Zur Erlangung des akademischen Grades eines
Doktors der Wirtschaftswissenschaften (Dr. rer. pol.)
von der Fakultät für Wirtschaftswissenschaften
der Universität Karlsruhe (TH) genehmigte

Dissertation

Using Unified Personal Information in Workspaces

Dipl. Wirtsch.-Inform. Olaf Grebner

Tag der mündlichen Prüfung: 16. Juli 2009

Referent: Prof. Dr. Rudi Studer

Korreferent: Prof. Dr. Peter Knauth

2009 Karlsruhe

Abstract

Knowledge work gains more and more importance for the workforce. Thereby knowledge workers (KWers) deal with information personally relevant to them and their work. Furthermore, they use tools to support their work like, e.g., desktop workspaces and specialized applications.

However, KWer support is hindered in these applications by personal information fragmentation, i.e., applications needed for a particular activity at hand keep their own set of personal information while not interconnecting this. The KWer faces redundant work to manage this information in several applications and lacks an overview on available information.

This thesis shows at hand of two application domains, personal task management and personal meeting management, how to address the personal information fragmentation problem for the support of particular, higher-level KWer activities across individual application boundaries. By using a common unified personal information model and building the applications on top of a personal information management (PIM) system like the recently proposed semantic desktops, e.g., the Nepomuk Social Semantic Desktop, one unified set of a KWer's personal information can be provided across individual applications in a workspace.

In particular, this thesis shows a framework with information models, reference architecture and example applications leveraging the former. The information model extends the unified personal information model in the particular domains, i.e., an ontology models respectively the personal task and personal meeting management domain. The reference architecture shows how to efficiently build applications which use unified personal information. The applications for personal task management (Kasimir) and personal meeting management (Nepomuk Meeting Manager) have been designed using this unified personal information and feature methods on how to leverage this information for the KWer.

The presented applications have been evaluated with end-users by combining both formative and summative evaluation techniques. Formative usability tests ensured the continuous incorporation of user feedback in the whole design process, ranging from early evaluating design ideas in paper-based prototypes over to evaluating these features in working prototypes of implemented applications. In a summative evaluation, 13 users worked throughout 3.5 weeks with a matured version of the prototype application.

This work represents to a large extent an interdisciplinary work among the research fields of semantic web technology, personal information management and human-computer interaction.

Content Overview

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Description
API	Application Programming Interface
AT	Activity Theory
CRM	Customer relationship management
CSCW	Computer-supported cooperative work
DAO	Data access object
DI	Domain independent
DS	Domain specific
Eds.	Editors
ERP	Enterprise resource planning
HCI	Human-computer-interaction
KM	Knowledge management
KWer	Knowledge worker
MB	Mapping Builder
MM	Meeting management
MMM	Meeting Management Model
MMS	Meeting Management Service
NAO	Nepomuk Annotation Ontology
NIE	Nepomuk Information Element
ORSD	Ontology requirements specification document
PIC	Personal information cloud
PIM	Personal information management
PIMO	Personal Information Management Ontology
POB	Personal Ontology Builder
PSEW	P2P Semantic Eclipse Workbench
PSI	Personal space of information
PUT	Personal Unifying Taxonomy
QP	Query Processor
RDF	Resource Description Framework
SCM	Supply chain management
SSM	Semantic Save Manager
TLM	Task List Manager
TM	Task management
TMM	Task Management Model
TMO	Task Model Ontology
TMOE	Task Model Ontology Extension
TMS	Task Management Service
URI	Uniform Resource Identifier
VO	Value Object
W3C	World Wide Web Consortium
WfMC	Workflow Management Coalition
XUL	XML User Interface Language

1 Introduction

In this section we explain the research background, the objectives and research questions of this thesis. Furthermore, we outline the resulting applications targeted to end-users, the software reference architecture and the information model. We present cornerstone literature to enable the reader a smooth entry into the research topic and explain the thesis organization to provide orientation to the reader throughout this thesis.

1.1 Research Background

Knowledge work is “performed by professional or technical workers with a high level of skill and expertise” [Davenport et al., 1996], so-called *knowledge workers (KWers).*

From an individual KWer’s personal perspective, a KWer pursues higher-level activities during the work [Moran&Zhai, 2007]. This includes core working activities like, e.g., creating a report, and supporting activities like, e.g., managing personal tasks and conducting a meeting.

For each of these activities, a KWer can use computer support. This involves a number of *information objects* and *applications* on a *workspace*, today mainly following a desktop metaphor. The KWer orchestrates these applications to execute the activity, i.e., uses several applications in a certain sequence, and passes information between them. For example in the case of creating a report, the KWer involves a word processor to write the report and an email client to exchange text snippets among the co-workers.

1.1.1 Personal Information Fragmentation Negatively Impacts KWer Activity Support in Workspaces

Today, the KWer’s *personal information is fragmented* across applications in the digital workspace, i.e., the *KWer’s personal information is scattered across the desktop and its applications* [Jones, 2008]. Current desktop systems and applications don't represent the personal information cloud in a unified and integrated form as the KWer expects it. For example, an email client folder containing emails has for the KWer no visible connection to a file system folder containing documents despite dealing with the same topic like for example a task. There is no integrated view across different applications for personal information which the KWer regards as related. Figure 1 sketches an example of this situation for a KWer in the role of an end-user using three applications and corresponding information to fulfill a research task. In this example, an engineer uses a calendar, an email client as well as a task management application to research the competitive situation for a car’s rear axles.

Workspace applications *manage only a particular type of personal information and lock it in* a proprietary storage. This causes one major type of personal information fragmentation. Karger mentions that these applications “often store their data in their own particular locations and representations, inaccessible to other applications" [Jones&Teevan, 2007, p. 127], so-called application information silos. Karger outlines this type of information fragmentation as paradox as this “information is fragmented by the very tools that have been designed to help us manage it” [Jones&Teevan, 2007, p. 127]. From an application developer view, Figure 1 sketches an example of this situation where several applications deal with isolated and possibly redundant pieces of the KWer’s personal information. In the example situation of researching the competitive situation for

rear axles, the engineer needs to manage representations of the same person in three different applications.

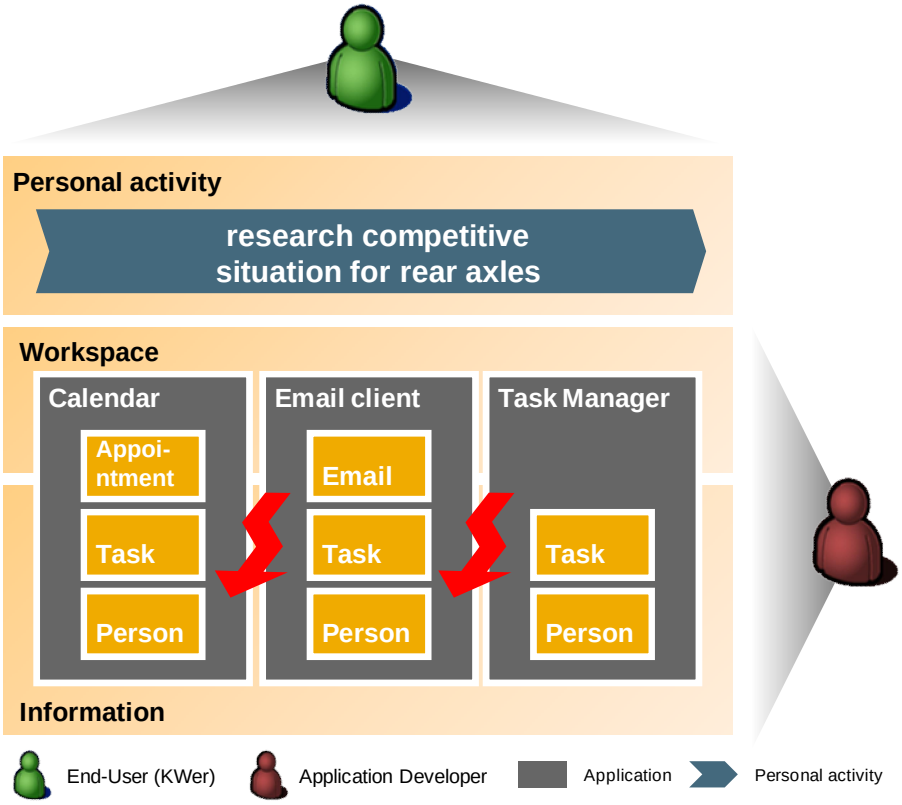


Figure 1: Example for personal information fragmentation and personal activity support.

Another related type of *information fragmentation is produced by the KWer herself*. Due to application information silos, the KWer is forced to fragment the own personal information when keeping it. This is due as each application only manages particular own information types, i.e., information silos, regardless of the KWer's intent for keeping the overall information. Bergmann reports this type of personal information fragmentation problem in the context of organizing project-related information, called project fragmentation problem [Bergman et al., 2006]. Figure 2 shows an example of "information related to a chemistry course" which is "fragmented into separate collections" across three applications. The involved applications' ability of managing each only a particular information element type forces the KWer to organize the information in a fragmented way.

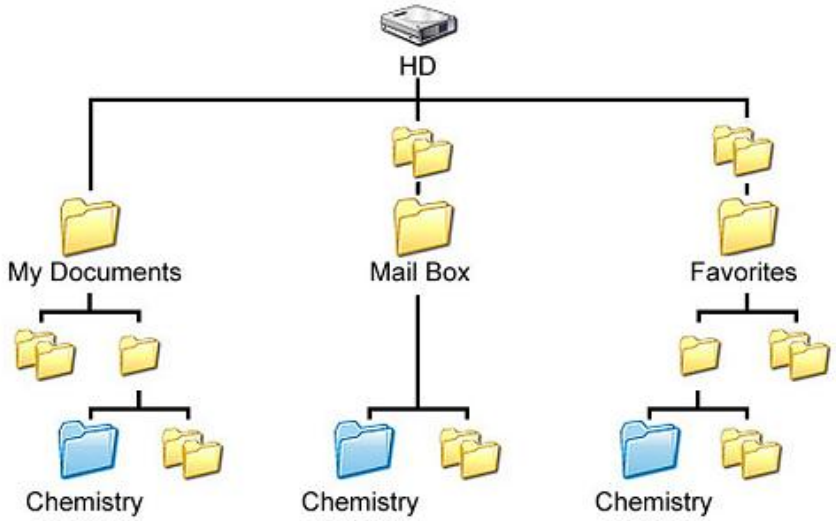


Figure 2: Personal information fragmentation example – project-related [Bergman et al., 2006].

The personal information fragmentation problem *negatively impacts KWer activity support* in several ways. Overall it is a major problem for digital workspaces. In detail, the described fragmentation leads to numerous problems. Karger describes general personal information management problems [Jones&Teevan, 2007, p. 127]:

- No single directory: No single directory exists to find all information about a particular topic.
- No possibility to link information: There's no way to link pieces of information about a particular topic to each other.
- Repetitive searches: Instead a KWer needs to start "multiple applications and perform numerous repetitive searches for relevant information" [Jones&Teevan, 2007, p. 127]. In addition, the KWer needs to decide "which application to look in" [Jones&Teevan, 2007, p. 127].
- Inconsistency: The KWer "may change data in one place (a new married name in the address book) and fail to change it elsewhere, leading to inconsistency that makes it even harder to find information" [Jones&Teevan, 2007, p. 127].

In particular KWer activity domains, for example personal task management activities as well as collaboration activities, "the fragmentation of commonly used information resources—first and foremost, e-mail—make it difficult for activity representations to adequately capture both local work products and communicative interactions" [Voida et al., 2008].

Overall, scattered personal information in application information silos "intensifies the cognitive load" [Ravasio&Tscherter, 2007, p. 275], especially as the KWer regards, like already mentioned, personal information as "*one single body of information*" [Ravasio&Tscherter, 2007, p. 275].

1.1.2 PIM Systems (Semantic Desktops) Manage Unified Personal Information

A *personal information management (PIM) system* provides an infrastructure for managing unified personal information on a computer. It is a basic system, i.e., it is a foundation for applications building on top of it. Recently, research established the term 'semantic desktop' for a dedicated type of PIM system. A semantic desktop PIM system represents personal information in a unified form using semantic representations like, e.g., the Resource Description Framework (RDF) [World Wide Web Consortium (W3C), 2009]. The goals for a semantic desktop PIM system are twofold. On the one

hand they store personal information accounting their semantics and on the other hand they query this personal information efficiently. It extends the operating system's personal information handling capabilities using Semantic Web technologies to enable an integrated management of the KWer's personal knowledge.

A PIM system provides a *unified representation for the KWer's personal information*. To achieve this, a unified personal information model performs personal information *integration around things meaningful to the KWer*. These things are the domain-specific objects the KWer is dealing with in her real-life like, e.g., people, locations, tasks and meetings. Such a unified personal information model provides the schema for this personal information. The unified personal information model is fully user-owned. It represents the KWer's subjective view on the world. The KWer can relate the things among each other according to her domain-specific knowledge. This unified personal information model then integrates the KWer's personal information according to the real-life things the KWer encounters. The information model features a *unified representation*. The unification takes place on metadata-level through, e.g., RDF [World Wide Web Consortium (W3C), 2009]. Being based on RDF, each representation of a domain-specific thing is a RDF resource which is identified by a Uniform Resource Identifier (URI). Furthermore, "all data is accessible and queryable as RDF graph" [Sauermann et al., 2005]. RDF enables linking and annotating resources. Ontologies represent a self-contained, flexible schema to model the personal information model and corresponding instances to represent the personal information of an individual KWer. There are numerous unified personal information models available like, e.g., Personal Information Management Ontology (PIMO) [Sauermann et al., 2007], OntoPIM [Katifori et al., 2005, personal Unifying Taxonomy (PUT) [Jones, 2008, p. 384] and the InfoTop Top-Level Ontology [Maier&Sametinger, 2008].

A PIM system features *functionality* to maintain this unified personal information and to support the stated goals. Thereby, a PIM system integrates artifacts from existing digital environments with the personal information representation, such as desktop information objects like emails, websites and files on a desktop computer. On a desktop, the interlinking and integration of desktop information objects with the KWer's unified personal information model is essential, as a KWer handles throughout the personal activities numerous desktop information objects in her existing applications. For example this includes files, websites and emails that are used in different activities. Furthermore, it enables the KWer to maintain and access this personal information and provides services to maintain and access the unified personal information across applications.

Nepomuk [Nepomuk Consortium, 2009a] [Groza et al., 2007] is an *example for such a PIM system*. Specifically, Nepomuk is an example of a *semantic desktop.* Sauermann defines this PIM system category as follows: "A Semantic Desktop is a device in which an individual stores all her digital information like documents, multimedia and messages. These are interpreted as Semantic Web resources, each is identified by a Uniform Resource Identifier (URI) and all data is accessible and queryable as RDF graph. Resources from the web can be stored and authored content can be shared with others. Ontologies allow the user to express personal mental models and form the semantic glue interconnecting information and systems. Applications respect this and store, read and communicate via ontologies and Semantic Web protocols. The Semantic Desktop is an enlarged supplement to the user's memory" [Sauermann et al., 2005].

These semantic desktop PIM systems involve three *data layers*, see Figure 3. On a *physical layer* computer devices store physical objects such as emails and files like, e.g., word processor documents

or spreadsheet documents. A *domain independent (DI) layer* represents domain independent objects from the physical layer, i.e., wraps existing physical objects and stores them in a unified form. A *domain* specific *(DS) layer* represents domain specific objects the KWer knows of as part of her personal information cloud.

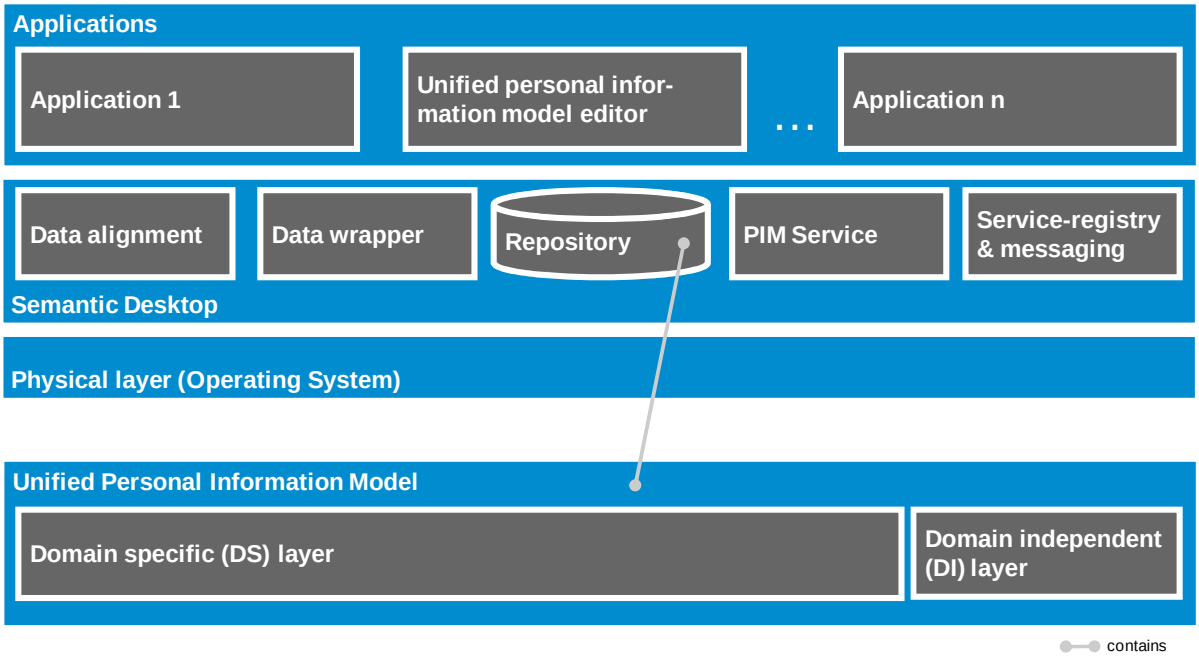


Figure 3: Generic architecture of a semantic desktop PIM system.

Semantic desktop PIM systems involve six *core modules*, see Figure 3. They enable with their modules the management of a unified personal information model and the integration of information objects into the unified personal information model, i.e., the KWer can reference these information objects from the unified personal information model representation. A "*unified personal information model editor*" module enables the KWer to interact with the model of the unified personal information representation, i.e., the unified personal information model. A *personal information management service (PIM Service)* provides functionality to query and modify both the unified personal information model as well as the KWer instances thereof. Consuming applications and other services can invoke this service. A *data alignment* module recommends mappings among entities on the DS layer or among DS layer entities and DI layer entities. A *data wrapper* crawls objects on the physical layer and creates for each found object a metadata representation in the DI layer. A *unified personal information repository* stores unified personal information as well as the corresponding model in a central place, i.e., both the DS layer and DI layer. A *service-oriented architecture* on the computer enables applications and other services to leverage the services of the PIM system.

1.2 Objectives and Research Questions

The objective of this thesis is to *improve KWer activity support in workspaces* by coping with personal information fragmentation through *leveraging the benefits of a unified personal information model*. Figure 4 sketches the to-be situation where a KWer can access and maintain a unified personal information repository by using applications in the workspace.

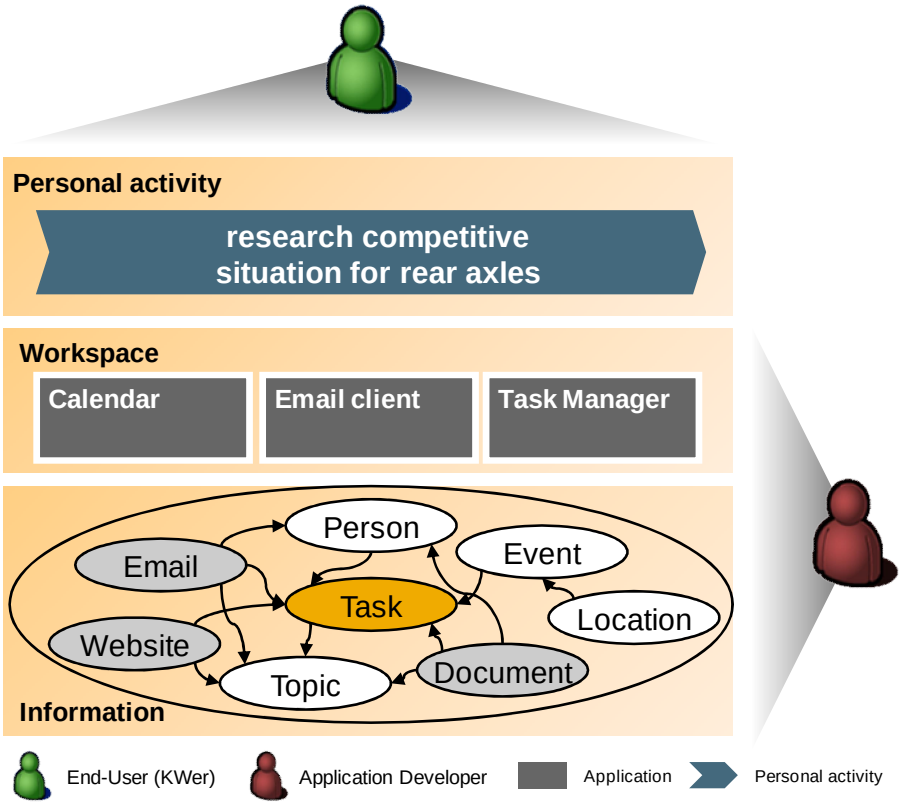


Figure 4: Example for personal activity support using unified personal information.

From an end-user's perspective this leads to the objective to design *applications using unified personal information for KWer activity support*, both in desktop applications and workspaces. We explore workspace-level application examples targeted to personal task management and personal meeting management, two of the higher-level personal support activities that a KWer conducts.

From an application developer's perspective, these applications require an *efficient implementation* and architecture to support the KWer well through the unified personal information model. And the *common, unified personal information model* needs to fulfill the requirements of the KWer's needs in the applications and workspaces.

Figure 5 provides an overview on the outlined objectives and details the underlying problems as well as derived research questions. The following sections explain these in detail from both the end-user's perspective (number 1a and 1b in Figure 5) as well from an application developer's perspective (number 2 and 3 in Figure 5).

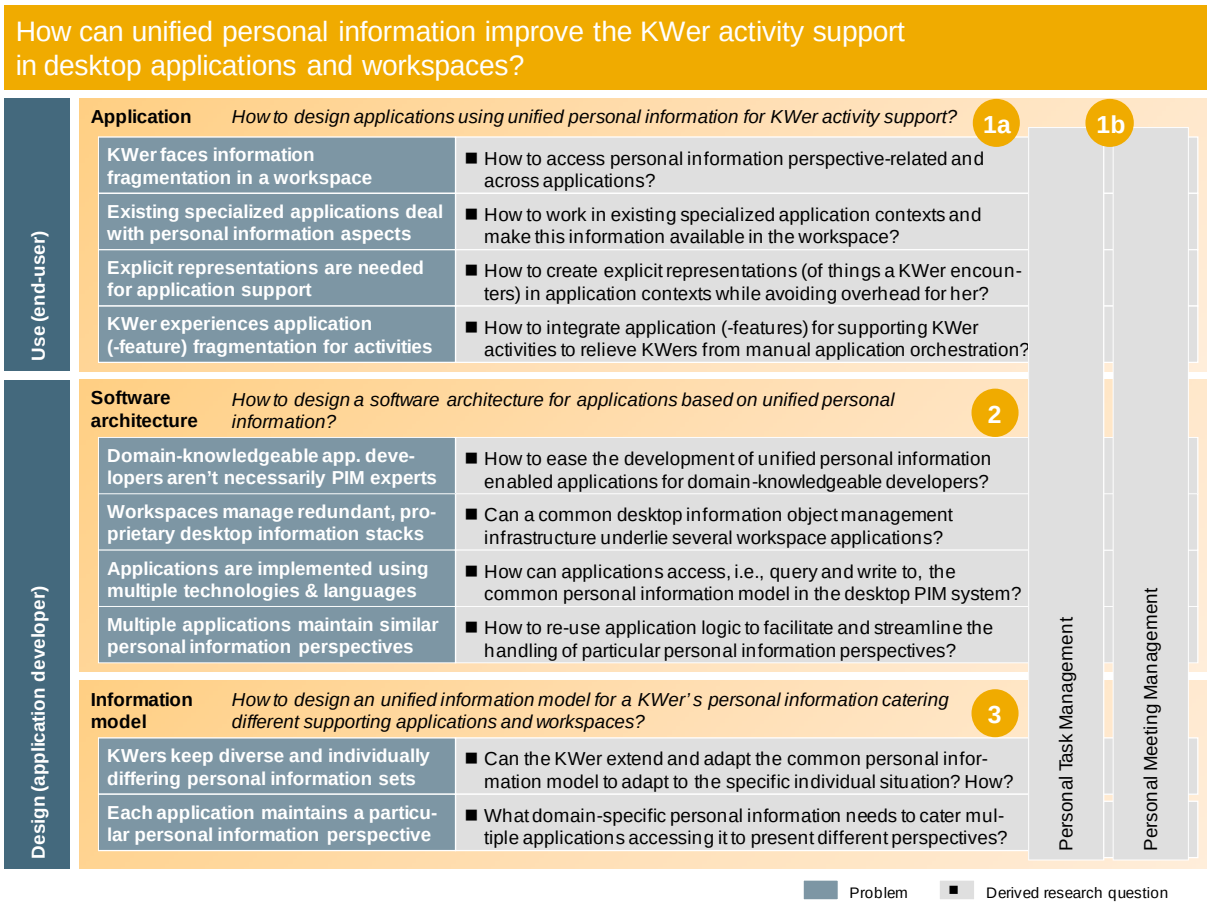


Figure 5: Identified problems and research question overview.

