its real, practical context. Based on this introductory thoughts, the subsequent sections are going to shine a light on elementary topics of CSCW in a more detailed fashion now. By this means, *awareness*, *work practices and nomadicity*, *knowledge sharing* and *groupware* are discussed.

Awareness

Computer supported cooperative work in general is a very lively process with a simmering risk for unpredictable outcomes. One key factor to decrease uncertainty is found in the concept of *awareness*. Kolfschoten et al. [3] start with the notion that the coordination of activites is a significant problem for people working together. This is especially true in a coalescing, globalised world, where groups for cooperative work are formed and changed rapidly. Without any kind of support from software, maintaining an overview of activities is hardly possible, as well as the task of selecting the right contact person is. For this reason, the idea of *group awareness* was born.

Dourish and Bellotti define awareness [4, p. 1] as *an understanding of the activities of others, which provides a context for your own activity.* Splitting it up, Gutwin et al. [5] present four types of awareness: group-structural awareness, social awareness, informal awareness, and workspace awareness. Here, group-structural awareness is concerned with the different roles and and responsibilities in a team. Likewise, social awareness affects the social context of a group, while informal awareness summarises the general knowledge about a team. Finally, workspace awareness conveys information about the current status of a workspace, and which steps had to taken for achieving this state. Tam and Greenberg [6] enhanced this conception towards asynchronous awareness, which refers to questions about the past too. Thereby, involved persons should gain a more

exact understanding of the history of individual and group activities. This is particularly important if participants are not active in a team continuously.

Thinking practically, tools for workspace awareness can be as simple as user lists, displaying who is active at the present time. Another instrument are multi-user scrollbars, which illustrate the current focus of work of other users. Benefiting from the advantages of these techniques on the one hand, possible drawbacks do occur on the other hand. One major issue is found in the interpretation of *privacy*: how much information about the activities of users should be disclosed by the system? Is it reasonable that users can control what information is shared, perhaps at the cost the system's functionality? Commonly, these decisions are taken based on the idea of reciprocity. Single users get only the information from other users, which they reveal to them also.

Kolfschoten et al. highlight the potential of emerging technologies like *augmented reality* as well [3, p. 108]. Raising the question, to which extent these innovative approaches can be covered by the proven concepts of awareness support, and what limitations may occur. For example, the study of a crime scene investigation scenario [7] indicates some upcoming challenges: investigators transfering a virtual image of the crime scene to remote experts feel controlled by them, while the experts sense, that they are missing something in not being physically present at the crime scene. This observation connects to another popular problem with awareness mechanisms. People are often confronted with an inappropriate amount of information in a specific situation. Therefore, it is vital to contextualise awareness support with respect to the concrete task domain. How this can be achieved, is visualised by a practical example now.

Bardram and Hansen propose the *AWARE architecture* for supporting *context-mediated social awareness amongst mobile, distributed and collaborated users, such as hospital clinicans* [8, p. 117]. Figure 2.1 provides an illustration of all components, which will be explained in a more detailed way subsequently.

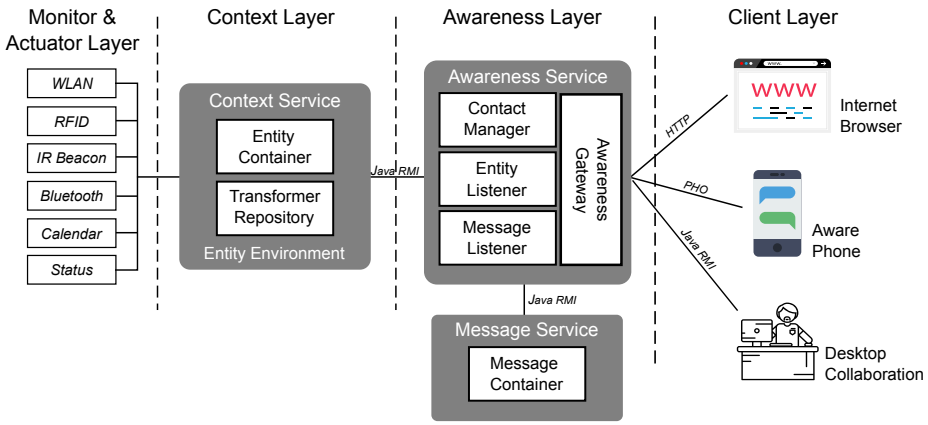

Figure 2.1: the AWARE system architecture [8, p. 117]

At the heart of it all, the concept states to combine CSCW system components for

establishing social awareness and ubiquitous computing components for acquiring context-awareness. In concrete terms, the architecture rests on a principle of four layers: the *client layer* uses the framework as a back-end system and comprises applications for the end-user. Whereas the *awareness layer* contributes the awareness service for literally keeping up an awareness of people, their social connections and how to get in contact with them. On the lower levels, the *context layer* and the *monitor and actuator layer* constitute the context-awareness infrastructure on which the entire AWARE framework is built [8, p. 117].

In an exemplarily sequence, the awareness service is utilised for managing information of subscribed users to the AWARE system first of all. It stores the associations between users for preserving social awareness, processing event-based notifications of changes within the contexts of users. For this reason, the contact manager offers a list of the contacts for each user. A message service allows contacts to exchange messages then. Whereas the context infrastructure monitors context cues in the environment of users. These cues can be displayed and prepared for further actions in the awareness service.

Summing it up, Bardram and Hansen set the following examples: person 1 relies on a list of contacts to be aware of the stored people' activities. If any context information with regard to person 2 is altered (e.g. changing the physical location), person 1 gets notified immediately. Likewise, person 3 receives a confirmation on its device, when person 4 reads a message sent earlier: all events are processed by the devices' applications properly and generate specific output like an updated user interface or notification pop-ups. [8, p. 119].

Work practices and nomadicity

Working is one central aspect in the life of humans. Throughout the evolution of mankind, performing tasks and activities for reaching certain goals has become a natural habit. While the main principles of work stay the same altogether, the methods do change regularly according to the developments of technology. Being closely related to the progress of the digital era, this is particularly true for information work, exemplarily in the service and education sector. Taking this domain as a starting point, Ciolfi and Pinatti de Carvalho [9] give weight to the potential of *nomadic practices* as a key property for such types of work.

Referencing the words of Davis [10], *work activities in certain professional contexts can and often must be detached from stable premises, and performed when and where it suits the workers' needs* [9, p. 119]. From the perspective of CSCW, this *modern nomadicity* leads to questions about the meaningful use of computing technologies to harmonise social and collaborative activities in and across different locations. Generally, De Carvalho identifies three different incentives for nomadic work practices [9, p. 121]:

- *choice*: e.g. if a person is willing to work at a certain location because of the comfort.

– *opportunity*: e.g. if a person decides to work at certain location because resources such as time or collaborators become available.

– *obligation*: e.g. when a person has to relocate to a specific site to carry out work, because certain resources are only available there.

Relating these factors of influence to information work, different conclusions can be drawn. First of all, it is the technology embedded in its environmental context, that determinates the degree of the personal freedom of work. Therefore, it is important to understand the possible advantages and disadvantages of CSCW for the respective domain. Setting a practical example, the permission to work at home raises the satisfaction of an employee by allowing flexible time management on the one side. But, on the other side, the results of the person's labour are not as scheduled as before, making it difficult for cooperative information workers to integrate it in the overall process.

Hence, one learning is that the application of CSCW always comes as a double-edged sword: CSCW per se is neither good nor bad, its actual value unfolds in the particular context. The crucial step manifests in finding the right balance for the interplay of the components (= humans and technology) of an organised system of work. Here, the concept of nomadicity is a key instrument to arrange elements at a level, which has never been achievable in the history of work practices before.

Knowledge and expertise sharing

Computer supported cooperative work as a whole unlocks great prospects for the exploitation of collective intelligence. When collaborating in a group, knowledge and expertise emerges from the synergies amongst people and their skill sets in theoretical as well as practical fashion. Based on this universal principle, Ackerman et al. are concerned with the implementation of concrete mechanisms for the sustainable sharing of knowledge in organisations [11].

To begin with, the idea of *knowledge sharing* points to the externalisation of knowledge by computational or information technology artifacts or repositories. On the contrary, the term *expertise sharing* is used for describing the ability of knowledgeable actors to get work done or to solve a problem *themselves*, without being supported by externalisations substantially. Following this differentiation, it is important to outline the core characteristics of the two concepts in the fields of CSCW.

For the matter of knowledge sharing, a repository model is the central aspect to build upon. Organisations create comprehensive repositories of what they know, including formal and informal information. People then can retrieve information by searching these sources of knowledge, or add information by asking and answering relevant questions. While this conception seems quite reasonable from a technical standpoint, it lacks several major cornerstones for the applicability in CSCW, as Ackerman et al. explain [11, p. 538].

In the first place, obtaining information for repository systems is recognised as a difficult challenge: the task of motivating people to add knowledge to a repository often fails on organisational issues. Balancing rewards and incentive structures with desired contributions is complicated by the competitive environment of organisations. Moreover, the social context of information reuse transcends the capabilities of repository models inherently: information has to be decontextualised by an author and recontextualised by the spectator for conceiving the information object in consequence. As an example, the specific location of a printer in an office may not be relevant, but the current network configuration certainly is. The more complex such informations are, the more difficult it is to find an appropriate description. Likewise, the recontextualisation itself could require a certain degree of expertise, since it is achieved as a situated, social action.

Another major challenge is discovered in the sustainability of repository systems: information objects and classification schemes have to be maintained in the long run, keeping everything up-to-date and consistent. On that score, people assess the process of adding meta-data costly in terms of time and labour. Therefore, categories with the lowest cognitive effort are chosen regularly, complicating information retrieval. Altogether, Ackerman et al. summarise with the statement, that the repository model encourages an objectified view of knowledge [11, p. 540]. Whereas considering information as a duality of process and object is concluded to be a more favourable approach.

This notion leads to the idea of *expertise sharing*: here, the priority is given to interpersonal communication of knowledgeable actors. Thus, the role of individuals as sources for the distribution of tacit knowledge increases in importance. Nonaka and Takeuchi qualify the term *tacit* in types of knowing, which are difficult or impossible to verbalise [12]. Hence, these competences are learned best via common experiences, where contact to other individuals is necessary for the full use of information.

Ackerman et al. take these thoughts to the next level by introducing two additional concepts, starting with the *community of practice (CoP)* [11, p. 547]. Generally, a CoP is specified as a group of people sharing a common practice by working together in a certain domain. For the viewpoint of CSCW, it is valuable to understand, how a characterising set of practices can be internalised within these teams: Lave and Wenger describe the process as a *legitimate peripheral participation* [13], where *participants in the group move from the periphery to the center, i.e. become increasingly knowledgeable* [11, p. 547]. More precisely, occasions for learning appear in working on similar issues in a similar way, when a common ground for cultivating expertise sharing is given. Once again, the overall meaning of context and situatedness can be identified in this patterns of thinking.

While the CoP refers learning opportunities to shared practices, the concept of *social capital* focuses on deriving collective abilities from social networks [11, p. 548]. In this case, the general attitude of people to help each others as well as the relational dimension of such interconnected systems are seen as basic resources for expertise sharing. Nahapiet and Ghoshal emphasise three core properties of social capital in consequence [14]:

– *structural opportunity*: who shares knowledge and how is it shared (infrastructure)?

– *cognitive ability*: what is shared?

– *relational motivation*: why and when do people take part in knowledge sharing?

Dealing with this questions, Ackerman et al. infer that *contextual knowledge is required to select the right source, strip off the information from its original context and then re-contextualize it according to the situation at hand* [11, p. 548]. At the heart of it, this knowledge is often tied to being aware or asking other people. For this reason, the main task of computerised systems in expertise sharing can be summarised in the seeking of appropriate people: using computational power and algorithms to connect the right persons at the right time, providing the most efficient way for the exchange of knowledge in a particular situation.

Groupware

Starting with a universal definition, *groupware* denotes application software for the practical realisation of the theoretical findings in the research domain of CSCW. General examples include systems for the management of knowledge, projects, workflows, contents and documents. As a common characterisation, these systems are used by groups of people for collaboratively working together towards specific goals. In this respect, the success of such technologies depends largely on the appropriate utilisation over time, adapting the groupware to the local contexts in the resepective organisations. Bansler and Havn discuss this observation as a key issue of CSCW [15].

In most cases, the implementation of groupware systems fails on a misinterpretation of installation and use by the responsible stakeholders. On one side, the required *attention, support and resources* for the introduction of groupware systems get underestimated [15, p. 55]. At the same time, users tend to re-invent the artifacts by developing novel uses [16, p. 367]. Accordingly, groupware is delineated as an *extreme fragile class* of technology for two reasons [17, p. 6].

First, these comprehensive solutions are always in competition with existent media [17, p. 6]. Bansler and Havn state, that users are not passive consumers of media. It is their choice to select the medium that they are most comfortable with for accomplishing a task at a certain point of time. Thus, new technologies for cooperative work are often introduced with the *handicap of the unknown*: when users experience problems with a novel system, they just switch back to approved mediums for continuing their work in an established fashion. Picking up this thought, humans are more concerned with the question of *how* they can deal with a programme personally, than *what* they possibly could achieve with it. In some way, this can be classified as a *protective mechanism*, which is quite common in decision making generally.

Second, agreeing on corporate conventions is vital for governing communication and collaboration [18]. Therefore, groupware needs consensus on the means of operation to be effectively deployed in the long term. Related to practice, an explicit and progressive

adjustment amongst technology and organizational context is required [19]. For example, this involves the training of users, the adaption of existing procedures, and refinement of the applied system itself.

All in all, Barner and Havn suggest to answer these challenges with the concept of *sensemaking* [15, p. 56]: mediating persons have to make sense of the technology by understanding it in conncetion to the specific, local context. Thus, from the designers point of view, groupware has to flexible and customisable, as well as constraining and enabling at the same time. Consequently, it should be developed in direct accordance with the future users, avoiding possible sources of error in advance and delivering an individually aligned product.

2.1.2 Interaction Design

Interaction Design can be defined as *the practice of designing interactive digital products, environments, systems, and services* [20, p. 31]. First and foremost, it is concerned with analysing the behaviour in human-computer interaction: which ways of interaction are possible and reasonable, how do humans react to different characteristics of design, what lessons can be learned for developing applications in a certain context as a result? In order to convey a compact impression of this wide-ranging subject, some selected topics in the fields of *information-based working* are considered in the following discourse.

Computer- versus paper-based tasks

Even though a world without computers is hardly imaginable anymore, the utilisation and exploitation of digital systems actually is a relatively recent achievement in the era of humans. Regarding the storage of information, people were used to work with physical materials for the longest time of their existence. Therefore, it is vital to understand, how computer-based tasks are equivalent to paper-based ones and which determinants have an impact on the usage of these two media, as Noyes and Garland discuss [21, p. 1352].

First of all, the superior relevance of *cognitive indicators* for the evaluation of task-specific performance is emphasised [21, p. 1361]: using partial indicators like reading speed and accuracy alone does not seem to be sufficient due to the complexity of the examined processes. For example, the nature of the computer screen imposes unique requirements to the human perception inherently. This concludes a different way of memory processing in comparison to other visual inputs. Subsequently, Wästlund et al. figure out, that paper-based working in general leads to a better consumption and production of information [22]. Reasons are found *in a greater level of experienced tiredness and increased feelings of stress* when working with virtual desktops [22, p. 389].

Noyes et al. address to this argumentation by presenting a comprehension task on either computer or paper and measuring cognitive workload as well as performance thereby [23]. As a result, significantly more workload on the effort dimension is encountered for the computer-based task again. Moreover, another discovery reveals in the applicability of cognitive workload measurement for distinguishing small differences in processing. This

in turn can be especially valuable in the context of more elaborated tasks, where problem solving or decision making is elementary.

Resuming their findings, Noyes and Garland stress the idea of *equivalence*: computer and paper-based working will hardly ever be conceived the same by people, as *two different presentation and response modes are being used* [21, p. 1371]. But it is plausible, that humans will increasingly be able to take advantage of this diversity by integrating the benefits and omitting the drawbacks of both domains in future. Whereas this progress goes hand in hand with the exploration and comprehension of the humans' cognition, as the improvements of the last decade have shown.

Problems and practices with desktop systems

In a more practical approach, Ravasio et al. focus on the problems human users experience in their daily work with computers [24]. To begin with, the opening statement raises the issue, that *the desktop metaphor as the de facto standard UI requires memorizing conventions and procedures instead of interacting intuitively and in a straightforward manner* [24, p. 157]. For the detection of concrete evidence, the study is based upon the aspects of *document classification* and *document retrieval* in consequence [24, p. 157].

As a first step, observed classification practices are discussed with regard to four elementary facets in the organisation of hierarchical filing systems [24, p. 163]:

- *archiving*: users have a strong need for archiving files. They rely on their archives as an information source, and do invest real effort in creating meaningful file systems structures accordingly. In addition, naming files in a reasonable way helps users to recognise their contents at first sight.
- *maintenance*: users integrate maintenance in their practices regularly. For example, they sort documents when a project ends and retain only the valuable ones.
- *use of hierarchical structures*: users create new (sub-)folders for keeping an overview. Document storage is seen as a continuous process with flexible, non permanent structures.
- *classifying information*: users spend substantial cognitive effort for the classification of documents, as well as the labelling of folders and documents.

Analysing these behaviours, Ravasio et al. are able to spot three main requirements of users for an improved operationality of personal computer systems [24, p. 170]:

- *Divide information belonging to users and to the system in a suitable manner*: users could profit from a clearer separation of their own data and system-owned data by reducing the cognitive workload for selecting appropriate folders or structures.

- *Provide small but potentially extremely helpful tools to manage information*: each automated function, that serves a certain (otherwise manual) task reliably, saves time and energy to the benefit of the user.
- *Integrate rather than separate information*: the natural trait of *networked information access* by human users is not supported sufficiently in the basic conception of (hierarchical) file systems.

Subsequently, the outcomes of the study confirm the presence of the three generic types of information, i.e. temporal, working and archived information [24, p. 170]. Furthermore, it is assumed, that *all the efforts invested in organizing, naming and maintaining the hierarchical file system structure are aimed at (1) engraving the information's content and context into the system, and (2) providing an overview in a single glance* [24, p. 170].

In their second subject, Ravasio et al. attend to the practices of document retrieval in virtual desktop environments. Most commonly, search activities start, when a user is not able to locate a specific piece of information immediately. Therefore, two modes of searching come into question [24, p. 170]:

- *Logical search*: the motivation for utilising a system-offered search tool is closely bound to the complexity of its usage. This seems quite natural, as humans tend to prefer the most comfortable solution for a given problem, minimising the cognitive effort.
- *Manual search*: looking for documents manually is the conventional strategy of users. They trust in *their own intelligence, memory capacity, and contextual knowledge* more than anything else. Thus, patterns for searching include *direct access*, *semantic proximity* and *exhaustive search*.

Drawing their conclusions, Ravasio et al. try to elucidate the reasons for this behaviour: overall, humans do realise on their part, that the application of search tools is as demanding as performing a manual search, cognitively and mechanically [24, p. 173]. More importantly, the *direct access strategy* turns out to be effective in most cases, approving its value on a regular basis. In addition, searching manually allows users to remediate their own knowledge of the viewed information domain. Contrary to these positive perceptions, the application of search tools is often associated with negative previous experiences, when something did not work in the desired fashion. Finally, these streams result in the need for a thoughtful design of document-retrieval solutions in general: it is more sensible, to adapt technology to the requirements of human users, than the other way around.

Task management of information workers

Pursuing the words of Bondarenko et al. [25], information workers are embedded loosely in a highly flexible and rapidly changing field of data processing, where the acquisition

of new skills is a constant demand. Digital workplaces have extended the space of possibilities dramatically, overcoming physical restrictions in many respects. Information is stored in a large number of formats and locations, allowing to be accessed from a variety of devices, at any point of time. Ultimately, this leads to the increase of communication load and data fragmentation, as well as incompatibilities between applied tools are a consequence. Therefore, conceptual knowledge of supporting document and task management of information workers is an elementary factor to solve emerging issues.

Establishing the context, Bondarenko et al. refer the definition of a task to be *an action or a series of actions towards some well-defined end, i.e., a change of state* [25, p. 469]. Whereas the tasks of information workers are often initiated by and do result in documents. Hence, it is reasonable to consider document and task management as two integral parts of one process [25, p. 470]: the task is at the center of the process, while various types of documents are utilised as resources for completing the task. This in turn suggests a personal perspective to the overall procedure, leading to the determination of requirements for such a digital management system. Based on the findings of three studies, four layers and their respective demands are identfied, as displayed in Table 2.1:

Task decomposition level	Supported process
Knowledge work	Document and task management
Unstructured workflow	Nondeterministic task flow
Multitasking	Frequent switching
Task suspension and resumption	Stable state creation

Table 2.1: four layers of task decomposition [25, p. 471]

In accordance to the wording of Bondarenko et al., the requirements are outlined as follows [25, p. 473]:

Supporting task management in knowledge work

- In order to preserve time or mental effort that should be devoted to primary tasks, a document-management system should require as little effort as possible to set up and maintain (least-effort principle).
- It is advisable to base the system on system(s) in use.
- A variety of existing ways of organising documents and tasks should be supported by the system, as opposed to imposing its own.

Supporting a nondeterministic task flow

- Task management in an unstructured workflow should be supported by a document management system by providing ad hoc collections of task-related documents as a representation of tasks.