1.2.1 Activity Support for KWers as End-Users

The central research question is to explore, whether and how *unified personal information can improve the KWer activity support* in desktop applications and workspaces (number 1a in Figure 5). On an application-level, the unified personal information model helps to address information fragmentation in the workspaces and therein located applications. The end-user needs to handle unified personal information in the applications and workspaces, i.e., the user interaction in the applications needs to take this information into account. From an end-user perspective, we in particular focus on the following questions:

- An application based on a unified personal information model can present all information related to a particular perspective in one place independent of different applications maintaining parts of this information. Does having all perspective-related personal information in one application user interface improve the KWer support for a certain activity, e.g., personal task management? How can such support look like in the application?
- Numerous specialized applications exist today on the desktop that each deal with a particular part of the KWer's personal information and are specialized to support a KWer's corresponding low-level well, e.g., an email client for communication. Can these existing applications be integrated using the unified personal information model, on the one hand, to enable the KWer to work in the context of respectively existing applications and, on the other hand, to make this information available beyond individual applications in the workspace? This question is crucial to address information fragmentation imposed by application boundaries.

- In some applications, no explicit representation of the personal things does exist. A personal thing is a thing that a KWer encounters at a certain time and that the KWer recognizes as distinct entity. For example, a KWer recognizes a task when seeing a particular email, but there is no explicit task representation. This prevents an efficient tool-support for these personal information things which the KWer knows of but not the computer. This often coincides with KWers applying workarounds in existing applications to support activities which the application doesn't primarily support as the application's designers didn't aim at these use cases. For example, KWers sending email messages to themselves to remind themselves of tasks. How can the KWer create representations of these personal things, with the help of the application, and avoid additional effort and overhead?
- A KWer manually orchestrates a number of applications during the execution of a higher-level activity (activity), i.e., experiences application (-feature) fragmentation in regard to a particular KWer activity. To address this, integrating applications and application features into one application context makes sense for KWers for activities which involve a number of applications and application features[1]. How is it possible to integrate applications or application features to support a particular KWer activity? How does presenting a better integrated workspace to the KWer improve KWer productivity? How can unified personal information contribute to this application integration?

To present *examples* of workspace-level applications leveraging a KWer's unified personal information, we focus on two higher-level supporting activities that a KWer conducts (number 1b in Figure 5). These activities are characterized by either intensive personal information usage or by involving a high number of applications, respectively.

- *Personal task management* is the KWer's personal activity of organizing the KWer's workload and managing the KWer's resource to achieve the personal goals in the best possible way. Application support for personal task management suffers from heavy information fragmentation across multiple applications. For example, task lists on the one hand and numerous information repositories with task-related information, but all disconnected.
- *Personal meeting management* is the KWer's personal activity of preparing, conducting and post-processing a meeting. Application support for this activity is characterized by a high number of low-level actions with different involved tools. The KWer needs to orchestrate these applications to prepare, conduct and post-process a meeting.

1.2.2 Activity Support through an Application Developer's Perspective

From an application developer's perspective, using unified personal information for KWer activity support in a workspace poses some challenges compared to current application development. To compare this situation with the current practice, Figure 6 sketches on the left the architecture principle of applications developed with current state-of-the-art methods. On the right, Figure 6 sketches the to-be architecture for applications leveraging unified personal information along with currently existing challenges.

[1] For example, ContactMap [Fisher&Nardi, 2007] integrates numerous application features for communication in a people-centric application helping the KWer to manage the personal social network. In another example, a KWer needs to orchestrate in a personal meeting management activity a number of applications to prepare, conduct and post-process a meeting.

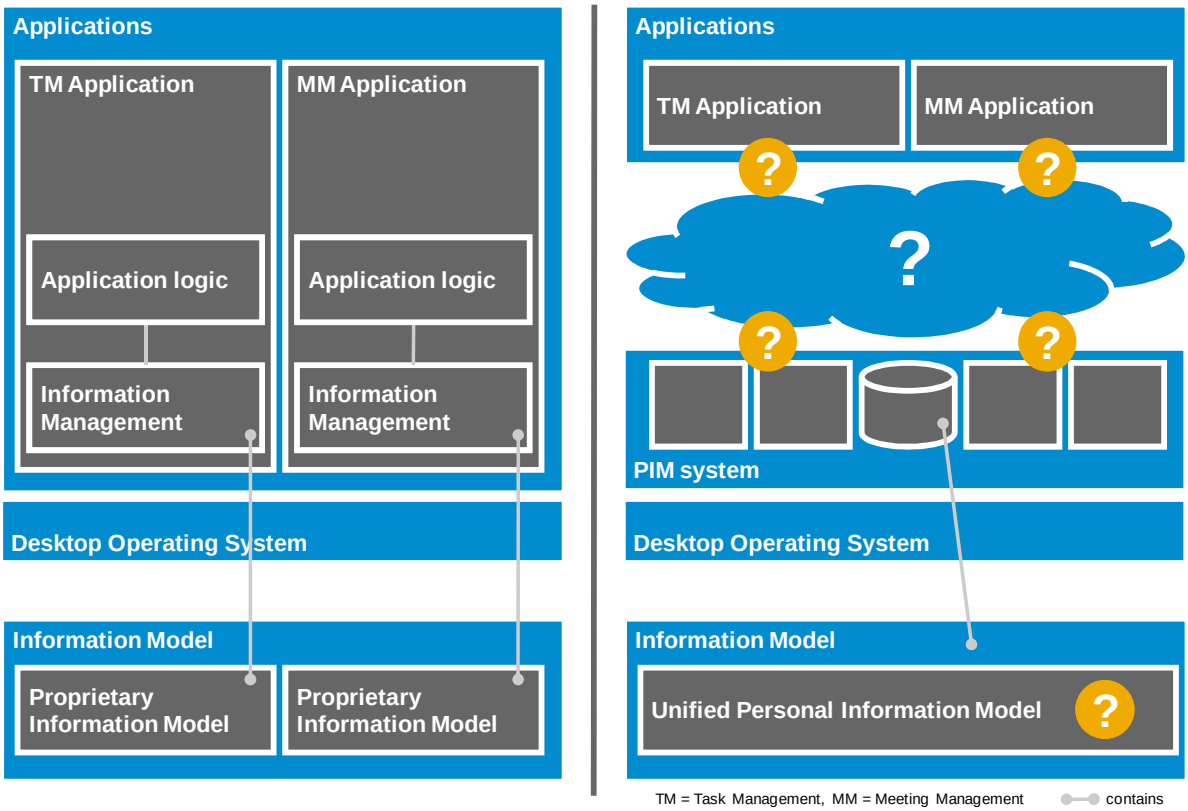


Figure 6: Comparison of state-of-the-art architecture (left) with to-be architecture (right).

We explore an *efficient implementation* of such an enhanced support (number 2 in Figure 5), i.e., the implementation of multiple interconnected applications and workspaces based on a common, unified personal information model. Managing a unified personal information model has become possible through the recent development of the semantic desktop PIM system as introduced by for example the Nepomuk PIM system. However, it is an open question on how to efficiently use this information. How to design an architecture which efficiently enables such an implementation for application developers? From an application developer's perspective, we in particular focus on the following questions:

- Developers of end-user applications are skilled in the respective domains but not necessarily experts in personal information management technology and related desktop architectures. How to ease the development of unified personal information enabled applications for domain-knowledgeable developers?
- Implemented workspace applications so far feature their own proprietary stack for integrating desktop information objects. Can a common desktop information object management infrastructure underlie several workspace applications?
- Multiple applications are implemented in different technologies and programming languages. How can applications, having a diverse technology background, access a KWer's unified personal information, i.e., query and write to the common personal information model and further services of the semantic desktop PIM system?
- Multiple applications need to access similar sets of a KWer's unified personal information, i.e., query and write to similar perspectives of the common personal information model. For example, possibly more than one application handles the same perspective on personal information like, e.g., maintaining the personal information "task" thing. How can the

application developer re-use application logic to facilitate and streamline the handling of particular personal information perspectives?

Subsequently an implementation needs to be based on an elaborated unified personal *information model* to be able to provide efficiently the above described application features (number 3 in Figure 5). Here, we explore the requirements such an information model needs to fulfill to match the KWer's activities in the applications and workspaces.

- KWers keep a diverse set of personal information, which differs individually from KWer to KWer. Can the KWer extend and adapt the common personal information model to adapt to the specific individual situation and individual support for personal activities? How?
- Each application supporting a specific personal KWer activity maintains a particular perspective on the KWer's unified personal information. For example, applications supporting a KWer's personal task management activity maintain a task-centric perspective on the KWer's personal information. Does the unified personal information model fit these requirements for applications supporting a KWer's personal activities? What domain-specific information needs to be available in the unified personal information model with regard to multiple applications accessing it to present different perspectives for the KWer?

1.3 Approach – Iterative Prototype Development

This work is scientifically located in design science. We followed an *incremental prototyping* approach, i.e., we developed prototypes in an incremental development and refinement process. The prototype development process we followed can be described as a stage / gate process and the prototype passes through the process from the initial idea to the final prototype usable by end-users. The development process consists of several phases, i.e., stages, where the prototype continually gets refined. We applied a four-stage process, see Figure 7. Each development process stage concludes with a user evaluation, which represents a gate in the prototype development process. The user evaluation serves as quality gate and the outcome determines the next stage, i.e., whether the prototype needs to run through, e.g., the design stage again or whether it advances to the next stage.

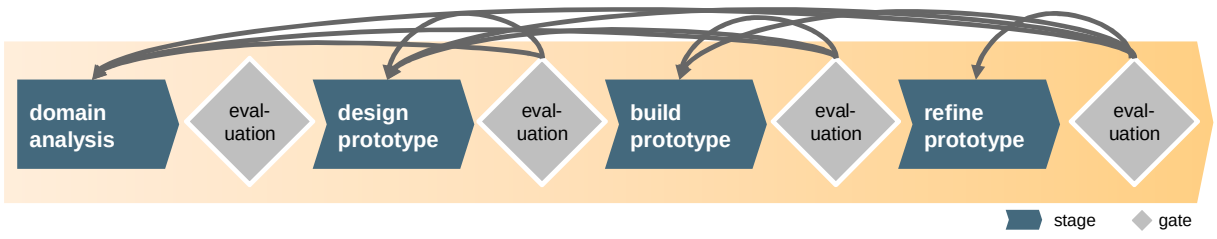


Figure 7: Prototype development process – stages and gates.

In the *domain analysis* stage, user interviews and literature research enable an understanding of the prototype domain. This stage results in reports with a domain description using scenarios and personas, resulting user requirements as well as the state of the art in the domain.

In the *design prototype* stage the first sketch of the prototype helps to clarify and develop several design ideas and alternatives. These sketches are 'low-fi' prototypes, i.e., paper prototypes or mockup screens later on in this stage. This coincides with the domain-specific specification of the prototype. This stage results in a set of prototype mockups which depict the application user

interface, prototype mockup up videos which explain the intended user interaction and a report on the domain-specific specification of the application.

In the *build prototype* stage the prototype is implemented by software engineers. Based on the specifications defined in the previous stage, the software engineers specify a system-specific specification, i.e., determine the technology and architecture used to implement the prototype. This serves as blueprint for the prototype implementation. The result of this stage is a 'hi-fi' prototype, i.e., a prototype that an end-user like a KWer can actually use and work with. It is common that the prototype runs several times through this stage to build several prototype modules.

In the *refine prototype* stage the prototype is continually refined with incremental improvements. These are based either on user feedback or based on testing results which identified flaws of the then current prototype. Result of this stage is an improved version of the hi-fi prototype.

Continuous *evaluations* after completing each stage enable throughout the whole design process a consistent integration of user feedback into the prototype. The early user involvement starting with detailed user and domain studies significantly increases the fit of the prototype to the user domain.

1.4 Results

This thesis contributes a *framework* for applying unified personal information for KWer activity-support across applications and workspaces, see Figure 8. The framework consists of three components, i.e., (1) applications supporting the KWer's personal activities, (2) the software reference architecture and (3) a unified personal information model. Each element addresses the corresponding problems and research questions as outlined above.

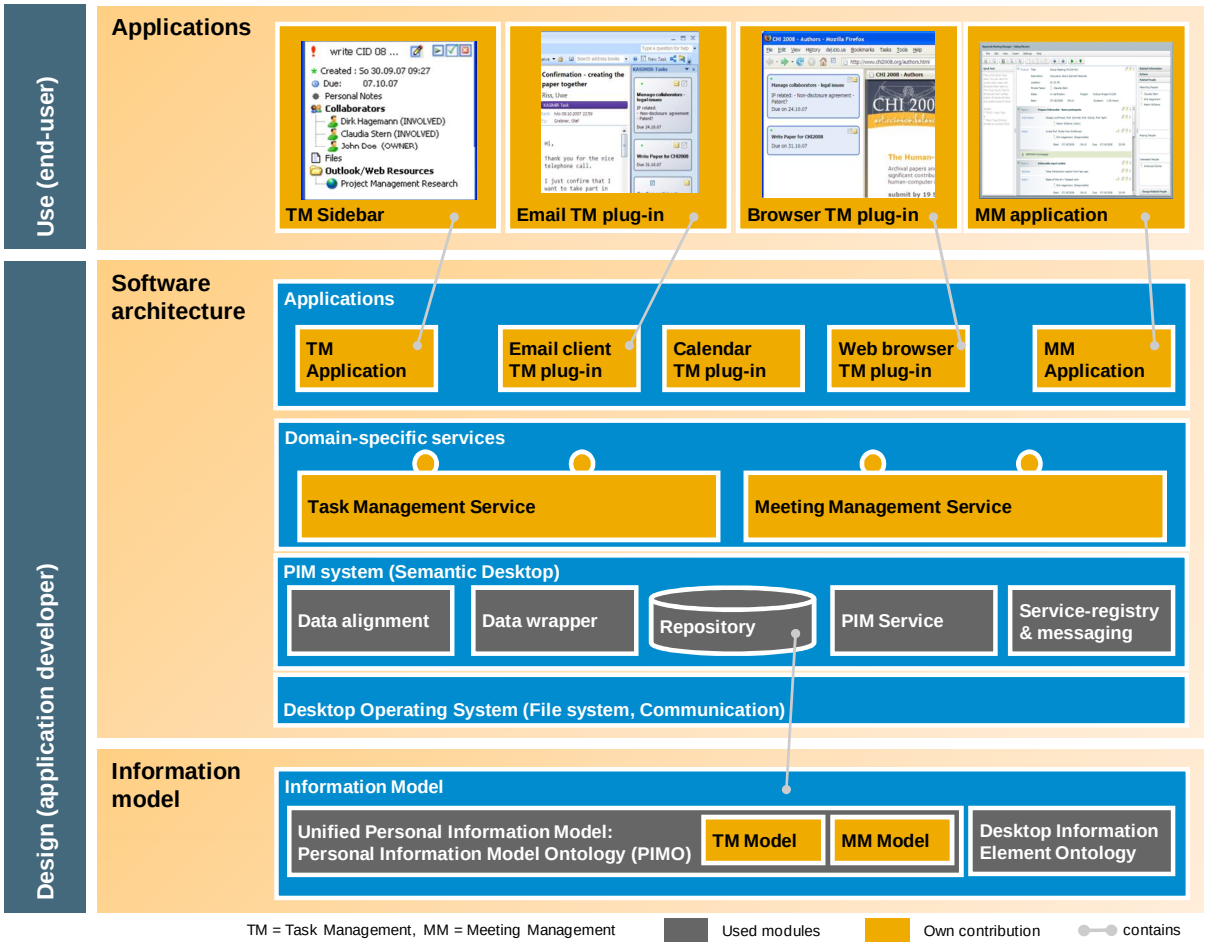


Figure 8: Framework – Unified personal information for workspace-wide KWer activity-support.

Parts of these results have been previously published in the following journals, books, conference or workshop proceedings and technical reports – some parts of this thesis are based on these contributions: [Grebner&Ebner, 2008], [Grebner&Riss, 2008a], [Grebner et al., 2008], [Grebner&Riss, 2008b], [Grebner, 2009], [Grebner et al., 2007a], [Ong et al., 2007b], [Ong et al., 2008], [Riss et al., 2008], [Riss&Grebner, 2008], [Grebner et al., 2006], [Grebner et al., 2007b], [Grebner et al., 2007c], [Ong et al., 2007a], [Riss et al., 2009], [Taylor et al., 2008]. Furthermore, the Task Management Model is publicly available as Task Model Ontology (TMO) at [Brunzel et al., 2008].

1.4.1 Workspace-Level Applications Leveraging Unified Personal Information

Fully functional prototype *applications* demonstrate how a KWer in the role of an *end-user can* use unified personal information to support her activities. We present three types of applications using unified personal information to support a KWer's personal activities. The presented applications target a KWer's personal support activities of personal task management (TM) and personal meeting management (MM). In the following, we show how these applications demonstrate concepts to address personal information fragmentation in the workspaces and therein located applications using a unified personal information model. Furthermore, we show how they address the sets of research questions defined above.

The *Kasimir* personal task management application [Grebner et al., 2008] enables a KWer to manage the personal task load. Thereby Kasimir allows the KWer to manage the task representations in conjunction with related personal information.

- The Kasimir personal task management application enables the KWer to manage beneath a task list the related personal information, i.e., a task information perspective with respectively involved desktop information objects and a social perspective with the people involved in the task.
- We show how a KWer can *manage personal information in specialized application contexts* and can *at the same time make this information available beyond the individual application*, by showing the example of Kasimir and the further following applications. In Kasimir the KWer interacts in a specialized task management application context. At the same time Kasimir makes the created tasks and related information available beyond itself through using the unified personal information.
- The Kasimir personal task management application demonstrates how an application designed from scratch can incorporate unified personal information and how it can present it to the KWer's advantage, see section 5.

The *SAP Task application plug-ins* [Grebner et al., 2008] enable the integration of task representations in other application's contexts. For example, the KWer can manage explicit task representations in the email inbox without additional effort.

- We explore and demonstrate a concept to *create explicit representations of personal things without overhead for the KWer* and directly in an application context, see section 6. We implemented plug-ins for the personal task management activity support for the three applications Microsoft Outlook Email, Microsoft Outlook Calendar and Mozilla Firefox demonstrating how a KWer can create tasks from emails, appointments and websites using known actions, see section 6.
- By using the SAP Task application plug-ins, the KWer interacts in a specialized task management application context within the application hosting a SAP Task application plug-in. At the same time the SAP Task application plug-ins make the created tasks and related information available beyond itself through using the unified personal information.
- The SAP Task application plug-ins demonstrate at the example of a KWer's personal task management activities how an application with limited extension possibilities like, e.g., the proprietary email client Microsoft Outlook, can incorporate unified personal information and present it to the KWer's advantage, see section 6.1.

The *Nepomuk Meeting Manager* application supports the KWer's personal meeting management activities. The Nepomuk Meeting Manager facilitates the orchestration of several applications on the KWer's request to prepare, conduct or post-process a meeting.

- With the Nepomuk Meeting Manager the KWer can see all personal information related to a meeting in one place. This again includes besides the meeting minutes a task information perspective with respectively involved desktop information objects and a social perspective with the people involved in the meeting.
- In the Nepomuk Meeting Manager the KWer interacts in a specialized meeting management application context. At the same time the Nepomuk Meeting Manager makes the created meetings and meeting protocols available beyond itself through using the unified personal information.

- The Nepomuk Meeting Manager demonstrates how an application designed from scratch can incorporate unified personal information and how it can present it to the KWer's advantage, see section 7.
- We explore and demonstrate the *integration of several application features into one application context* that follows a foundational information concept, i.e., the application functionality is aligned around this information concept. At the example of the KWer's personal meeting management activity, we show the Nepomuk Meeting Manager as an application which integrates communication and task management features. It aligns the functionality around the concept of a meeting. Other specialized desktop applications, like, e.g., Microsoft Outlook for composing and sending emails or the Kasimir task management application for creating tasks, provide these features.

Each of the presented *applications supporting personal task management and personal meeting management activities excel* through the use of unified personal information and the presented concepts to utilize it in their respective domain compared to state-of-the-art. The Kasimir application, in conjunction with the SAP Task application plug-ins, supports the KWer's personal task management activities. The Nepomuk Meeting Manager supports the KWer's personal meeting management activities.

1.4.2 Software Reference Architecture

A *software reference architecture* shows how application developers can efficiently implement applications based on unified personal information, see Figure 8. The proposed software reference architecture consists of three layers.

First, facing the end-user, *functionally-oriented workspaces and applications* each support a particular type of a KWer's activity. They each represent a distinct perspective on the common underlying unified personal information set. These workspaces and applications can be implemented using multiple programming languages and technologies.

Second, a set of *domain-specific services* handles the access to a particular perspective on a KWer's personal information from the application layer to the unified information model. The domain-specific service's functionality provides the needed parts of the personal information to domain-knowledgeable developers. This way, the user experience of each workspace can be designed solely relying on the domain and completely independent from the underlying (operating) system and associated metaphors and paradigms.

Third, using a *PIM basic system* we can access numerous desktop services and thus re-use desktop functionality without replicating it in each workspace. Here we use the semantic desktop *Nepomuk* [Groza et al., 2007] as example for a service-oriented PIM basic system. The Nepomuk semantic desktop provides a service-oriented desktop architecture and services. The PIM basic system manages the unified personal information model representing the KWer's personal information cloud and offers a PIM Service providing generic read and write functionality for the unified personal information model. Nevertheless, it is user-owned but formally represented using a set of ontologies. Furthermore, it encapsulates core operating system functionality in services like, e.g., handling desktop information objects such as for example, emails, websites and files.

Using a PIM basic system enables an *evolutionary architecture.* It helps to transition current desktop workspaces to the proposed multiple functional workspaces by replacing step-by-step their

redundant proprietary stacks for managing a KWer's personal information. Offering a modularized architecture enables this evolution of existing applications. With putting this architecture in place, today's commercial workspaces can transition into the here outlined multiple functional workspaces without dismissing existing workspaces and investments.

The domain-specific services ensure a *consistent handling of personal information across applications*. When multiple applications access, i.e., query and write to the common unified personal information model, a common set of application logic in the domain-specific services facilitates the handling of a particular personal information perspective like, e.g., the domain-specific services for personal task management, see section 10. According to our experience, despite having a formalized and shared conceptualization in place, i.e., an ontology, ambiguity in using the information still does exist and this leads to incompatibility problems across application. In this regard, using only a unified information model is not enough for efficiently using this information across applications.

1.4.3 Information Model

We use a unified representation of the KWer's personal information cloud as a *common abstract information model*. This unified personal information model is at the heart of the reference architecture and is stored in the repository of the PIM basic system, see Figure 8.

We use a formalized representation of this model, the Personal Information Model Ontology (PIMO) [Sauermann et al., 2007] to enable machine processing on it by the domain-specific services, see section 9. It features a unified representation through the RDF representation and it is integrated along the KWer's personal things. This represents the KWer's mental model, i.e., the personal, subjective view on the world. These information entities of the PIMO are the domain-specific entities of the personal information cloud that the KWer deals with during pursuing her activities, i.e., like for example people, tasks, locations, events or topics. The KWer can explicitly interlink the entities with each other which are related in the KWer's view, e.g., because they were helpful for the same task.

This unified personal information model needs to be adapted to fully enable applications that support a KWer's activities to work with it. We show the adaptations at hand of the two example use cases of personal task management and personal meeting management and explain how this can be reproduced for further domains. The Task Management Model (TMM) is a conceptual representation of tasks and related information for use in personal task management applications. It is part of the KWer's unified personal information model. The Task Model Ontology (TMO) represents an agreed, domain-specific information model for tasks and covers personal task management use cases, see section 9.1. In the *Task Model Ontology (TMO)* we extend the task concept of the personal unified information model to meet the requirements for the support of the personal task management activity. The Meeting Management Model (MMM) is a conceptual representation of meetings and related information for use in personal meeting management applications for KWers. It is as well part of the KWer's unified personal information model. Its representation bases on the PIMO ontology and thus represents an agreed, domain-specific information model for meetings and related information and covers the personal meeting management use cases as defined in section 7.4.

1.5 Cornerstone Literature

The *literature* dealing with personal information management, personal information fragmentation and ways to resolve this problem is overseeable. Especially for application support for a KWer's higher-level activities corresponding application boundaries there are few literature resources. This research takes place at the intersection of the research disciplines of human-computer-interaction

(HCI) and information management science. This concerns as well the research on software applications dealing with a personal point-of-view, i.e., applications that focus and optimize support from a KWer's perspective and across potential application boundaries.

Jones and Teevan outline in "Personal Information Management" [Jones&Teevan, 2007] the current personal information management research both from a human-computer-interaction (HCI) and information science perspective.

Jones gives in "Keeping Found Things Found" [Jones, 2008] a concise overview on personal information management. This includes theoretical parts on personal information and a personal information management process as well as presents related personal information management research results.

Kaptelinin and Czerwinski give in "Beyond the Desktop Metaphor" [Kaptelinin&Czerwinski, 2007a] an extensive overview on the state-of-the-art of workspace design from a HCI perspective. Though the focus is on workspaces, i.e., digital work environments, there is overlap to the KWer's personal information management.

The PhD thesis of Richard Boardman "Improving Tool Support for Personal Information Management" [Boardman, 2004] analyses in depth the information fragmentation of personal information on the desktop. Thereby Boardman discusses application-centric vs. workspace-level application integration to overcome the application-induced information fragmentation.

1.6 Thesis Organization

This thesis is structured in four building blocks. Figure 9 shows an overview of these building blocks and the structure of the related sections.

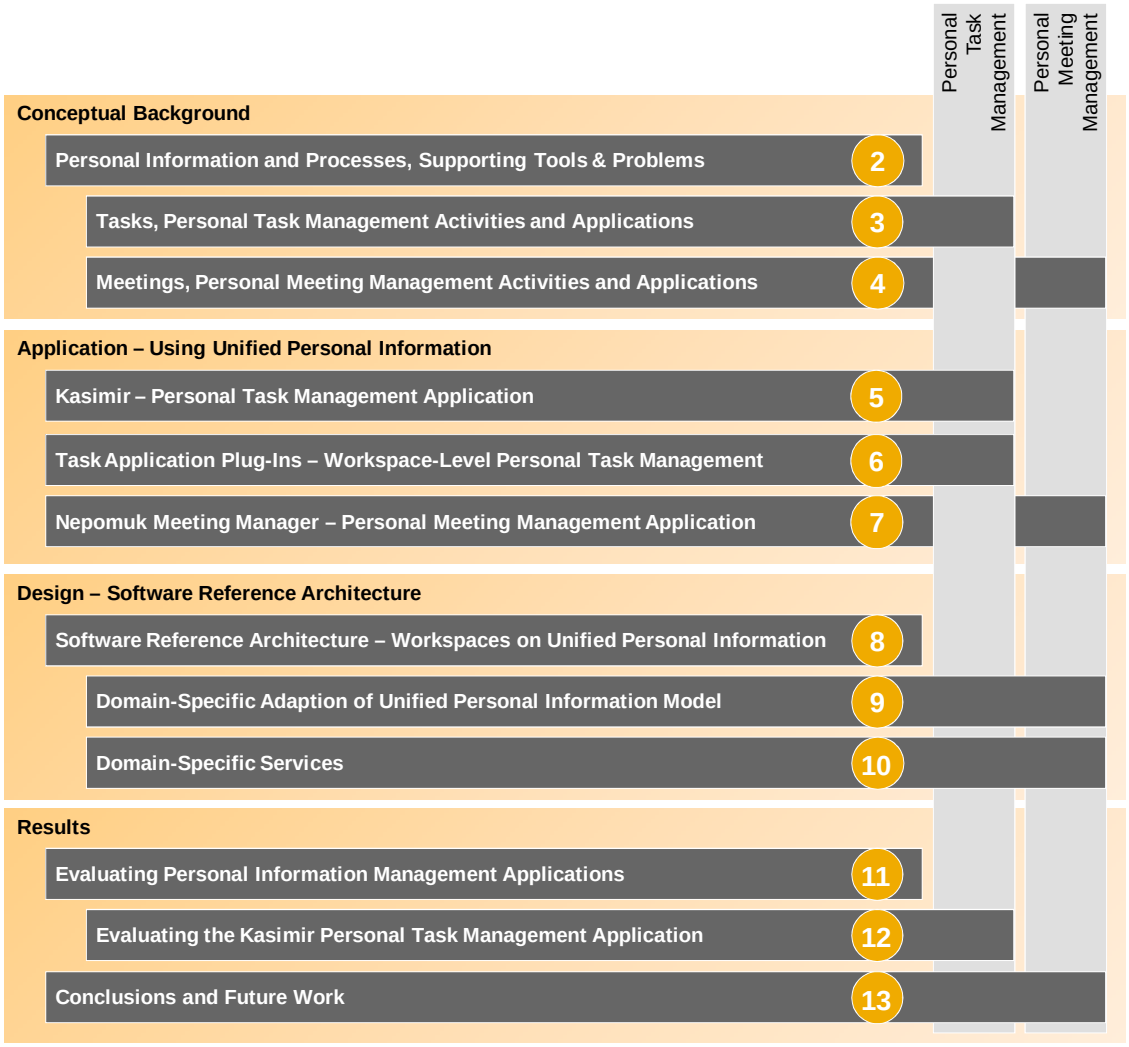


Figure 9: Thesis structure.

In the "*Conceptual Background*" building block we focus in section 2 on the knowledge worker, personal information, personal processes and supporting tools. We explain a KWer's personal information and the personal, higher-level activities a KWer conducts to achieve her goals, i.e., the personal processes. We present digital work environment, i.e., workspaces, and contained applications as supporting tools for the KWer's higher-level activities. We highlight the problem of the fragmentation of personal information in applications within the workspace. Integration and unification are principle approaches addressing the information fragmentations. We then show at hand of the supporting activities of personal task management and personal meeting management in section 3 and 4 the more specific aspects of personal processes, involved people and related application support.

In the "*Application – Using Unified Personal Information*" building block we present three applications using unified personal information to support a KWer's personal activities. In section 5 we present the Kasimir task management application enabling a KWer to manage the personal task load. Thereby Kasimir allows the KWer to manage the task representations in conjunction with related personal information. In section 6 we present the SAP Task application plug-ins enabling the integration of task representations in other application's contexts. For example, the KWer can manage explicit task representations in the email inbox without additional effort. In section 7 we present the Nepomuk Meeting Manager application to support the KWer's personal meeting

management activity. The Nepomuk Meeting Manager facilitates the orchestration of several applications on the KWer's request to prepare, conduct or post-process a meeting.

In the "*Design – Software Reference Architecture*" building block we present in section 8 a software reference architecture that enables an efficient realization of workspace-level applications that leverage unified personal information. The software reference architecture consists of the three levels of applications, domain-specific services and the PIM system managing the tailored unified personal information model. In section 9 we go into the details of the domain-specific adaptation of the unified personal information model. We present the Task Model Ontology (TMO) modeling task representations as extension for the unified personal information model as well as the Meeting Management Model as domain-specific adaption of the unified personal information model. In section 10 we detail the domain-specific services at the example of the Task Management Services and the Meeting Management Services. These services enable a re-use of application logic to build applications supporting a KWer's corresponding personal activities.

In the "*Results*" building block we present in section 11 an elaboration on how to evaluate personal information management applications. In section 12 we present in detail how we evaluated the Kasimir personal task management application. We conducted several formative usability studies and one long-term evaluation. In section 13 we show a summary of the presented work and present open issues and directions for future work. We highlight how the problems and research questions mentioned in this introductory section have been addressed.

We leverage the two personal supporting activities of personal task management and personal meeting management as continuous examples throughout this thesis and across the thesis building blocks and sections. We present for these two use cases the complete realization through the stack of application, domain-specific services and the information model as well as partly the evaluation.

PART I – CONCEPTUAL BACKGROUND

ACKNOWLEDGEMENTS AND PUBLICATIONS

- The research on state-of-the-art in personal task management applications in section 3.4 has been published in the Nepomuk project deliverable D10.1 SAP Scenario Report [Grebner et al., 2006]
- Tobias Ackermann contributed to section 4 with his study thesis [Ackermann, 2008] on meeting management and the Nepomuk Meeting Manager – meeting management foundations.

2 Personal Information and Processes, Supporting Tools & Problems

In the following sections we introduce the knowledge worker (KWer), the KWer's personal information, the KWer's personal activities and workspaces supporting the KWer's activities. As well, we highlight the problems of current workspaces and related applications for supporting a KWer's activities. Figure 10 sketches these elements in relation to each other.

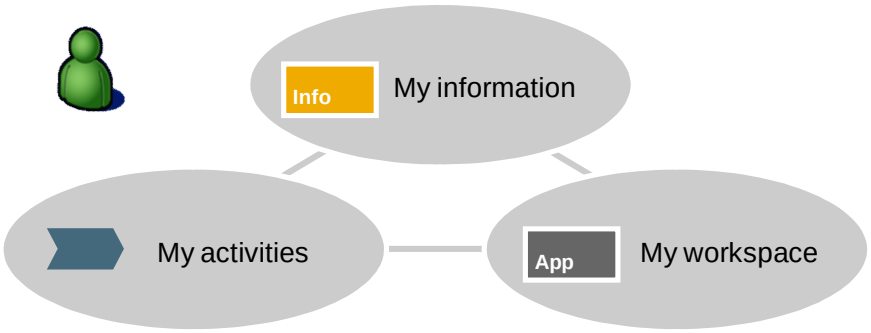


Figure 10: Overview on a KWer's personal information, processes and workspace.

2.1 Knowledge Work and Knowledge Workers (KWers)

Knowledge work or synonymously *knowledge-intensive work* is "characterized by variety and exception rather than routine" [Davenport et al., 1996]. Knowledge work is "performed by professional or technical workers with a high level of skill and expertise" [Davenport et al., 1996]. These so-called *knowledge workers* perform knowledge work to a certain extent, i.e., not only occasionally. They work in highly dynamic environments. A project manager or sales representative are examples for those knowledge workers. They solve problems depending on individual influence factors and have different methods of working on a task. Like their work, their cooperation and communication relationships are dynamic. For example, knowledge work represents activities in research, product development, training, consulting or management activities for strategy development and planning [Goesmann, 2002, p. 34]. The following four attributes[2] characterize knowledge work:

- Manifold results: knowledge workers solve problems and generate results dependent on individual influence factors.
- Different methods of performing work: knowledge work is characterized by spontaneous change of the behavior and the mode of performing work.
- Low dependency on used/stored information: knowledge workers depend only on a low fraction on electronic or paper-based information for the execution of their tasks.
- Dynamic cooperation and communication relationships: New cooperation and communication relationships arise frequently because of dynamic team structures. They change or are terminated with the same pace.

In order to distinguish knowledge work from other types of work, based on the presented characteristics, some differentiating aspects are given: [Moore, 1999]

- "The primary raw material in the knowledge work process is information.

[2] In a field study by [Kidd, 1994] cited in [Goesmann, 2002, p. 33f]

- The primary product of the knowledge work process is information to which value has been added by the knowledge and problem-solving skills of the knowledge worker.
- Knowledge work is mentally rather than physically intensive.
- There is a heavy reliance on the knowledge and creativity of individuals, even in collaborative group settings.
- Knowledge workers are not easily interchangeable or replaceable[3] like assembly line workers due to different levels of personal knowledge, innate problem-solving skills, creativity, personality, and style." [ibid.]

Knowledge work gains more and more importance in today's business environment. Drucker [Drucker, 1993] highlighted this already in 1993 by nominating the increase of knowledge worker productivity, in a magnitude similar to the increase of the manual worker productivity achieved within the last century, as the biggest challenge of the century.

2.2 Personal KWer Activities in Knowledge Work

Seen from a KWer's personal perspective, a KWer conducts *higher-level activities* [Moran&Zhai, 2007] to achieve individual goals as part of her daily work. A KWer's activities, in short for higher-level activities, are highly variable and typically in the KWer's to-do list.

These activities are defined from a personal perspective and thus resemble personal 'processes'. Compared to them, a *business process* or synonymously *process* is from an organizational perspective a "set of one or more linked procedures or activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships" [Workflow Management Coalition (WfMC), 1999, p. 10].

We first present the possible personal activities. Then we show the characteristics of these personal activities as they represent knowledge-intensive activities.

2.2.1 Personal Value Chain: Personal Primary Activities and Support Activities

The KWer performs activities in a *personal value chain* in analogy to an enterprise's value chain [Porter, 1998], see Figure 11. The KWer's activities can be categorized into the two types of *personal primary activities* and *personal support activities*.

[3] Goesmann [Goesmann, 2002, p. 33f.] contrasts the knowledge worker with a specialist being bound to more external rules, procedures and guidelines. The specialist has a rather continuous method of performing work and is mostly bound to organizational structures with tight responsibilities and space.

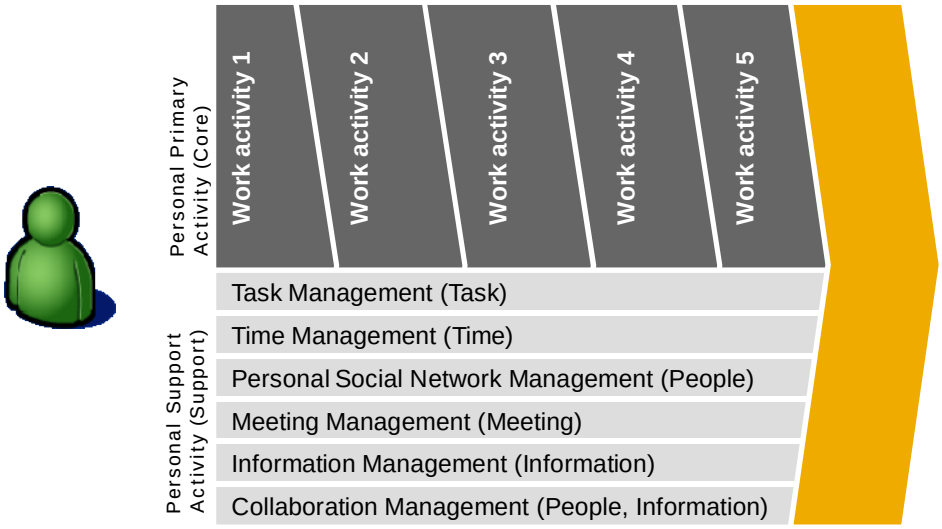


Figure 11: Personal primary and supporting activities.

Personal primary activities are the working activities where the KWer actually works towards achieving the individual goals. They correspond to the enterprise's line functions. They depend largely on the KWer's job role and the KWer's individual skills and capabilities. By executing personal primary activities, the KWer produces the output that she gets rewarded for, either materially in the form of payment or immaterially in the form of personal satisfaction. For example, a software researcher would consider research as personal primary activity. These activities that she performs in her day-to-day work include, e.g., creating scientific reports & papers, gathering user requirements in interviews and building a prototype. Drilling into a more detailed level, writing a scientific paper itself has several primary sub-activities, like for example do literature research, gather co-writers, organize document versions. This shows that the granularity differs depending on the KWer's requirements and the chosen perspective. Similar activities can be indentified for example in the professional domain for project managers, sales representatives etc. In the domain of the KWer's personal environment, e.g., media management can be a personal primary activity consisting of watching videos, listen to music, managing music and video files and shopping for new movies and music.

Personal supporting activities enable the KWer to manage herself and things that support her. Only when managing herself as scarce individual resource the KWer can efficiently execute the primary activities. They correspond to the enterprise's staff functions. They are independent of the KWer's job role as they concern each individual KWer. These supporting activities are cross cutting primary in the execution through their support for primary activities. They as well vary in their concrete instantiation from KWer to KWer depending on, e.g., the KWer's job role, gathered experience, felt importance and skill level. For example, KWers with a busy schedule and workload feel a more urgent need to perform efficient task and time management than KWers whose workday leaves sufficiently enough time to cope with the workload. Again, the granularity differs depending on the KWer's requirements and the chosen perspective. We currently have identified several facets of personal management and present them in the following personal supporting activities at hand of KWers in a professional domain. This list covers the major supporting activities but does not claim to be exhaustive. In brackets the respectively mainly involved information item is shown, which we explain in detail in section 2.4.4.

- Task Management (Task): Plan, structure and prioritize a set of tasks.
- Time Management (Time): Plan and control available time.

- Personal Social Network Management (People): Build personal social network, maintain the network and activate nodes within the network as needed [Fisher&Nardi, 2007, p. 171].
- Meeting Management (Meeting): Prepare, conduct and post-process meetings.
- Information Management (Information): Collect, organize, browse and retrieve information.
- Collaboration Management (People, Information): Communicate and interact with people.

The KWer's personal activities always include information processing activities. Information processing activities include the KWer's (re-)acquiring, processing, organizing, and (re-)distributing of information [Jones, 2008, p. 60]. The information processing activities represent an orthogonal, information-centered perspective on the activities that a KWer conducts, compared to this activity-centered perspective.

2.2.2 Characteristics and Implications of Working with Knowledge-Intensive Activities

Knowing the *characteristics of working with knowledge-intensive activities* is important to evaluate the demand for knowledge-intensive activity support by application systems. Several characteristic attributes are identified by Schwarz et al. [Schwarz et al., 2001] based on the characteristics of knowledge-intensive work presented by Kidd [Kidd, 1994]: [Schwarz et al., 2001]

- *Unpredictable patterns of behavior*: The highly variable, ad-hoc arising behavior patterns require the definition of individual and situational process patterns serving as comprehensible guidelines.
- *Communication-oriented*: Highly variable communication networks and use of manifold media is typical. Domain-specific knowledge is contained in documents, emails and drafts. This requires the integration of these communication channels.
- *Interdisciplinary*: The expertise of many fields and line of businesses is the prerequisite for work on knowledge-intensive activities. This requires fast ability to ascertain information and easy access to background information especially for new or inexperienced employees.
- *Information-bulky*: The, often superficial, processing of huge amounts of information and documents is typical.
- *Argumentation-based*: The proceeding is a continuous exchange of arguments and negotiating along core questions, key requests, option for actions and optimization criteria.
- *Iterative*: The problem solution process is iterative and incremental by nature requiring the support for cycles and stepwise refinement.

The presented characteristics for working on knowledge-intensive activities strongly indicate the relevance of *supporting knowledge work with software* and computer systems. Especially the involvement of communication and collaboration platforms requires software support.

2.3 Personal Information and Personal Information Cloud

An *information item* is according to Jones and Teevan "a packaging of information in a persistent form that can be acquired, created, viewed, stored, grouped (with other items), moved, given a name and other properties, copied, distributed, moved, deleted, and otherwise manipulated. Examples of information items include: (1) paper documents, (2) electronic documents and other files, (3) email messages, (4) Web pages, or (5) references (e.g., "shortcuts" or aliases") to any of the above" [Jones&Teevan, 2007, p. 7].

Each information item has a particular *information form*, or information type, like for example paper documents, files, email messages and web bookmarks [Jones&Teevan, 2007, p. 7].

We refer to information items in a computer-supported information form as *information elements*. All information items represented in a computer supported environment possess a subset of all possible information forms, i.e., the information forms which can be presented in a computer. These information forms include for example files, websites and emails.

2.3.1 Personal Information

Personal information is information related to an individual KWer. The relation can be characterized in several dimensions as "there are several ways in which information can be 'personal'" [Jones, 2008, p. 33]. From an individual KWer's perspective information "can be owned by, about, directed toward, sent by, experienced by, or relevant to 'me'" [Jones, 2008, p. 33]. Table 1 lists these different categories of personal information and gives examples.

	Relation to "me"	Examples
1	Controlled by (owned by) me	Email messages in our email accounts; files on our computer's hard drive
2	About me	Credit history, medical, web browsing, library books checked out.
3	Directed toward me	Phone calls, drop-ins, TV ads, web ads, pop-ups.
4	Sent (posted, provided) by me	Email, personal web sites, published reports and articles.
5	(Already) experienced by me	Web pages that remain on the Web. Books that remain in a library. TV and radio programs that remain somewhere in 'broadcast ether'.
6	Relevant (useful) to "me"	Somewhere "out there" is the perfect vacation, house, job, lifelong mat. If only I could find the right information!

Table 1: Categories of Personal Information [Jones, 2008, p. 34].

These categories are not free of intersections as they represent different, but not orthogonal perspectives on a person's personal information. In "their union, they exclude very little" [Jones, 2008, p. 35] of a KWer's possible personal information.

Personal information in the category "*controlled by (owned by) me*" is the "information a person keeps, directly or indirectly (e.g., via software applications), for personal use" [Jones, 2008, p. 34]. Personal information in the category "*about me*" is "information about a person but available to and possibly under the control of others" [Jones, 2008, p. 35]. Personal information in the category "*directed toward me*" is information that is directed to a person and includes, e.g., emails arriving in the inbox, pop-up notifications of the incoming email and the ringing telephone [Jones, 2008, p. 35]. Personal information in the category "*sent (posted, provided) by me*" is "information set by the person (or posted or published)" [Jones, 2008, p. 35]. Personal information in the category "*(already) experienced by me*" is information "experienced by a person" [Jones, 2008, p. 35]. This includes both information under the person's control and information not under the person's control like, e.g., "pages a person views on the Web" [Jones, 2008, p. 35]. Personal information in the category "*relevant (useful) to "me"*" is information that is relevant or useful to a person. This "category cuts across others to include subsets of the information we control, information we've experienced before, and also new information we've never seen before" [Jones, 2008, p. 35]. For example, this includes an "article that is perfect for a report we're writing" [Jones, 2008, p. 35].

2.3.2 Personal Information Cloud and Perspectives on It

Moran defines a *personal information cloud* (PIC), i.e., "the 'working set' of information that is relevant to the individual and his work" [Moran&Zhai, 2007, p. 338], i.e., the information an individual KWer manages for herself and deals with in the activities she executes. Jones defines the same information set as the *personal space of information (PSI)*, focusing at the core on information that is under the KWer's control: "Personal information, in each of its senses, combines to form a single personal space of information (PSI) for each individual. A person has only one PSI. At its center, a person's PSI includes all the information items that are, at least nominally, under that person's control" [Jones&Teevan, 2007, p. 10]. Jones adds that a person each has "only one PSI" [Jones, 2008, p. 45] and that the "PSI is external to the person" [Jones, 2008, p. 45].

The KWer regards the personal information as "*one single body of information*" [Ravasio&Tscherter, 2007, p. 275], complementary to the personal information cloud. "Ravasio et al. (2004) indicate that users explicitly desire linking: 'most interviewees expressed the need to have their information linked together, e.g., article author and respective address book entry, or citation and cited article, etc.' (p. 169)" [Jones&Teevan, 2007, p. 143] [Ravasio et al., 2004].

Personal information collections are well-defined perspectives on the KWer's personal information. Boardman defines a collection of personal information to be "a self-contained set of items. Typically the members of a collection share a particular technological format and are accessed through a particular application" [Boardman, 2004, p. 15]. "PICs are personally managed subsets of a PSI. PICs are 'islands' in a PSI where people have made some conscious effort to control both the information that goes in and also, but usually not necessarily, how this information is organized. PICs can vary greatly with respect to the number, form, and content coherence of their items" [Jones&Teevan, 2007, p. 12]. For example, this includes "project-related information items that are initially dumped into a folder on our notebook computer and then organized over time" or a "carefully maintained collection of bookmarks to useful reference sites on the Web" [Jones&Teevan, 2007, p. 12].