- The system should allow to arrange documents/collections of documents along with the task flow and supply an overview of task-related collections.
- The system should make task-relevant representations across collections of documents possible.
- The system should provide an overview of task-related document collections and their state without additional interaction by the user.
- Implicit planning and prioritising among tasks or task-related document collections should be enabled by the system.

Supporting frequent task switching as a result of multitasking

- The system should support task suspension and resumption in the environment.
- The system should support the preservation of the current state of environment, reflecting the task state.
- The system should allow user-defined changes in the representation of the current state of the task.

Supporting task suspension and resumption by encoding the task state

- The system should support encoding of semantic judgments with respect to the task state in cues available in the digital domain.
- Extract and visualise relevant semantic data about a document's usage: interaction possibilities, information presentation, ownership of a document, intended audience, period of use, frequency of use.
- The system should provide extracted information on a granularity level required by the user and in user-specified terms, and allow to customise the level of granularity.
- The system should allow the user to show and change higher-level concepts and information about future use for task-related documents or document collections, such as importance, confidentiality and action demand.
- Concerning the required user effort for offloading and uploading information about the task state to and from the environment, the system should be adaptable and request as well as present only the necessary information.

Eventually, it can be summarised, that the promotion of the *least-effort principle* is essential to any (document) management system. Because the effort of these systems increases with the number of tasks an information worker carries out at the same time, supporting task switching and keeping everything up-to-date [25, p. 480]. Hence, it is

reasonable, to take these possible problems already in the planning and development of such a system into consideration, rather than reacting to them costly when it is deployed. Accordingly, the system should be designed from a perspective, where the perceived user effort for operating it is in the focus of interest [25, p. 480]. By this means, time and mental capacity of information workers can be saved and utilised for other tasks, increasing the overall productivity while retaining personal contentment.

2.1.3 Information storage and retrieval

Within the course of evolution, the production of information has become a natural characteristic of mankind. As a result, humans are engaged with collecting and storing information since the beginning of their existence. Consequently, the idea of organising contents for profiting from their retrieval on a later occasion is a long-serving concept, and only the ways and means of accomplishing it have changed. By the development and use of innovative media, people are able to increment the amount of accessible information significantly. Up to the point, where the artless capabilities of individuals are not sufficient for rationally processing the loads of data anymore. This again can be described as the central question accompanying the enforcement of digital systems: how is it possible to exploit the manifold opportunities of digitalising information, while avoiding the closely-related risks?

Manning et al. approach to this assignment in supplying the following definition: *information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)* [26, p. 1]. For a further distinction, information retrieval systems can be divided in three categories by the scale of their operations [26, p. 2]:

- *Web search* covers searching mechanisms over a myriad of documents stored on millions of interconnected computers.
- *Personal information retrieval* is concerned with finding and classifying information on personal computers.
- *Enterprise, institutional, and domain-specific search* focuses on retrieval within collections like corporation's internal documents. Here, documents are stored on centralised file systems commonly, as well as a couple of responsible computers establishing the search over the collection.

Seventeen theoretical constructs of information retrieval

Jansen and Rieh try to identify the underlying theoretical constructs in the fields of *information searching and information retrieval* for building a general foundation [27]. As a starting point, two nested frameworks distinguish between *behaviors when people are using information systems, and systems that support, afford, and enable the behaviors* [27, p. 1518].

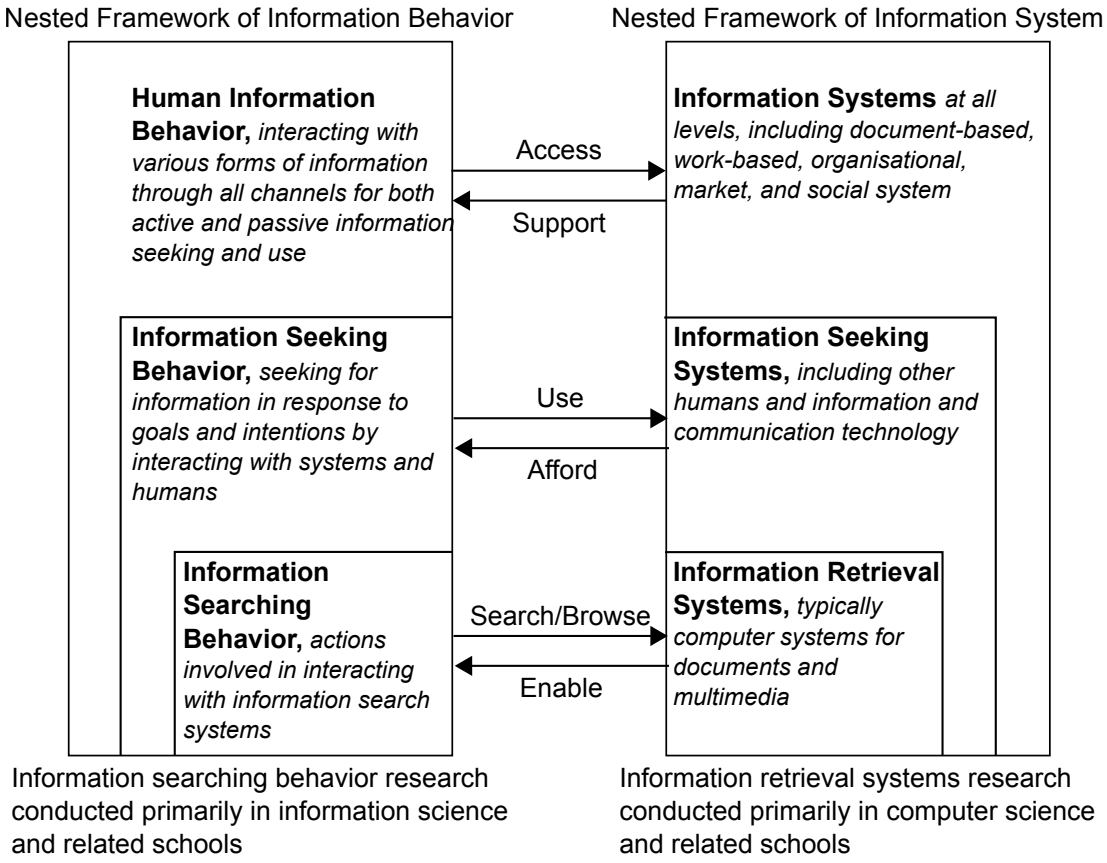

Figure 2.2: framework of human information behavior and information systems [27, p. 1518]

Based on the formulation of Figure 2.2, Jansen and Rieh are in a position to derive possible constructs from a broadly conceived literature review which also touches the domains of communication, learning and economics. Then, it is feasible to extract all constructs describing substantial concepts in the domains of information searching and information retrieval [27, p. 1522]. Ultimately, Jansen and Rieh characterise their designated seventeen constructs in the following manner [27, p. 1522]:

- *Multiple definitions of information*: information is a fundamental concept with a spectrum of definitions.
- *Hierarchical relationship of information*: viewed from the perspective of an information searcher, a hierarchical relationship among data, information, knowledge, and wisdom can be identified.
- *Perceived benefit of information*: it is unlikely, that an information system will be utilised, if it is more troublesome for a user to have information rather than not to have it.

- *Relevance*: relevance is an elementary criterion in the process of evaluating the performance of searching or retrieval.

- *Information representation*: it is possible to represent information algorithmically by the sum of its attributes.

- *Information ranking*: information, that addresses a demand for information (as expressed by a query) can be ranked in order of some predicted measures (e.g. relevance, usefulness, freshness, authority, etc.).

- *Document similarity*: if a document responds to a given query, similar documents will be relevant too.

- *Uncertainty principle*: users try to resolve uncertainty in knowledge by the process of information searching.

- *Principle of least effort*: users of information systems will conduct a series of actions and operations that requires the least perceived effort to locate the desired content.

- *Searching as an iterative process*: a searching process consists of several steps of interaction with information systems and is completed when the particular need of information is satisfied.

- *Interaction*: interactions amongst users and systems are elementary in the context of information searching and retrieval.

- *Information provision*: in the course of a certain task, users take advantage from provided information to accomplish it.

- *Preference of channel*: when attempting to obtain information, people have preferences of media and technologies.

- *Information obtainability*: the use of information is directly proportional to how easy it is to obtain.

- *Query*: the information need of a user is formulated as a question and converted into a query that is accepted by an information retrieval system subsequently.

- *Neutrality of technology*: the presented content of information retrieval systems is unbiased.

- *Memex vision*: technology is the solution to make information available to people.

2.2 Document management systems

Recapitulating the findings of Chapter 2.1, human society created an overwhelming world of (digital) information, which in turn can only be mastered with the development and utilisation of supportive artificial tools. Thus, the key to success reveals in the applicability of such technological frameworks: people need electronical systems, that incorporate their natural behaviour and cognition. Based on this universal notion, it is possible to carve out individual solutions for specific domains in order to accomplish certain tasks.

Document management systems (DMS) on their account are designed for the assistance in information-based working environments. Accordingly, the capture, organisation, storage and retrieval of documents is addressed to begin with [28, p. 13]. In the course of the following subchapters, a more detailed explanation of the term and its concept is given. For this purpose, a brief overview of the history of DMS introduces the topic. Then, the actual scope and essential features of DMS are discussed. Finally, their areas of application get examined, complementing the overall picture with the presentation of some practical examples.

2.2.1 History

Although humans probably were not able to fully unterstand it from a scientific point of view, they already realised the potential of information storage millennia ago. Therefore, first, yet primitive document management systems are already found in the writings on the walls of caves. Progressing in skills and technologies, the methods advanced to more elaborated scroll systems and books over a large period of time. Whereas these changes did have significant impact back then, the real breakthrough of document management systems proceeded in the course of the last centuries, as Mancini outlines in eight theses [29, p. 30]:

The filing cabinet

Invented in the late 1800s, the filing cabinet allows to store paper documents in file folders. As simple as this concept may be, it eases the wole process of managing documents considerably by establishing a replicable local order.

The server

While proving to work well, the overall idea of the filing cabinet is delimited in terms of storage space: the more documents are filed, the harder it is to retrieve particular documents and sort new ones. This constraint is surmounted by the emergence of computing: distributed client/server architectures make it possible to store documents electronically, transferring it to a digital world of theoretical infinity.

The personal computer

With the prevalence of personal computers, individual users are empowered to take part in the whole process of document management. Still, this innovative step leads to deficiencies in the way documents are organised and secured at the very beginning.

Electronic document management systems

Evolving from the unstructured nature of distributed personal computers, electronic document managements systems (EDMS) gain popularity in the 1980s. Again, it takes time to shape these complicated systems in the sense of user-friendliness and general appropriability. This process reaches to the present, pursuing the goal of complete collaboration amongst variously skilled people in different roles.

The search engine

Facing the problem of digitally scattered documents, search engines are integrated into DMS as a basic cornerstone. By this means, the sophisticated task of information retrieval is simplified eminently, enabling users to find any document in the system within seconds.

The scanner

Settling in a world of paper documents, EDMS do profit tremendously from the improvements of computer scanners in the mid-1980s. As a result, informations from previously separated channels can be integrated into one holistic system: ensuring flexibility in workflows and supporting the different types of humans' awareness.

The cloud

By the composition of a global internet-structure, organisations and their members do not have to host and access data locally any longer. Instead, the whole DMS infrastructure can be decentralised in the *cloud*, leading to concepts as the software-as-a-service (SaaS) model: here, software is ready to go on demand, permitting users to create, share and edit documents uncommited to time or place. Additionally, this innovation reduces the expenses for DMS overall.

The smart phone

From the users' perspective, smart phones and mobile devices in general come in handy to take advantage of electronical document management: assembling a comprehensive scheme, the computer extends the natural capabilities of the human by processing loads of information, and provides the essential documents *at the fingertips* in return.

This notion probably summarises the potential future of document management best, as the term is in a constant state of flux. Ultimately, it is all about information, and how

people are able to perceive it in a specific context. Therefore, technology is only a means to an end and a manifestation of its time.

2.2.2 Scope

Despite their volatile nature, document management systems share some elementary characteristics and are classifiable to the greatest extent. At the heart of it, Kampffmeyer defines document management as the *database-supported management of electronical documents* [30, p. 3]. For a better differentation to the techniques of the past, this process is delineated as *electronical document management (EDM)* in consequence.

Principal features of EDMS are visualised folder structures, checkin/checkout, versioning, manipulation of metadata and the use of search engines [30, p. 3]. This scope of *classical EDMS* is determined by the ISO-norm DFR 10166 roughly and illustrated in the following example of Kampffmeyer:

One major application of EDMS is found in the electronic file, where related informations from various sources are brought together to a single instance [30, p. 3]. Hence, document management in its closest sense is concerned with nothing more than the aggregation and subsequent presentation of information in documents. Whereas this functionality is embedded in a more complex, dynamic environment of *enterprise content management* then again:

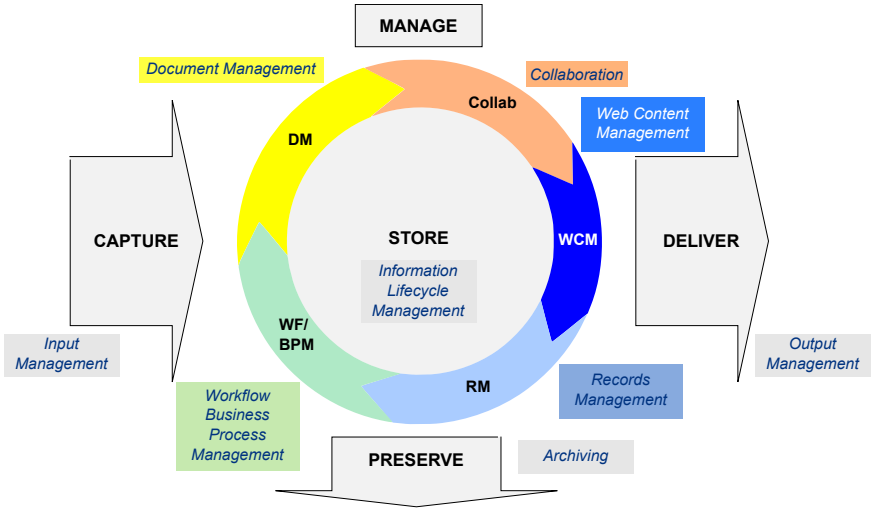

Figure 2.3: the ECM components model, Project Consult [31, p. 24]

Commonly, the term *document management* as such is mixed with the displayed concept of *enterprise content management* in Figure 2.3 on a number of occasions: in this broader meaning, EDMS are also concerned with the further processing of documents. Here, documents are understood as *enabling mechanism* or *currency* in a lifecycle of collaboratively achieved tasks. Either way, it boils down to the *document* being the basic component to focus on.

Electronical documents

Kampffmeyer determines the *electronical document* as *any types of unstructured informations, which are encapsulated in a file as an enclosed entity, existing in a data management system* [30, p. 5]. Thus, there are two different options of contributing documents to an EDMS:

- Incorporating documents, that are created with the help of computer systems (e.g. a textfile or a table from a database).
- Incorporating documents, that are converted from an analogue to a digital form (e.g. a scanned image or a video transferred from a camera).

Accordingly, documents can be assembled from one or more single objects (pieces of information): this diploma thesis for example is composed of text, graphical images, references and much else. On that score, it is possible to carry out a further categorisation of documents [30, p. 6]:

- *Elementary components* consist only of one type of object (e.g. a textfile, an image).
- *Compound documents* include several types of objects (e.g. text, tables, hyperlinks).
- *Container documents* as superior items can be used to store elementary and/or compound documents as well as additional information like metadata.

Summing it up, the main characteristic of any document is its contextual value: human users do process (electronical) documents in specific situations, attended by different expectations and intentions. Likewise, EDMS as related tools of choice are only a part of an overall setting, and connected to other influential systems thereby. Thus, it is important to assess EDMS also by their modular capabilities, integrating with emerging environments: estimating the real potential that an EDMS can add to a given set of conditions.

Integration needs of electronical document systems

To start with, Leikums stresses the particular meaning of documents [32]: all informations contained by a document were created for a purpose and do belong to a certain *area of activities* (in an organisation) accordingly. Hence, it is reasonable to allocate documents to these instances, facilitating the interoperability between different types of information systems. By this means, the circulation and life cycle of documents is adapted to the particular processes of work [32, p. 194].

Illustrating this idea by a concrete example, an organisation could occupy a department for financial affairs, where human employees have to create reports on a regular basis.

In this context, only a small amount of the organisations' overall documents are required, as well as a combination of typical operations for processing them. Consequently, human users do profit from an individually attuned solution, providing the information and methods they need in the given situation. This suggests to build information systems from *bottom-up* rather than *top-down*: combining modular systems, which have proven to be efficient in their domain, to an overall information system.

Altogether, Leikums concludes in the finding, that three groups of systems, whose integration with the EDMS is possible, do exist [32, p. 203]:

- systems, whose integration is mandatory or very necessary
- systems, whose integration is advisable and would improve their administration and usage, and enhance processes within the institution
- systems, whose integration is necessary only in particular cases or for particular institutions

As a result, it is notable, that EDMS integration options with other information systems should already be considered and applied in the development phase of the document management system [32, p. 196].

2.2.3 Essential features

In accordance to the principle of modularity, EDMS do share some key properties, which qualify them as a distinct module to fulfil a particular scope of functions. Raynes identifies these *real benefits of an EDMS* in the provision of the following features [33, p. 304]:

- *a process for check-in, check-out*: a mechanism, that permits only on user at a time to modify a document, figuratively a lock.
- *version control and audit trail*: techniques, that constitute the recording and monitoring of all changes made to a document in the course of time.
- *document review*: allow users to add comments to a document without changing the document itself indeed.
- *security processes*: controlling the groups of users that can access documents, based on the particular context.
- *organisational processes*: methods of collecting documents in related groups, typically folders.
- *free-text searching*: means for identifying and retrieving documents as a function of the text they contain.

- *metadata*: enabling the acquisition of information associated with the document, such as the author, the title or the date it was created.
- *workflow*: establishing a controlled way of guiding documents from one user to another.
- *imaging*: methods for converting paper documents to an electronic format, e.g. scanning.
- *publishing*: make the compilation of documents in consistent collections possible and distribute them to their target clients.

2.2.4 Areas of application

While the implementation of any EDMS rests on theoretical foundations overall, some crucial insights still can be acquired in the course of its practical realisation only. Paying attention to this thought, the following subsection discusses possible areas of application, and moves the focus to the main purpose of this diploma thesis thereby: developing a software prototype for the management of student records at the Vienna University of Technology.

Kampffmeyer points out the high effort of establishing EDMS in organisations above all [30, p. 11]: on the one hand, financial expenses for the implementation and maintenance are indispensable, while on the other hand the cognitive resources for administration should not be neglected too. Thus, the installation of an EDMS is a decision in principle and changes the whole environment of its application, initiating a process, that can not be revoked easily.

Giving credit to this concerns, the relevant question in turn is, how enterprises or academical institutions are able to profit from EDMS in concrete terms. Sprague therefore identifies three generic functions in organisations, that are especially receptive to EDM [34, p. 40]:

- Application areas, that depend on the document as the primary mechanism for getting the work done.
- Application areas, that are susceptible to emerging document technologies inherently.
- Application areas, that can generate business value from the utilisation of EDM technologies and approaches in general.

Based on this definition, Sprague attempts to carve out a more process-orientated perspective by relating the functions to the common tasks of organisations. As a result, seven generic categories of EDMS applications producing value in supporting the organisation can be derived [34, p. 40]:

- Improving the publishing process
- Supporting organisational processes
- Supporting communication among people and groups
- Improving access to external information
- Creating and maintaining documentation
- Maintaining corporate records
- Promoting training and education

Students management at Austrian universities

Addressing Austrian universities as the primary domain of interest, the *federal act on the organisation of universities and their studies* suggests the use of EDMS by the following statutes at least [35]:

§ 2. The guiding principles to be observed by the universities in pursuance of their objects are:

VII: national and international mobility of students, graduates, and university scientific and artistic staff

VIII: collaborative relationships between members of the university

XII: economics, economical and expedient management of finances

§ 3. Within their sphere of action, the universities fulfil the following tasks:

VI: internal co-ordination of scientific research (and the advancement and appreciation of the arts) and teaching at universities

VII: promotion of domestic and international co-operation in research and teaching, and the arts

VIII: promotion of the use and practical application of their research findings, and of community involvement in efforts to promote the advancement and appreciation of the arts

X: maintenance of contacts with graduates

XI: information of the public on the performance of the tasks of the universities

Examples and challenges of document management in practice

This subchapter presents a couple of implemented EDMS solutions in order to convey a more practical impression of the previous findings altogether. It is debated, how EDMS are realised with regard to their respective context, and which possible challenges have to be taken into account overall.

Ranabahu et al. introduce *Kino, a set of tools that streamline the document management process in life science domains* [36, p. 1]. To begin with, the scope of the system is outlined by two use cases in the context of biologists: scientific workflows and document annotations.

While the process of genome sequencing *determines the complete DNA sequence of an organism's genome*, it is not sufficient for *providing any information on genes, their location, or their functions* [36, p. 3]. To achieve this goal, biomedical researchers have to analyse the genomic sequence individually with the help of data repositories. Thus, the tasks of such an analysis are based on web services commonly and can be part of a service oriented workflow. One major handicap in this regard appears while searching for services in catalogs, where imprecise terms and tags make it difficult to find the appropriate service in a reasonable amount of time [36, p. 4].