Representations of domain-specific things that the KWer encounters in real-life are one subset of the entity representations in the KWer's personal information cloud. Domain-specific things are a notion for objects that the KWer experiences in real-life and deals with while executing her activities. These domain-specific things are part of the KWer's mental model, i.e., the KWer's subjective view on the world. These things are meaningful to the KWer in real-life and the KWer is aware of them as individual objects, e.g., persons, locations, tasks and meetings. The KWer has a particular relation to these things, e.g., a KWer knows a particular person or possesses a particular book. Their information representation is part of the personal information cloud, i.e., are the entities of the personal information cloud. These may be subjective and as such only meaningful to the KWer herself, like for example for particular tasks, topics and documents. It can be in any of the personal information categories mentioned above. For example, a KWer's personal information cloud contains persons the KWer knows of, locations the KWer has been to, tasks the KWer has executed, topics the KWer talked about or events like meetings that the KWer attended.

2.3.3 Personal Information Cloud as Part of the KWer's Personal Activities

When working on a particular personal activity, the KWer deals with entities of the KWer's personal information cloud. Kaptelinin points out that activities "often involve the use of *many types of information resources* managed across a range of applications. The resources relevant to a particular

task [personal activity] may be received via email (e.g., messages, attachment files, and embedded links) or may be created by the user herself" [Kaptelinin&Boardman, 2007, p. 303f].

A KWer handles a *particular personal information perspective for each activity type*. In personal *primary activities* personal information is involved beneath other, non-personal information. A personal primary activity doesn't put any constraints on the personal information, all available personal information can be relevant depending on the nature of the activity.

In personal *supporting activities*, however, mostly personal information is involved. Supporting activities focus through their specialized orientation on a particular perspective on personal information. This means that these activities involve personal information related to a foundational concept like, e.g., a task, person or meeting. These foundational concepts are depicted in brackets in Figure 11 and explained in section 2.4.4. For example, the personal task management activity focuses on tasks and related information. The personal social network management activity focuses on people and related information.

In both activities, personal information is involved in several roles: The KWer leverages personal information as input, processes this information in these activities and the results of these processes represent as well personal information. These information entities are involved in the higher-level task independent of any system representation.

2.4 Digital Workspaces Support a KWer's Activities

KWers use *integrated digital work environments*, i.e., workspaces, to support the execution of their personal activities, today mainly commercial workspace implementation follows the *desktop metaphor* [Kaptelinin&Czerwinski, 2007b, p. 1]. Integrated digital work environments are "environments based on a coherent set of principles supporting a coordinated use of tools and resources across various tasks and contexts" [Kaptelinin&Czerwinski, 2007b, p. 7]. Workspaces "contain information objects such as documents, messages, images, and music, and applications that support the production and consumption of information objects during communication, writing, and reading" [Kaptelinin&Boardman, 2007, p. 306].

Multiple applications run within a workspace and offer specialized functionalities to accompany the workspace. Each application provides specialized functionalities through its own user interface to the KWer. In a workspace, multiple applications can provide redundant functionality with minor differences, e.g., in the user interface, user interaction or performance. For example, both the Microsoft Internet Explorer and the Mozilla Firefox web browsers enable the KWer to browse the web. However, KWers have a particular preference for a particular application in the same functionality category because one or more features better suit the KWer's diverse needs.

The KWer interacts with the application using each application's user interface to access the application's functionality. The KWer's action to invoke and use certain application functionality is called a *lower-level KWer activity*. A lower-level activity is an activity the KWer conducts in the context of a particular application, for example sending an email in an email client application. These lower-level activities are "integral parts of larger-scale activities that span the physical and digital domains" [Kaptelinin&Boardman, 2007, p. 306], i.e., the KWer's higher-level tasks presented in section 2.2. Thus each lower-level activity contributes to a KWer's personal activity (either primary or supporting), but as such each lower-level activity doesn't satisfy the goal which a KWer pursues with the activity.

Seen from a higher-level, the activities "taken place in digital work environments [themselves] are integral parts of larger-scale activities that span the physical and digital domains. Accordingly, digital workspaces are integral parts of larger-scale physical-virtual environments [Kaptelinin&Boardman, 2007, p. 306].

Looking at the support for a KWer's higher-level activities, there are two perspectives on workspaces and the therein involved applications for the designer of such software. Figure 12 depicts the application-centric perspective and the workspace-level perspective using an email application as example.

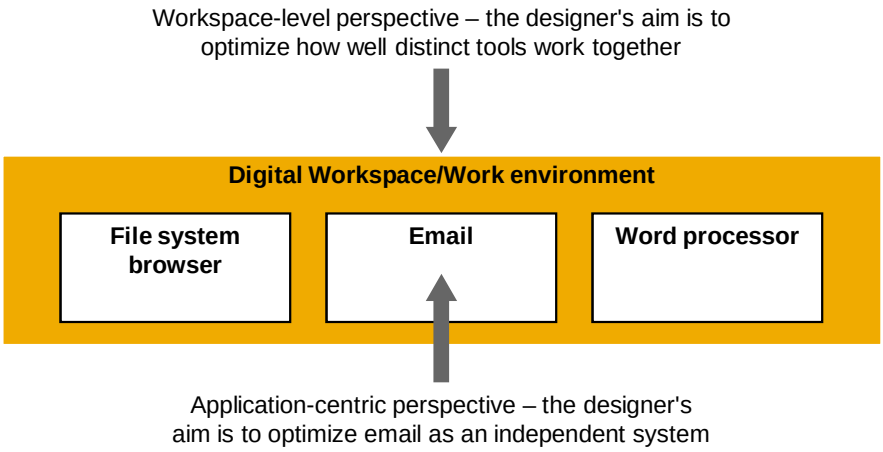


Figure 12: Application-centric and workspace-level perspectives [Kaptelinin&Boardman, 2007, p. 305].

From an *application-centric perspective*, the "designer's primary aim is the optimization of an independent application. The main design concern is what features and functions should be added to the application to make it more powerful" [Kaptelinin&Boardman, 2007, p. 305]. This includes for example "expanding email, making it more sophisticated and powerful by adding advanced features and functionalities" [Kaptelinin&Boardman, 2007, p. 295].

The contrasting *workspace-level perspective* is "based on the analysis of user needs at a higher-level, that of the digital workspace as a whole" [Kaptelinin&Boardman, 2007, p. 305]. Here the "designer's aim is to optimize how well the distinct applications work together within a workspace as a whole. A workspace can be defined as spatial, temporal, and logical organization of sources that support higher-level tasks" [Kaptelinin&Boardman, 2007, p. 305]. This includes for example actions to "improve integration between email and other applications" [Kaptelinin&Boardman, 2007, p. 295].

2.4.1 Workspace-Level Support through Manual Application Orchestration

Kaptelinin highlights that a "key aspect of performing higher-level tasks is therefore the *coordination of multiple applications*, such as email and the file system explorer. When carrying out higher-level tasks people need to switch between their digital subhabitats" [Kaptelinin&Boardman, 2007, p. 303f]. Again, higher-level tasks are "tasks that can be meaningfully defined independently of the applications with which they are carried out" [Kaptelinin&Boardman, 2007, p. 303]. "The emphasis on supporting high-level activities, involving a variety of applications, gives priority to flexible solutions based on dynamic constellations of tools" [Kaptelinin&Boardman, 2007, p. 306]. For example, "when people use email in support of higher-level tasks, [...] the employment of multiple applications is the rule rather than the exception" [Kaptelinin&Boardman, 2007, p. 306]. There the KWer typically invokes and *uses a number of applications* throughout the execution of one activity by

conducting a number of low-level activities. The KWer manually controls the sequence of the invocations. Due to the dynamic nature of the activity there is no formally specified path to execute a particular activity, i.e., the KWer manually orchestrates a number of applications to fulfill an activity. For example, a personal primary activity like, e.g., researching the competitive situation for rear axles in an engineering company requires the KWer to conduct several low-level activities in corresponding applications, see Figure 13.

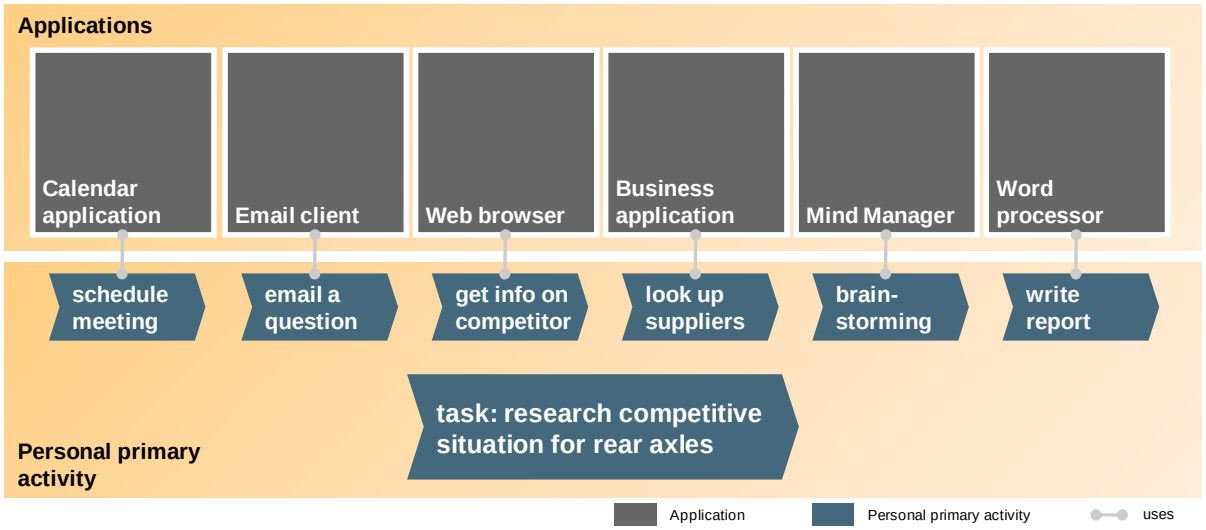


Figure 13: Example of a personal primary activity.

Of note is that even highly integrated business software applications often are only one part of the applications the KWer uses in a workspace to execute a personal activity. Business software applications are for example supply chain management (SCM), enterprise resource planning (ERP) or customer relationship management (CRM) applications. As well, a personal supporting activity like, e.g., managing a meeting on a discussion about the identified competitors requires the KWer to conduct several low-level activities in corresponding applications, see Figure 14.

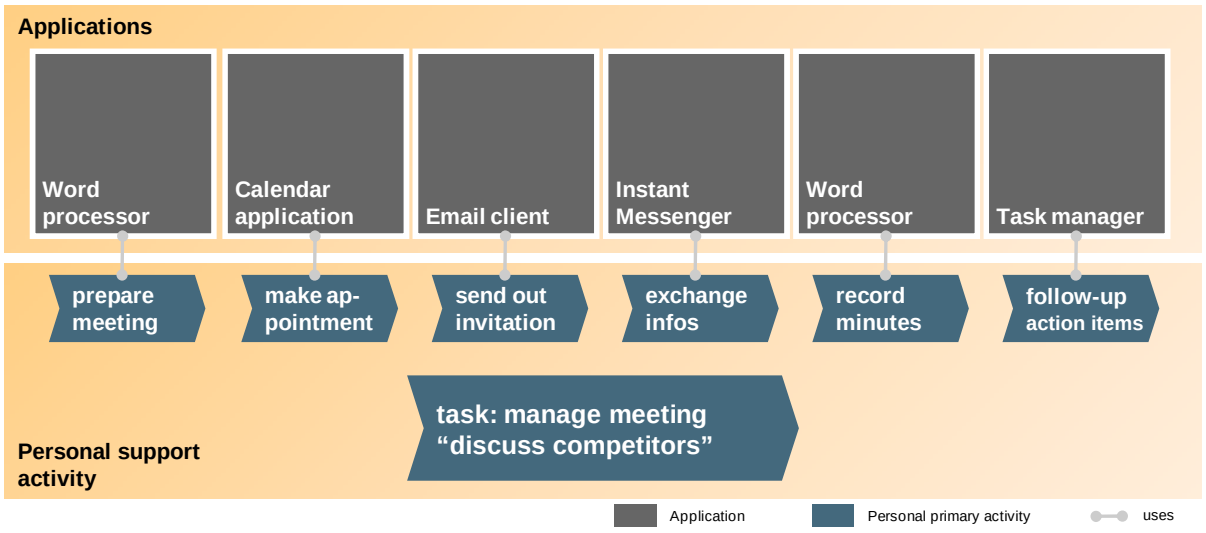


Figure 14: Example of a personal support activity.

Categorizing the applications that support a KWer's activities helps to understand, what applications a KWer actually uses and potentially can use. There are several types of applications when looking at the available applications to support a KWer's activities. The applications can be categorized along

two dimensions, the content structure they deal with and the degree of centralization of these applications. Figure 15 shows the main types of applications along these two dimensions. This perspective assumes that a KWer works within an enterprise.

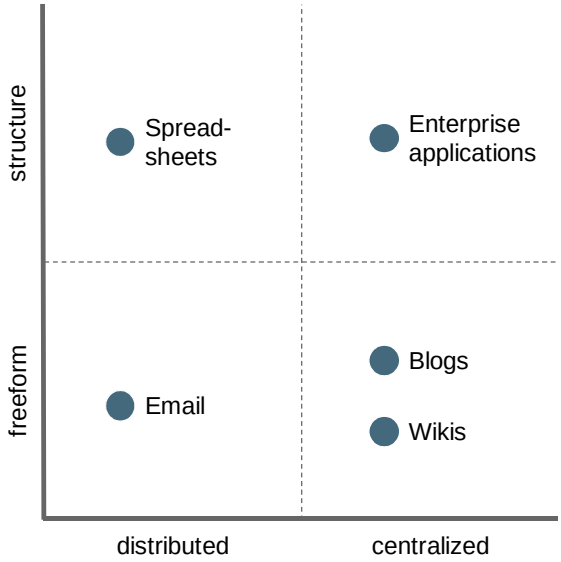


Figure 15: Application types categorized by Structure and Centralization. [Heuser et al., 2007, p. 48].

2.4.2 Limitations of Application-Centric KWer Activity Support

An *application-centric integration is limited* for certain higher level tasks. For example, an application like email covers only one part of potential collaboration support, i.e., as a "highly successful tool for asynchronous and primarily textual communication" [Kaptelinin&Boardman, 2007, p. 307]. However, much "collaboration takes place in collocated teams, where the emphasis is on face-to-face formal and informal meetings, phone calls, and so forth, rather than email communication" [Kaptelinin&Boardman, 2007, p. 307]. "Essential as it is, email is often just one of the diverse technologies used to support collaboration" [Kaptelinin&Boardman, 2007, p. 307].

Kaptelinin further explains this problem domain at hand of an email application. "For instance, in the TaskMaster system users can manually add resources from other applications to email. The usefulness of the system depends, therefore, on how many additional resources the user needs to add manually. If most resources are included automatically, the overhead may be insignificant. In contrast, if the user needs to spend substantial time manually adding resources to thrasks[4], the system may become unusable. The amount of manually added resources will depend on the applications people use outside email. In an empirical study conducted by Czerwinski, Horvitz, and Wilhite (2004) it was found that tasks, which can be described as 'email', constitute 23 percent of all tasks in the group of knowledge workers taking part in the study. The large proportion of 'non-email' tasks (i.e., tasks that are not explicitly described as 'email') can be interpreted as an indication that subjects carried out most of their task outside email" [Kaptelinin&Boardman, 2007, p. 304] [Czerwinski et al., 2004].

The KWer's situation depicted for the support of higher-level activities is different from a KWer, who conducts knowledge work on a relatively stable activity involving few applications and little personal information. I.e., for such an activity type few applications like for example an integrated business

[4] Remark: A thrask is in the TaskMaster system a combination of an email thread and a task.

software application provides the main source of information and represents the KWer's main workspace for the activity. From a KWer perspective, such work mainly involves one or more business software applications with often an integrated user interface. These applications are usually integrated among each other from the manufacturer, i.e., the KWer can use these applications in one client. This client transfers the information between several windows and centrally integrates information which thus is consistently available in the whole application. Compared to the above described situation of primary and support activities, a KWer doesn't, or at least with a significantly reduced amount, need to work with a number of applications, exchange information among these applications and keep and overview on this information. In addition, the amount of personal information involved in these processes is fairly low as these processes are largely based on organizational processes.

2.4.3 Alternative Workspace Designs to Better Support KWer Activities

However, today's desktop systems face the major drawback that there is *insufficient support for a KWer's higher-level activities* on workspace-level. Today's commercial desktop systems are "still designed primarily for document management" [Ravasio&Tscherter, 2007, p. 277]. Applications running on the desktop provide specialized functionalities suitable for different low-level tasks. However, it is up to each individual KWer to get results satisfying an activity out of the combination of available documents and desktop applications.

Human-computer-interaction research addresses this problem by *proposing multiple workspace alternatives* targeted each at a specific type of a KWer's activities instead of following a single workspace with the desktop metaphor [Moran&Zhai, 2007] [Kaptelinin&Czerwinski, 2007b, p. 6f]. Each of these workspaces function-wise supports a particular type of the KWer's activities, e.g., managing tasks or conducting meetings. Moran summarizes that "in the future a variety of advanced visual representations may be adapted to specific problem domains and different device forms, complementing a conventional desktop metaphor" [Moran&Zhai, 2007, p. 340].

Common to most of these workspaces is their *personal activity-orientation and the user interaction following a foundational concept*. First, each presented workspace represents a functional perspective that mainly targets a specific purpose, i.e., a set of a KWer's personal higher-level activities. The workspaces assemble workspace functionality according to a foundational concept and a KWer activity in contrast to the prevalent commercial workspaces which assemble their functionality along the boundaries of the therein contained applications. Second, the user interaction of the workspaces is designed around a primary foundational concept [Kaptelinin&Czerwinski, 2007b, p. 7] [Ravasio&Tscherter, 2007, p. 283f], i.e., the presentation of their functionality to the KWer follows a foundational concept and not a metaphor like the desktop-metaphor. Foundational concepts which predominantly form the respective workspace view are tasks and projects, collective activities, people, time, content like, e.g., document attributes and context like, e.g., personal roles. For example, ContactMap [Fisher&Nardi, 2007] offers its functionality in a people-centered way, i.e., it uses the foundational concept of people.

The presented *workspaces differ in the supported KWer activity type*. A part of the workspaces target a KWer's primary activities, i.e., general knowledge work, others target a KWer's supporting activities.

Among the workspaces targeting primary activities, a portion of them are specialized environments that offer functionality related to a certain domain, profession, job role or specialized user need. This means for the KWer that a specialized functional layer in the work environment offers support for

the KWer's particular personal primary process. The KWer therein has access to a dedicated set of functionality helping her to achieve her goals for the particular process. One example is the Research Desktop [Microsoft Cooporation, 2009], a digital integrated work environment whose integrated functionality targets researchers in their core production process of conducting research by, e.g., providing bibliography management and personal social network management embedded into the workspace.

The remaining workspaces addressing the KWer's primary activities provide general activity support without imposing any constraints on the domain of the activities. For example, Giornata [Voida et al., 2008] provides an activity-centered general purpose workspace with enhanced activity-related information and collaboration management.

Workspaces addressing the KWer's supporting activities provide supporting functionality for a dedicated set of these supporting activities. The user interaction correspondingly centers on a suitable foundational concept. For example, ContactMap [Fisher&Nardi, 2007] targets with the foundational concept people the personal process of managing the personal social network, i.e., building the personal social network, maintaining it and activating nodes within the network as needed [Fisher&Nardi, 2007, p. 171].

2.4.4 Overview on Alternative Workspace Designs

The presented research prototypes target different KWer activities and focus on different foundational concepts. Table 2 and Table 3 show known prototypes according to the respectively supported personal primary and supporting activities. These overviews are based on input from Ravasio and Tscherter [Ravasio&Tscherter, 2007, p. 283f].

Personal Primary Activity	Personal information perspective	Explored in Prototype
General primary activities • Use a workspace and applications	Task-centered (Task) • Involved information • Involved People	Giornata [Voida et al., 2008]
General primary activities • Use a workspace and applications	Time-centered	Lifestreams [Freeman&Fertig, 1995]
General primary activities • Use a workspace and applications	Content-centered view • "documents and other pieces of information" [Ravasio&Tscherter, 2007, p. 283f]	Presto [Dourish et al., 1999] Haystack [Adar et al., 1999]
General primary activities • Use a workspace and applications	Context-centered view • "the influence of situational issues and past actions on the present goals and activities of an individual user" [Ravasio&Tscherter, 2007, p. 283f].	Personal role management [Shneiderman&Plaisant, 1994] Kimura system [Voida et al., 2002]

General primary activities, focused domain-specifically on research • Query libraries • Manage bibliography • Compile information for reports	Multiple foci, e.g., • People-centered • Task-centered • Documents-centered	Microsoft Research Desktop [Microsoft Cooporation, 2009]

Table 2: Overview workspace prototypes for personal primary activities.

Personal Supporting Activity	Personal information perspective	Explored in Prototype
Manage personal tasks • Manage and prioritize a task list • Manage related information and people	Task-centered (Task)	Personal Role Management [Shneiderman&Plaisant, 1994], Soylent [Fisher&Dourish, 2004] Task Gallery [Robertson et al., 2000] TaskMaster [Bellotti et al., 2002] UMEA [Kaptelinin, 2003]
Manage personal time • Plan and control personal time spending	Task/Time-centered (Task) • Involved Information • Involved Tasks	PTM Personal Task Manager [Personal Task Manager, 2009]
Manage personal social network • Get overview on people • Maintain personal social network • Acquire new people	Person-centered (Person) • Involved Information • Involved Tasks	ContactMap [Fisher&Nardi, 2007] [Nardi et al., 2002a]
Manage personal meetings • Prepare, conduct and post-process a meeting	Meeting • Involved Information • Involved People	JourFixe [3ado AG (triado), 2009] Adobe Acrobat Connect [Adobe Inc., 2009] (See section 4.6 for further applications)
Manage Personal Information • Collect, organize, browse and retrieve information	Information-centered (All) • Overall personal information	Tiddly Wiki [UnaMesa Association, 2009] MediaWiki [Wikimedia Foundation Inc., 2009a]
Manage personal collaborations • Communicate and interact with people	Person/Information-centered (Person) • Involved Information • Involved Tasks • Involved People	Microsoft Outlook [Microsoft Corporation, 2009a]

Table 3: Overview workspace prototypes for personal supporting activities.

2.4.5 Examples for Higher-Level Activity Support on Workspace-Level

Exemplarily, we present two prototype designs in detail for each type of KWer activity. Giornata targets a KWer's primary activities and ContactMap targets the KWer's supporting activity of personal social network management.

Giornata [Voida et al., 2008] supports the KWer in the general work as individual and in collaboration with other people. Giornata thus targets a KWer's personal primary activities and thereby doesn't limit its support to a particular domain.

As part of the general work support for higher-level activities, Giornata supports the personal information management for a particular activity and supports the KWer's collaboration process.

Giornata's main perspective on the KWer's personal information is *activity-centric*, i.e., a task representation is the foundational concept of the prototype interface. Further involved information objects (and attributes) are people, other activities and information elements of an activity.

Giornata enables the KWer to manage for each activity a corresponding workspace with a desktop visualization, i.e., Giornata provides for each KWer activity a workspace and desktop, see Figure 16. For each activity, Giornata manages a task representation, which can be unnamed in case that KWer cannot assign a name yet. An information container is integrated into the desktop visualization which keeps information elements like, e.g., files, for the particular activity ("Per-Activity Resource Storage"). As well, through a collaboration sidebar, the KWer can interact with people related to the activity ("Contact Pallete"). The KWer can switch between task representations using a hotkey combination like it is known for switching between application windows in commercial desktop workspaces ("Virtual Desktop Manager").

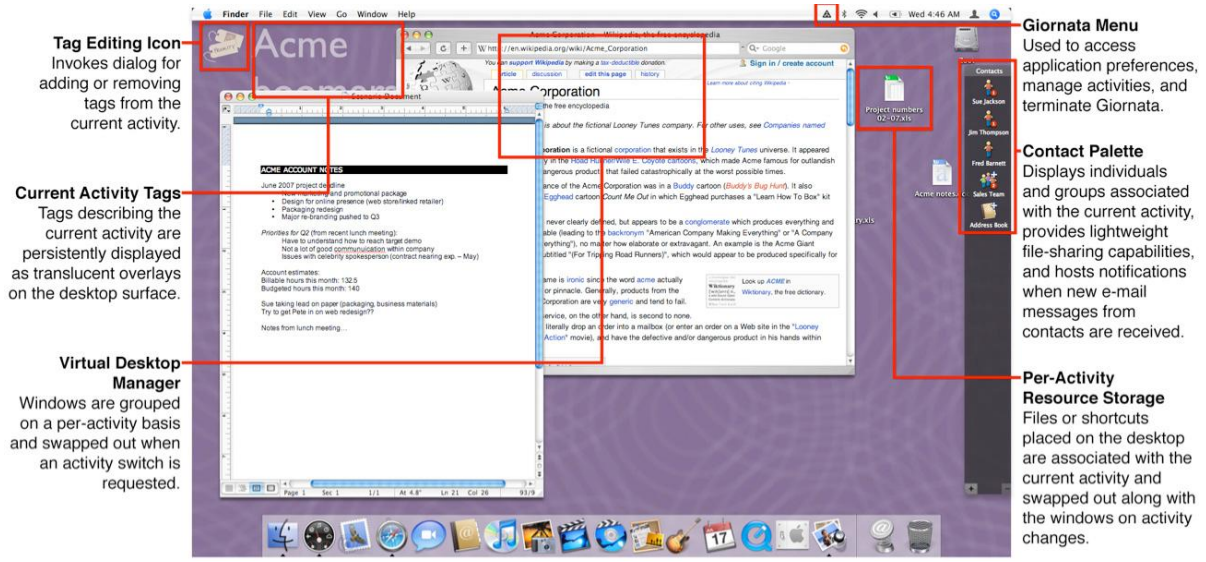


Figure 16: Giornata user interface – activity-centered information management [Voida&Mynatt D., 2009].

ContactMap [Fisher&Nardi, 2007] [Nardi et al., 2002a] supports the KWer in managing the personal social network. ContactMap thus targets a specific KWer activity, a personal supporting activity. As part of this activity, it covers the three interpersonal activities of "building social networks", "maintaining the networks" and "activating nodes within the networks as needed" [Fisher&Nardi, 2007, p. 171]. As part of these activities the KWer can manage the communication and information related to the involved people.

ContactMap acknowledges that "personal social networks are critical resources in today's economy (Nardi, Whittaker, and Schwarz 2002)" [Fisher&Nardi, 2007, p. 173]. Nardi, Whittaker and Schwarz [Nardi et al., 2002b] found that "workers were careful managers of their personal social networks" [Fisher&Nardi, 2007, p. 173]. Users manage "these tasks with communication and contact-

management tools" [Fisher&Nardi, 2007, p. 171], however these tools have several drawbacks, from information loss of message attachments to separate storage and presentation of different tools' communication histories. Thus, "users attempt to work around the problems of current tools" [Fisher&Nardi, 2007, p. 171]. This repurposing of tools still leads to unsatisfying results and Fisher and Nardi come to the conclusion that the "social workspace is incompletely supported by current technology, no matter how many workarounds users devise" [Fisher&Nardi, 2007, p. 171].

ContactMap's main perspective on the KWer's personal information is people-centric, i.e., a person is the foundational concept of the prototype interface. Further involved information objects (and attributes) are people, relationships between the people, activities and information elements.

ContactMap enables the KWer to manage personal social networks, i.e., by "displaying the contacts in the user's social network and providing functionality relevant to those contacts" [Fisher&Nardi, 2007, p. 173]. Figure 17 depicts the ContactMap application. Depicted on the right, the KWer's personal social network is shown through the persons grouped by their relationship to the KWer. On the left, the information details are shown for a selected person and the KWer can interact with that person, e.g., to initiate a phone call or send an email. This way, ContactMap "integrates communication and information management in a single user interface" [Fisher&Nardi, 2007, p. 174].

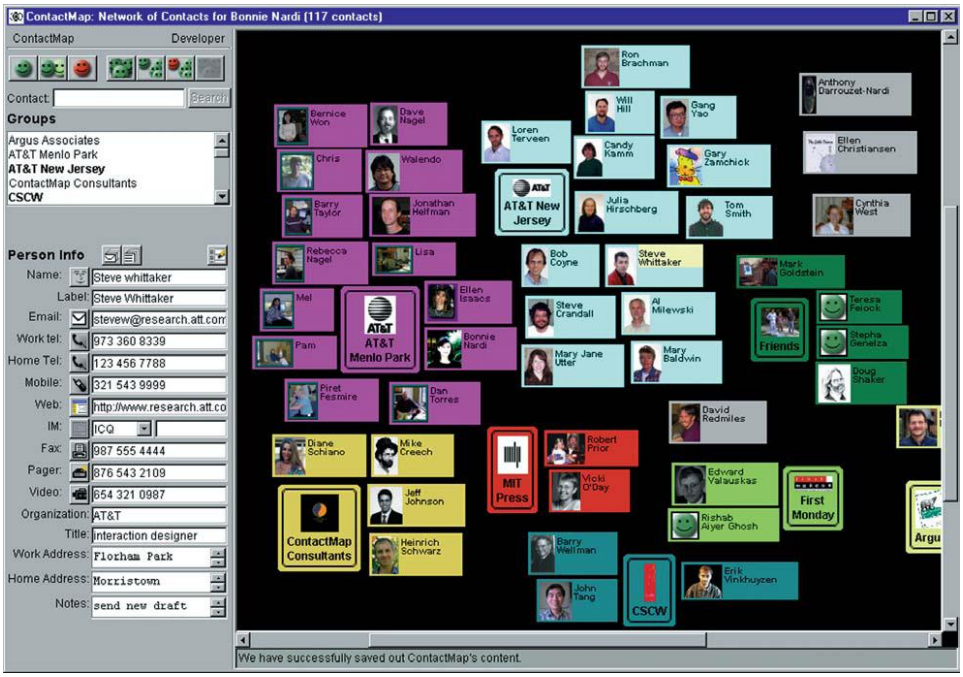


Figure 17: ContactMap visualization of a KWer's personal social network [Nardi et al., 2002a].

2.5 Personal Information Fragmentation across Workspace Applications

Information fragmentation of personal information across applications in the workspace is a major problem for KWer activity support. For example, this scattered personal information in application information silos "intensifies the cognitive load" [Ravasio&Tscherter, 2007, p. 275], especially as the KWer regards, like already mentioned, personal information as "one single body of information" [Ravasio&Tscherter, 2007, p. 275].

In general, information fragmentation "is a pervasive problem in personal information management (PIM)" [Jones&Teevan, 2007, p. 127], even in real-life and thus without computer involvement. For example, this is shown when even "a seemingly simple decision, such as whether to say "yes" to a

dinner invitation, often depends upon information from several sources - a calendar, a paper flyer, Web sites, or a previous email conversation" [Jones&Teevan, 2007, p. 127]. Information fragmentation “will always be a problem but, like information overload, can assume forms ranging from relatively benign to purely malignant" [Jones, 2008, p. 392].

Information fragmentation at the boundaries of physical and digital representations will always be present but can be mediated to some extent with computer-support. For example, when the KWer involves in an activity both information processed with computers and information only available in the real-world like, e.g., paper-based books. The information in the physical artefact can be transferred into the computer, e.g., through scanning the book, and is then available digitally.

2.5.1 Personal Information Fragmentation Types in Workspaces

The KWer’s *personal information is fragmented* across applications in digital workspaces, i.e., the *KWer’s personal information is scattered across the desktop and its applications* [Jones, 2008]. Current desktop systems and applications don't represent the personal information cloud in a unified and integrated form as the KWer expects it, i.e., the personal information is fragmented across applications in the workspace. E.g., an email client folder containing emails has for the KWer no visible connection to a file system folder containing documents despite dealing with the same topic.

Workspace applications *managing only a particular type of personal information and locking it in* a proprietary storage causes one major type of personal information fragmentation. Karger mentions that these applications “often store their data in their own particular locations and representations, inaccessible to other applications" [Jones&Teevan, 2007, p. 127]. Karger outlines this type of information fragmentation as paradox as this “information is fragmented by the very tools that have been designed to help us manage it” [Jones&Teevan, 2007, p. 127]. Jones as well sees here "many examples of seemingly avoidable information fragmentation" [Jones, 2008, p. 392]. Jones mentions as example the failed attempt of referencing an email in the file folder hierarchy as the file system doesn’t support inserting shortcuts from, e.g., Microsoft Outlook to other items than files. Karger furthermore mentions the KWer’s "need to manipulate multiple pieces of information in ways that cross application boundaries. Users like to gather information in ways that cross application boundaries. Users like to gather information objects into groups that will be used together, and organize or annotate those groups in ways that will make it possible to retrieve them based on their intended usage. The artificial separations imposed by separate application data models can interfere with this organizational urge" [Jones&Teevan, 2007, p. 129].

Another related type of *information fragmentation is produced by the KWer herself*. Due to application information silos, the KWer is forced to fragment the own personal information when keeping it. This is due as each application only manages particular own information types, i.e., information silos, regardless of the KWer's intent for keeping the information. Bergmann reports this type of personal information fragmentation problem in the context of organizing project-related information, called project fragmentation problem [Bergman et al., 2006]. Figure 18 shows an example of "information related to a chemistry course" which is "fragmented into separate collections" across three applications. The involved applications' ability of managing each only a particular information element type forces the KWer to organize the information in a fragmented way.

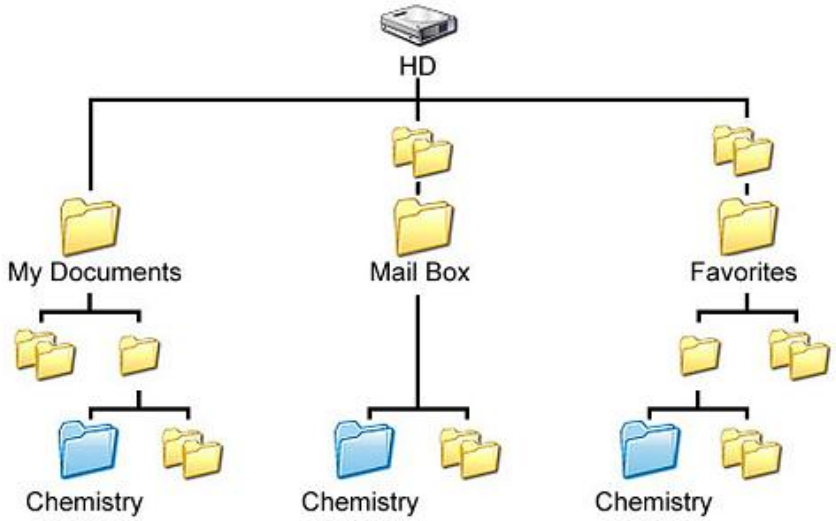


Figure 18: Personal information fragmentation example – project-related [Bergman et al., 2006].

In addition, the fragmentation of personal information can occur across several devices, where each device maintains its own storage. Jones mentions that information "is further fragmented as new computers and new devices replace older versions" [Jones, 2008, p. 159][5].

2.5.2 Negative Impact on a KWer's Activity Support in Workspaces

The personal information fragmentation problem negatively impacts KWer activity support in several ways. Overall it is a major problem for digital workspaces. In detail, the described fragmentation leads to numerous problems.

Karger describes general personal information management problems [Jones&Teevan, 2007, p. 127]:

- No single directory: No single directory exists to find all information about a particular topic.
- No possibility to link information: there's no way to link pieces of information about this topic to each other.
- Repetitive searches: Instead a KWer needs to start "multiple applications and perform numerous repetitive searches for relevant information" [Jones&Teevan, 2007, p. 127]. In addition, the KWer needs to decide "which application to look in" [Jones&Teevan, 2007, p. 127].
- Inconsistency: The KWer "may change data in one place (a new married name in the address book) and fail to change it elsewhere, leading to inconsistency that makes it even harder to find information" [Jones&Teevan, 2007, p. 127].

In a particular domain of supporting a KWer's activities, for example for personal task management as well as collaboration activities, "the fragmentation of commonly used information resources—first and foremost, e-mail—make it difficult for activity representations to adequately capture both local work products and communicative interactions" [Voida et al., 2008].

Overall, scattered personal information in application information silos "intensifies the cognitive load" [Ravasio&Tscherter, 2007, p. 275], especially as the KWer regards, like already mentioned, personal information as "one single body of information" [Ravasio&Tscherter, 2007, p. 275].

[5] Here we don't consider this problem type further. It adds the dimension for versioning and maintenance of personal information over time. We focus here on the current KWer's information.

2.6 Unification and Integration – Resolving Information Fragmentation

Both unification and integration are needed to overcome information fragmentation in digital workspaces.

Integration can be defined as the "act or process of making whole or entire. [Webster's, 2009a] and the verb integrate means to "form into one whole; to make entire; to complete; to renew; to restore; to perfect" [Webster's, 2009b]. *Unification* can be defined as "the act, process, or result of unifying: the state of being unified" [Merriam-Webster Inc., 2009].

Jones distinguishes integration and unification. "With *integration*, pieces fit together to make a more perfect whole, but still retain their identity as separate pieces, somewhat like a mosaic. With *unification*, the pieces blend together and, at least with respect to the focus of unification, the pieces lose the ability to act independently of one another" [Jones, 2008, p. 373].

Both unification and integration feature particular benefits and costs. The goal is to find the right balance between how much information integration and unification addresses the prevalent information fragmentation without erasing useful boundaries among the information. "Even as these schemes [of information integration and unification] remove separations that cause us problems, they may inadvertently remove separations that are useful, even essential, to our practices of PIM" [Jones, 2008, p. 372]. Jones states on the ratio of unification or integration that is needed or more important for synergy that mostly "integration [...] but a little unification is needed as well" [Jones, 2008, p. 373]. This complies with Karger stating that unification is "not a complete solution, but an enabler of solutions" [Jones&Teevan, 2007, p. 151].

Looking at the *personal information unification benefits and costs* in "general, the benefits of unification need to be weighed against its costs" [Jones, 2008, p. 373].

The *benefits of unification* are primarily the "unified use of structured storage across applications" [Jones, 2008, p. 373]. "If different applications surrender autonomy in their ways of storing and using some kinds of information (e.g., contact information) and, instead, make uniform use of a single system-level function for the write and read of this information, then corrections, updates and the uniform application of an auto-complete feature become much easier" [Jones, 2008, p. 373].

The *costs of unification* are that the use "of a shared store means a surrender of control (e.g., to the operating system) and possibly a degradation in performance as well" [Jones, 2008, p. 373]. Jones sees for unification the requirement of "a single, shared store. Coordinated use of separate stores is not likely to cut it. Imagine, for example, the combinatorial nightmare that could follow from an attempt to communicate and synchronize changes between several different stores" [Jones, 2008, p. 373].

2.6.1 Unification

Unification means to bring "information together to some useful purpose. To achieve this goal, any unification approach must choose some 'least common denominator'" [Jones&Teevan, 2007, p. 130]. "There are many possible common denominators, each imposing different constraints and offering different benefits" [Jones&Teevan, 2007, p. 130].

"Using such a common denominator is in tension with applications' needs for rich, specialized representations of their content. Rich representations let applications offer powerful, domain-specific operations. The tradeoff is that a simplified shared representation lets applications interact

in a unified fashion with data from many applications and domains without needing to understand a multitude of rich representations" [Jones&Teevan, 2007, p. 130]

Karger argues that "the key step for such unification is the choice of an *appropriate common API* - a representation of the data and interfaces to it that can be accessed by all applications. Such a Application Programming Interface (API) must be simple enough not to discourage application builders from supporting it". Karger mentions "a few APIs: pixels, text, files, groups, annotations, and links" [Jones&Teevan, 2007, p. 150] and discusses "the representations those APIs offer and the actions they enable" [Jones&Teevan, 2007, p. 150]. Table 4 shows an overview on possible forms of unifications and explains possible actions for the KWer enabled by each form of unification.

Technique	Offered by	Operations	Enable
Visual unification	window manager, wincuts, embedding	layout, tile, show, hide	simultaneous view of information
Standard datatypes	text, files	cut/copy/paste, read/write	unified searching, data transfer
Metadata		refer, describe	list below
Grouping	directories, TaskMaster, Presto	add/remove items from group	organize, browse, simultaneous view
Cross-reference	OLE, COM, WWW	embed, traverse	simultaneous view, orienteer
Attribute/value	XML, Presto	annotate, query	annotate, search, organize, browse
Relations	RDF, Databases, Haystack		record relationships, unified search

Table 4: Different unification techniques (offered by different software systems) [Karger, 2007, p. 132].

Of note is, that a "unified organizational framework" [Jones&Teevan, 2007, p. 151] doesn't inevitably mean to squash all information "into a 'one size fits all' (and therefore ill-fitting) organizational structure" [Jones&Teevan, 2007, p. 151]. Karger continues to emphasize that it "is important to distinguish, however, between a unified organizational system and a single organizational system. Even if it is possible to create heterogeneous collections, there is still likely to be value in seeing a heterogeneous collection of, say, all of a user's email. Since references to an individual item can appear in multiple places, we can simultaneously maintain today's application-driven data partitions (to the extent that they are useful) and also offer task-specific, cross-application collections of the same information" [Jones&Teevan, 2007, p. 151].

Besides the unified information model, the interface supporting the model is of high importance. "A unified data model does no good if it sits inert on disk; interfaces must be designed to exploit the unified model" [Jones&Teevan, 2007, p. 151]. Karger further mentions that the key is user interaction with the unified information. "Much research is ongoing to find the right metaphor or paradigm for letting people interact with their unified information" [Jones&Teevan, 2007, p. 151]. Karger mentions for example the workspace prototypes Lifestreams, Presto, ContactMap, TaskMaster, Universal Labeler and UMEA, as presented in section 2.4.4.

Of the presented unification possibilities, Kaptelinin and Czerwinski find four principle designs applied in these workspace prototypes to achieve a unified personal information model by integrating different information hierarchies [Kaptelinin&Czerwinski, 2007b, p. 8]. These are:

- "Using the same logical attributes across different types of information objects ([Freeman&Gelernter, 2007], [Karger, 2007])" corresponds to metadata-based unification.
- "Linking information objects to roles, contacts, or projects ([Plaisant et al., 2007], [Robertson et al., 2007], [Fisher&Nardi, 2007], [Voida et al., 2007], [Kaptelinin&Boardman, 2007])" corresponds as well to metadata-based unification.
- "Organizing windows into groups ([Robertson et al., 2007])" corresponds to visually-based unification.
- "Maintaining a uniform structure across existing hierarchies ([Kaptelinin&Boardman, 2007])" corresponds to metadata-based unification. They demonstrate for example the workspace-level connection of personal information entities across desktop applications by connecting corresponding file system folders and email client folders.

Karger concludes that common to all of these workspace prototypes "is the recognition that the partition of information induced by diverse data formats and solitary applications is not the organizational metaphor consistent with people's uses of their information, and that a unified data model is a critical contribution toward organizing and presenting information to the way they want it" [Jones&Teevan, 2007, p. 151f].

2.6.2 Integration

"PIM poses a special kind of data integration challenge since personal information – for example, the information a person needs to complete a current project – is often drawn from several sources represented in several different formats" [Jones&Teevan, 2007, p. 112]. As example, Catarci et al. present a situation where a KWer "needs information that comes from his email, his address book, and his pictures" [Jones&Teevan, 2007, p. 112]. "Such different information needs to be integrated and reconciled in order to be efficiently accessed" [Jones&Teevan, 2007, p. 112] by a KWer.

Jones categorizes different kinds of personal information integration [Jones, 2008, p. 379f]:

- *Visual integration*. "Computers provide several alternatives for comparable viewings of digital information, including the computer desktop, a folder listing of existing files (or email messages or web references), and the window displays of opened documents, email messages, and web sites. Our view of items can act as a powerful extension to our limited internal working memory for information" [Jones, 2008, p. 379].
- *Feature integration*. "Features, within and across applications, should work together according to our activities" [Jones, 2008, p. 379].
- *Integration of task, time, and information management*. "It should be possible to group and tag information according to task and time of anticipated use" [Jones, 2008, p. 379].
- *Integration of activities and experiences (context)*. "Activities should be integrated with elements of the context (location, time, sound, sight) in which they occur" [Jones, 2008, p. 380].
- *Integration of activities with each other*. "Look for ways that one activity might leverage another" [Jones, 2008, p. 380]. "For example, we plan projects and we organize project-related information. The to-do list or the outline we create as we plan a project can also provide a basis for the organization of project-related information. Can good organization emerge as a by-product of planning" [Jones, 2008, p. 380]?
- *Integration of the new and the old*. "For example, people may have considerable time and energy invested in existing folder hierarchies and other organizations. Moreover, these

> organizations and supporting applications are used in many ways that are not well understood. Consequently, a new tool has a better chance of success if it is able to build on these organizations and extend the functionality of existing applications rather than forcing a leap to an entirely new way of doing things" [Jones, 2008, p. 380].

Integrating applications is possible from two different perspectives, i.e., the above presented workspace-level and application-centric perspectives. This means to enhance support for higher-level activities by either integrating additional functionality into an application or by integrating several applications to work together.

First, application-centric integration tries to integrate all relevant functionality needed to support a particular higher-level task into one application, e.g., by enhancing an existing application. However, this approach has limits. Kaptelinin and Boardman [Kaptelinin&Boardman, 2007, p. 295ff] explain that that it doesn't make sense to put too much functionality into one application, as the primary functionality is then overloaded and the user's attention is distorted. For example, an email client can be enhanced with task management functionality, but Kaptelinin and Boardman found that it may distract the quick reading through email and thus overall dilute the key strengths of email. In addition, when the KWer needs to pick functionality from more than one application, productivity is hindered as these applications usually are not integrated and thus information loss can occur and needs to be mitigated.

Second, the *workspace-level integration* of applications connects applications through glue-applications that provide information exchange between the applications. Boardman presented one possible integration among several applications across their boundaries by copying corresponding folders between the file system and an email client [Kaptelinin&Boardman, 2007, p. 295ff]. The benefit of this approach is that a set of specialized applications interacts together on the desktop while the information is integrated. On the downside of this particular approach, this third-party integration with an additional application taking care of the information integration results in synchronization issues and thus leads to user acceptance problems.

2.7 PIM System (Semantic Desktop) for Managing Unified Personal Information

A *PIM system* provides an infrastructure for managing unified personal information on a computer. Recently, research has established the term 'semantic desktop' for a special type of PIM system which represents unified personal information using semantic representations like, e.g., RDF [World Wide Web Consortium (W3C), 2009].

From an information management perspective, PIM Systems that integrate and unify a KWer's personal information mainly follow two *goals*. On the one hand they store personal information accounting their semantics and on the other hand they query this personal information efficiently.

A PIM system provides a *unified representation for the KWer's personal information*. The *unification* takes place on metadata-level through, e.g., RDF [World Wide Web Consortium (W3C), 2009]. A corresponding unified personal information model provides a schema for this information. The unified personal information model focuses on things relevant to the KWer, i.e., the unified personal information model performs personal information *integration* around the KWer's things.

A PIM system features *functionality* to maintain this unified personal information and to support the stated goals. A PIM system integrates artifacts from existing digital environments into the personal information, such as desktop information objects like emails, websites and files on a desktop computer. Furthermore, it enables the KWer to maintain and access this personal information and provides services to maintain and access the unified personal information across applications.

2.7.1 PIM System Requirements for Personal Information Integration

A PIM system that integrates and unifies personal information needs cover several data integration aspects. In this context, data integration is challenging as personal information "is often drawn from several sources represented in several different formats" [Jones&Teevan, 2007, p. 112]. Catarci et al. mention *important issues for data integration* on personal information:

- *Personal information structuring.* For setting up a logical framework for data integration, "the process begins by specifying the mediated schema to be accessed by users and the data sources. The relationships between sources and the mediated schema are called *mappings*. The mappings are needed for query processing when the query execution engine will reformulate a query proposed over the mediated schema into appropriate queries over the data sources" [Jones&Teevan, 2007, p. 113].
- *Data extraction*. *Wrappers* need to be constructed in the process. They "are responsible for transforming the data at the sources into a form that makes them suitable for use in the data integration system" [Jones&Teevan, 2007, p. 113]. Catarci et al. further specify that in particular, "it is common practice to require that wrappers produce data that all conform to the same data model, such as the relational data model, thereby enabling all further processing on the data to be performed in a single data model" [Jones&Teevan, 2007, p. 113].
- *Schema matching*. "The goal of this element is to provide tools that take two schemas as input and produce a semantic mapping between elements of the two schemas" [Jones&Teevan, 2007, p. 114].
- *Quality*. The notions of "data quality, quality of answers [...] and data cleaning [...]" [Jones&Teevan, 2007, p. 114] need to be incorporated in the data integration setting. Since "personal information may come from an unreliable source, or may be obsolete, it is important to associate with data a notion of quality. For example, the address book may be considered very reliable, whereas information extracted from emails coming from unknown contacts should be considered much less reliable" [Jones&Teevan, 2007, p. 114].