Changing the focus to the case of document annotation, a genome database is at the centre of attention. Concerning this matter, a collection of genomes needs to be updated and annotated constantly by the collaborative effort of several groups of scientists and bioinformaticians [36, p. 4]. Here, the main challenge arises from the *lack of integration across tools*, burdening the maintenance of documental descriptions.

Trying to resolve the mentioned problems, Ranabahu et al. propose a modular system called *Kino*, illustrated in Figure 2.4:

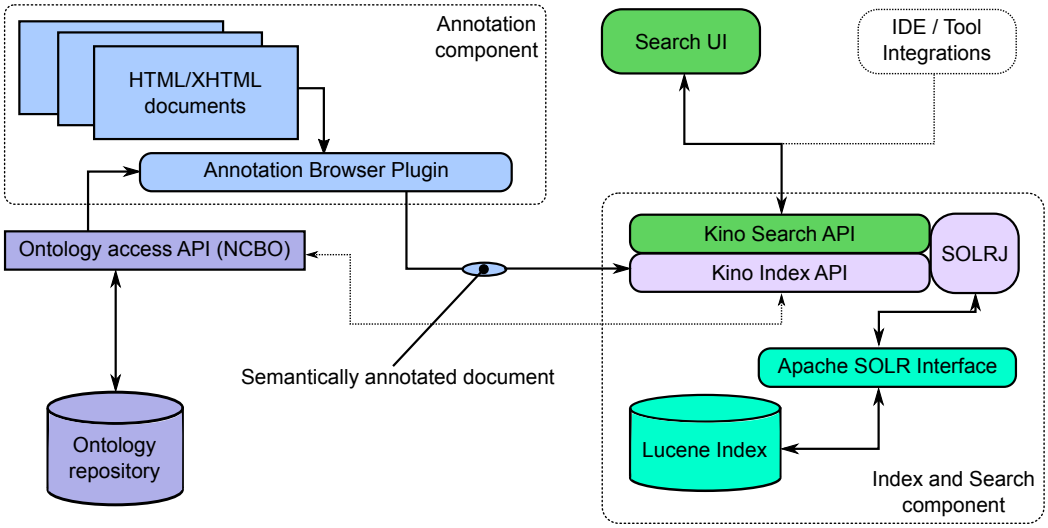

Figure 2.4: system architecture of Kino, major components [36, p. 4]

Generally, the depicted system is based upon a workflow of three steps [36, p. 5]:

- *Annotation*: users are able to provide annotations via various tools (e.g. browser plugin, website or an integrated development environment). When the annotations are added, the augmented document is submitted to the indexing engine.
- *Indexing*: Apache SOLR is used for indexing the documents. It can be installed independently and offers multiple interfaces for client programs.
- *Search*: a Javascript driven Web UI offers a typical search engine interface and allows to filter the results.

Jadid and Idrees report on the application of EDM at a construction site [37]. Starting with the observation, that civil engineering traditionally requires a quantity of documents like drawings or technical specifications. Commonly, the content of these documents slightly changes over time and needs to be distributed to all relevant staff members subsequently. Enhancing the long-established hardcopy based change management, EDMS can contribute particular advantages in this context [37, p. 2]:

- *Changing the way of propagation*: project engineers are able to build customised networks for exchanging documents electronically, controlling the flow of information.
- *Faster document-centered processes*: parallel document reviewing and approval saves resources (e.g. reducing the time of documents' cycles; no need for physical presence in a meeting at a certain location).
- *Flexible alteration of documents*: documents can be updated, archived and restored in a central repository, maintaining a general knowledge base.
- *Availability of documents*: project participants are able to retrieve documents individually and independent of time and place.
- *Security of documents*: unauthorised document access is prevented by the use of a role-based security mechanism.

By this means, Jadid and Idrees identify the situational availability of relevant information as the critical factor of success in the dynamic environment of construction areas: several entities are involved throughout the entire process and do depend on diverging versions of documents for their respective tasks.

On the scale of things, the specified examples of EDMS applications impart some additional conclusions: first of all, EDM as a concept has a high scalability and can be utilised in nearly any domain due to its ability of information representation. Nevertheless, this does not imply that EDM should be utilised in each possible case. Hence, the real

challenge is found in the anticipation, if and how an EDMS is able to add value in a concrete area of application. Individual approaches help to minimise risks, but require further effort on the other hand. Thus, the whole subject of EDMS in fact is still evolving and many facets have yet to be explored.

2.3 Agile user-centered processes

The development of a software (prototype) for the management of student records at the Vienna University of Technology suggests a more classical interpretation of EDMS in the first place. Information is stored, retrieved and manipulated by users in different roles, performing their individual tasks as parts of an collaborating environment. While many EDMS would claim to cover this *basic functionality* out of the box, several essential details can only be identified and adapted by the initial inclusion of the client users indeed. For this reason, it is important to discuss what an appropriate development approach could look like.

Salah et al. highlight the potential of integrating *agile development processes and User Centered Design (UCD)* [38, p. 1]. On the one hand, agile software development methods are designated to overcome the perceived limitations of plan-driven methods by equilibrating the need for a process [38, p. 1]. User Centered Design then again *is a set of techniques, methods, procedures and processes as well as a philosophy that places the user at the centre of the development process* [38, p. 1]. Thus, combining these two strategies helps developers in conceiving the needs of the potential users of their software and equips them with a flexible tool to incorporate the volatile requirements in an iterative development process [38, p. 1].

Consequently, Salah et al. conduct a systematic literature review (SLR) of challenges, practices and success factors in the domain of agile user centered design integration (AUCDI). The research is based on the alphabetically listed keywords in Table 2.2 [38, p. 1]:

Agile	UCD
Agile Development	Human-Centred
Agile Method	Usability
Agile Practice	Usability Engineering
Agile Project	User Centred Design
Extreme Programming	User Experience
Scrum	User Interaction
	User Interface

Table 2.2: keywords for SLR process on AUCDI [38, p. 2]

As a next step, the resulting findings can be summarised in categories and analysed accordingly [38, p. 5].

Lack of allocated time for upfront activities

First of all, Salah et al. notice that the usage of agile methods limits upfront planning activities, because one key characteristic is found in the ability to react on changing requirements. Furthermore, incremental agile development can lead to a dissected *feature by feature* development, which may result in a fragmented user interface that misses an overall structure and vision.

In order to cope with these possible issues, a seperate pre-development stage of *upfront design* is recommended. This stage can be utilised for facilitating a holistic system view by deriving requirements, understanding users, their goals and the context of application.

Difficulty of modularisation/chunking

Design chunking is understood as dividing design into smaller pieces, i.e. design chunks. These modular components are used for adding further elements to the overall design in a stepwise fashion. Concerning this matter, difficulties are identified in the determination of the appropriate amount of interaction design work per iteration, as well as ascertaining the chronological order of design chunks per se can be a problem.

Feasible solutions are described in the definition of well attuned design goals, where large or complex features can be published in single releases, chunking design into concrete features. Besides, it may be advisable to move activities related to the user experience to a later point in the development life cycle.

Performing usability testing

Usability testing in general is concerned with the measurement of the performance of average users processing prepared system tasks. Challenges can arise in the methods of usability testing, scheduling usability testing, accessing users for usability testing and the costs of running usability sessions.

To tackle these problems, Salah et al. suggest prearrangement for user research, applying low fidelity prototypes and remote ways of usability testing. Apart from that, valuable opportunities for testing the software's usability do occur when iterations and releases are finished. Supplementary, it can be sensible to utilise an existing user pool, that contributes usability testing in the role of a development partner.

Lack of documentation

Salah et al. mention, that agile approaches strive to produce a minimum of documentation overall, though documentation is vital for the estimation and implementation efforts in turn, i.e. properly integrating agile development and user-centered design. Moreover, the lack of a suitable requirements documentation can lead to confusion regarding deliverables.

Resolving this shortage of documentation, the application of wikis, use cases, scenarios, personas, sketches, wire frames and design patterns is advisable.

Bringing Chapter 2 to a conclusion, it can be summarised that the process of document management is more tailored than ever before. While the capabilities of EDMS advance steadily, complexity increases as well. Thus, the growing number of adjustable variables for an operable system raises the probability of failure in many contexts. Therefore, it is important to rely on a solid foundation of knowledge, which allows to continuously assess and reflect the actions to be taken.

As well as the presented theoretical background is not exhaustive at all, it should comprise some valuable insights into the addressed topic of document management and its surrounding environment. Based on these findings, a project description concerning the development of a software prototype for the management of student records at the Vienna University of Technology can be established in Chapter 3 now.

CHAPTER 3

Project description

This chapter describes the planning and design of the software prototype in advance of discussing its practical implementation in Chapter 4 then. For the sake of elaborating a final draft, several consecutive steps have to be taken into account altogether: starting with the *initial situation* at the Vienna University of Technology, general parameters can be determined and refined in the concrete *requirements* of the new DMS accordingly. An adequate *market sounding* is carried out in consequence, leading to the selection of an appropriate basic system as groundwork for the prototype. Ultimately, the comprehensive *concept* of the novel software will be presented.

3.1 Initial situation

First of all, the relevant information for constituting the project's context is gathered by meeting persons in charge at the Vienna University of Technology and reviewing internal documents, which are provided additionally. By this means, the actual scope of the work is carved out in a coordinated activity with the responsible stakeholders. Clarifying, *what* is the particular problem in the current situation, and *which measures* are desired for solving it in reference to the given general conditions.

3.1.1 Scope and context

On a greater prospect, the underlying idea states to develop a *holistic document management system* for the Vienna University of Technology overall: allowing each entity, to take advantage of the new system within the range of its own tasks, as well as benefiting from the possible collaboration that emerges as a whole. In terms of this thesis, the originating step of *developing a functional prototype for the subarea of one entity* is covered, i.e. a *proof of concept* to build upon.

Introducing the concrete field of deployment, the *Admission Office* of the Vienna University of Technology moves to the focus of interest: this organisational entity is responsible *for all matters concerning the admission of students*, notably the administration of applications to various degree programmes. As time goes by, the corresponding procedure has been revised periodically, adapting the infrastructure and workflows to upcoming requirements. But, to approach the major problem in this context, the transition from paper-based document management to state ot the art EDM never took place thoroughly. Thus, the systems and methods in use are in many aspects not as efficient as they could be, burdening the process for all persons involved.

In a more detailed analysis, the discussed subject should be addressed from three different *views*:

- *student's view*
- *employee's view*
- *dean's view*

Each perspective represents a group of humans with specific demands to be satisfied. Hence, the main goal of the prototype manifests in *enabling individuals to perform their tasks in the most comfortable and efficient way while preserving the quality of the initial request.* Put into practice, it is key to optimise the particular workflows of the participants above all by providing *a suitable set of methods and tools* to them: allowing people to *experience* a direct improvement in the course of their activities and raising the acceptance for the novel system consequently.

As a result, any action taken by a user should *contribute to the user's contentment on the one hand, and lead to the emergence of additional value in the entire system on the other hand.* Related to practice again, the following example illustrates this facilitating principle in the given context of the Admission Office: every time an employee reviews the application of a student with the help of the EDMS, a standardised, yet flexible workflow comprises a series of actions, that are *familiar* to the employee. Furthermore, this process fosters the respective *student record* (= a set of documents and metadata) by updating the relevant parts of information in the background of the system automatically. Therefore, an EDMS at large is able to divide complex operations into smaller steps, which in turn can be handled by humans and their cognitive skills best. By this means, the rendered effort of users is reinforced by the system and generates a significantly higher revenue, than it would ever be possible in combining *manual* activities.

Based on this considerations and the environmental conditions, it is reasonable to structure the implementation into two general components overall:

- a rather simple, but effective user interface at the *frontend*
- a fully equipped electronic document management system at the *backend*

One cornerstone of this setup is the resulting modularity, as illustrated in figure 3.1: besides its elementary purpose of storing and retrieving data, the EDMS serves as a *construction kit* for the functionality provided by the user interface. Thus, the frontend needs to incorporate just the methods and representations, that are truly required at a certain point of time for a specific task. Accordingly, numerous lightweight user interfaces can be developed and customised for different areas of application, while only a single EDMS has to be administrated *behind the scenes*.

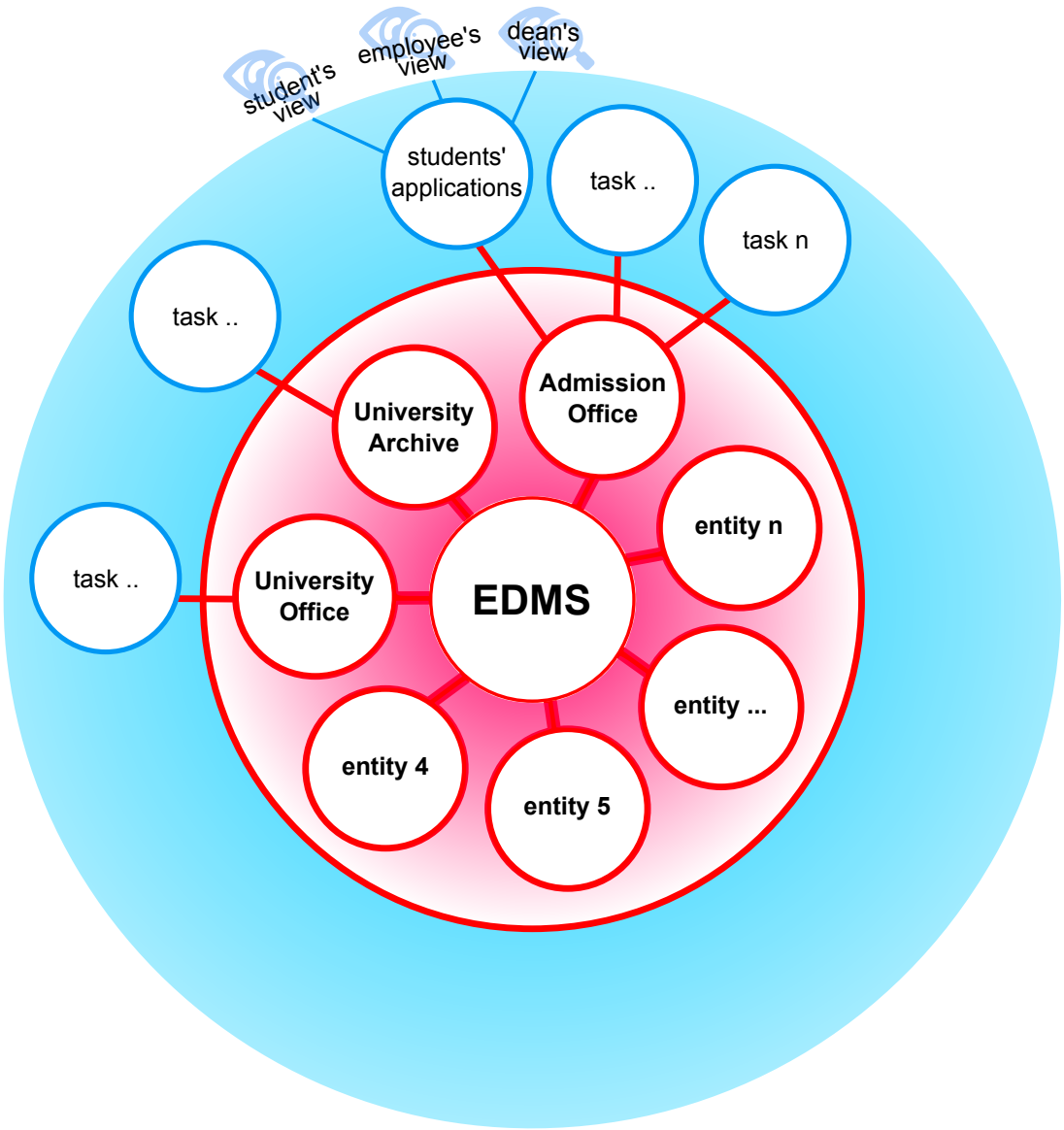

Figure 3.1: EDMS-centred modular design

3.1.2 Established DMS at the Vienna University of Technology

Comprising a multitude of smaller organisational entities, information management has been an integral part within the Vienna University of Technology since decades. Each entity is concerned with an area of responsibility and employs several people in different roles. These persons are qualified to process the assigned tasks and get in contact with the staff of other departments occasionally. Even though this sounds like the ideal breeding ground for computer supported cooperative work, the enforcement of a uniform, superior strategy could only be accomplished in parts up to now. Reasons may be found in the overwhelming complexity of such a project, missing resources or the incompatibility of individual requirements.

As a natural consequence, a wide range of isolated solutions for document management has grown in the course of time: this includes classical filing cabinets, storage on File Transfer Protocol (FTP) servers, as well as more elaborated software systems and workflows. In the overall context, each entity has established a DMS that meets its demands sufficiently and could be afforded by the available capabilities. On the long term, however, it is very likely that this imbalance sacrifices quality for sheer applicability.

Thus, a conclusion would advise to focus on a much more information-based approach on the whole: assuming, that information is the source and means to an end of any task, a collectively utilised EDMS should be the answer to all questions. Because the relevant information is already there, and only needs to be restructured and allocated thoughtfully, to unite the fragmented entities into a self-organising system. This holistic system then again could be utilised to provide a consistent interface for handling the communication with externals in an efficient way.

3.2 Requirements

Considering the previous findings as general scope of action, the practical implementation of the software prototype calls for a precise definition of *feasible* requirements subsequently: determining the essential aspects that have to be realised for embedding the prototype in the existing environment at the Vienna University of Technology. From a developers point of view, it is important to start with a set of minimum requirements as well: leaving enough flexibility and space for incremental improvements, that may result from the further collaboration with the customer.

3.2.1 Requirements for the EDMS

The EDMS shall build the fundament for the design of the Admission Office's user interface. Furthermore, it should constitute the possible implementation of additional user interfaces for other entities in future. Thus, the main requirements are:

- *quantity structure*: only a high-performance system comes into question; it has to cope with approximately 2.1 terabyte data in 530.000 documents from more than

32.000 users when launched.

- *On-premises deployment*: all documents have to be stored on internal servers of the Vienna University of Technology.
- *application programming interface (API)*: the EDMS has to provide a comprehensive API, that supports the programming languages of Java and Ruby.
- *database system*: the EDMS must be connectable to a preexisting Oracle Relational Database Management System (RDBMS).
- *financing*: ideally, the EDMS should be open source; in terms of licensing, payment per user is more favourable than payment per document.
- *workflow management*: the EDMS should allow the customisation of workflow management.
- *metadata*: the EDMS should aid the flexible configuration of metadata.
- *semantic search*: the EDMS has to facilitate the application of semantic search techniques.

3.2.2 Requirements for the Admission Office's interface

The user interface of the Admission Office is the visible end product presented to the respective users, i.e. *the tool that enables the completion of their tasks.* Hence, the requirements are prepared in close cooperation with persons in charge at the Admission Office: elaborating, which functionality *really is needed* for the different groups of users in the concrete context, based on many years' experience of direct involvement.

Student's view

- *initial login*: enable the login with username and password; schedule a deadline for the sending of the study application and display a confirmation request as well
- *study application*: show a list of required documents for the study application and provide a mechanism for the uploading of documents (*university qualification evidence*, *matriculation certificate*, *other documents*); uploaded documents should be taggable (= associating with catchwords, creating metadata); when the relevant documents are successfully uploaded, students must be able to send their study application
- *reaction to call for improvements*: permit re-login and the same procedure described in *study application*

Employee's view

- *list view of open study applications for bachelor programmes*: show a list of currently open bachelor programme's applications; show a list of (expired) calls for improvements (= change requests)
- *review open study applications for bachelor programmes*: enable the selection of the current state of the review; display all available states in a lookup table; automatically convert uploaded documents into Portable Document Format (PDF); maintain an embedded logfile for capturing each manipulation and the name of the responsible employee, recording the full reviewing process
- *call for improvements*: generate documents in Rich Text Format (RTF) based on templates and master data of students and studies, relevant templates are *call for improvements* and *notifications*; store the generated documents as part of the respective student record in the EDMS
- *review of equivalence*: make the review of authenticity (*authentic* or *forged*) and equivalence (*university qualification evidence* or *matriculation certificate*) of uploaded documents possible; assign an appropriate tag, if the document is equivalent
- *admission*: generate a *notification* for the admission of the student to the desired bachelor programme; conclude the pre-enrolment record and transform it into a student record; lock the student record for further changes by the student

Dean's view

- *list view of open study applications for master or doctoral programmes*: show a list of currently open applications to master or doctoral programmes
- *review open study applications for master or doctoral programmes*: display the comprehensive students record of the currently reviewed study application
- *compile statement*: enable the creation and sending of a *statement* for the reviewed application to a master or doctoral programme; facilitate the review of equivalence and selection of exams to achieve equivalence for the given study application
- *search mechanism*: provide a search mechanism for student records; search criteria are *student's name* and *reference number*; display the content of the retrieved student record

Postconditions

- *successful notification*: students have successfully received the *notification* for the admission to their desired study

– *active student record*: depending on the result (*notification*) of the review of the student's application, either a pre-enrolment record (if declined) or a student record (if successful) does exist in the EDMS

Altogether, the listed requirements represent a *constitutive agreement* on the performance of the software prototype. Nevertheless, it is quite likely that ideas for additional or slightly modified features emerge within the development process, and may be realised to improve the overall functionality.

3.3 Market sounding

Based on an objective assessment, two reasons suggest the application of an already established electronic document management system, rather than designing an own: First, the provided functions of an EDMS are generically enough to be adapted in a customised software. Second, and of greater importance, an individual composition obviously would lack the quality of a professional, market-proven solution. Thus, it is necessary to perform a *market sounding* to reach the next goal on the way to the final concept: selecting an appropriate EDMS as fundament for the self-developed prototype.

3.3.1 Evaluation

Technically, the market sounding and resulting evaluation can be classified as a *subjective, non-exhaustive web search* by the author of this thesis. Hence, the assessment of each EDMS mainly rests on information, and directly testing the software, if possible. Overall, the findings are summarised in several categories:

– *product name*

– *open source*: whether all functionality and the source code of the EDMS is freely available, which software license is granted.

– *quantity structure*: is the product able to cope with considerable amounts of data, or is it designed for smaller organisations?

– *applicability*: if the product meets the particular requirements for the given project, or essential functions are missing.

– *sustainability*: estimation of the product's future applicability, i.e. if it is still actively developed and maintained, or discontinued to some extent (e.g. support is no longer provided).

– *overall impression*

Each category is evaluated from a minimum of one star up to a maximum of three stars. The *overall impression* does not automatically result from the partial ratings, because some details may have a greater significance in the whole context. Figure 3.2 presents the outcomes of the evaluation, sorted alphabetically by the name of the products.

Product Name	Open Source	Quantity Structure	Applicability	Sustainability	Overall Impression
Alfresco Community Edition	★★★	★★★	★★	★★	★★
bitfarm-Archiv DMS GPL-Edition	★★★	★	★	★★	★
Casebox	★★	★	★	★★	★
Kimios	★★	★	★	★	★
KRYSTAL DMS Community Edition	★★★	★	★	★	★
LogicalDOC Community Edition	★★★	★	★	★★★	★
Mayan EDMS	★★★	★	★★	★★★	★★
Nuxeo Platform	★★★	★★★	★★★	★★★	★★★
ONLYOFFICE Community Server	★★	★	★	★★★	★
OpenDocMan Community Version	★★★	★	★	★★	★
OpenKM Community Version	★★★	★	★	★★★	★
OpenProDoc	★★	★★	★	★★	★★
SeedDMS	★★★	★	★	★★	★

Figure 3.2: market sounding, evaluation

Already the first stage of the market sounding reveals enough evidence, that the product range and quality of open source solutions should be sufficient for the intended purpose. Thus, fee-based software is only researched as reference. Overall, open source EDMS may be distinguished in two groups for the most part:

On the one hand, market leaders offer *community editions* of their enterprise products. Commonly, these scaled-down versions are restricted in terms of functionality (e.g. less features, one database management system compatible at all) and the quantity of possible users or documents. On the other hand, groups of independent developers deploy more lightweight EDMS, which are targeted at smaller businesses in general. Here, the main problem relates to the sustainability of such products, because the responsible persons may end their project at any time. Altogether, it has to be concluded, that each reviewed EDMS can be applicable in a certain environment, but one system definitely stands out in the particular context: *Nuxeo Platform*.

3.3.2 Selected software: Nuxeo Platform

Above all, the unique characteristic of *Nuxeo Platform* is found in its business model: an established company provides a sophisticated EDMS free to use and refine. Revenues are generated from *Nuxeo Studio*, an easy-to-use graphical interface for the configuration and customisation of Nuxeo Platform, as well as the professional support of the product. Hence, there is no limitation in any regard: all features and functionality is at the hands of the developer that is willing to become acquainted with the underlying framework.

Considering the technical perspective, Nuxeo Platform comprehends the advantages of a modular architecture: functionality is understood as a result of interchangeable components (*bundles*), rather than being built rigidly into the structure of the system. This modular design allows to incorporate changing or growing requirements in a flexible way and supplies a variety of extension points to connect to.

Conveying a more pratical overview, Nuxeo includes, but is not reduced to the following set of features and principles [39]:

- *Content* is a magnitude larger in size and throughput than ever before, consists of complex, structured objects and manifests in an exponentially increasing amount of types and delivery channels.
- Nuxeo Platform is designed for a *data-driven world*, *built to be extended* and *tested for a critical workload*.
- The *Content Repository* features a rich content model, relationships, querying and search, a versioning policy, access control as well as pluggable data and file persistence.
- A flexible *Representational State Transfer Application Programming Interface* vulgo *REST API* presents full repository create, read, update and delete (*CRUD*) support with more than 130 operations and several native client libraries (e.g. Java).
- Massive scalability can be achieved by using either NoSQL (MongoDB) or relational (PostgreSQL, Oracle, MySQL, etc.) database systems.
- A benchmark on *performance* confirms one billion documents and 6.000 queries per second on a single cluster, with up to 14.000 documents processed per second.

According to this basic capacities, Nuxeo Platform can be utilised for any use case from pure document management to outright groupware activities. Thus, Nuxeo may be summarised best as an *enabling software system*, that works out of the box at the frontend, but is also capable to be placed as bedrock at the backend.

3.4 Concept of the prototype

The *concept* of the software prototype, i.e. the implementation referred to as *Student Records Manager* (SRM) henceforth, considers all previously discussed components and assembles them into a functional system now: describing, *how the Student Records Manager is set up in order to be able to fulfil the agreed requirements.*

Starting with the *big picture*, the SRM is designed as a *three-tier architecture*. This structure allows to separate systems into modular layers and raises the flexibility of the overall application thereby. Each layer serves a certain function of the system, and can be modified in terms of the function itself, or the individual components that produce the function. Thus, a *loose coupling* between the layers reduces dependencies while preserving interchangeability. Transposing this generic concept, Figure 3.3 provides the concrete composition of the SRM in an universal illustration:

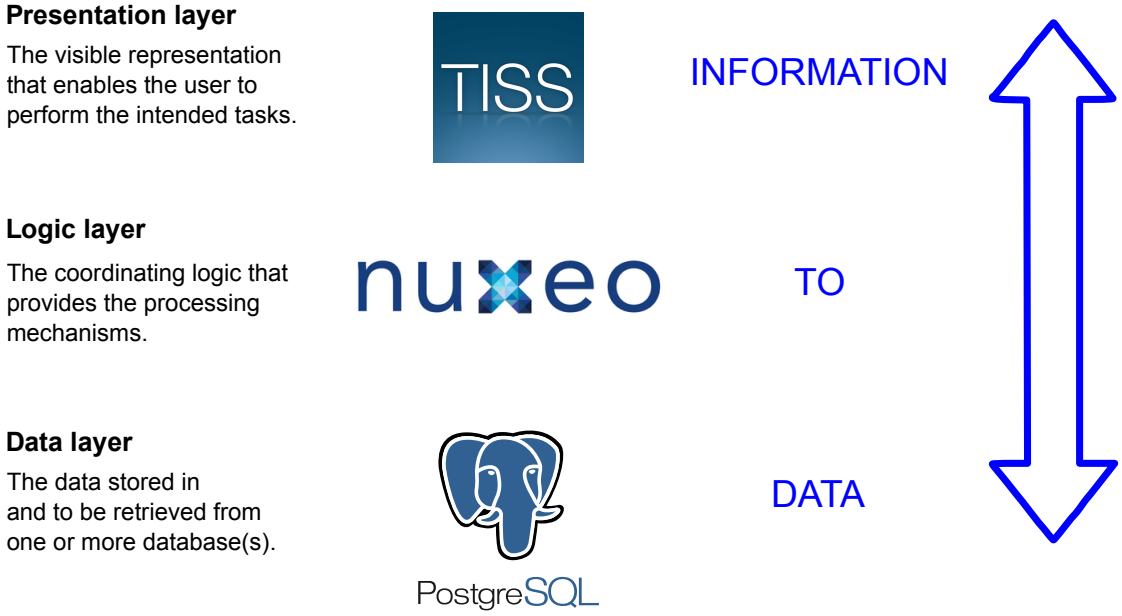

Figure 3.3: the software prototype as a three-tier application

On an abstract level, the purpose of the SRM is nothing else than *transforming information to data and data to information* vice versa: that is, making sense of the retrieved data in the presentation layer to display the information users require for accomplishing their tasks, as well as dismantling the manipulated and newly user-generated information to store it in one or more database(s) then again. Consequently, the following subchapters outline a more detailed view on the three layers.

3.4.1 Presentation layer

The *presentation layer* builds the interface to the human user and is called *user interface* accordingly. More specifically, the deployment as a *web application* allows users to access the presented information and functionality via a web browser on different devices, ensuring platform independence.

Moreover, it is advisable to design and style the respective web pages in a fashion, which is familiar to the target audience. For this reason, the *cascading style sheets* (CSS) of the established TU Wien Information-Systems and Services (TISS) are utilised. Thus, the novel user interface integrates seamlessly into the known environment of users and minimises their need for further training.

Corresponding to the different roles and tasks of users, three distinct user interfaces (views) are considered in addition. This subdivision paves the way for an even more tailored *presentation* for each group of people, increasing the *practical usability* of the SRM. Finally, the exhaustive list of use cases comprises:

The student's view

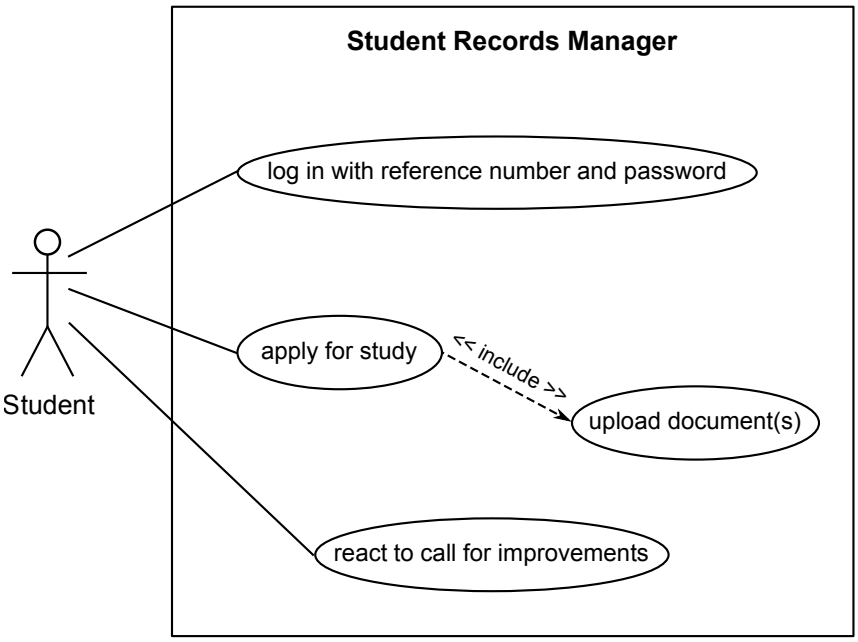

Figure 3.4: SRM, possible use cases of students

log in with reference number and password: students (for simplicity, this group includes persons, that apply for a study, and may not be qualified students up to this point) log in to the system with their reference number as username and birthday as password; in case of the initial login, a deadline for the sending of the study application is created,

displayed and must be confirmed by the student to progess to the use case *apply for study*; the system has to verify the authenticity of the login data

apply for study: students see their master data, educational background and proposed field of study; in order to be able to send their study application, students must upload a matriculation certificate (if citizen of EU-/EWR-member state) or an university qualification evidence (if not citizen of EU-/EWR-member state) first; students may re-upload their matriculation certificate or university qualification evidence; students may upload (several) other documents; students may add tags to one or more of the uploaded documents and may remove their added tags; students may send their study application within the displayed deadline; the system has to convert uploaded documents into Portable Document Format (PDF)

react to call for improvements: in addition to the functionality of the use case *apply for study*, students are informed with a notification document, which states the reason of the call for improvements, and see their previously uploaded insufficient document(s)

The employee's view

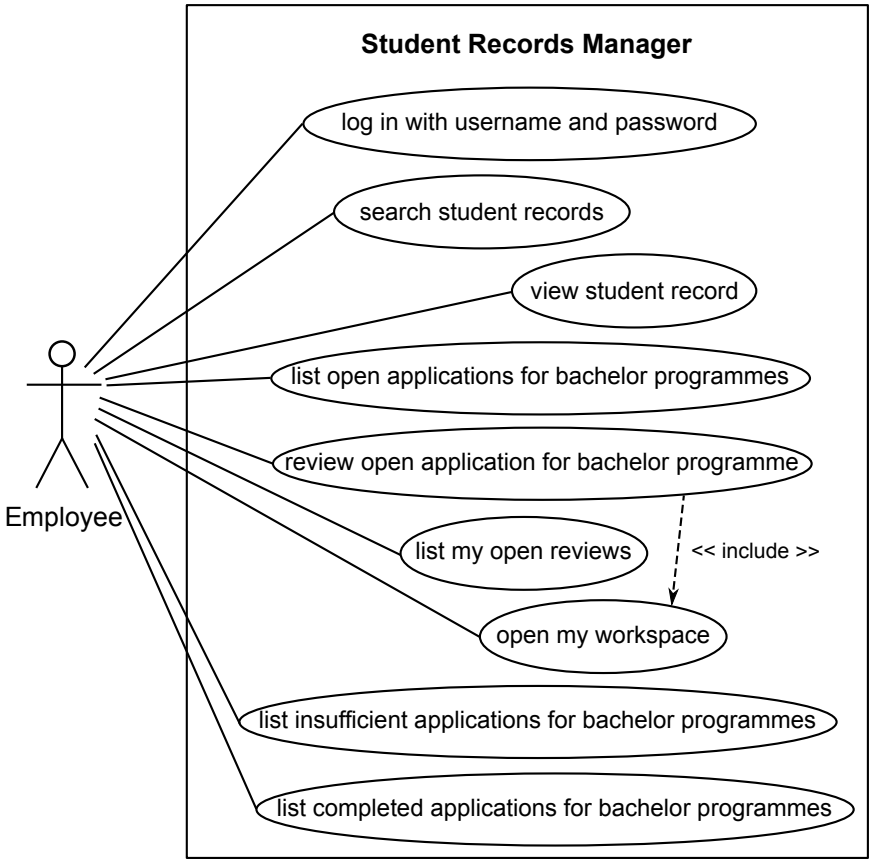

Figure 3.5: SRM, possible use cases of employees

log in with username and password: employees log in to the system with their username and password

search student records: employees may search for student records by one or more criteria, i.e. the name of the student (exact name or part of a name), reference number of the student and state of the student record; employees may open a retrieved record according to the use case *view student record*

view student record: employees see the most recently opened student record in this session, displaying the master data, educational background and information of the student's proposed/completed studies, as well as the current state of the student record; additionally, the uploaded documents of the student and the related issued documents of the Admission Office are disclosed

list open applications for bachelor programmes: employees see a list of open applications for bachelor programmes, displaying the student's reference number, first name, last name, state of the student record as well as the date of the application's expiration (deadline) for each entry; employees may open a listed application as a function of the use case *open my workspace*

review open application for bachelor programme: employees review an opened bachelor programme's application in their workspace; the system has to disable the application for the review by other employees, as long as it is opened; employees see the applicant's master data, educational background, proposed field of study and uploaded documents; employees may add or remove documents' tags; for each document, employees select one result concerning its authenticity and equivalence; the system adds an adequate tag to the particular document; employees may add an optional comment for the changes made; employees see a log of all changes ever made by employees to the application; employees may save or abort the review and are able to complete the review, when all required documents (= matriculation certificate or university qualification evidence) are evaluated; employees choose a result for the completed review, either granting the admission, declining the admission or calling for improvements by the student; if completed, the system has to create and attach a complying notification document to the student record and/or update the application's deadline; the system has to change the student record's state according to the employee's actions; the system has to save the currently selected values; the system provides a table to look up all possible states of a student record

list my open reviews: employees see a list of opened applications to bachelor programmes, whose review they have started, but not yet aborted, saved or completed; the list displays the student's reference number, first name, last name, state of the student record as well as the date of the application's expiration (deadline) for each entry; employees may open a listed application under the terms of the use case *open my workspace*

open my workspace: employees see the content of the most recently opened student application in this session, displaying the applicant's master data, educational background, proposed field of study and uploaded documents; employees may start or continue the review as per use case *review open application for bachelor programme*

list insufficient applications for bachelor programmes: employees see a list of insufficient applications for bachelor programmes, displaying the student's reference number, first name, last name, state of the student record as well as the date of the application's expiration (deadline) for each entry

list completed applications for bachelor programmes: employees see a list of completed applications for bachelor programmes (either admission granted or admission declined), displaying the student's reference number, first name, last name, state of the student record as well as the date of the completion of the review for each entry; employees may open a listed application according to the use case *view student record*; employees may narrow or widen the listed applications by setting a timescale (7, 14, 30, 60 or 365 days)

The dean's view

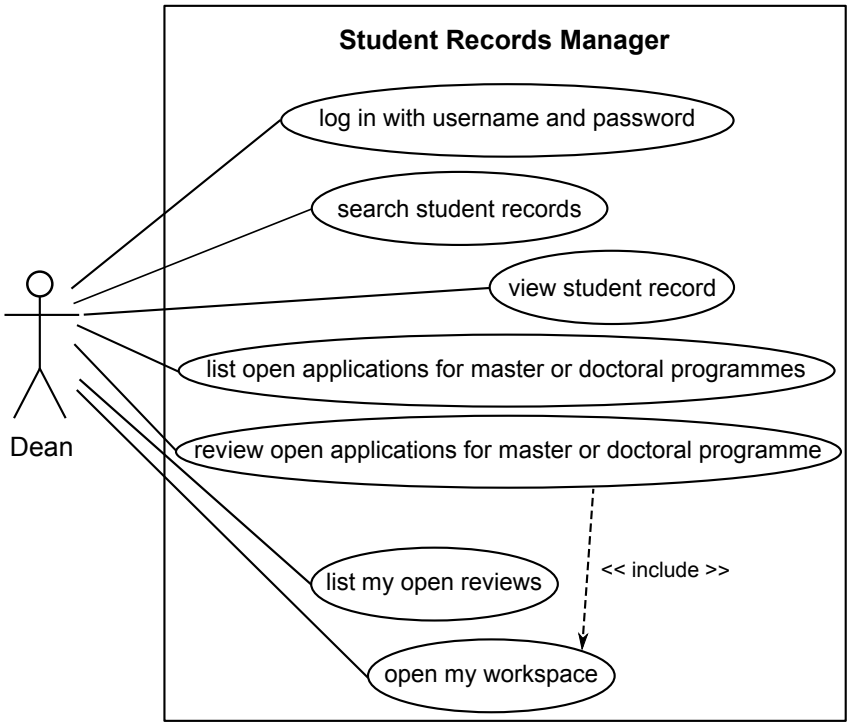

Figure 3.6: SRM, possible use cases of deans

log in with username and password: deans log in to the system with their username and password

search student records: deans may search for student records by one or more criteria, i.e. the name of the student (exact name or part of a name), reference number of the student and state of the student record; deans may open a retrieved record according to the use case *view student record*

view student record: deans see the most recently opened student record in this session, displaying the master data, educational background and information of the student's proposed/completed studies, as well as the current state of the student record; additionally, the uploaded documents of the student and the related issued documents of the Admission Office are disclosed

list open applications for master or doctoral programmes: deans see a list of open applications for master or doctoral programmes, displaying the student's reference number, first name, last name, state of the student record as well as the date of the application's expiration (deadline) for each entry; deans may open a listed application as a function of the use case *open my workspace*

review open application for master or doctoral programme: deans review an opened application for a master or doctoral programme in their workspace; the system has to disable the application for the review by other deans, as long it is opened; deans see the applicant's master data, educational background, proposed/completed field of studies and documents (= the student record so far); deans select values for a predefined set of fields to assemble their overall statement on the application; deans may save or abort the review; deans may complete the review, when the values for all required fields are selected; if completed, the system has to create and attach a complying statement document to the student record; the system has to change the student record's state according to the dean's actions; the system has to save the currently selected values

list my open reviews: deans see a list of opened applications to bachelor programmes, whose review they have started, but not yet aborted, saved or completed; the list displays the student's reference number, first name, last name, state of the student record as well as the date of the application's expiration (deadline) for each entry; deans may open a listed application under the terms of the use case *open my workspace*

open my workspace: deans see the content of the most recently opened student application in this session, displaying the applicant's master data, educational background, proposed/completed field of studies and documents; deans may start or continue the review as per use case *review open application for master or doctoral programme*

3.4.2 Logic layer

Nuxeo Platform is applied as *middleware* in the logic layer. Here, it provides the necessary business logic and structures to manage the allocation of data between its surrounding layers. In order to fulfil this *mediating* function, a *content repository* consitutes the elementary part of the system:

Basically, the repository is utilised for storing *documents* and offers advanced functionality to create, manipulate or delete these documents. Describing the special characteristic of the Nuxeo repository, a document is defined as a *set of fields*, and not just a simple file [40]. Figur 3.7 illustrates this concept.