2.7.2 PIM System Examples: Nepomuk Semantic Desktop and OntoPIM

We present the PIM system architecture at hand of two PIM System examples, whereas in the following, we consider only the Nepomuk PIM system in detail.

Nepomuk [Nepomuk Consortium, 2009a] [Groza et al., 2007], an acronym for 'Networked Environment for Personalized, Ontology-based Management of Unified Knowledge', is a PIM system also known as '*Social Semantic Desktop'*. It represents a "comprehensive solution for extending the personal desktop into a collaboration environment which supports both the personal information management and the sharing and exchange across social and organizational relations" [Nepomuk Consortium, 2009b].

Nepomuk is an example of a *semantic desktop.* Sauermann defines this PIM system category as follows: "A Semantic Desktop is a device in which an individual stores all her digital information like documents, multimedia and messages. These are interpreted as Semantic Web resources, each is

identified by a Uniform Resource Identifier (URI) and all data is accessible and queryable as RDF graph. Resources from the web can be stored and authored content can be shared with others. Ontologies allow the user to express personal mental models and form the semantic glue interconnecting information and systems. Applications respect this and store, read and communicate via ontologies and Semantic Web protocols. The Semantic Desktop is an enlarged supplement to the user's memory" [Sauermann et al., 2005].

OntoPIM [Katifori et al., 2005] is another PIM system with similar characteristics to support the two mentioned main goals. First, to "store any object of interest according to its semantics, or in other words, to relate it to the concepts of the personal ontology, where an object may be a mail, a document, a picture, or any other type of data" [Jones&Teevan, 2007, p. 118]. Second, to "effectively query such a personal ontology according to the user's needs and preferences" [Jones&Teevan, 2007, p. 118].

2.7.3 PIM System Architecture

From a software architecture perspective, a PIM system consists of several data layers, i.e., representations of personal information models, and of modules handling this information. The modules cover the PIM System requirements for personal information integration mentioned above. Table 5 gives an overview of these data layers and modules. Thereby it shows whether the two example PIM systems support these.

PIM system	Nepomuk	OntoPIM
Data layers		
Physical layer	Physical objects of computer, e.g., files, websites, emails, ...	Physical objects of computer
Domain independent (DI) layer	Nepomuk Information Elements (NIE) Ontology [Nepomuk Consortium, 2009c]	DI layer (with DI objects representing physical objects)
Domain specific (DS) layer	Personal Information Model (PIMO) Ontology [Sauermann et al., 2007]	DS layer (with DS objects, referencing DI objects where needed)
Modules		
Unified personal information model editor	PIMOEditor [Sauermann, 2009]	Partly through • Personal Ontology Builder (POB) • Mapping Builder (MB)
PIM Service	PIMO Service [Sauermann&Klinkigt, 2009]	Query Processor (QP)
Data alignment	LocalDataAlignment [Sauermann et al., 2009a]	-
Data wrapper	Aperture Data Wrapper [Aduna B.V.&DFKI GmbH, 2005]	Partly through • Semantic Save Manager (SSM)
Unified personal information repository	Local Storage, based on Sesame2 [Aduna B.V., 2009a]	N/A
Service-oriented architecture	Nepomuk service registry and messaging [Nepomuk Consortium, 2009d]	-

Table 5: PIM System data layers and modules, with examples OntoPIM and Nepomuk.

Figure 19 shows the generic semantic desktop PIM system architecture based on the described data layers and modules. These will be further explained in detail below. Applications using the semantic desktop PIM system are depicted above the semantic desktop PIM system.

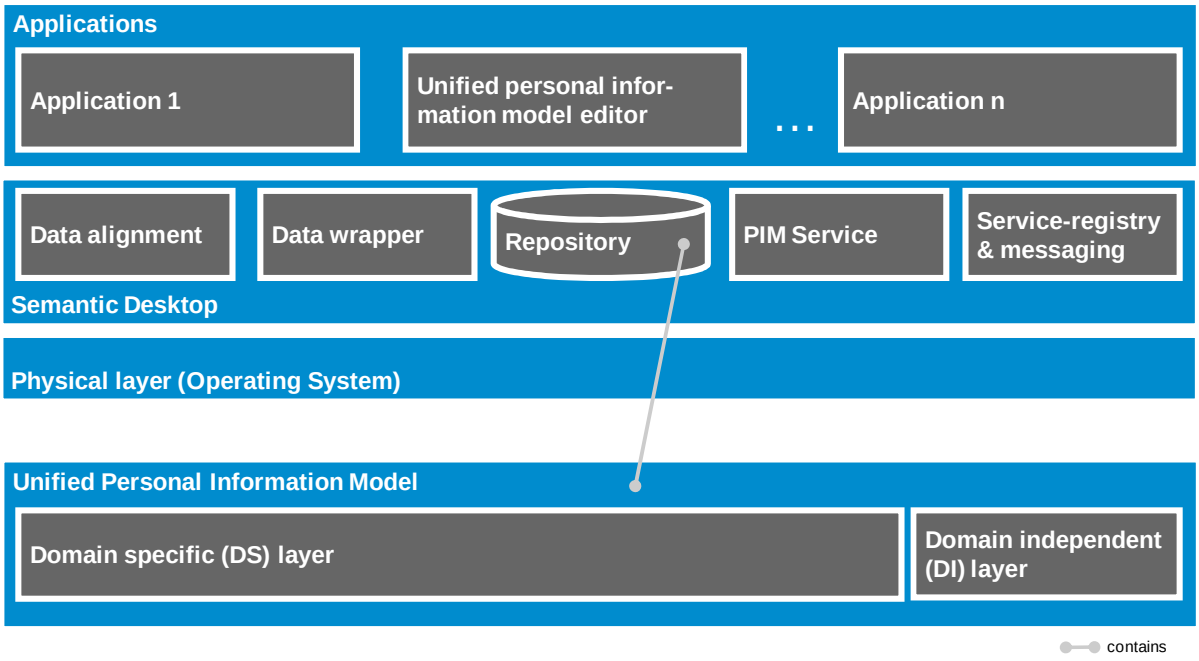


Figure 19: Generic architecture of a semantic desktop PIM system.

2.7.4 PIM System Data Layers

PIM systems involve three *data layers*. On a *physical layer* computer devices store physical objects such as emails and files like, e.g., word processor documents or spreadsheet documents, see 'Physical layer' in the architecture view in Figure 19. These information items following a specific form are part of the KWer's personal information. Nepomuk bases on a computer with a desktop-based workspace. It thus covers emails from email clients, files in the file system as well as web-based resources on the desktop. OntoPIM refers to this as physical layer, "storing files or relational tables or any other physical objects that can be stored on a PC" [Jones&Teevan, 2007, p. 118].

A *domain independent (DI) layer* represents domain independent objects from the physical layer, i.e., wraps existing physical objects and stores them in a unified form. The unified personal information repository as introduced in the next section stores the DI layer, see 'DI layer' in the architecture view in Figure 19. Nepomuk uses the Nepomuk Information Elements (NIE) Ontology [Nepomuk Consortium, 2009c] to represent metadata of, e.g., emails, appointments, files as well as videos and audio files in a unified form. OntoPIM refers to this as "first wrapper layer (DI layer) representing domain independent objects from the physical layer, such as Alex's address book entries, emails, documents, photos, and so on" [Jones&Teevan, 2007, p. 118]

A *domain* specific *(DS) layer* represents domain specific objects the KWer knows of as part of her personal information cloud. Catarci et al. further mention that such "a representation is intended to be completely independent of the physical representation of information" [Jones&Teevan, 2007, p. 119]. The DS layer references physical objects by referencing a representation from the DI layer. The unified personal information repository as introduced in the next section stores the DS layer, see 'DS layer' in the architecture view in Figure 19. Nepomuk uses the Personal Information Model (PIMO) Ontology [Sauermann et al., 2007] to represent the KWer's personal information objects in a unified

form based on RDF. It represents an integrated personal information model as it focuses on the domain-specific things a KWer encounters during conducting activities. OntoPIM refers to this as "second wrapper layer (DS layer) representing domain specific (DS) objects, such as family representations or invitation cards, in the running scenario. Note that references in OntoPIM correspond to DS object attributes" [Jones&Teevan, 2007, p. 119].

2.7.5 PIM System Core Modules

PIM systems involve six *core modules*. A "*unified personal information model editor*" module enables the KWer to interact with the model of the unified personal information representation, i.e., the unified personal information model. In Nepomuk, the PIMOEditor [Sauermann, 2009] enables the KWer to view and modify the unified personal information model represented by the PIMO ontology. In OntoPIM, the KWer interacts with the 'Personal Ontology Builder (POB)' [Jones&Teevan, 2007, p. 119] to build her own personal ontology represented in the DS Layer. Furthermore, the 'Mapping Builder (MB)' "allows the user to create and modify the types [of] his DS objects" [Jones&Teevan, 2007, p. 120]. However, in OntoPIM the user interaction focuses on building an ontology, whereas the PIMOEditor is more targeted to non-information-modeling-aware end-users.

A *personal information management service (PIM Service)* provides functionality to query and modify both the unified personal information model as well as the KWer instances thereof. Consuming applications and other services can invoke this service. The PIM service exposes this functionality through an API to allow other applications and services to access it. Nepomuk offers the PIMO Service [Sauermann&Klinkigt, 2009] to query, modify and populate the PIMO on the DS layer. In OntoPIM, the 'Query Processor (QP)' processes and answers "the queries posed by the user over the personal ontology" [Jones&Teevan, 2007, p. 121].

A *data alignment* module recommends mappings among entities on the DS layer or among DS layer entities and DI layer entities. This can happen automatically in case of an unambiguous mapping, i.e., recommendations with a confidence level of 100%, or by incorporating user feedback in case of confidence levels below a 100%. This module automates tedious metadata alignment tasks like for example mapping email sender representations on the DI layer to their corresponding person representations in the in the DS layer of the unified personal information. Further use cases consist of cleaning and streamlining metadata on the DS layer. Nepomuk uses the LocalDataAlignment framework [Sauermann et al., 2009a] to provide such services. OntoPIM has no comparable functionality.

A *data wrapper* crawls objects on the physical layer and creates for each found object a metadata representation in the DI layer. Nepomuk uses the Aperture Data Wrapper [Aduna B.V.&DFKI GmbH, 2005]. It crawls physical objects and creates a metadata representation in the DI layer. Aperture for example includes file system objects like files and folders and email client objects like emails, appointments, contacts and tasks. Aperture supports both Microsoft Outlook and Mozilla Thunderbird. In OntoPIM, the Semantic Save Manager (SSM) performs a so-called 'Semantic Save' on a physical object, i.e., "(1) invokes the operating system in order to save [the physical object] "o" in the file system, (2) creates the appropriate DI abstraction of "o", and (3) links it to the corresponding wrapper" [Jones&Teevan, 2007, p. 121].

A *unified personal information repository* stores unified personal information as well as the corresponding model in a central place. In Nepomuk, the Local Storage [Nepomuk Consortium, 2009e]

module is a semantic information repository storing RDF representations based on Sesame2 [Aduna B.V., 2009a]. The literature on OntoPIM doesn't specify such a repository.

A *service-oriented architecture* on the computer enables applications and other services to leverage the services of the PIM system. As well, this allows to make the architecture extensible, i.e., new services can enrich the existing PIM system functionality. The service-oriented architecture mainly consists of the service-registry, i.e., a catalogue for PIM system services. Using the service registry, potential consumers like applications and other services can discover and connect to the PIM system services. A messaging service enables the inter-service communication. Nepomuk uses an OSGI-based service registry [Nepomuk Consortium, 2009d] and a XMPP-protocol based messaging service [Nepomuk Consortium, 2009f]. OntoPIM has no comparable functionality.

3 Tasks, Personal Task Management Activities and Applications

In the following sections we introduce in detail the notion of a task and its representation, the KWer's involvement roles in a task, the KWer's personal task management activities, involved information as well as the supporting applications supporting the KWer's personal task management activities. Figure 20 sketches these elements in relation to each other.

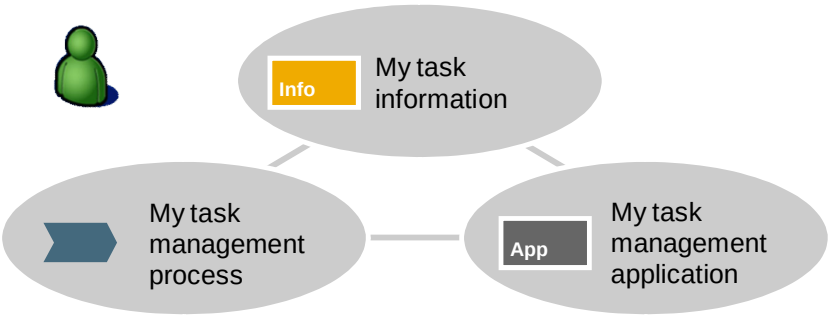


Figure 20: Overview: A KWer's task-related information, processes and applications.

3.1 Defining a Task

The concept of a *task* is in this context a KWer activity that the KWer needs to execute in order to accomplish a goal. In terms of Activity Theory (AT) [Engeström, 1999], the concept of an action represents the task concept used here. Figure 21 sketches the concepts used in AT and points out the relation of an action (here equivalent to the used task concept) to an activity. An activity can be described as a three layered hierarchy as depicted in Figure 21 [Leont'ev, 1978].

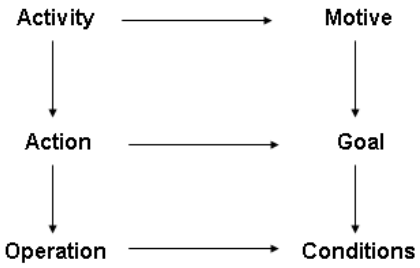


Figure 21: Hierarchical conception of an activity.

In AT every interaction of a subject with the world can be described as an activity if it is motivated by a particular need of the subject. The motive stimulates the subject to perform the respective activity, even if often in an unconscious way. The motive can be represented by an object the subject strives for. However, there are cases in which the motive and the concrete object of an activity differ. In this case, we distinguish activity and action where the former is related to the motive while the latter is related to the object [Leont'ev, 1981]. For example, the goal of a KWer's action could be to write a book while my motive could be to acquire reputation by the publication of this book. Finally, actions can consist of operations. These are routine processes of which the goal might not be aware of anymore. In the sphere of knowledge, the distinction between actions and operation corresponds to the difference between explicit and implicit knowledge as described by Polanyi [Polanyi, 1967].

For the KWer the *boundaries between two tasks are fluid* as tasks cut across each other. The importance and extent, i.e., required effort by the KWer, of a task change dynamically. For example, an originally small task grows while working on it into a bigger issue where the executing KWer needs

to plan in more detail. Vice versa, for example, a task with larger extent may turn out as not important anymore and thus doesn't need to be executed or may turn out as a simple task which can be executed in a few minutes.

A KWer manages *task representations* to remind herself of the task, better manage a task's execution or meet time constraints imposed by a task. A task may have an *explicit or implicit representation*. A task representation is a representation of a KWer activity. An *explicit task representation* exists for example when a KWer keeps a task representation as an entry of a to-do list. An explicit representation of a task can have different granularities. For example, for an activity lasting for one week to create a report, the task representation can consist of a to-do list item "create report" or can consist of several to-do list items making up the task like, e.g., "prepare report", "create report" and "finalize report". The granularity level of the modeling by the KWer depends on several factors, such as the intended use of the task representation, the task importance, the task urgency and intended quality of the task results. Besides explicit task representations there are *implicit task representations*. An implicit representation of a task is an object or the representation of an object which reminds the KWer's of a task in her subjective view. For example, when the KWer sees a document on the desktop, she knows that she still needs to work on that document until the end of the week.

In the context of executing a task, a KWer deals with different information and people. From a KWer's point-of-view, a task is the integrating element for both information and people in this context, see Figure 22 for an illustration.

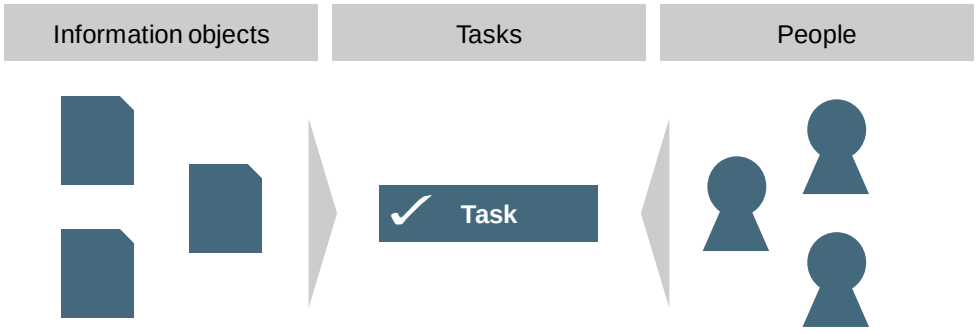


Figure 22: Task as integrating element for information and people.

A KWer involved in a task can have different *roles* with regard to a task. A KWer can be *involved* in a task. When distinguishing a KWer's role for a task by the responsibility type, the so-called RACI diagram enables to derive the four roles 'Responsible', 'Accountable', 'Consulted' and 'Informed' [Wikimedia Foundation Inc., 2009b]. A KWer *responsible* for a task is the one who actually works on the task to achieve the task's goals. This role is as well referred to as *task owner*. Alternatively the KWer can supply resources or support others to achieve the goal. A KWer *accountable* for a task is the one who is "ultimately accountable to the correct and thorough completion of the task" [Wikimedia Foundation Inc., 2009b]. Thereby one person often can task both roles of being accountable and of being responsible. A KWer *consulted* for a task is the one "whose opinions are sought" [Wikimedia Foundation Inc., 2009b] and two-way communication is possible. A KWer *informed* for a task is the one who is "kept up-to-date on progress" [Wikimedia Foundation Inc., 2009b] and only one-way communication takes place.

3.2 Task-Related Activities of a KWer in the Task Management Activity

The KWer's *personal task management process* helps the KWer to plan, structure and prioritize the personal set of tasks.

The personal task management process is closely related to other personal supporting activities. Figure 23 shows the KWer's personal task management activity in the context of related personal activities.

Figure 23: Personal task management activity at the intersection of related personal activities.

The *personal time management* activity helps the KWer to plan and control the available personal time. Managing the personal tasks is always tightly related to the management of the personally available time as the managed resource is the in both case the KWer herself. The *personal goal / priority management* activity helps the KWer to define the personal goals and priorities and optionally align it with organizational goals relevant to the KWer. Depending on a KWer's goals, independent whether these are of organizational or personal nature, the priorities for personal tasks need to be set. The *personal information management* activity helps the KWer to collect, organize, browse and retrieve personal information. Personal information is involved in the KWer's tasks and the KWer organizes this information to have it available for use in the respective tasks. The *collaborative task management* activity helps the KWer to work together with other people on a task. The part of a collaborative task that is directly KWer relevant for the KWer, and with which the KWer interacts, has the status of a personal task for the KWer.

The KWer's *personal task management* activity consists of several sub-activities. Figure 24 gives an overview on the next, more fine-granular, level of personal task management activities. These activities don't have a linear dependency, as the KWer can execute them as needed in various ordering.

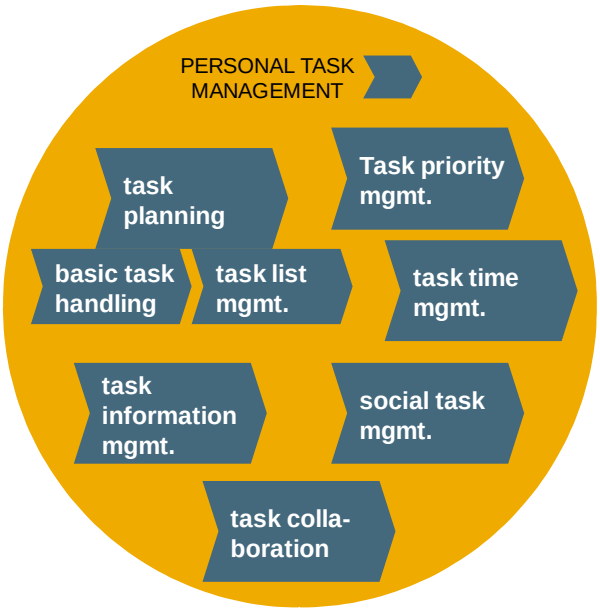


Figure 24: Personal task management activities.

The *task planning* activity helps the KWer to structure the workload and perform work decomposition, i.e., breaking down and categorizing tasks. As part of the task planning activity, a KWer deals in the *basic task handling* activity with organizing her tasks, e.g., by creating and populating a task representations. Furthermore, the KWer deals in *task list management* activity with handling a personal to-do list where the KWer manages a list of personal tasks that are due for execution or are already executed.

The *task priority management* activity helps the KWer to maintain the priorities coming with a task. The *task time management* activity focuses on the time needed to execute a task and the KWer can assign a task's due date to manage the time-related aspects of work.

The *task information management* activity helps the KWer to handle information needed for executing a task. This includes categorizing tasks to re-find them and organizing needed information objects such as papers or magazines. The *social task management* activity helps the KWer to organize and track the persons involved in a task.

The *task collaboration* activity focuses on collaboration in the task domain. This means, that KWers can delegate tasks to each other, can perform and task tracking and conduct information sharing.

3.3 Task Management Experience and Skill of KWers

KWers conducting personal task management can be categorized into two dimensions regarding their task management-related experiences and tool usage skills as well as the importance of their personal task management activities for them, see Figure 25.

On the one hand, *task management activity experience* denotes the level of experience with the process of managing personal tasks, e.g., the knowledge of methodologies like, e.g., Getting Things Done [Allen, 2003]. This includes the *tool usage skill* which denotes the KWer's degree of familiarity with task management-related tools. Overall the KWer's individual level ranges from novices who are new to the subject to power users who have extensive experience in conducting personal task management activities and in using task management-related tools.

On the other hand, the *personal task management activity importance* for the KWer's job role is an important factor. This ranges from an overall low importance of personal task management activities for the KWer's job role over to overall high importance for the KWer's job role, e.g., due to significant task overload. The importance implicitly influences the frequencies of conducting task management activities. A low importance implies infrequent task management activities whereas a high importance implies frequent task management activities.

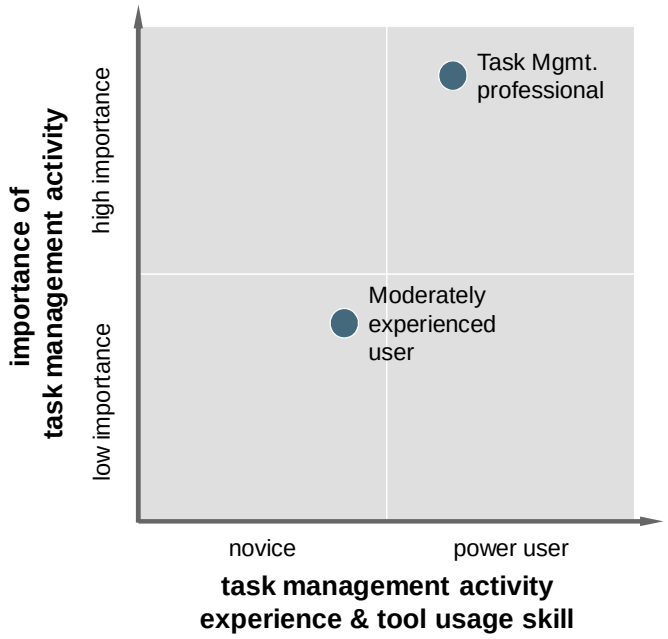


Figure 25: KWer profile with personal task management activity importance and experience.

Of the many possible presented experience and importance profiles that a KWer can incorporate, two KWer profiles are common, i.e., task management professionals and moderately experienced users.

Task management professionals are a group of people with high interest in professional self management for which they see personal task management as key success factor. This reflects the high importance of their personal task management activities for their job roles. In addition, their interest in personal task management is usually independent of their job roles and driven by their wish and need to optimize their work life. They have good experience and methodological skill in managing their tasks and the surrounding processes. They are power users in their task management-related tools and often prefer to use the keyboard if possible to increase their operating speed.

Moderately experienced users are a group of people, who occasionally manage their tasks. They have less interest, less experience and less methodological skill in managing their tasks compared to task management professionals. This is because they can fulfill their job as well without an elaborated task management, i.e., task management relatively has a lower importance for them. They are moderately experienced with regard to task management-related tool use.