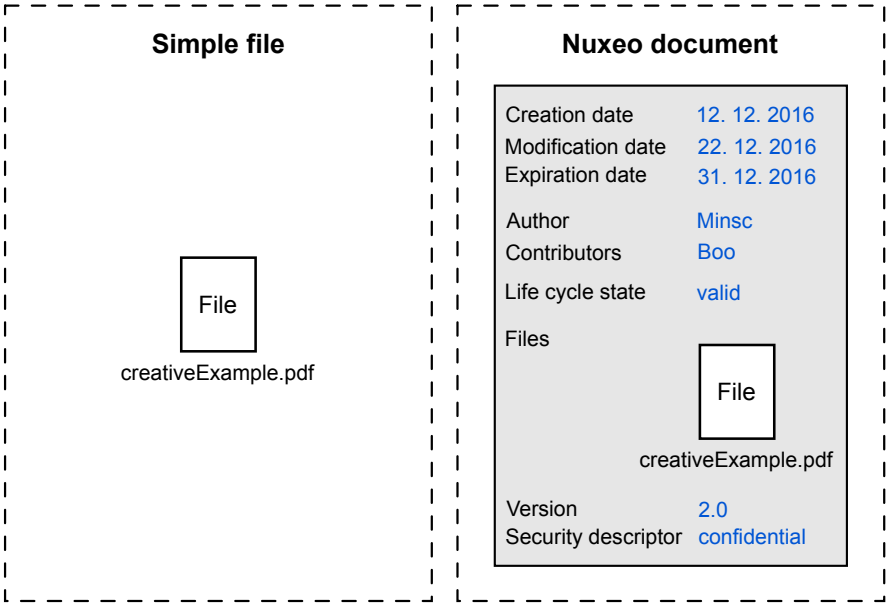

Figure 3.7: Nuxeo document concept

Fields may be simple fields (string, integer, boolean etc.), simple lists or complex types. Whereas a file per se represents a special form of a complex field, containing a binary stream, filename, mime-type and size. Thus, a document can hold zero, one or several files, and even a folder is viewed as a document due to its captured metadata (title, creation date etc.).

Inside the Nuxeo repository, each document is classified by a *document type*. The document type specifies a name, a base document type, a set of facets, as well as a set of schemas. These *schemas* can be used to define the structure of the document and metadata blocks.

Overall, the repository and its documents can be accessed directly via a REST API, or additionally supplied client libraries (for Java, PHP, Python, iOS, Android etc.) of Nuxeo Platform. By this means, the presentation layer's web application is able to connect to the repository, providing the required documents (= structured information) and operations for the users' tasks (= their use cases).

On the other hand, the Nuxeo repository applies a service for managing the *persistence* of documents. This *Nuxeo Visible Content Store (VCS)* capitalises on an RDBMS that keeps the data at the backend, i.e. the data layer. It is *pluggable* and allows to store data in standard SQL databases, using a natural object mapping to tables, supporting full-text search if possible.

3.4.3 Data layer

Linking to the logic layer, Nuxeo Platform supports several *relational database management systems*. The Student Records Manager shall incorporate *PostgreSQL*, as it is one of the most established *open source* systems in this domain. Likewise, it would be feasible to use the *Oracle* or *MySQL* RDBMS in a refined version of the software prototype: the functionality of Nuxeo Platform stays the same one way or another, and just a part of the configuration has to be changed.

Eventually, the data layer *completes* the concept for the Student Records Manager. As a result, all *general conditions* of the project are determined successfully and the *implementation* of the prototype may start.

CHAPTER 4

Implementation

This chapter outlines the *genesis* of the Student Records Manager. Thus, it comprises everything, that needs to be done for *transforming the theoretical project description into a practical realisation embodied by the software prototype*. Starting with a general overview of the *development process*, each *development stage* is examined in detail subsequently. The resulting *overall setup and structure* gets summarised accordingly, illustrating the entire composition. Finally, the user interface as the visible end product of the *Student Records Manager* is discussed in relation to the initial requirements of the project.

4.1 Development process

Software development on the whole can be considered as a *sequence of consecutive steps, that supplement each other in order to produce a certain piece of software*. Hence, it is a list of tasks to be accomplished by the combination of tools and techniques from the state of the art. In this respect, the presented process in Figure 4.1 displays *only one of many possible solutions* which may be feasible in the given context:

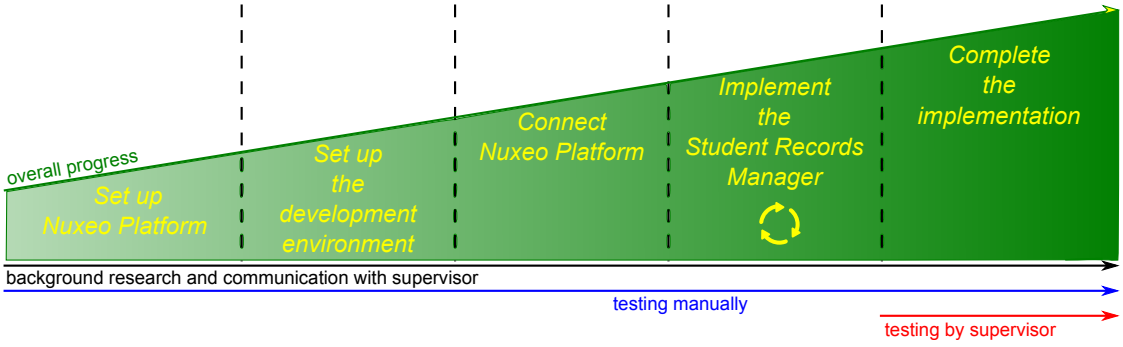

Figure 4.1: development process of the Student Records Manager

Each task has a certain goal, which forms the *transition* to the next stage at the same time: by this means, it can be guaranteed, that particular *milestones* are reached and therefore provide the essential groundwork for the further development. In addition, several background activities are performed to *assure the quality* of the overall process, reflecting and adapting the work practices if necessary.

4.2 Development stages

In the course of the following subchapters, the series of actions as well as all required information for succesfully building the *Student Records Manager* is disclosed. Thus, the revealed process can be regarded as a *reproducible approach* that may support the development of similiar applications in future to some extent.

4.2.1 Set up Nuxeo Platform

First and foremost, it is advisable to try out Nuxeo Platform and verify its functionality: because this prefabricated software is at the heart of the SRM and constitutes all subsequent measures to be taken. Hence, any problem that is detected and solved at this early stage saves a lot of effort at a later point of time.

Related to practice, this concrete steps are realised:

- a *local server* is built: reusing an older personal computer, that is enhanced by a new hard disk for being able to simulate the storage of a large amount of data.
- Debian 7 (Wheezy) is installed as *operating system*: being popular for network servers, available for free and compatible with Nuxeo Platform in general.
- Nuxeo Platform 7.10 is installed on the operating system: using the appropriate Debian installer distributed by Nuxeo, which is based on Apache Tomcat and supplies all required third-party dependencies.
- PostgreSQL 9.4 is installed on the operating system: a sophisticated open source RDBMS, which is compatible with Nuxeo Platform 7.10.
- PostgreSQL 9.4 is configured to work with Nuxeo Platform 7.10: creating the role and database for Nuxeo and some further minor actions

Having established the basic setup of Nuxeo Platform, the most important parts of its comprehensive documentation are studied. In this way some theoretical knowledge about the elementary concepts and structures of Nuxeo is acquired, as well as directly applied by analysing the previously installed system (understand, how Nuxeo Platform is assembled; manually starting and stopping the service on the server etc.).

4.2.2 Set up the development environment

At the beginning of stage two, *TISS*, the applied *information system* of the Vienna University of Technology moves into the focus: it represents the normative system, where the SRM is embedded. Thus, there are several rules and technical conditions, that have to be met, when developing a new *component* in this context. Consequently, the *supervisor* as link to the department of Campus Software Development (CSD), which is responsible for maintaining TISS, provides the following source material:

- *static resources*: a package, which contains the static resources (CSS, icons, images etc.) for building a TISS project, as well as an implemented showcase that should be used as guideline for the development of such a project.
- *data encapsulation*: a project, that includes a bundle of classes, which can be utilised for generating simulated master data of students, because it is legally not possible to incorporate original data.

Based on the specifications of this materials, the definite *development environment* for the SRM is set up as a whole:

- a local personal computer with Windows 7 as operating system serves as general development platform.
- NetBeans IDE 8.1 is installed on the operating system, acting as a software development platform that supports the development of different Java application types.
- Java EE (Java Platform, Enterprise Edition) is applied in consequence to facilitate the practical implementation of SRM: a platform consisting of several services, protocols and APIs which provide the functionality for the development of multitiered, web-based applications.
- Apache Maven is used for literally *building* the web application archive (WAR file), i.e. a java archive (JAR file) subjected to distribute a collection of resources (Java Servlets, Java classes, XML files, static web pages etc.), which *constitute the web application.*
- Apache Tomcat is utilised to *deploy* the resulting web application on the local machine.
- Mozilla Firefox is the favoured web browser to access the web application.

In order to get a more sophisticated impression of the different concepts, further documentation on the addressed topics is consulted as a next step. Eventually, an introductory Jave EE project is implemented successfully, preparing the ground for a thorough analysis of the supplied TISS showcase subsequently. Finally, everything is sufficiently understood and in the right place to advance to the next stage of development.

4.2.3 Connect Nuxeo Platform

Being equipped with a working local server on the one hand, and a reliable development environment on the other hand, the purpose of stage three is to *relate* this previously independent systems to each other, i.e. establishing a temporary connection between them over a local area network (LAN).

As a result, it is possible to access and test Nuxeo Platform from the development platform. This is accomplished via the standard web interface of Nuxeo Platform 7.10, as illustrated in Figure 4.2:

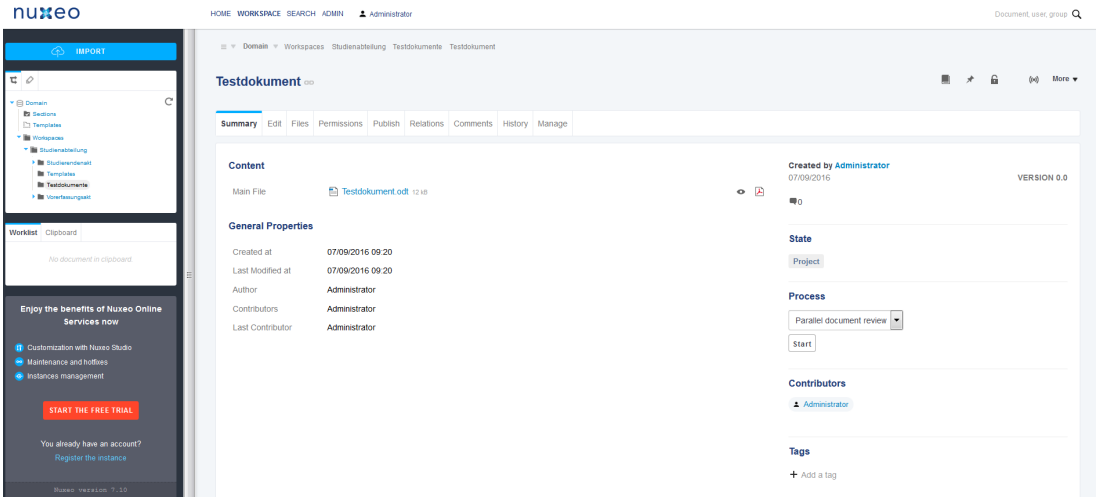

Figure 4.2: standard web interface of Nuxeo Platform 7.10

Several basic operations are tested, e.g. creating documents, uploading files to documents, adding tags to documents, changing metadata of documents. By this means, the complexity and scope of Nuxeo Platform is assessed from a user's perspective, i.e. *experiencing* the application in a practical context.

Furthermore, the Nuxeo Bulk Document Importer service is utilised to fill the Nuxeo repository initially with approximately 30 000 documents. According to that, the search engine is explored and pursuing actions are taken, e.g. altering the version of documents, deleting documents, trying to retrieve (deleted) documents in a certain version et cetera.

Nuxeo Java Client

While the *web* interface is designed for presenting the contents and functions of the Nuxeo repository directly to the end user, an additional *application programming interface* (API) allows to integrate the Nuxeo repository *remotely* in other applications, i.e. the Student Records Manager.

Paying attention to the development environment, the *Nuxeo Java Client* provides a Java client library for the Nuxeo Platform Representational State Transfer (REST) APIs.

Version 1.0 of this library is compatible with Nuxeo Platform 7.10 and therefore imported into the Maven-based project in NetBeans:

```
import org.nuxeo.client.api.NuxeoClient;

String url = "http://10.0.0.2:8080/nuxeo";
NuxeoClient nuxeoClient = new NuxeoClient(url, "Administrator", "Administrator");
```

As a consequence, the Repository API and Automation API of Nuxeo Platform can be used within the Student Records Manager to *instruct Nuxeo Platform to perform certain operations*. For example, a student record (= document with metadata that may contains other documents with metadata and files) is created and stored in the Nuxeo repository:

```
// Create folder to hold documents for studentsRecord with ID "studentsRecordID"
Document folder = new Document(studentsRecordID, "Folder");

folder.set("dc:title", studentsRecordID);
folder.set("dc:source", "Vorerfassungsakt erstellt");
folder.set("dc:expired", date);
folder.set("dc:format", "nicht in Bearbeitung");

nuxeoClient.repository().createDocumentByPath(admOffice+"/Vorerfassungsakt", folder);
```

Another method could be called to retrieve all student records with a particular state in return:

```
@Override
public List<Document> getStudentsRecordsWithState(String state) {
        List<Document> studentsRecords = new ArrayList<Document>();

        Documents docs = nuxeoClient.header("X-NXProperties", "*").repository().query(
        "SELECT * FROM Folder WHERE dc:source='" + state + "'");

        for(Document child : docs.getDocuments()) {

        studentsRecords.add(child);
        }

        return studentsRecords;
}
```

Depending on the state of the student record, it may be useful to be able to change its storage location:

```
@Override
public void moveDocument(String studentsRecordID) {

        Document directoryToMove = nuxeoClient.header("X-NXProperties",
        "*").repository().fetchDocumentByPath(admOffice+"/Vorerfassungsakt/"
        + studentsRecordID);

        Document directoryToMoveTo = nuxeoClient.header("X-NXProperties",
        "*").repository().fetchDocumentByPath(admOffice+"/Studierendenakt");

        directoryToMove.setPropertyValue("dc:expired", "2050-01-01");
        directoryToMove.updateDocument();

        nuxeoClient.automation("Document.Move").param("target", targetDirectory).param(
        "name", studentsRecordID).input(directoryToMove).execute();
}
```

By testing this basic functionality in the first place, some valuable lessons are learned in advance. All things considered, the *preparation* for the implementation is completed at the end of stage three.

4.2.4 Implement the Student Records Manager

The important preliminary activities are to *enable* the implementation; the major part of development is found in stage four nevertheless: *implementing the Student Records Manager* is a *manual* process that involves the selection of an appropriate software architecture, as well as the thoughtful elaboration of its contents. In this respect, the following subchapters start with the presentation of the *Model View Controller architecture*, and work the way through all relevant components of the original software prototype.

Model View Controller architecture

Referring to a more common definition, the *Model View Controller* architecture can be described as a three-tier architecture on *application level*: this means in effect, that the software application's maintainability is improved by a *separation of concerns*, i.e. organising its elements logically into the model, view and controller layer. As illustrated in Figure 4.3 and explained subsequently, each layer serves a specific purpose, while the orientation of the connections facilitates the control of the application's complexity:

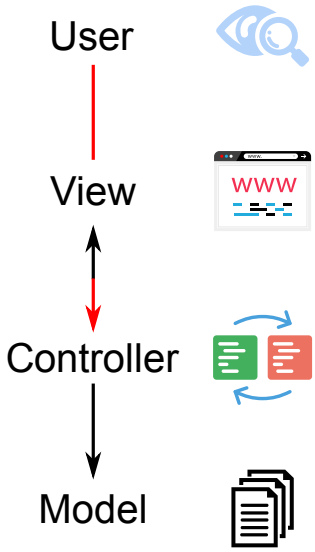

Figure 4.3: MVC components and their interplay

The persistent data of the application is managed in *models*. On the other hand, views are (visual) representations of the models' data, that can be understood by the user then again. In between, *controllers* are used to manipulate the models and update the views based on the user's interaction. Thus, the number of direct connections is minimised in order to provide a clear and modular application structure overall.

Related to practice, the adaption of the MVC architecture may be considered best as an *enabling* starting point, where the *structural rules* for the implementation are determined.

Figure 4.4 outlines the concrete segmentation of the Student Records Manager:

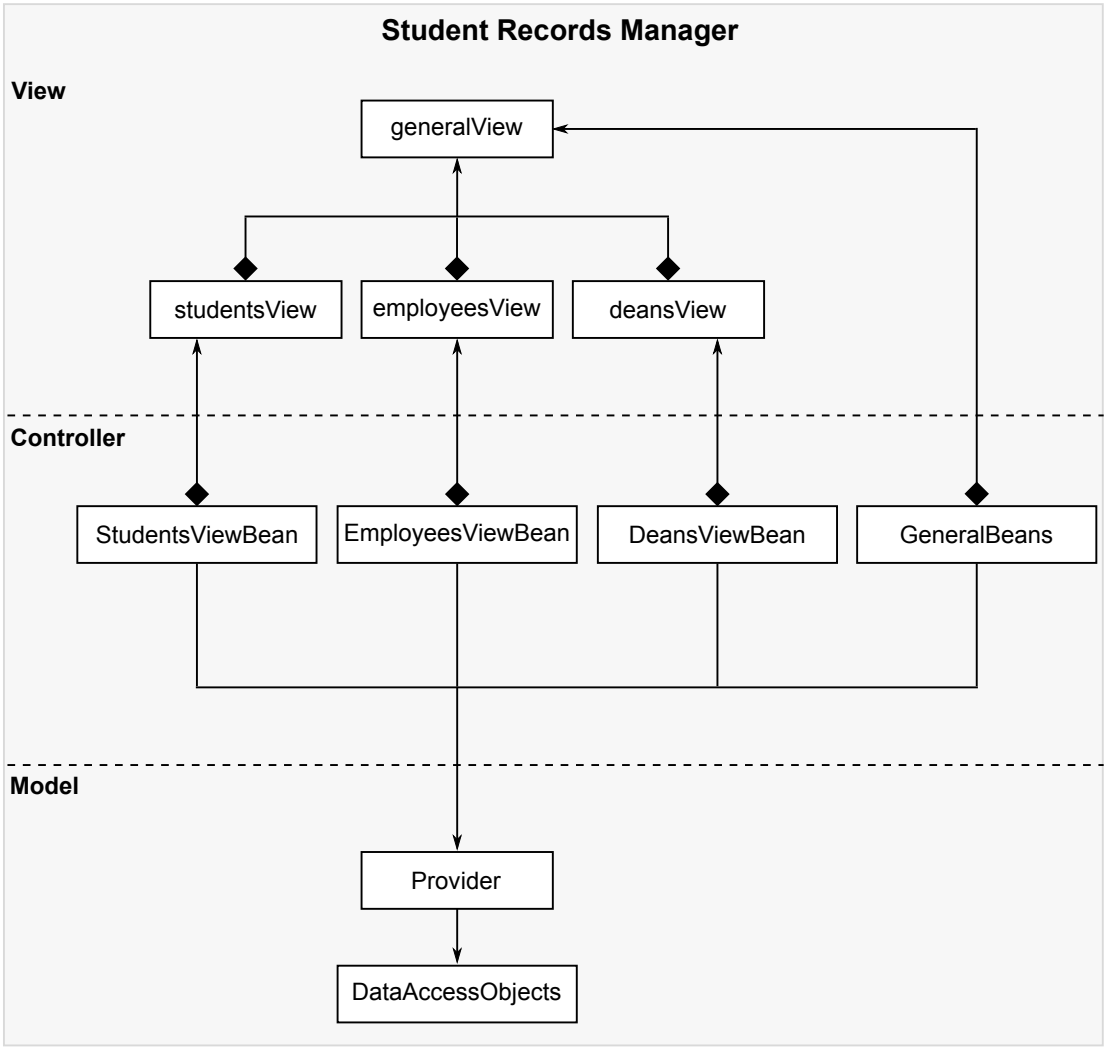

Figure 4.4: diagram of the overall structure of SRM

Retaining a certain degree of abstraction, *Facelets* is the default view handler technology for *JavaServer Faces*. This web template system makes use of valid XML documents to form views, i.e. the XHTML format is applied to create a document, which can be processed by the user with the help of a web browser. Hence, the *generalView* comprises documents, that frame the general structure of the representation, while the documents of the *studentsView*, *employeesView* and *deansView* are designed for the specific requirements of the users' tasks. Each view is administrated by a controller *bean*, i.e. a class that contains the logic to update the dynamic parts of the XHTML documents and transform the user interactions into concrete operations on the models. In this respect, a provider class offers methods to manipulate the stored data at the backend of the application, and

retrieve it vice versa.

Eventually, the big picture of the Student Records Manager's technical implementation is *one possible interpretation* of the MVC architecture. Thus, an architecture or pattern depicts a way to a solution, and is not the solution itself. According to that, a more detailed characterisation of the application's contents is supplied in the following passages.

Model

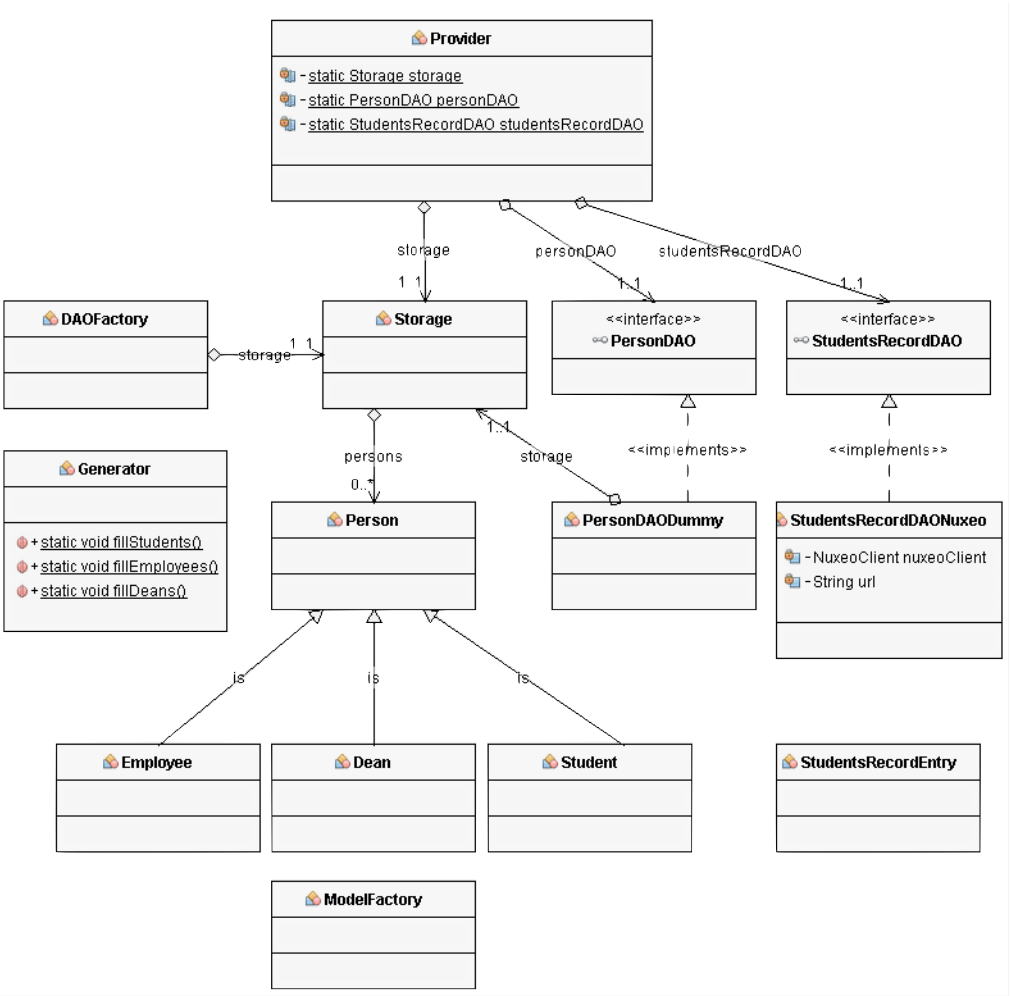

Figure 4.5: class diagram of the Model layer

Figure 4.5 displays all classes of the model layer, as well as some attributes and operations are mentioned to clarify the underlying concept. To begin with, the *Provider* (class) constitutes the connection to the controller layer. Therefore, it creates a *Storage*, which is handed over to the *DAOFactory*. Consequently, a *PersonDAODummy* and a *StudentsRecordDAONuxeo* are fabricated with the aid of the *DAOFactory*. This allows the

Provider to *provide* methods for the modification of *Person* objects on the one hand, and to give instructions to the Nuxeo Platform on the other hand. For this purpose, the previously introduced *Nuxeo Java Client* is integrated in the *StudentsRecordDAONuxeo* class. In addition, the *Generator* is called in the *Provider* to populate the *Storage*. This is necessary to create instances of the *Students*, *Employees*, *Dean* classes, what in turn enables the login of users in their roles and simulates the master data of students. A *StudentsRecordEntry* is an object, that holds combined data from the *Storage* and Nuxeo Platform (mainly used for the list representations within the views).

In this context, *Data access object* (DAO) is applied: a *software design pattern*, that separates the application from the persistence mechanism. The main idea of this pattern states to distinguish, *what* data access is required by the application (= public interface of the DAO), and *how* these needs may be served with a particular database scheme (= implementation of the DAO). By this means, only the DAO implementation has to be modified in case of a change to the persistence mechanism, preserving the maintainability and flexibility of the overall application.

View

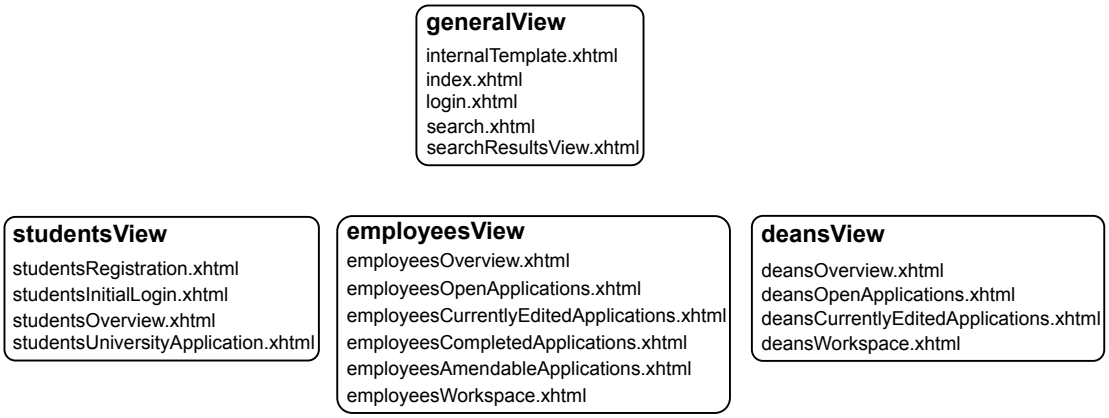

Figure 4.6: views of the Student Records Manager

Figure 4.6 summarises all documents (= web pages), that make up the view (= visual representation) of the Student Records Manager together. Basically, all these newly created Extensible Hypertext Markup Language (XHTML) documents build upon the prefabricated TISS showcase by incorporating *internalTemplate.xhtml*: this document can be described as a *template web page*, which capitalises on further resources (page fragments, Cascading Style Sheets, icons, images, JavaScripts) and thereby specifies the general design and style to be applied for all web pages.

A document (= a single view) consists of a structured sequence of JavaServer Faces (JSF) and XHTML *tags*, which enclose static contents (text, images, icons etc.) as well as dynamically allocated data from the model layer. Concerning this matter, the Java *Expression Language* enables the communication between the view and its controller,

i.e. integrating the return values of the controller bean's methods and objects into the document. Finally, XHTML output is rendered as a function of the tags and contents and may be viewed in a web browser by the user then again. Additional libraries like *PrimeFaces* extend the design options by providing extra tags and components.

Because a view is perceived *visually*, it should be tested by a *human user* above all. Comprehending the presentation is a cognitive process, i.e. small changes in a view can have major consequences on the interpretation and execution of a task. Thus, this part of development mainly is an *artistic activity with reference to best practices* and not directed towards a well-defined goal.

Controller

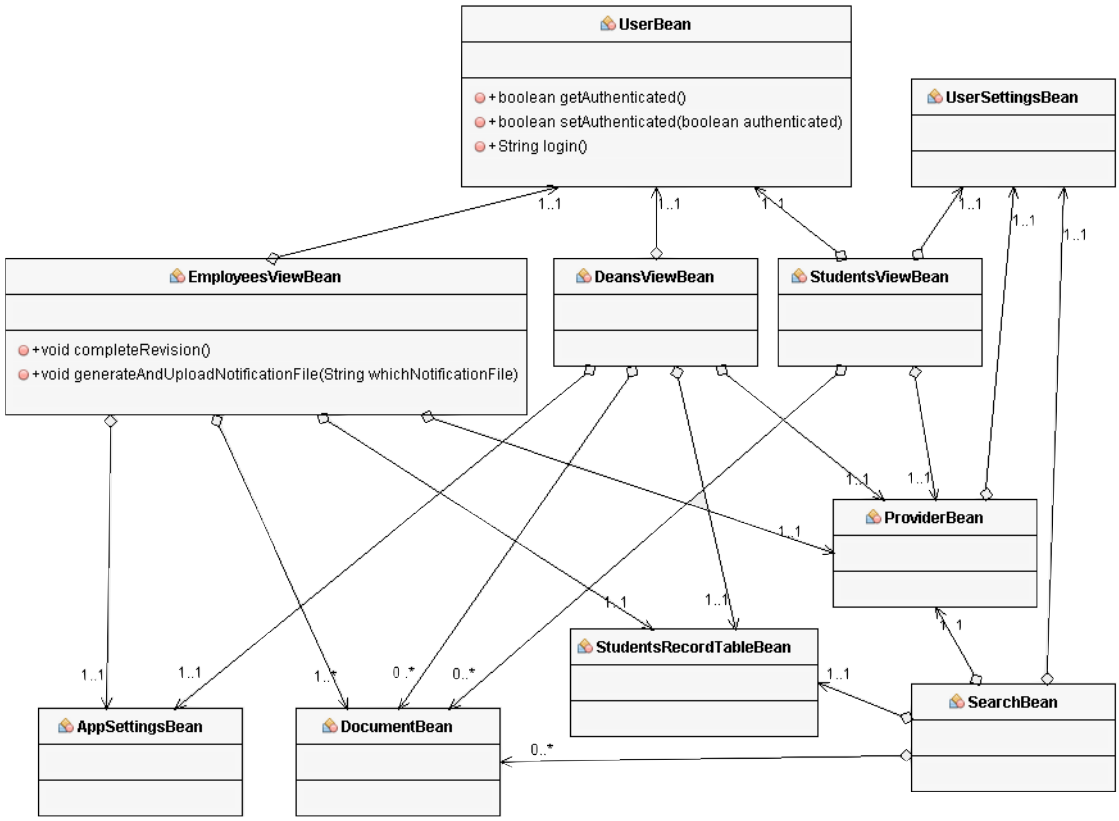

Figure 4.7: class diagram of the Controller layer

Figure 4.7 shows all classes of the controller layer, as well as some exemplary attributes and operations to be discussed subsequently. In a nutshell, these classes are referred to as *managed beans* (by the JSF framework) and take advantage of the Java EE platform's *Contexts and Dependency Injection* (CDI) services: a bean therefore is a *source of contextual objects* that define the application's state and logic, i.e. it temporarily stores

(changing) attributes at deployment time and *controls* the processing of data based on the user's input.

Thus, each bean has a certain *scope* and can be *injected* into other beans, providing access to its attributes and operations (= variables and methods) in their current state. Related to practice, a user may fills in the login credentials on the corresponding web page (view) and clicks the button *login*. This action triggers the method *login()* of the *UserBean*, where the user's login data is retrieved from the *Storage*. If the stored password matches the entered password of the user (which is saved in a variable of the UserBean), the variable *authenticated* is set to *true* in the the UserBean, the user successfully logs in to the application and gets redirected to the appropriate starting page.

Because the UserBean is *SessionScoped*, the current data of the bean is present until the user logs out again. Consequently, the UserBean can be injected into further beans: for example, the *DeansViewBean* includes the name of the dean (= logged in user) in the statement file, which is automatically generated when the revision is completed (= the user clicks the button *completeRevision* on the web page *deansWorkspace.xhtml*).

In this way, all beans are created to serve a specific *purpose* within the application, i.e. offering a set of methods and variables to process and hold data for a certain period of time. One bean is responsible for one particular view, and may aid other beans and views.

List of third party software packages

As the Student Records Manager is not built from scratch and utilises existing *software packages* (e.g. the Nuxeo Java Client), which *depend* on additional software packages for their part, a lot of *third party software packages* have to be incorporated by the application in order to work properly. From a developer's perspective, unnecessarily extensive dependencies *should be avoided* in general, but *may not be avoidable* in practice. This dilemma is labeled *dependency hell* commonly. Nevertheless, Table 4.1 lists all third party software packages, that are required to compile and run the Student Records Manager:

Name	Scope
JDK 1.8	Java platform
activation-1.1.jar	compile
bval-jsr303-0.3-incubating.jar	compile
commons-beanutils-1.8.3.jar	compile
commons-collections-3.2.1.jar	compile
commons-digester-1.8.jar	compile
commons-fileupload-1.3.jar	compile
commons-io-2.4.jar	compile
commons-lang-2.4.jar	compile
commons-lang3-3.3.2.jar	compile
commons-logging-1.1.jar	compile
freemarker-2.3.25-incubating.jar	compile

geronimo-annotation_1.1_spec-1.0.jar	compile
geronimo-atinject_1.0_spec-1.0.jar	compile
geronimo-interceptor_1.1_spec-1.0.jar	compile
geronimo-jcdi_1.0_spec-1.0.jar	compile
geronimo-validation_1.0_spec-1.1.jar	compile
guava-18.0.jar	compile
jackson-annotations-2.6.3.jar	compile
jackson-core-2.6.3.jar	compile
jackson-databind-2.6.3.jar	compile
javassist-3.12.0.GA.jar	compile
jsoup-1.6.1.jar	compile
jstl-1.2.jar	compile
log4j-api-2.4.1.jar	compile
log4j-core-2.4.1.jar	compile
mail-1.4.7.jar	compile
myfaces-api-2.1.7.jar	compile
myfaces-extcdi-core-api-1.0.5.jar	compile
myfaces-extcdi-core-impl-1.0.5.jar	compile
myfaces-extcdi-jsf20-module-api-1.0.5.jar	compile
myfaces-extcdi-jsf20-module-impl-1.0.5.jar	compile
myfaces-extcdi-message-module-api-1.0.5.jar	compile
myfaces-extcdi-message-module-impl-1.0.5.jar	compile
nuxeo-java-client-1.0.jar	compile
okhttp-3.0.0-RC1.jar	compile
okio-1.6.0.jar	compile
openwebbeans-impl-1.1.4.jar	compile
openwebbeans-jsf-1.1.4.jar	compile
openwebbeans-resource-1.1.4.jar	compile
openwebbeans-spi-1.1.4.jar	compile
openwebbeans-web-1.1.4.jar	compile
primefaces-3.5-manubu3.jar	compile
relative-resource-handler-1.0.0-TISS_RC2.jar	compile
retrofit-2.0.0-beta4.jar	compile
scannotation-1.0.2.jar	compile
slf4j-api-1.6.1.jar	compile
tomahawk20-1.1.10-tiss-1.jar	compile
geronimo-servlet_3.0_spec-1.0.jar	provided
bval-core-0.3-incubating.jar	runtime
myfaces-impl-2.1.7.jar	runtime
commons-beanutils-core-1.8.3.jar	runtime
commons-codec-1.3.jar	runtime

Table 4.1: list of required third party software packages

Internationalisation

Moving the focus away from the factual *technical* implementation of the Student Records Manager gradually, it is important to understand, how the resulting application is *perceived by the end users* in reality. Ultimately, the whole software and its development process would be rather pointless, if the target audience could not profit from the final product. One key characteristic of the application's presentation is the *language* of the displayed contents: the more translations are provided (by the machine), the less languages have to be learnt (by humans), and the better the usability is (overall).

A common solution in this context is the creation of static resource files: text is stored in modular strings and loaded into the specified places of the web pages at runtime. Thus, the relevant resource file gets selected during program execution based on the user's *locale* settings. Figure 4.8 shows a snippet of the Student Record Manager's resource file for English translation and one corresponding integration:

Messages_en.properties

```
#documents data
addTag = Add tag
allDocuments = All documents
currentlyUploadedDocument = Currently uploaded document
description = Description
documents = Documents
documentsOfAdmissionOffice = Documents of the Admission Office
insufficientDocuments = Insufficient documents
issuedDocuments = Issued documents
mcAndUqeTitle = Matriculation certificate/University qualification evidence
noDocumentUploaded = No document uploaded yet
otherDocuments = Other documents
removeTag = remove tag
requiredDocuments = Required Documents
```

studentsUniversityApplication.xhtml

```
<h3>#{amsg['mcAndUqeTitle']}</h3>
<div class="twoCols">
    <fieldset>
        <ul>
            <li>
                <h4>#{amsg['currentlyUploadedDocument']}</h4>
            </li>
            <li>
                <h:panelGroup rendered="#{empty studentsView.UQEFromStudentsRecord}">
                    <h:outputText>#{amsg['noDocumentUploaded']}</h:outputText>
                </h:panelGroup>
```

Figure 4.8: example of language allocation

Altogether, German and English are supported by the use of *ErrMessages_de.properties*, *ErrMessages_en.properties*, *Messages_de.properties* and *Messages_en.properties*.

4.2.5 Complete the implementation

The last stage of development is concerned with the initial *release* of the Student Records Manager, i.e. enabling other persons the convenient testing of the software prototype. In order to be accessible over the Internet, Nuxeo Platform 7.10 (installed as a comprehensive *virtual machinge image*) and the Student Records Manager application (deployed via Tomcat) are *hosted* on an already existing private server, which is configured appropriately.

As a next step, various types of test data are created within the system, allowing each user to explore the application in different roles (= student, employee or dean) and states. Consequently, the supervisor of this diploma thesis puts the Student Records Manager through its paces and proposes several modifications for the *fine tuning* of the prototype.

This procedure of adjusting and testing the implementation again is repeated multiple times, until the Student Records Manager gets *approved* by the supervisor finally. The current development process ends at this point, and the prototype is ready for the first *evaluation*, as discussed in Chapter 5 in greater detail.

4.3 Overall setup and structure

Recapitulating the genesis of the Student Records Manager, two illustrations shall convey the *essence* of the overall process. At first, Figure 4.9 provides an abstract visualisation of the *setup*, which is chosen to develop and run the Student Records Manager:

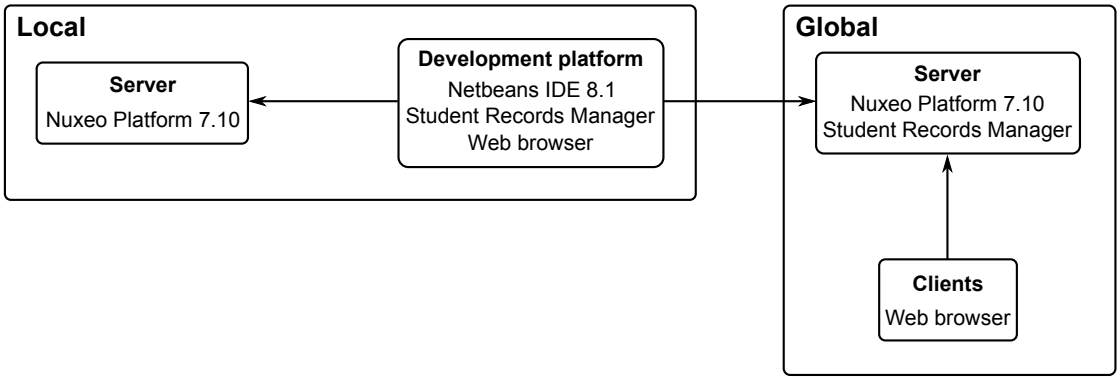

Figure 4.9: recap of the Student Record Manager's basic setup

Narrowed down to the substantial parts, the Student Records Manager is developed on a local computer, and may be deployed locally, or globally across the internet, using Apache Tomcat in each case. Nuxeo Platform 7.10 establishes the backend of the whole application, and a web browser is subjected to display the presentation at the frontend. Of course, this *setup* is not set in tablets of stone: it is just *one possible combination of hardware and software*, that *enables* the development and deployment of the Student Records Manager.