3.4 State-of-the-Art in Personal Task Management Support

A number of different types of applications represent the state-of-the-art in supporting a KWer's personal task management activities. A KWer uses a number of applications throughout the planning and execution of a task. Figure 26 gives an overview on the existing application types.

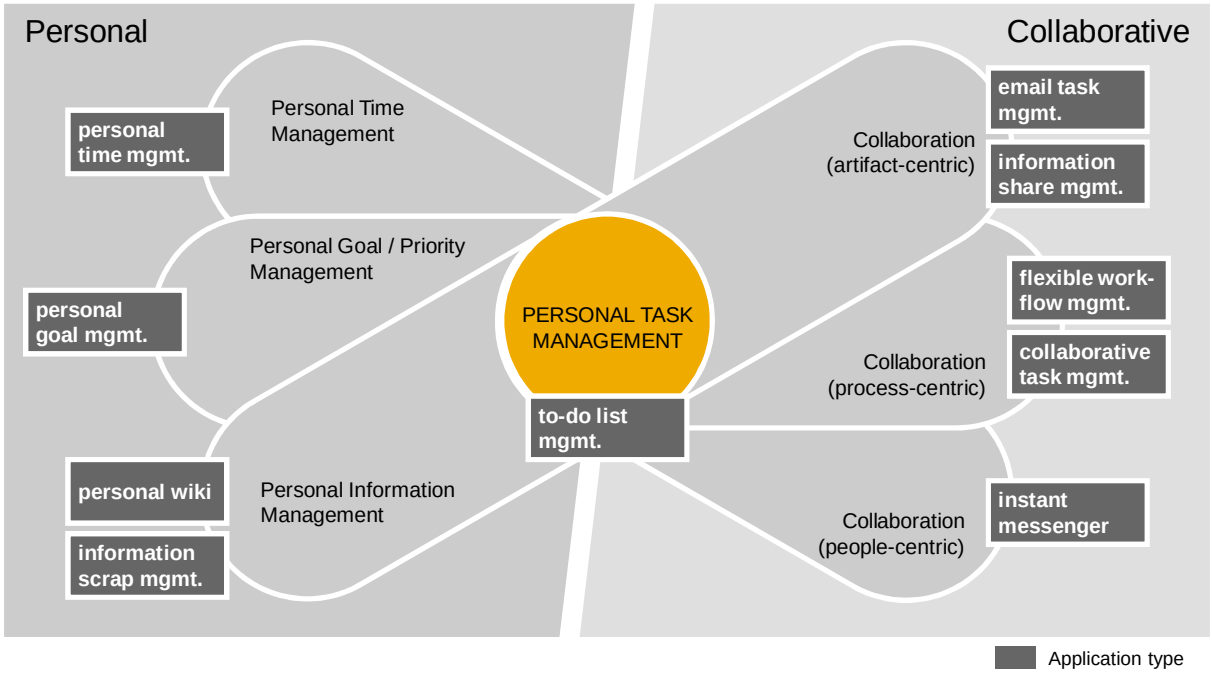


Figure 26: Application types for personal task management support.

The here presented applications and types tackle the KWer's personal task management activities from an individual point-of-view, i.e., these applications are designed to support an individual rather than an organization.

To-do list management applications manage lists of a KWer's tasks and at their core support the KWer's personal task management activity, for examples see section 3.4. There are client-based and web-based applications.

Personal time management applications enable the KWer to track the time spent on tasks and projects, for example see Personal Task Manager [Personal Task Manager, 2009]. Thereby they keep a record of tasks and projects the KWer worked on or plans to work.

Personal goal management applications help the KWer to manage the personal and personally relevant organizational goals, see for example see ThinkingRock [Avente Pty Ltd, 2009]. They allow the KWer to model the goals and structurally follow them up by aligning and executing tasks to each goal.

Personal wiki applications enable the structured collection of personally relevant information, for example see MonkeyGTD [Baird, 2009]. Wiki pages can represent tasks and using special categories for these pages the KWer can manage task categories in a structured way for as such unstructured content.

Information scrap management applications allow the KWer to collect and organize small information snippets called information scraps, for example see Microsoft OneNote [Microsoft Corporation, 2009b]. These information snippets can contain or represent task information for the KWer.

Instant messenger applications help the KWer to exchange text messages in real-time, for example see Skype [Skype Limited, 2009]. These communication facilitators allow the KWer to communicate during a task execution and to informally delegate tasks to peers.

Collaborative task management applications provide a number of people access to a common set of tasks, see for example Mindquarry [Mindquarry GmbH, 2009]. Often these appear in the form of shared tasks lists along with synchronization and collaboration functionalities like instant messaging or telephony integration.

Flexible work-flow management applications enable the KWer to create workflows ad-hoc and with this control the execution of particular activities among humans and computers, see for example FRODO [Abecker et al., 2001]. Through a so-called work list, these applications deliver a KWer's tasks and the KWer can manage and input tasks into these applications.

Information share management applications provide particular KWer information to collaborating peers, for example see Microsoft SharePoint [Microsoft Corporation, 2009c]. In the context of the KWer's personal task management activity, a KWer can share task-relevant information for each task with the dedicated target users.

Email applications allow the KWer to manage personal tasks by e.g. sending themselves reminder emails for particular tasks, for example see Microsoft Outlook [Microsoft Corporation, 2009a] and section 3.4.3. Numerous attempts have been undertaken to optimize the task management functionality in email applications, see section 3.4.3 for details.

In the following we exemplarily detail a number of application types and explain their support for a KWer's personal task management activities.

3.4.1 Groupware Approaches

Microsoft Outlook [Microsoft Corporation, 2009a] is a groupware application in heavy use. It features personal task management functionality. It is thought as a tool to work on personal tasks as integrative part of calendar and email functionality. Here we give a short summary of the main features that this tool provides with respect to personal task management. These features include:

- Creation and deletion of *individual tasks* providing space for a description of the task.
- *Status administration* with five predefined states that can be set by the user.
- *Priority* definition with the three predefined values *high*, *normal* and *low*.
- *Completion* value described as percentage.
- *Due date* and *start date* can be defined.
- Manual setting of a *reminder* for a specific task.
- Definition of a *Task Owner*.
- *Schedule time in the calendar* for a task. This can be done by dragging the task item into the calendar. A time slot with the task data is automatically generated.
- *Assignment* of the task to another user. A copy of the task can be kept in the requester's task management. Moreover, you can request a Status report from the assigned user.
- Task assignee can *accept or reject tasks*.
- *Forwarding of tasks* is possible.
- The *attachment of related documents* is possible.
- A *task list* is provided in which the order can be rearranged.
- It is possible to assign a task to a specific *category*.
- *Connection to email* is realized in the way that the user can drag mail items to the task folder which leads to the generation of a task item. Task requests are also sent by email.

3.4.2 Web-Based To-Do List Approaches

There exist several web-based applications that fulfill basic task management functionalities. Web-based applications often leverage dynamic elements of the Web 2.0 [O'Reilly, 2005].

A *basic task management application* consists of creating tasks, assigning tasks to projects and prioritizing them. Additionally, these applications enable the tracking of the task status. These applications provide basic task management functionality without extensive integration with email or other applications, i.e. they represent rather a stand-alone task management application.

For example, *TaskFreak!* [Ozier, 2009] is a simple web-based application representing basic task management functionality which implements the 'Getting things done' methodology of David Allen [Allen, 2003]. Task are shown in a to-do list and may be categorized and assigned to projects. However, it provides no extensive integration with email or other applications.

A more elaborated approach is *Voo2do* [Rura, 2009a] representing a web-based support application for collaborative work using simple *web-based to-do lists*. It maintains a central view on all existing tasks, a so-called dashboard, and keeps track of due dates. The tasks may be organized into projects and contexts. Tasks are assigned to a project. A context represents a "group of projects" that can be viewed together [Rura, 2009b], i.e., a context represents a filter on existing projects like e.g. home, work, or any other context. A context can be extended to a collaborative context. For projects in such a context, the person assigned to a task is shown. The current implementation only supports the tacking of this person, while future versions should incorporate task sharing between several users [Rura, 2009b]. Beyond the personal task management functionality, Voo2do features a *shared task list*, so-called 'Public Task Lists', allowing for the creation and hosting of a public web page that displays selected tasks of a user dynamically. It offers read-only access and can be restricted by a password [Rura, 2009b]. A further feature is a report on deadlines which allows users to monitor tasks with upcoming deadlines.

An *extended task management application* features additional functionality, e.g., regarding the integration with its environment, such as the integration of emails, is provided for example by web-based groupware solutions. In the following, several examples are presented by pointing out some special functionality.

WebEx WebOffice [Cisco Systems Inc., 2009] is a web-based office suite integrating the task management functionality. This means, that tasks can be assigned to other WebOffice users as well as to external users that have access to these tasks after having registered to the user's WebOffice account.

Spongecell [Spongecell L.L.C., 2009] is a web-based calendar application that allows sharing of calendar entries as well as import and export functionalities. It is designed for simplicity and as such it supports the creation of a calendar event included in an email by forwarding an email to a central email address. Spongecell then searches through this email in order to extract possible calendar entries. Additionally, Spongecell offers a side-bar, so-called Spongebar, creating calendar entries by entering an event in a free text form. E.g., "Soccer practice every other Tues at 6" would create a corresponding calendar entry.

Central Desktop [Central Desktop Inc., 2009b] is a further web-based groupware-style application supporting team collaboration, task and project management. Central Desktop connects the

milestones from the project management with task lists of task management as illustrated in Figure 27. This enables the consolidation of tasks and project milestones [Central Desktop Inc., 2009c].

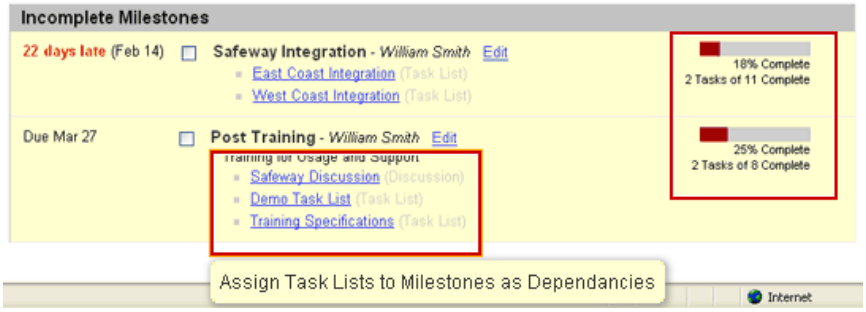


Figure 27: Task lists assigned to milestones [Central Desktop Inc., 2009a]

3.4.3 Email Task Management

One of the main tools to organize knowledge work is emailing. Although this is mainly designed as a pure communication tool, it often substitutes task management tools due to the lack of alternatives. However, this means that the tool is used for a purpose for which it is not designed. Most typical task management functionalities are missing but its flexibility makes people overlook this lack. Based on the widespread use of email for task management there is a long-lasting discussion going on how to improve email to fulfill task management needs.

Gwizdka [Gwizdka, 2002] introduced the email interface *TaskView* that was based on TimeStore developed by Baecker and his colleagues [Yiu et al., 1997]. TimeStore mainly describes a two-dimensional representation of email items in terms of time and sender, allowing for a simpler overview of the arrived email items. In TaskView this schema was enhanced by introducing an icon representation for items, additional information provision per item, temporal forward and backward navigation. The actual message text is shown by double clicking on the task icon. On the basis of TaskView several empirical studies were performed that led to the following observations:

- A graphical overview as provided by TaskView only partially improves the efficiency in handling emails.
- TaskView only provides considerable benefit for people with high visual memory.

Bellotti and colleagues [Bellotti et al., 2002] developed Task Master as an approach to deal with the insufficiencies of email as task management tool. They started to develop TaskMaster as an email application designed for task and project management. They observed that interdependent tasks relied on threads of messages, files, and links, which they treated as collections called *thrask*s. The design is based on the identification of 7 major pain points:

- Tracking concurrent actions
- Prioritization of actions received by email
- Managing activity over time
- Supporting deadlines and reminders
- Bringing together related items
- Switching between applications
- Providing an overview as most important point

TaskMaster tries to tackle these problems offering the following features: a warning bar for deadlines; action clusters; task-specific contact lists. Bellotti concluded from the evaluation of

TaskMaster that users like the idea of using email for task management but that such an application must also provide enough functionality to support task related activities.

A further study on the central role of email for task management based on TaskMaster was performed by Bellotti et al. in 2003 [Bellotti et al., 2003]. In this investigation they found an additional categorization of emails:

- emails related to tasks that require only a simple and rapid response taking only a few seconds;
- more complex rapid- or extended-response tasks that are interdependent, partially with obligations that depend on tasks of others. They are often related to email threads.

Based on the results they suggested embedding task management tools directly into the inbox. In this respect email tools should be rethought, not in terms of messaging but in terms of activities to be performed.

In 2004 Bellotti et al. [Bellotti et al., 2004] suggested a *Task List Manager* (TLM) application to support personal task management, based on the results of their previous studies. The application was designed to help users to organize their work items enabling the following activities:

- Capturing the personal daily tasks
- Planning and execution of simple tasks
- Prioritization, management, and reasoning about tasks
- Assistance to improve, user observation, and reflection on activities
- Recording annotations, ideas, and action items
- Support for planning and problem solving
- From case studies related to task management they concluded that most of the task management was connected to emailing and electronic calendars. In their studies they could identify the following strategies to handle tasks:
- *Task Vistas* – these are simple lists of to-dos which allow the user to get an overview of all relevant tasks
- *Informal Priority Lists* – these are to-dos lists that focus on those tasks that have to be accomplished in the near future concentrating on the most urgent actions first
- *State Tracking Resources* – this describes the state tracking of many similar and simultaneously executed tasks
- *Time Management* – this becomes important when time becomes the limiting factor
- *Value Execution* – In this strategy a task serves different goal at the same time in order to optimize the outcome of a task.

From the observed strategies they derived various aspects that should be supported by TLM:

- Viewing of entire Task Vistas from different perspectives
- Highlighting of the most important work items
- Capturing of task histories and states
- Management of time constraints to support workload distribution
- Registration of temporal and procedural work regularities
- Freedom for users to skip tasks
- Offering more benefits using the TLM than there are costs for the additional administrative activities
- Explication of value extension opportunities
- Description and modeling of social relations

- Capturing of notes and task lists away from the desk.

In 2005 Bellotti and Ducheneaut [Bellotti et al., 2005] continued their work with an overview article on the insights in email-centric task management. In this review they formulated six key challenges for email-based task management:

- Keeping track of various concurrent actions
- Marking the right items as important and outstanding
- Managing work items that are extended over time or keep track of threads of activity
- Management of deadlines and reminders
- Collocation of related items and event-based collation of documents and discussions
- Providing a task oriented overview for extended work activities, e.g., entire projects

In this context they described the challenges of knowledge work as the capability to accomplish various simultaneous goals by using email: (1) to keep themselves up to date; (2) to respond to requests in time; (3) to track requests to others; (4) to make deadlines; (5) to process incoming information; (6) to distribute information, and (7) to be prepared for upcoming events. Knowledge work is therefore characterized by *multi-tasking* several activities mainly via email. They observed a considerable overload of work resulting from the speed with which work items could be processed electronically and from the increased connection opportunities between different persons [Sproull&Kiesler, 1991]. The overload is also increased by the fact that it is not the mere quantity of emails but in particular the quality of emails that aggravates the problem. This includes complex tasks that require the coordination with others. Their conclusion was that only the incorporation of task management functionality in the inbox can lead to a significant improvement regarding the overload problem.

3.4.4 Machine Learning Approaches

CALO Stardust [Carnegie Mellon University – Human-Computer Interaction Institute, 2007a] is a prototype targeting the KWer support for personal task and information management activities. Human computer interaction design researchers designed it to demonstrate the capabilities of the machine learning techniques developed in the CALO project [SRI International, 2008] for personal information management. CALO Stardust features a sidebar showing a KWer's tasks and related information. Its features are task prioritization, event and incoming email notification, meeting coordination among multiple individuals and automation of repetitive tasks [Carnegie Mellon University – Human-Computer Interaction Institute, 2007a]. Unlike the previously mentioned prototypes, CALO Stardust learns the associations between the KWer's "tasks, files, contacts and other relevant information automatically" [Carnegie Mellon University – Human-Computer Interaction Institute, 2007b] as the KWer works on the computer. For example, CALO Stardust presents based on observed user behavior "related resources such as websites or files" [Carnegie Mellon University – Human-Computer Interaction Institute, 2007b] when a KWer works on a particular task. The KWer doesn't need to manually create and model the associations between tasks and related information, but can give feedback to suggestions made by the CALO Stardust system.

4 Meetings, Personal Meeting Management Activities and Applications

In the following sections we introduce in detail the notion of a meeting, the KWer's involvement roles in a meeting, the KWer's personal meeting management activities, involved information as well as the supporting applications supporting the KWer's personal meeting management activities. Figure 28 sketches these elements in relation to each other.

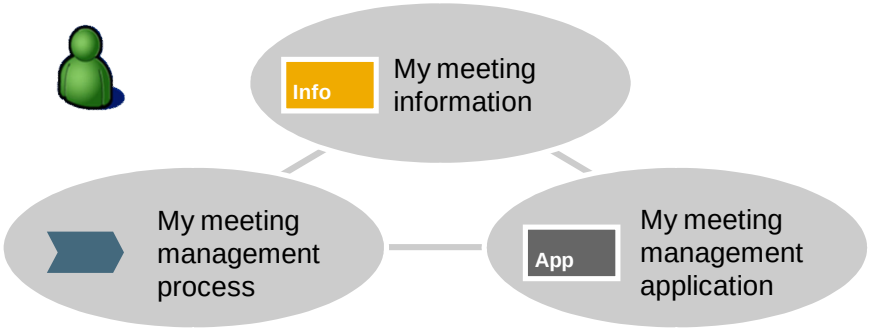


Figure 28: Overview: A KWer's meeting-related information, processes and applications.

4.1 Defining a Meeting

Meetings[6] are "essential to structure and coordinate work in organizations" [Antunes&Costa, 2003]. A meeting is defined as "a number of people assembled together, usually at a pre-stated date and time, to discuss a topic for the purpose of presenting information, swaying opinion, formulating a decision, practicing a skill, and/or developing a plan of action. Those at the meeting may belong to the same group, to different groups, or perhaps not to any group at all. A meeting might be called by an individual or by an organization. Usually the people meeting convene together physically within a designated area. Sometimes, however, meetings are held by people thousands of miles apart via telephone conference calls or video conferencing." [Kaliski, 1999]

While the given definition lists different activities involved in a meeting's context, it is also possible to list different meeting genres. Antunes and Costa mention that "planning meetings, project meetings, briefings, brainstorms, welcome meetings and workshops are just few examples of meeting genres common in organizations" [Antunes&Costa, 2003].

There are different *stakeholders to a meeting*. A KWer involved in a meeting can have different *roles* with regard to a meeting.

- A *meeting participant* takes part in a meeting and is interested in the meeting information and outcome.
- An *informed person* is not required to attend the meeting in person but is interested in the information distributed in the meeting and its results.
- A *meeting organizer* is responsible for setting up the meeting and assuring the meeting can be conducted.
- A *minute taker* writes up a protocol during and after the meeting.

[6] The here explained meeting definition will be used throughout this document.

- A *moderator* guides the meeting participants through the meeting and ensures that the meeting follows its agenda.

A *meeting protocol*[7], also known as *meeting minutes,* is "the instant written record of a meeting or hearing" [Wikimedia Foundation Inc., 2009c]. The "primary function of minutes is to record the decisions made" [Wikimedia Foundation Inc., 2009c]. However, besides recording the meeting results as well the proceeding of the meeting can be interesting for stakeholders in the meeting. A meeting protocol captures the things happening in a meeting where a stakeholder is interested in. Subsequently meeting minutes typically contain information of the meeting agenda, actual discussed topics, discussions, decisions, defined actions and responsible and involved persons. Meeting protocols "often give an overview of the structure of the meeting, starting with a list of those present, a statement of the various issues before the participants, and each of their responses thereto" [Wikimedia Foundation Inc., 2009c]. The actual content of the meeting minutes depends on the purpose of the meeting protocol. There are several purposes for taking meeting minutes. Meeting minutes can be recorded for informative purposes, e.g., to record the status quo as achieved in the meeting. Furthermore, meeting minutes can be recorded because legal requirements, e.g., to record the exact proceeding, decisions and comments of government or executive meetings.

4.2 Meeting Classification

Meetings can take various forms, and a number of classification schemes have been proposed to capture the different conditions and situations. In general, meetings can be classified by the following criteria, which are used to categorize possible meeting situations, see Table 6.

Criteria	Possible Values
place	same, different
size of group	small, large
structuredness	unstructured, structured
areas of group work	planned, spontaneous
duration	brief, long

Table 6: Meeting Classification Criteria.

According to Streitz et al. [Streitz et al., 1994], the most obvious and prominent dimensions of cooperation of people are time and place. Because the word "meeting" implies "to meet", it is not possible to use the time dimension because meeting cooperation always takes place at the same time. There can be no single meeting taking place at different times (disregarding different involved time zones).

The second dimension mentioned by Streitz et al. [Streitz et al., 1994], p*lace*, can have the discrete values of *same* and *different*. It is also possible to use ordinal values that are higher, the more different the locations are. Meetings with four people in one room and a fifth person participating via telephone call are less different than five people from five different countries participating in a web conference. Ellis et al. [Ellis et al., 1989] also distinguish between meetings that take place at one location ("face-to-face" meetings) and "distributed" meetings where the participants' locations differ.

Another dimension characterizing meetings mentioned by Streitz et al. [Streitz et al., 1994] is the *size of group*. While the smallest possible meeting involves only two persons (*small*), there can be workshops with 500 and more participants (*large*). According to Panko and Kinney [Panko&Kinney,

[7] We use the terms protocol, minutes, meeting protocol and meeting minutes interchangeably.

1995], most meetings are dyadic and triad. More than three quarters of all meetings have a size of group of two or three persons.

Li et al. [Li et al., 1999] divide meetings by their *structuredness* into *unstructured* and *structured* meetings. Like the place dimension, it is also possible to have ordinal values for different levels of structuredness. While completely spontaneous meetings can be considered to be unstructured and completely pre-planned meetings to be structured, there can be meetings that only have a brief agenda with some degrees of freedom. For example, meetings that consist mostly of brainstorming topics on a pre-planned agenda are between unstructured and fully structured meetings. The lower the structuredness, the higher the degree of freedom to go into unplanned topics and details.

Weber et al. [Weber et al., 1997] distinguish between two different *areas of group work*. *Planned* and *spontaneous* meetings need different types of supporting systems. Planned meetings can be modeled a priori because they often proceed in a predefined way and have a recurring appearance. Spontaneous meetings cannot be modeled before they take place because of the lack of static patterns. This means, that there can be pre-meeting phases before the actual meeting takes place, but they are optional because meetings can also develop ad-hoc.

Panko and Kinney [Panko&Kinney, 1995] distinguish by *duration* between *brief* and *long* meetings. Almost three quarters of all face-to-face meetings last 30 minutes or shorter while only three percent of them take more than two hours.

4.3 Meeting-Related Activities of a KWer in the Meeting Process

A KWer's *personal meeting process* describes the KWer's activities to conduct and manage a meeting. This section describes the identified meeting process phases and the therein executed meeting-related activities. The identified activities form together the meeting process.

The meeting process consists of several phases. There is a pre-, in- and post-meeting phase based on the meeting itself as time perspective fixture, see Figure 29.

Figure 29: Meeting process phases.

Meeting-related activities are activities that a KWer conducts to coordinate or manage a meeting, see Figure 30 for an overview. These meeting-related activities are explained in detail below. In contrast to this, work-related activities are activities a KWer conducts independent of meetings and comprise, e.g., conducting brainstorming or giving a presentation.