The technical implementation of the Student Records Manager is summarised by means of an example: starting at the *user interface*, a snippet of the employee's workspace is shown in Figure 4.10. All documents of the currently opened student's university application are listed and have to be evaluated to complete the review. In this case, only the required university qualification evidence is uploaded and tagged with the string *test*, which could be removed by the employee:

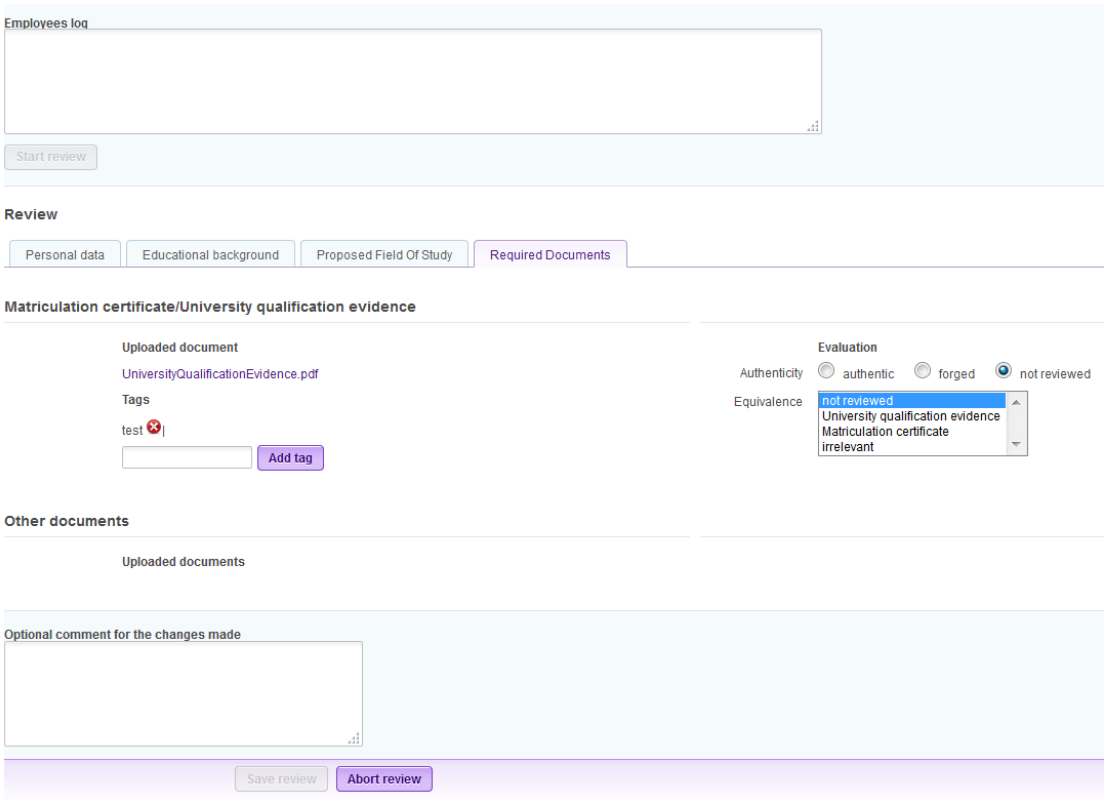

Figure 4.10: recap of the Student Record Manager's structure, user interface

Tracing back the source of the tag, the appropriate *view* of the presentation has to be examined, i.e. a part of *employeesWorkspace.xhtml* as seen in Figure 4.11:

```
<ui:repeat value="#{ProviderBean.getTagsFromDocument(employeesView.studentsRecordID,
                    'UniversityQualificationEvidence')}" var="tagsUQE">
    #{tagsUQE} 
```

Figure 4.11: recap of the Student Record Manager's structure, view

With the help of *<ui:repeat>*, all returned elements of the method specified in *value="..."* are written out at runtime. The *#{...}* indicates the usage of Java *Expression Language*, i.e. the relevant method in the *controller* is invoked accordingly. Figure 4.12 reveals the accountable code in *ProviderBean.java*:

```
public List<String> getTagsFromDocument(String studentsRecordID, String whichDocument) {

    List<String> tags = Provider.getTagsFromDocument(studentsRecordID, whichDocument);

    return tags;
}
```

Figure 4.12: recap of the Student Record Manager's structure, controller

As the fetching of tags is a quite simple scenario within the application, only the corresponding method in *Provider.java* has to be called. In terms of the big picture, the *model* layer is reached by now, as illustrated in Figure 4.13:

```
public static List<String> getTagsFromDocument(String studentsRecordID, String whichDocument) {
    return studentsRecordDAO.getTagsFromDocument(studentsRecordID, whichDocument);
}
```

Figure 4.13: recap of the Student Record Manager's structure, model

It is evident, that the method *getTagsFromDocument(...)* passes on a list of *strings*: therefore, it instructs the implementation of the data access object (via its implemented interface) to return a *certain data type*, which can be processed by the application. Figure 4.14 illustrates this principle:

```
public class StudentsRecordDAONuxeo implements StudentsRecordDAO {
⋮
@Override
public List<String> getTagsFromDocument(String studentsRecordID, String whichDocument) {

    List<String> tagList = new ArrayList<String>();

    try {

        if (whichDocument.contains(".")) {
            whichDocument = whichDocument.substring(0, whichDocument.lastIndexOf("."));
        }

        whichDocument = studentsRecordID + "_" + whichDocument;

        Documents doc = nuxeoClient.repository().query("SELECT * FROM Document WHERE dc:title='"
                + whichDocument + "'");

        // "Tagging" contains only tags, which are currently related to an existing document
        Documents tags = nuxeoClient.repository().query("SELECT * FROM Tagging WHERE relation:source='"
                + doc.getDocument(0).getUid() + "'");

        // here we get tags, which are also stored as "documents", and select their titles
        for (Document child : tags.getDocuments()) {

            tagList.add(child.getTitle());
        }
    } // thrown by Nuxeo, if the document does not exist (= if no document is uploaded yet)
    catch (Exception ex) {

        tagList.add("");
    }
    return tagList;
}
```

Figure 4.14: recap of the Student Record Manager's structure, data access object

Thus, the implementation of the DAO could be replaced without affecting the other components of the Student Records Manager. Nuxeo Platform has its own persistence mechanisms to store and relate tags then again, and guides the process down to its utilised database system in final consequence.

Although the presented example covers a less complex element of the Student Records Manager, the general procedure is similiar for the majority of features. Ultimately, it is all about transforming information to data and recombining data to information in turn.

4.4 The Student Records Manager

Eventually, the apparently most important part of the Student Records Manager is *the user interface*: it depicts the *enabling* component, which should allow people to perform their individual tasks in a *more efficient way* than (ever) before. This means in practice, that the application fails regardless of its theoretical potential, if the interface does not meet *the users' expectations*. Hence, it is reasonable to empathise with the role of a user in each stage of development, i.e. incorporating the viewpoint of a developer *and* the perspectives of possible end users in their environmental contexts. Consequently, some aspects of this *connected thinking* are discussed in the following subchapters.

4.4.1 Elementary concepts

Overall, the material understanding of the term *design* is inspired by the work of Norman [41]: essentially, each perceived object of the world has an *affordance* for humans, i.e. it suggests one or more particular actions to be taken with it. Thus, the elements of the Student Records Manager's user interface shall be *clear and straightforward* to the user, ruling out misinterpretations by the design itself. In this regard, all elements have to serve a certain purpose, and the presentation is as *minimalistic* as necessary to *provide* the required functionality.

If some operations may not be obvious in the first place, information about their *intended usage* is embedded in each web page additionally. One core idea of this *transparent* design is to *introduce* users into the application's ways of working, i.e. motivate them to comprehend the underlying processes and technologies at least partially. Because it is much more favourable for users to get an *abstract impression* of their (daily) tasks, rather than memorising and repeating only the specific series of actions, which are needed to perform the tasks. On the whole, the user interface *just* has to comply with the *natural behaviours* of humans.

Related to practice, finding the right balance between *automation* and *manual human actions* is key to any software. In case of the Student Records Manager, a simple *lifecycle* principle is applied to preserve this conformity: each student record has *one state at a time*, which defines its logical position within the system. These states in turn are assigned automatically by the system in consequence of the interactions done by the users.

Stating a real example, an employee may choose to grant the admission for a bachelor programme application (= click the appropriate button in the user interface) according to the preceding review of the documents' equivalence and authenticity: therefore, the state of the student record (= state of the corresponding folder in the Nuxeo repository) changes from *bachelor programme application* to *student record* automatically.

Based on this concept, student records appear in one or more lists presented in the user interface: thus, the background idea proposes that a single student record is always stored at a *certain location* and ready to be processed further on from this point. In this way, individual operations can be joined together and grouped to larger tasks. The following description illustrates the operations of the (conceivable) task "*handle bachelor programme application*":

If the student (applicant) confirms the first registration, a student record with the student's reference number (= UserID) is set up in the system and gets the state *pre-enrolment record created* consequently. When the applicant has uploaded the required documents and sends the completed application, the state of the student record switches to *bachelor programme application*. The list of the employees' web page *open applications* shows all student records with the state *bachelor programme application* in turn. Next, an employee opens the application within the workspace, does the review, and chooses *call for improvements* as result of the evaluation. The student record's state changes to *bachelor programme call for improvements*. Now, the applicant is able to upload the required (improved) documents and send the application again. The state of the student record changes to *bachelor programme application* anew, and the application can be reviewed by the employee once more. Finally, the admission to the bachelor programme is granted and the student record remains as a fully qualified *student record* in the system henceforth. As a conclusion to the discussed conception, Table 4.2 lists all (prototypical) states, that are possible for a student record within the scope of the Student Records Manager:

Initial state	Trigger	In role	Subsequent state
	confirm registration	student	pre-enrolment record created
pre-enrolment record created	send application	student	bachelor programme application
bachelor programme application	grant admission	employee	student record
bachelor programme application	decline admission	employee	declined
bachelor programme application	call for improvements	employee	bachelor programme call for improvements
bachelor programme call for improvements	send application	student	bachelor programme application
master or doctoral programme application	complete review	dean	statement compiled

Table 4.2: student record lifecycle and states

By this means, a student record acts as a *living object*, whose *existence* constitutes the execution of tasks by transporting a *meaningful* set of information to humans. In order to complement the overall appearance of the user interface, a short analysis on the most important features of each *view* is carried out in the next sections.

4.4.2 Student's view

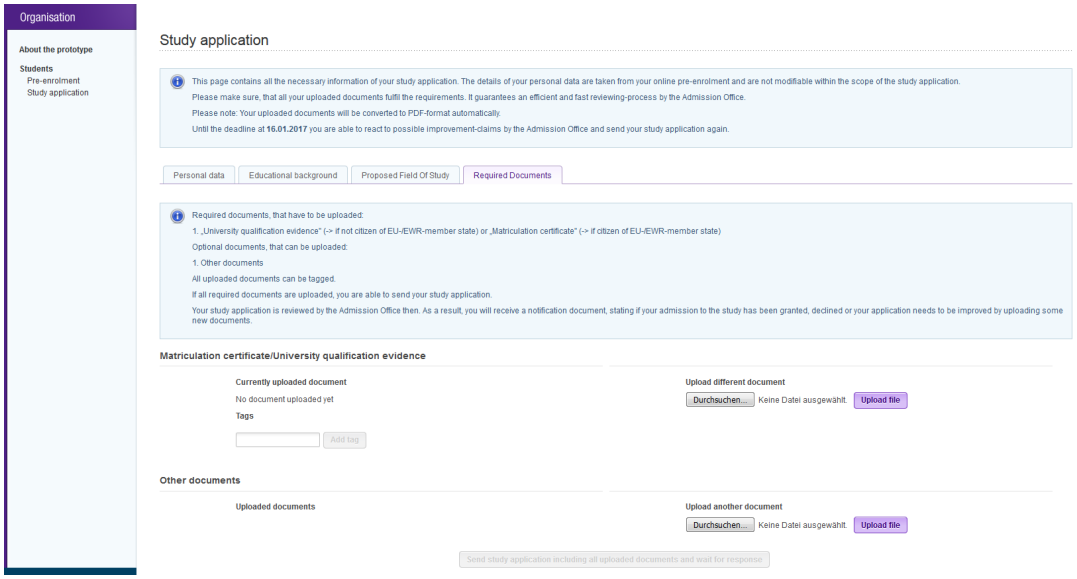

Figure 4.15: user interface for conducting a bachelor programme application

As shown in Figure 4.15, the main intent to be accomplished with the help of the student's view is the *provision of relevant information*: a student uses the system to apply to a bachelor progamme, i.e. it serves a certain, well-defined purpose. In this regard, it is substantial to *know*, which documents must be uploaded, and whether the application is successful, or not. Thus, the view is only of temporarily interest for the student, and should include the *minimal information*, that is required for the completion of the task. One mechanism in this context is the presentation of *state-dependent* information, as illustrated in 4.16:

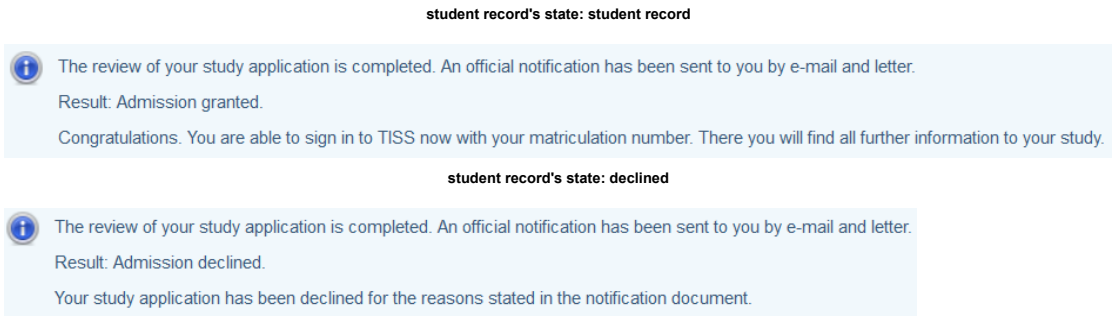

Figure 4.16: displaying state-dependent information within the user interface

Hence, the student always has the *full* information on the current state of the application and may react accordingly.

4.4.3 Employee's view

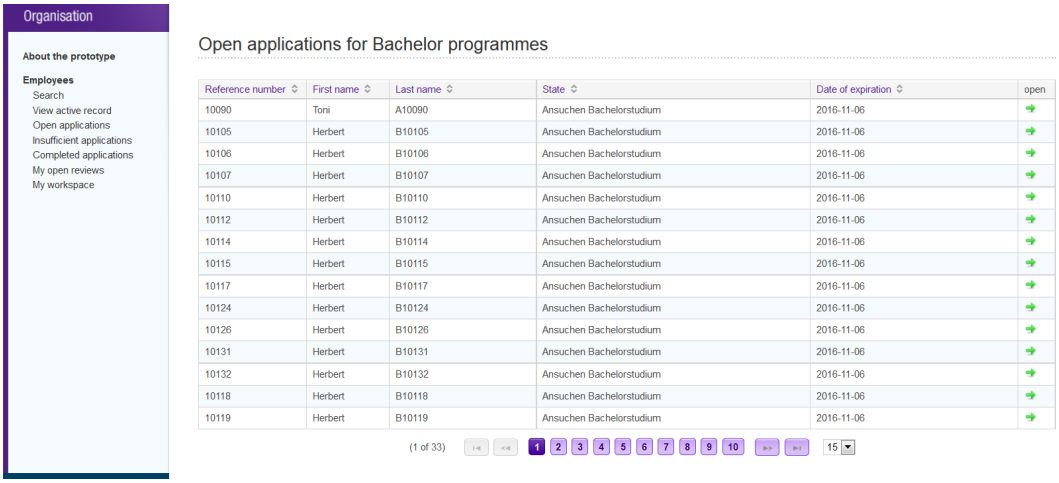

Figure 4.17: user interface, open applications for bachelor programmes

Referring to 4.17, the employee's view rests upon *lists* in principle. Each list aggregates student records in a specific state and satisfies a particular function for the employee (for reasons of uniformity, states have a German naming within the prototype, as some other elements do). Based on the type of the list, employees may be able to open and review a student record in their workspace, or explore its contents (= master data and documents) by the use of a separate web page (*view active record*).

In addition, a search engine allows to retrieve student records from the Nuxeo repository, i.e. generating *individual* lists of student records as a starting point for further operations. Figure 4.18 outlines this functionality:

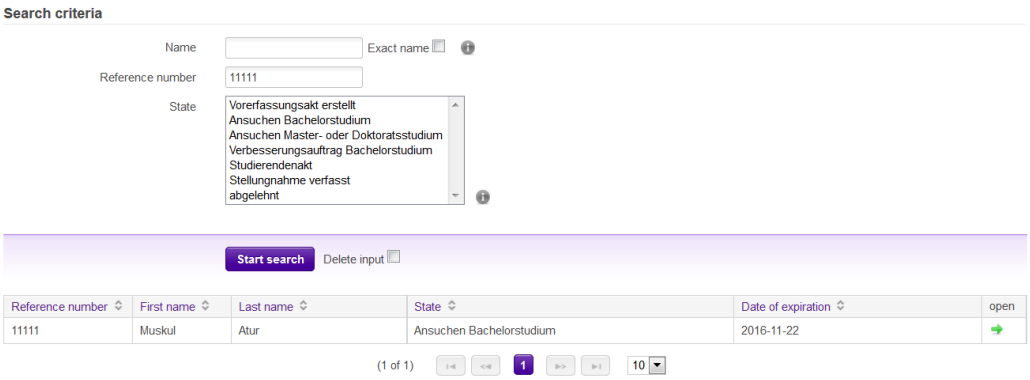

Figure 4.18: user interface, search engine

Thus, employees do have a certain leeway to customise the workflows according to their preferences. On the other hand, the workspace guarantees, that only *one* student record is *actively* reviewed by an employee at a time.

4.4.4 Dean's view

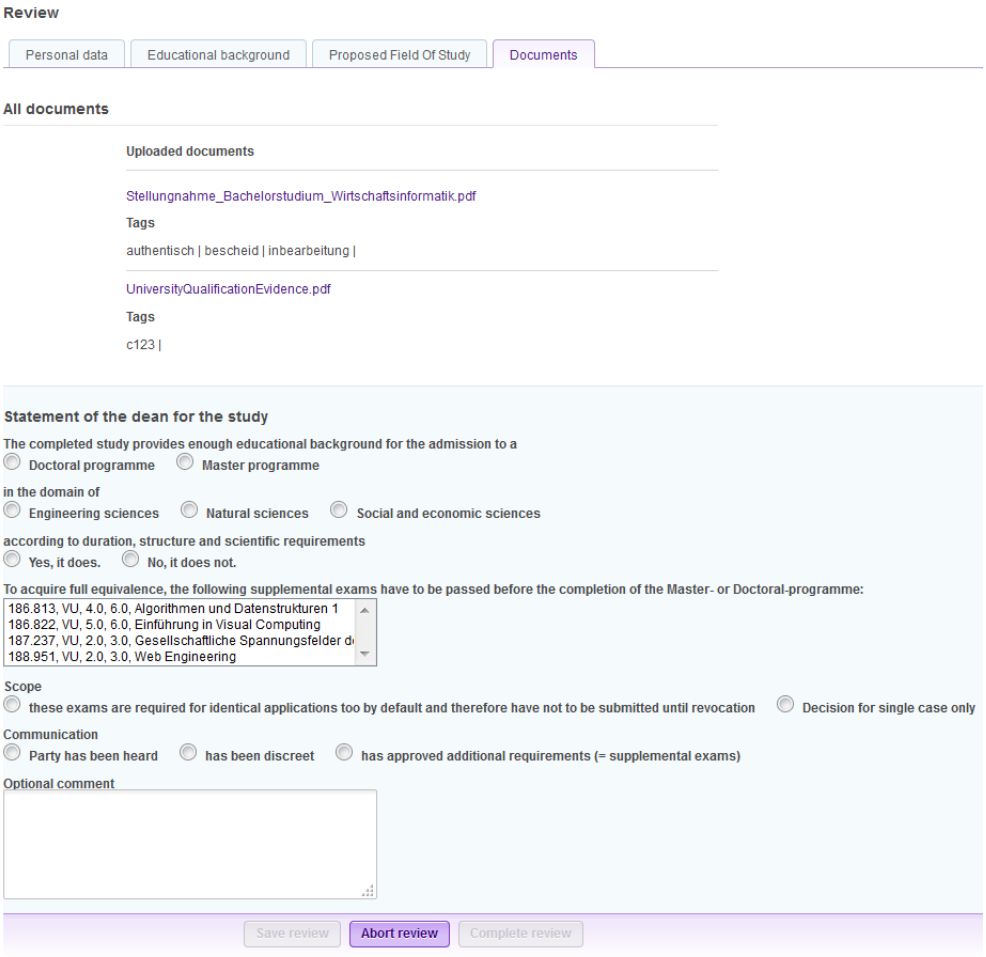

Figure 4.19: user interface, reviewing a master programme application

As shown in 4.19, the dean's view can be described as a *mixture* of the employee's and student's views' concepts: deans use the system on a regular basis, and do focus on the same task for the most part, i.e. composing a statement about a student's application to a master or doctoral programme.

Therefore, several lists can be utilised, as well as a central workspace does exist. Overall, deans spend significantly less time within the system than employees, but should obtain the same degree of information. In order to ensure an efficient process, the required information for writing a *justified* statement is available at a glance, as well as the statement document is generated and appended to the student record automatically.

4.4.5 Limitations and further development

Although the Student Records Manager runs stable in its testing environment, and gives a promising impression altogether, it represents a software *prototype* with several *limitations*: first and foremost, it is hardly possible to anticipate all (technical) requirements of the *real* application environment, where the SRM could be integrated into in future. Hence, it can not be assumed, that the prototype is operable *out of the box*. Reasonable efforts would be necessary to connect the SRM with existing persistence mechanisms (e.g. incorporating the original master data of students), as well as a consistent security policy has to be established. Furthermore, some methods of the prototype may create *bottlenecks*, which must be resolved to optimise the response time of the application. In terms of *accessibility*, additional adaptions could improve the inclusion of people with special needs, i.e. providing interfaces for speech output and other useful tools, that facilitate the perception of the application.

Considering the pure content and functionality of the Student Records Manager, manifold enhancements are imaginable: advanced *automation chains* could increase the overall productivity of the SRM by attaching supplementary processes to manual actions of users. For example, the statements of deans would be printed out automatically at the *Admission Office* in consequence of a completed review. Employees may scan paper documents, that are allocated to the corresponding student records directly. Moreover, the generic *list* concept allows the flexible gathering and presentation of (the contents of) student records for nearly any purpose. Thus, the Student Records Manager offers a number of modular approaches, which could be refined in succeeding versions of the software.

CHAPTER 5

Evaluation

This chapter reports on the *evaluation* of the software prototype. Eventually, the Student Records Manager is a *product* designed for a specific goal, i.e. the *requirements* determined in cooperation with the customer in the project planning. Therefore, the development process describes the *way* to the final product, but not the *quality* of the product itself. This is the point, where the *evaluation* comes into effect: a procedure, which permits an objective assessment on the *performance* of the product. By this means, it is possible to draw *realistic* conclusions in respect of an appropriate future strategy for the product. Thus, the evaluation is a tool that aids *decision-making* and suggests *immediate measures to be taken*. Put into practice, some valuable hints on the humans' *perception* of the Student Records Manager can be obtained and harnessed to improve its general usability.

5.1 Approach

As the Student Records Manager is launched for the first time, it seems legit to conduct a *summative* evaluation above all. This *result-orientated* evaluation approach is aimed at providing information on the suitability of the prototype's *core operating principles*, i.e. it states, if the basic mechanisms of the SRM are understandable for users. Hence, it is advisable to decouple the SRM from the *application context*, enabling *testers* to focus on the pure functionality, instead of getting lost in or influenced by details of the content maybe. Thus, a subset of *five* people, which are not associated with the *Admission Office* at the Vienna University of Technology, is randomly selected in the beginning. As a next step, the link (= Uniform Resource Locator, URL) to a *online questionnaire* is sent to the testers. The corresponding web page contains only *minimal information* about the purpose of the Student Records Manager and the valid address (URL) to access the application via a web browser.

Consequently, all testers are in a position, where they instantly have to *make sense* of the Student Records Manager in their current situation: as a result, it can be estimated, *how*

the elementary functions of the application are *perceivable* by means of the different views. In the course of this *explorative process*, testers are invited to answer the questionnaire and evaluate the prototype thereby.

5.1.1 Questionnaire

As shown in Figure 5.1, the *questionnaire* is tailored to the specific characteristics of the Student Records Manager and consists of *ten* questions in total. Besides the first question, which is ought to supply information about the testers, and the optional last question on how to improve the prototype, each question shall be answered according to a *grading scheme* from *one* (best) to *five* (worst):

Figure 5.1: evaluation questionnaire

5.2 Results

Once the testers have provided their answers, it is possible to present and discuss the overall results of the evaluation. To begin with, Figure 5.2 summarises the testers' *experience using software and Document Management Systems in particular*:

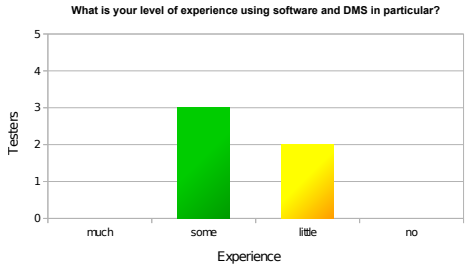

Figure 5.2: evaluation, results, question 1

Obviously, the group of testers is comprised of *average computer users*, which are no experts in the fields of document management on the one hand, but should have a general idea of some concepts and certain practical skills on the other hand. Thus, the final results may be quite representative and could be referenced for further evaluations of the Student Records Manager in future.

Considering the information about the testers as a starting point, all findings of this initial evaluation are categorised and analysed in the following subchapters.

5.2.1 Functionality of the prototype

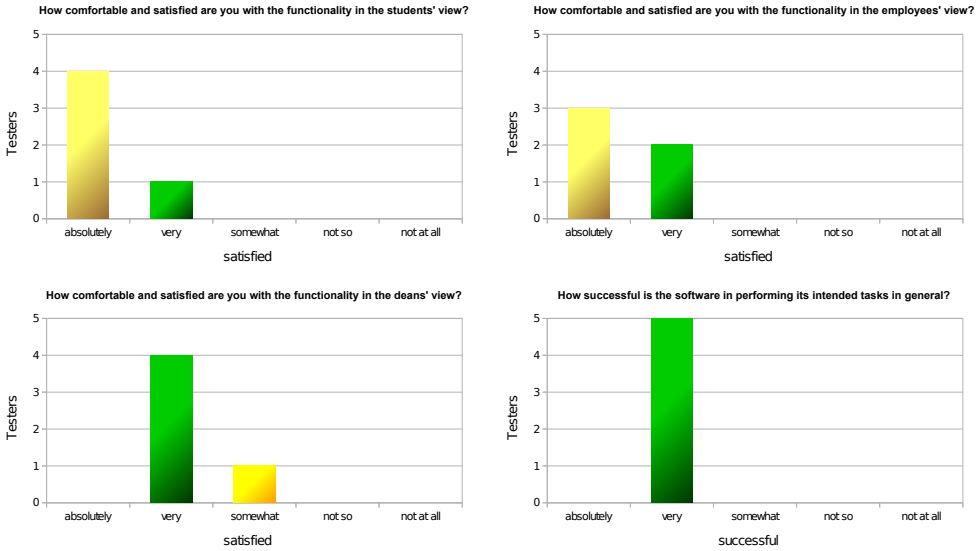

Figure 5.3: evaluation, results, question 3, 4, 5 and 6

Based on the bar charts of Figure 5.3, it is concluded, that the core functionality offered by the Student Records Manager should be *sufficient* to meet the demands agreed with the customer. Especially the student's view seems to approximate to an optimal solution already, while the dean's view performs worse in comparison. This observation may be put down to the peculiarity of the dean's workspace, where rather specific knowledge is required to *compose the statement* on a student's appliction to a master or doctoral programme.

The functions of the employee's view then again appear to be reasonably comprehensible for the testing persons, i.e. the *list concept* is supposed to be a reliable mechanism. Interestingly, all testers are in complete agreement, that the software does have *additional potential* to be exploited in terms of the presented functionality: one interpretation of this assessment may be, that the prototype operates well *in the aggregate*, but could profit from improvements *in detail*.

5.2.2 Usability of the prototype

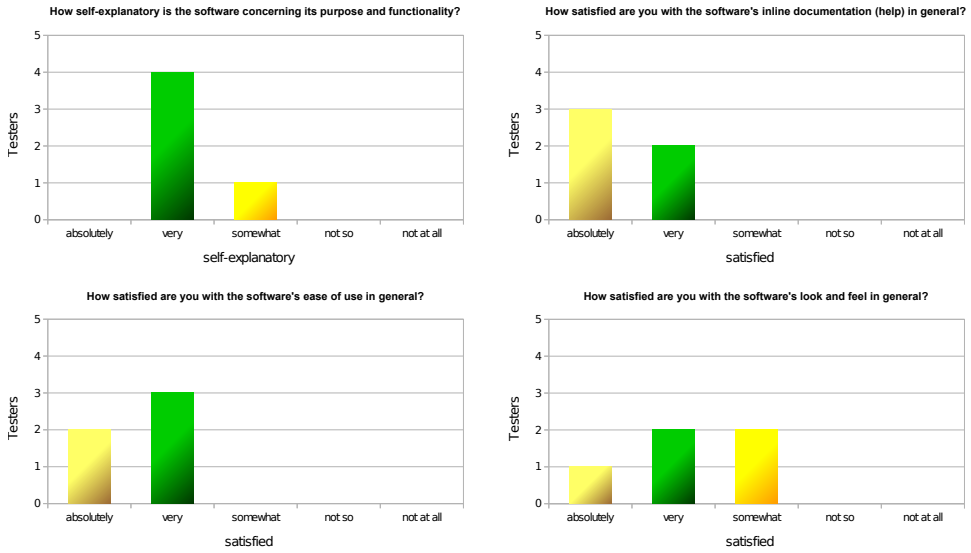

Figure 5.4: evaluation, results, question 2, 7, 8 and 9

Figure 5.4 reveals the evaluation results with regard to the *usability* of the Student Records Manager, i.e. the testers' *subjective experience* while examing the functionality of the prototype. In general, the testing persons seem to be able to use the software quite *intuitively*: therefore, the *layout and design* of the views (= web pages, XHTML documents) appears to be acceptable, as well as the text elements are understandable.

Thus, it could be argued, that the *inline documentation* is not explicitly necessary in the first place, but does contribute to a positive impression overall: if the occasion arises, users would rely on the *provided help*. Likewise, the testers confirm the *ease of use* of the Student Records Manager, i.e. there is no need for humans to adapt to the user

interface in an unfamiliar way, because it can be perceived *naturally* by them. Finally, one lesson should be learned from the evaluation of the software's *look and feel*, which may be considered as *too old-fashioned* for some testers.

5.2.3 Improvement of the prototype

Altogether, the current version of the Student Records Manager is rated fairly well, and should have no particular shortcoming, that has to be remedied *in principle*. Nevertheless, the testing persons submit several valuable suggestions for the enhancement of the prototype, which are taken up and elaborated in the following notions consequently.

Generally, it could make sense to include an additional type of help in the views: this concept may be summarised best as *direct feedback to user actions*. The basic idea states, that any action (= usage of functionality, e.g. click on a button, entering text into an input field) of a user can cause a certain effect on the system. This concrete outcome in turn could be displayed *directly* in the view, i.e. in spatial or logical proximity to the respective trigger. Related to practice, green check marks may be rendered besides the corresponding selection box, if an employee has chosen a result for the review of a document's authenticity. Or a text field with useful information will be presented to the employee, when the review of a student's university application is completed (currently, employees see only the blank page of their workspace then). All these *little assistances* can be comprised as *visual cues*, that facilitate the interaction process in a more dynamic way, than static contens may do. Of course, this does not mean to *overload* the presentation with irrelevant information.

On the contrary, one interesting thought proposes a *pure graphical user interface*, with no text at all: obviously, the representation of document names and master data of students certainly requires a textual description in any case, but some aspects could be more perceptible by the use of (universal) *symbols*. This approach has the potential to be analysed from the perspective of *cognition and workload* at least.

Further comments of the testers refer to an extended functionality of the Student Records Manager, e.g. an *internal messaging system*, where employees are able to communicate (synchronously and asynchronously) with other participants. Students perhaps ask questions concerning their university application, while deans could request employees to feed specific data (e.g. a certificate of a student) into the repository: overall, this could be a reasonable mechanism to foster *collaboration*.

Ultimately, the next step in the *evolution* of the Student Records Manager should be the implementation at the *Admission Office*, which constitutes the intended area of application for the software product. As a consequence, a supplementary *evaluation* by the employees of the Admission Office would be meaningful in order to attune the prototype to the prevalent *context*. As soon as the application is in full use, a *formative evaluation* could be scheduled to preserve its factual quality, i.e. adapting the software on a regular basis to the changing requirements if necessary.

CHAPTER 6

Conclusion

This chapter is designated to conclude the *origination* of the Student Records Manager by *reflecting* on the previous work and its results. Recapitulating the course of events, everything starts with the *human notion*, that the Vienna University of Technology as a sum of its parts could profit from a consistent electronic document management system. Consequently, a *constitutive document* is elaborated in cooperation with responsible stakeholders to *translate* the rather *indefinite* objective to a set of feasible goals: the draft for the *Student Records Manager* is born. This prototypical piece of software shall be formed to prove the *viability* of the overall conception.

In the beginning, it is quite difficult to estimate the effort and the required measures for the task, because there obviously is no preexisting *manual* on how to accomplish the assignment. Hence, the most important part of work is concerned with *researching and planning* an appropriate process, that *should* allow to produce the desired result eventually. Thus, from a personal point of view, the process itself is more valuable than the practical product in the final analysis, as it conveys skills on a *meta level*: once the *big picture* is understood, only the components have to be selected and arranged to create a functional system for a specific purpose.

In this way, it is possible to define certain *milestones*, which help to maintain the motivation over time, and build the Student Records Manager step-by-step. Of course, *work remains work*, and *there are phases of procrastination*, where *many things should be done, but nothing could be done* by the individual human being: in this regard, taking a break deliberately often seems to be more difficult than keeping up a steady, but ineffective workflow. Altogether, it turns out to be vital to *identify* the goal of the work, and devote the same attention to each task, as an apparently insignificant detail may be of greater importance in the further proceeding.

The power of *researching* should not be neglected too, because nearly anything imaginable does already exist or has been done before de facto. Thus, *only the necessary information*

has to be retrieved in order to combine it to a new manifestation: by this means, *Nuxeo Platform* is discovered at a later stage of the market sounding, and easily could have been missed in a less thoroughly enquiry. Investing effort in advance to study and comprehend the tools used for the implementation of the Student Records Manager helps to save time subsequently as well: plain and simple, *thinking ahead* is better than thinking, when it is too late.

One particular interesting experience is related to the *rapid progress* of the applied technologies: several new versions of different tools and software packages are published during the development of the prototype. This can mislead to the fallacy, that the own working process in comparison takes way too long and never reaches a decent quality, although it already *has a sufficient quality by objective aspects.* To counteract this *natural strive for optimisation*, it is crucial to *vary the focus interest* constantly, i.e. avoiding to get stuck in details.

Thus, as long as humans are involved in a *process*, it is hardly possible to predict the true outcome. Therefore, it can only be guessed, how much potential of the Student Records Manager (and Nuxeo Platform) could be exploited in a *real-life scenario*: first and foremost, the performance of an EDMS-based system depends on the *seriousness and consequence* of its implementation. It is a tool, that needs to be utilised and cultivated on a long term, or does fail within a short period of time, if the required conditions are not met initially.

A major advantage in this respect could be the fact, that the Student Records Manager would be customised *in-house* by the department of Campus Software Development: in theory, this should enable the implementation of a very tailored solution in close cooperation with the *Admission Office*. Again, the actual quality of the result would be a product of the effort both parties are willing to invest in the relationship.

Overall, the Student Records Manager shall be created to provide *additional value* to people and incorporate their natural behaviours. Hence, the application should simplify the execution of human tasks while preserving the quality of each task. Consequently, users are supported by the *automation* of particular processes, but do have the *obligation* to employ their *human reason* equally. This is especially true for the the handling of *information-based* tasks, e.g. the review of a student's university application.

In this sense, it will be very interesting to see, which strategies are chosen to cope with *Big Data* in future: ultimately, *data depict a capture of time, and time will ever be superior to data.* Thus, the universal challenge arises in transforming data into meaningful information and allocating the information to the appropriate entity *at a certain point of time.*

List of Figures

List of Tables

Bibliography

[1] Kjeld Schmidt and Liam Bannon: "Constructing CSCW: The first quarter century", *Computer supported cooperative work (CSCW) 22.4-6: 345-372, Springer, 2013.*

[2] Irene Greif: "Computer-Supported Cooperative Work: A Book of Readings", *Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1988.*

[3] Gwendolyn L. Kolfschoten, Thomas Herrmann and Stephan Lukosch: "Differentiated awareness-support in computer supported collaborative work", *Computer Supported Cooperative Work (CSCW) 22.2-3: 107-112, Springer, 2013.*

[4] Paul Dourish and Victoria Bellotti: "Awareness and coordination in shared workspaces", *Proceedings of the ACM 1992 conference on computer-supported cooperative work: 107-114, ACM, 1992.*

[5] Carl Gutwin, Saul Greenberg, and Mark Roseman: "Workspace awareness in real-time distributed groupware: Framework, widgets, and evaluation", *Proceedings of HCI on People and Computers XI.: 281-298, Springer, 1996.*

[6] James Tam and Saul Greenberg: "A framework for asynchronous change awareness in collaborative documents and workspaces", *International Journal of Human-Computer Studies 64.7: 583-598, Elsevier, 2006.*

[7] Ronald Poelman, Oytun Akman, Stephan Lukosch and Pieter Jonker: "As if being there: mediated reality for crime scene investigation", *Proceedings of the ACM 2012 conference on computer-supported cooperative work, ACM, 2012.*

[8] Jakob E. Bardram and Thomas R. Hansen: "Context-based workplace awareness", *Computer Supported Cooperative Work (CSCW) 19.2: 105-138, Springer, 2010.*

[9] Luigina Ciolfi and Aparecido Fabiano Pinatti De Carvalho: "Work practices, nomadicity and the mediational role of technology", *Computer Supported Cooperative Work (CSCW) 23.2: 119-136, Springer, 2014.*

[10] Gordon B. Davis: "Anytime/anyplace computing and the future of knowledge work", *Communications of the ACM 45.12: 67-73, ACM, 2002.*

[11] Mark S. Ackerman, Juri Dachtera, Volkmar Pipek and Volker Wulf: "Sharing knowledge and expertise: The CSCW view of knowledge management", *Computer Supported Cooperative Work (CSCW) 22.4-6: 531-573, Springer, 2013.*

[12] Ikujiro Nonaka and Hirotaka Takeuchi: "The knowledge creation company: how Japanese companies create the dynamics of innovation", *Oxford University Press, New York, USA, 1995.*

[13] Jean Lave and Etienne Wenger: "Situated learning: Legitimate peripheral participation", *Learning in Doing: Social, Cognitive and Computational Perspectives, Cambridge University Press, UK, 1991.*

[14] Janine Nahapiet and Sumantra Ghoshal: "Social capital, intellectual capital, and the organizational advantage", *Academy of management review 23.2: 242-266, AOM, 1998.*

[15] Jørgen P. Bansler and Erling Havn: "Sensemaking in technology-use mediation: Adapting groupware technology in organizations", *Computer Supported Cooperative Work (CSCW) 15.1: 55-91, Springer, 2006.*

[16] J.H. Erik Andriessen, Marike Hettinga and Volker Wulf: "Introduction to special issue on evolving use of groupware", *Computer Supported Cooperative Work (CSCW) 12.4: 367-380, Springer, 2003.*

[17] Claudio Ciborra: "Chapter 1: Introduction: What Does Groupware Mean to the Organizations Hosting it.", *Groupware and teamwork: invisible aid or technical hindrance?, John Wiley & Sons, New York, USA, 1996.*

[18] Gloria Mark: "Conventions and commitments in distributed CSCW groups", *Computer Supported Cooperative Work (CSCW) 11.3-4: 349-387, Springer, 2002.*

[19] Wanda J. Orlikowski, JoAnne Yates, Kazuo Okamura and Masayo Fujimoto: "Shaping electronic communication: the metastructuring of technology in the context of use", *Organization science 6.4: 423-444, INFORMS, 1995.*

[20] Alan Cooper, Robert Reimann and Dave Cronin: "About face 3: the essentials of interaction design", *John Wiley & Sons, New York, USA, 2007.*

[21] Jan M. Noyes and Kate J. Garland: "Computer-vs. paper-based tasks: Are they equivalent?" *Ergonomics 51.9: 1352-1375, Taylor & Francis, 2008.*

[22] Erik Wästlund, Henrik Reinikka, Torsten Norlander and Trevor Archer: "Effects of VDT and paper presentation on consumption and production of information: Psychological and physiological factors", *Computers in Human Behavior 21.2: 377-394, Elsevier, 2005.*

[23] Jan M. Noyes, Kate J. Garland and Liz Robbins: "Paper-based versus computer-based assessment: is workload another test mode effect?", *British Journal of Educational Technology 35.1: 111-113, John Wiley & Sons, 2004.*

[24] Pamela Ravasio, Sissel Guttormsen Schär and Helmut Krueger: "In pursuit of desktop evolution: User problems and practices with modern desktop systems", *ACM Transactions on Computer-Human Interaction (TOCHI) 11.2: 156-180, ACM, 2004.*

[25] Olha Bondarenko, Ruud Janssen and Samuël Driessen: "Requirements for the design of a personal document-management system", *Journal of the American Society for Information Science and Technology 61.3: 468-482, John Wiley & Sons, 2010.*

[26] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze: "Introduction to information retrieval", *Cambridge University Press, UK, 2008.*

[27] Bernard J. Jansen and Soo Young Rieh: "The seventeen theoretical constructs of information searching and information retrieval", *Journal of the American Society for Information Science and Technology 61.8: 1517-1534, John Wiley & Sons, 2010.*

[28] Jörg Dandl: "Dokumenten-Management-Systeme – Eine Einführung", *Arbeitspapiere WI, Nr. 9/1999, Johannes Gutenberg-Universität, Mainz, 1999.*

[29] John Mancini: "8 reasons you need a strategy for managing information", *AIIM, 2009.*

[30] Ulrich Kampffmeyer: "Dokumentenmanagement", *PROJECT CONSULT, Hamburg, Germany, 2005.*

[31] Ulrich Kampffmeyer: "EIM Enterprise Information Management"", *PROJECT CONSULT, Hamburg, Germany, 2013.*

[32] Toms Leikums: "A study on electronic document management system integration needs in the public sector", *International Journal of Advances in Engineering & Technology 5.1: 194-205, IJAET, 2012.*

[33] Michael Raynes: "Document management: is the time now right?", *Work study 51.6: 303-308, Emerald, 2002.*

[34] Ralph H. Sprague, Jr.: "Electronic document management: Challenges and opportunities for information systems managers", *MIS Quarterly 19.1: 29-49, University of Minnesota, 1995.*

[35] Bettina Perthold-Stoitzner: "Universitätsgesetz 2002 - UG", *Manzsche Sonder-Gesetzausgaben, MANZ'sche Wien, 2016.*

[36] Ajith Ranabahu, Priti Parikh, Maryam Panahiazar, Amit Sheth and Flora Logan-Klumpler: "Kino: a generic document management system for biologists using SA-REST and faceted search", *Proceedings of the Fifth IEEE International Conference on Semantic Computing, IEEE, 2011.*

[37] Mansour N. Jadid and Mobin Idrees: "Electronic document management system (EDMS) in civil engineering projects", *6th Construction Specialty Conference, University of British Columbia, 2005.*

[38] Dina Salah, Richard F. Paige and Paul Cairns: "A systematic literature review for agile development processes and user centred design integration", *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, ACM, 2014.*

[39] "Nuxeo Fact Sheet", *Nuxeo, 2016.*

[40] "Developer Documentation Center: Nuxeo Server", *Nuxeo, 2016.*

[41] Don Norman: "The Design of Everyday Things: Revised and Expanded Edition", *Basic Books, Perseus Books Group, USA, 2013.*