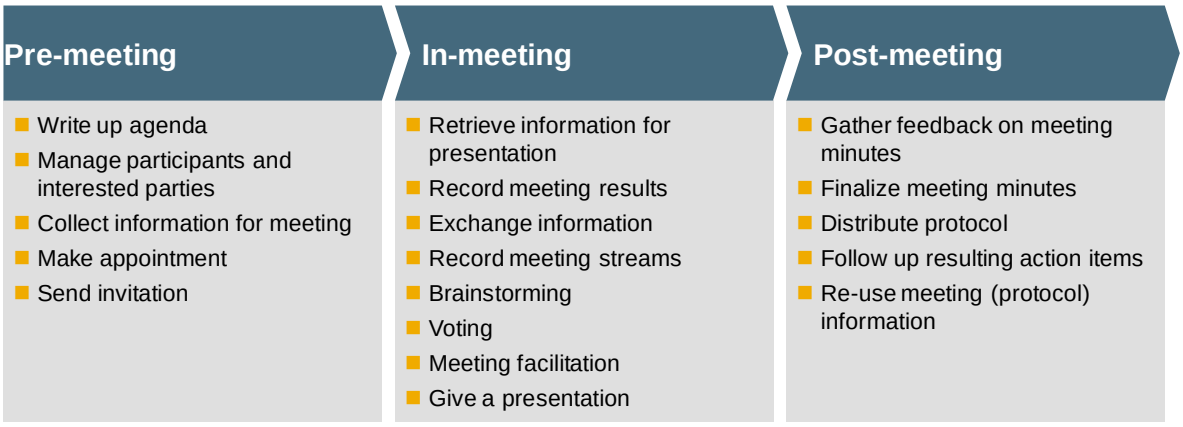


Figure 30: Meeting process activities by meeting phase.

These activities represent the KWer's individual perspective on meetings, i.e., not the global perspective on e.g. an organization. Meeting-related activities are performed by an individual KWer in an individual, meeting mode [Fuchs et al., 2001], i.e., the KWer either works on her own, optionally with computer support, or is located in a meeting setting, i.e., presents to people in a meeting, or interacts with people.

Meeting activities occur in all phases of the meeting process, i.e., the pre-, in- and post-meeting phase. Each meeting-related activity can be assigned to at least one of the meeting phases. In most cases there is a one-on-one relationship. However, some activities can be executed in two meeting phases, for example gathering feedback on the meeting minutes can happen in both the in-meeting as well as the post-meeting phase. In such a case, the activity is described here with the most probable phase.

The KWer can execute the described meeting-related activities to profit from the experience of managing a meeting that is codified in the meeting process. All activities are optional, i.e., the KWer can select depending on the situation which steps of the meeting process are appropriate to execute. The sequence, in which the activities are presented here, corresponds to the ideal sequence of execution, but the KWer can re-organize the sequence to match the particular meeting situation.

The meeting process described below is the result from findings gathered by an extensive literature review[8], combined with an analysis of state-of-the-art meeting support software applications, see section 4.6. The here presented meeting process can be considered as a reference process for preparing, conducting and post-processing meetings from a KWer's personal point-of-view. It represents a so-called best practice for meeting management.

Table 7 gives an overview on the activities the meeting process consists of and presents the therein involved information items and supporting applications.

Phase	Activities	Involved information items	Applications (e.g.)
Pre-meeting	Write up agenda	Agenda - Topic items	Word processor
	Manage participants and interested parties	People (Participants, interested parties)	Calendar application, e.g., Microsoft Outlook [Microsoft Corporation, 2009a]

[8] See section 4.1, 4.2 as well as [Ackermann, 2008] for an overview on the results, i.e., the there presented meeting characteristics and meeting process.

	Collect information for meeting	Information objects (e.g., files, websites, emails), Tasks from previous meetings (e.g., unfinished tasks)	File Manager, web browser, email client, word processor
	Make appointment	Appointment People (Participants, interested parties)	Calendar application, e.g., Microsoft Outlook [Microsoft Corporation, 2009a]
	Send invitation	People (Participants, interested parties) Message, e.g., email messaging	Email client
In-meeting	Retrieve information for presentation	Meeting Information	File Manager
	Record meeting results	Protocol - Action items - Meeting information	Word processor
	Exchange information	Screen sharing Messages	Meeting support, Instant messenger
	Record meeting streams	Audio and/or video streams	Meeting Audio / Video Capture
	Brainstorming	Information objects (e.g., files, websites, emails),	Mindmapping
	Voting	Clusters, rankings and voting results	Workshop support
	Meeting facilitation	Agenda - Topic items Protocol - Action items - Meeting information	Workshop support
	Give a presentation	Information objects (e.g., files, websites, emails),	Presentation manager
Post-meeting	Gather feedback on meeting minutes	Protocol, participants and interested parties	Word processor, Email client
	Finalize meeting minutes	Protocol	Word processor
	Distribute protocol	Protocol, participants, interested parties and messaging	Email client
	Follow-up resulting action items	Protocol - Action items (tasks)	Task Manager
	Re-use meeting (protocol) information	Protocol - Action items - Meeting information	File Manager

Table 7: Overview on meeting-related activities, involved information and applications.

4.3.1 Pre-Meeting Phase (Preparation)

All activities in the pre-meeting phase target the preparation of the actual meeting.

In the '*Write up agenda*' activity the KWer in the role of a meeting planner plans the meeting and thereby collects and arranges the topics, which need to be discussed or presented during the meeting. The KWer defines the meeting agenda based on these identified topics which represent the meeting's structure.

In the '*Manage participants and interested parties*' activity the KWer in the role of a meeting planner defines the meeting audience, i.e., names the meeting participants and interested people. Interested people are people who do not attend the meeting but still get informed about the results. Usually the meeting planner sets up a list with people related to the meeting which includes the meeting participants and interested people.

In the '*Collect information for meeting*' activity the KWer gathers information needed to prepare and conduct the meeting. For example, this can be textual information or with computer support this can be information objects, like, e.g., websites, files and emails.

In the '*Make appointment*' activity the KWer coordinates the meeting appointment, i.e., the meeting's location and time, to ensure that the meeting participants and optionally interested persons are available when the meeting takes place. Using a calendar application an appointment shows up in the KWers' calendar and contains information on the meeting location as well as the start and end time. For example, a standardized iCalendar [Dawson&Stenerson, 1998] invitation file can be used to create an appointment in a calendar application like Microsoft Outlook [Microsoft Corporation, 2009a].

In the '*Send invitation*' activity the KWer invites the participants to the upcoming meeting and notifies interested persons by sending an invitation message. The KWer usually sends an invitation after the preparation of the meeting, i.e., when the other in activities in the pre-meeting phase are executed. With computer support, the KWer, e.g., sends the invitation via email. This email contains, besides information on the place and time of the meeting, the agenda, details on the meeting and all relevant task information as attachment.

4.3.2 In-Meeting Phase

In the in-meeting phase the people get together and the actual meeting takes place. It consists of a number of activities:

In the '*Retrieve information for presentation*' activity the KWer accesses meeting information, i.e., opens information related to the meeting to get needed insights during the meeting. As well, the KWer can review the planned meeting and complete the information needed for the meeting.

In the '*Record meeting results*' activity the KWer takes minutes to capture the meeting's results as well as to document the exchanged information. The protocol is written by the minute taker and group decisions are made. The KWer records meeting results in the form of a textual protocol. In particular meeting results represent beneath the exchanged information in the meeting along the meeting's topics as well as, e.g., the taken decisions by the meeting participants and defined actions items. For example action items have usually a responsible person associated and optionally a due date.

In the '*Exchange information*' activity the KWer exchanges information during the meeting with the meeting's participants. This happens in written or spoken form. In distributed meetings this includes computer support for sharing computer screens using screen sharing applications, transmitting real-time audio and video streams of the meeting in a meeting support application and the exchange of messages, for example with an instant messaging application.

In the '*Record meeting streams*' activity the KWer can trigger the capturing of developments and meeting progress in an audio and video stream throughout the whole time the meeting takes place.

This enables all participants to better recall the meeting situation later on. This can take place next to a textual protocol of the meeting's results in the meeting protocol.

In the '*Brainstorming*' activity the KWer can participate in a brainstorming session. In meetings the participants can conduct a brainstorming session to, e.g., generate ideas, discover new trends and generate solution alternatives for a particular problem. Supporting applications try to stimulate creativity by, e.g., offering so-called mind-maps to take notes in a non-linear order.

In the '*Voting*' activity the KWer can set up and conduct a voting to collect the participant's opinions on a particular topic. A voting is especially useful in large groups of, e.g., 20 or more participants as in such situations the meeting time is not sufficient to enable each participant to utter her opinion in a speech or the like. When voting is supported by computer applications, the voting applications feature graphically visualize clusters, enable rankings or present voting results.

In the '*Meeting facilitation*' activity the KWer can act as moderator to guide the participants through the meeting and create a constructive meeting atmosphere.

In the '*Give a presentation (Work-related)*' activity the KWer presents information to the meeting participants. Thereby the KWer can use computer support such as slides in a presentation application.

4.3.3 Post-Meeting Phase (Post-Processing)

In the post-meeting phase the KWer post-processes the meeting and its results. It consists of a number of activities:

In the '*Gather feedback on meeting minutes*' activity the KWer in the role of the minute taker asks meeting participants to review the meeting minutes taken by her. All participants and interested parties are invited to collaboratively gather feedback on the meeting minutes and thus verify the protocol. This includes sending out the meeting protocol to the participants, requesting feedback on the meeting protocol and collecting the feedback.

In the '*Finalize meeting minutes*' activity the KWer in the role of a minute taker finalizes the meeting protocol. Thereby the minute taker optionally incorporates received comments and feedback on the protocol. As well, the KWer can create a summary and action item overview for better readability. Finalizing a protocol includes producing and publishing a readily readable version of it, e.g., with computer support this means creating a layouted PDF file, website or document. After finalizing the protocol no further changes to the protocol are allowed to ensure compliance. When a protocol has been finalized, the KWer archives it and indexes it so that it is retrievable at a later stage.

In the '*Distribute protocol*' activity the KWer can distribute a final version of the meeting protocol among the meeting participants and interested people. The finalized protocol is distributed by either the minute taker or the meeting planner.

In the '*Follow-up resulting action items*' activity the KWer can review and execute actions which have been defined in the meeting the KWer participated. Actions are the most important artifacts of meetings and are recorded in the meeting protocol. The KWer, e.g., needs to track unfinished actions so that they do not get lost before completion and the KWer is aware of their status. The KWer needs to remind other people to work on and complete actions for which they are responsible. With computer support, a repository can store these actions and the KWer can retrieve, filter and filter the action's representations.

In the '*Re-use meeting (protocol) information*' activity the KWer can use the information gathered in the meeting and the information written down in the meeting protocol in other work activities. With computer support, applications can make use of information contained in the meeting protocol. These applications can parse single protocol items and process them based on their type. For example, a project management system collects all items that are of the type "Milestone".

4.4 Meeting Management Experience and Skill of KWers

KWers conducting personal meeting management can be categorized into two dimensions regarding their meeting and meeting management-related experiences and tool usage skills as well as the importance of their personal meeting management activities for them, see Figure 31.

On the one hand, *meeting management activity experience* denotes the level of experience with the process of managing personal meetings, i.e., with the process of preparing, conducting and post-processing meetings. This includes the *tool usage skill* which denotes the KWer's degree of familiarity with meeting management-related tools. Overall the KWer's individual level ranges from novices who are new to the subject to power users who have extensive experience in conducting personal meeting management activities and in using meeting management-related tools.

On the other hand, the *personal meeting management activity importance* for the KWer's job role is an important factor. This ranges from an overall low importance of personal meeting management activities for the KWer's job role over to overall high importance for the KWer's job role, e.g., due to significant meeting overload. The importance implicitly influences the frequencies of conducting meeting management activities. A low importance implies infrequent meeting management activities whereas a high importance implies frequent meeting management activities.

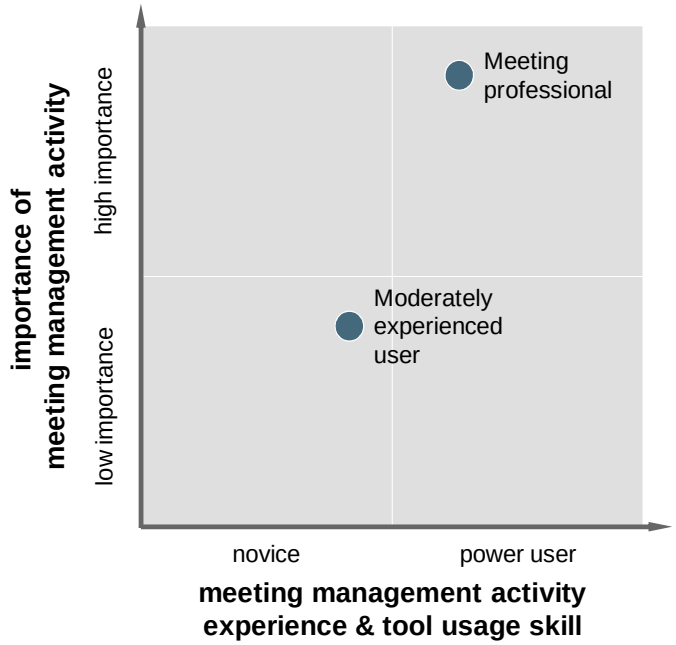


Figure 31: KWer profile with personal meeting management activity importance and experience.

Of the many possible presented experience and importance profiles that a KWer can incorporate, two KWer profiles are common, i.e., meeting professionals and moderately experienced users.

Meeting professionals are a group of people, usually in the role of, e.g., project managers and consultants, who participate in many meetings per week, have much experience with meetings and

are power users in their meeting-related tools. This reflects the high importance of their personal meeting management activities for their job roles. They have good experience and methodological skill in conducting meetings and the surrounding processes. They are power users with regard to meeting-related support, e.g., in collaborating with a team assistant or secretary to organize the meeting appointment. As well they are power users with regard to their computer-supported meeting-related tools and prefer to use the keyboard if possible to further increase their speed. They have experience in using wikis, i.e., they are familiar with textual markup syntax like used in wikis. They try to take the minutes as fast as possible and welcome any shortcuts in completing a protocol.

Moderately experienced users are a group of people, who occasionally take meeting minutes. They have less interest, less experience and less methodological skill in conducting meetings and the surrounding processes compared to meeting professionals compared to meeting management professionals. This is because they can fulfill their job as well without an elaborated meeting management, i.e., meeting management relatively has a lower importance for them. They are moderately experienced with regard to meeting management-related tool use. For example, they like to use the mouse and seldom use keyboard shortcuts. They know the concept of drag and drop and use it whenever possible.

4.5 Involved Information Items

The meeting process involves a number of information artifacts. Table 8 explains all identified information items in detail and depicts in which meeting-related activities these information items are involved.

Information item	Meeting-related activities the information item is involved
Agenda	The agenda is a list of topics, which need to be discussed or presented during the meeting.
People (Participants, interested parties)	Participants are people who attend a meeting while interested people do not attend a meeting but still want to be informed about the meeting's results.
Messaging	The activity needs a messaging information item in order to transmit the information when messages or documents are sent to different people.
Appointments	Appointments are a fixed mutual agreement for a meeting, which takes place at a specific place and time.
Protocol	The protocol consists of topics and protocol items. A good structuring of the protocol allows efficient post-processing.
Audio and/or video streams	Streams are another artifact of meetings. They can sometimes provide missing information which is not contained in the protocol.
Clusters, rankings and voting results	People use graphical creativity techniques when making decisions like, e.g., clustering or ranking and create respective artifacts.
Action items	Action items store information about what some responsible person has to do in order to fulfill the action's goal.
Protocol items	Protocol items are all items contained in the protocol. The items can be of a specific type like action, information, problem or decision. In order to efficiently post-process a protocol, an application must handle these information items.

Table 8: Information items involved in the meeting process.

4.6 State-Of-The Art in Meeting Management Support Applications

There are a number of applications supporting meetings and the meeting management process. Table 9 lists these types along with an example application and their meeting-relevant functionality.

Category	Example Application	Meeting-Relevant Functionality
Groupware System	Microsoft Outlook [Microsoft Corporation, 2009a]; Microsoft Exchange Server [Microsoft Corporation, 2009d]	Send invitation, distribute protocols, store appointments and action items
Word Processor	Microsoft Word [Microsoft Corporation, 2009e]; Microsoft PowerPoint [Microsoft Corporation, 2009f]	Take meeting minutes in the form of text documents
Creativity Support	Mindjet Mind Manager [Mindjet Corporation, 2009]	Take meeting minutes in the form of mind maps
Information Snippet Management	Tiddly Wiki [UnaMesa Association, 2009]	Take meeting minutes in the form of information snippets
Wiki	MediaWiki [Wikimedia Foundation Inc., 2009a]	Collaboratively take minutes which are stored in the wiki
Chat Application / Instant Messenger	Skype [Skype Limited, 2009]	Take notes in the context of a phone call / instant messaging discussion
Web Conferencing	Adobe Acrobat Connect [Adobe Inc., 2009]	Take notes in the context of a web conference
Meeting Audio/ Video Capturing	SoniClear Enterprise Recorder [Trio Systems L.L.C, 2009]	Record audio and/or video streams during a meeting
Workshop Support	Digital Moderation [Teambits GmbH, 2009]	Voting and decision making during large workshops
Meeting Manager	JourFixe [3ado AG (triado), 2009]	Support meetings during all phases of the meeting's lifecycle

Table 9: Application types supporting meetings and meeting-related activities.

A *groupware* application provides functionality for e-mail, calendaring and note taking in workgroups. Groupware systems are a common part of enterprise desktop environments, but they are general collaboration support software that is not focused on meeting support. They can be used to write invitations and to distribute protocols, but they do not provide domain knowledge to support the meeting process and they cannot be used to take minutes during a meeting.

A *word processor* is an application, which provides the user with tools needed to write, edit and format text and to save it as a document. Word processors are general tools, which have no specific knowledge about meetings, but can be used to take minutes during a meeting. The protocols can be structured by using different styles for topics and items. Topics can be formatted as a heading while items are "normal" text. Presentation applications support the same type of activities, as they support as well taking minutes during a meeting besides enabling the KWer to present slides to the meeting audience.

Creativity support applications, for example mind mapping tools, offer structured note taking during meetings. Hierarchy of the tree's nodes creates the structure. These tools are often used for information visualization and discovery of new knowledge.

Information snippet management applications allow using a computer like a scratch pad. In contrast to word processors, they allow structuring the content in more ways. One document is grouped into sections and subsections which can contain multiple pages of items. The applications provide methods for ordering and searching information contained in the documents.

Wikis are a type of collaborative applications which allow web pages to be created and edited using a Web browser. They are often used as a tool for knowledge management because they allow linking each information item with other existing information.

Chat applications and *instant messengers* provide chat rooms where multiple people can send each other textual messages, which appear ordered by time, by the sequence of their entry. The list with all messages of different chat contributions belonging to a meeting can be considered the protocol.

Web conferencing applications are chat applications which have some meeting domain knowledge and which offer support for having meetings. They allow sending invitations to each participant and it is possible to define roles like “presenter” for the attendees.

Meeting audio/video capturing applications typically record audio and video streams of a presentation, capture still images of slides shown on a projector and save images of whiteboards [Lee et al., 2004]. Some of these applications just require pressing the start and stop button while others can be actively used and allow linking textual notes to positions in the recorded streams. The focus however is on the recording of the audio and video streams.

Workshop support applications are targeted on large audiences of up to 500 people. Most of the time they are used in situations when there is only one presenter talking to a larger audience. The applications are often used by all participants and provide different views for the presenter and the audience.

Meeting manager software tries to cover all phases of a meeting’s life cycle. The applications in this category have the most domain knowledge of all presented categories. They support planning meetings and agendas, sending invitations, writing minutes and distributing the protocols.

The above presented types of meeting support applications each target each a number of activities of the meeting management process. Figure 32 gives an overview on the particular meeting process phases supported by an application type.

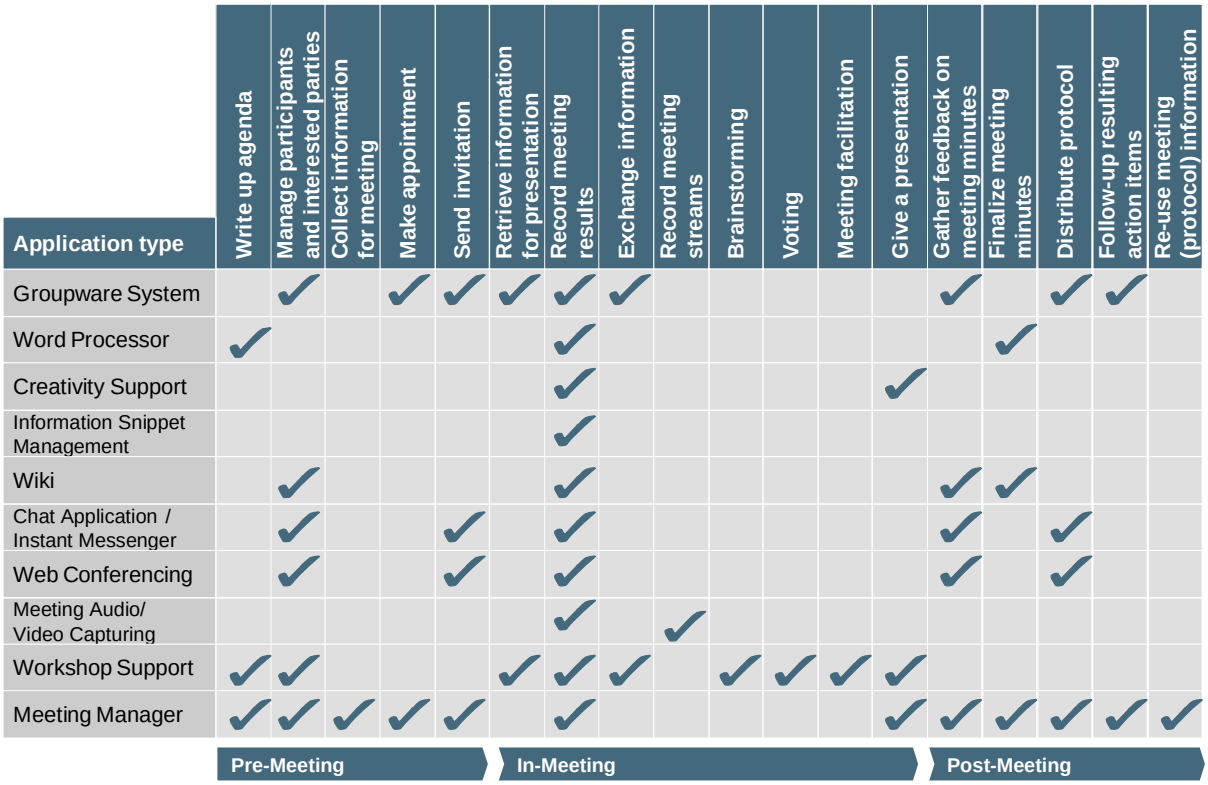


Figure 32: Detailed overview of supported meeting activities by meeting support applications.

PART II – APPLICATION – USING UNIFIED PERSONAL INFORMATION

ACKNOWLEDGEMENTS AND PUBLICATIONS

- The work on Kasimir and the SAP Task plug-ins presented in section 5 and 6 has been published in [Grebner et al., 2008] and [Grebner&Riss, 2008a].
- Special thanks goes to the students Christian Herlambang, Daniel Felix Müller, Karol Ratajczak, Benedikt Schmidt, Karanjit Singh and Fan Zhang for contributing to the Kasimir development.
- Tobias Ackermann contributed to section 7 with his study thesis [Ackermann, 2008] on meeting management and the Nepomuk Meeting Manager and for contributing to the Nepomuk Meeting Manager development.

5 Kasimir – Personal Task Management Application

The Kasimir application is a lightweight application designed to support the KWer's personal task management. The core design principle was simplicity regarding the task management features, i.e., these features are arranged in a way that allows easy handling.

In the following, we show first an overview on the Kasimir application. Then we show the Kasimir features for its three perspectives, i.e., for the personal task management perspective, the task information perspective and the social task perspective. Then we show the architecture and implementation of Kasimir and highlight the KWer's benefits of unified personal information within Kasimir.

5.1 Kasimir Application Overview

Kasimir enables the KWer to handle task representations, i.e., the explicit representations of the KWer's tasks in conjunction with related information and related people, see as well section 3.1. Kasimir integrates these three perspectives using a KWer's task representations and accordingly offers support functionality for the KWer:

- The *task perspective* allows the KWer to handle tasks, provides a task overview to the KWer and enables the KWer to break down the KWer's workload and structure it into individual task representations and prioritize them.
- The *task information perspective* handles information related to the task, i.e., shows task-related information and enables the KWer to attach needed emails, websites and files to the task representation as well as enables taking of notes that are stored with the task representation.
- The *social task perspective* handles people involved into a task and supports collaboration on the task among the involved people.

Kasimir has been *designed as sidebar* which resides next to open desktop applications to be available in the whole workspace, see Figure 33. The KWer needs task management functionality in the whole workspace and thus task management needs to be available across applications.

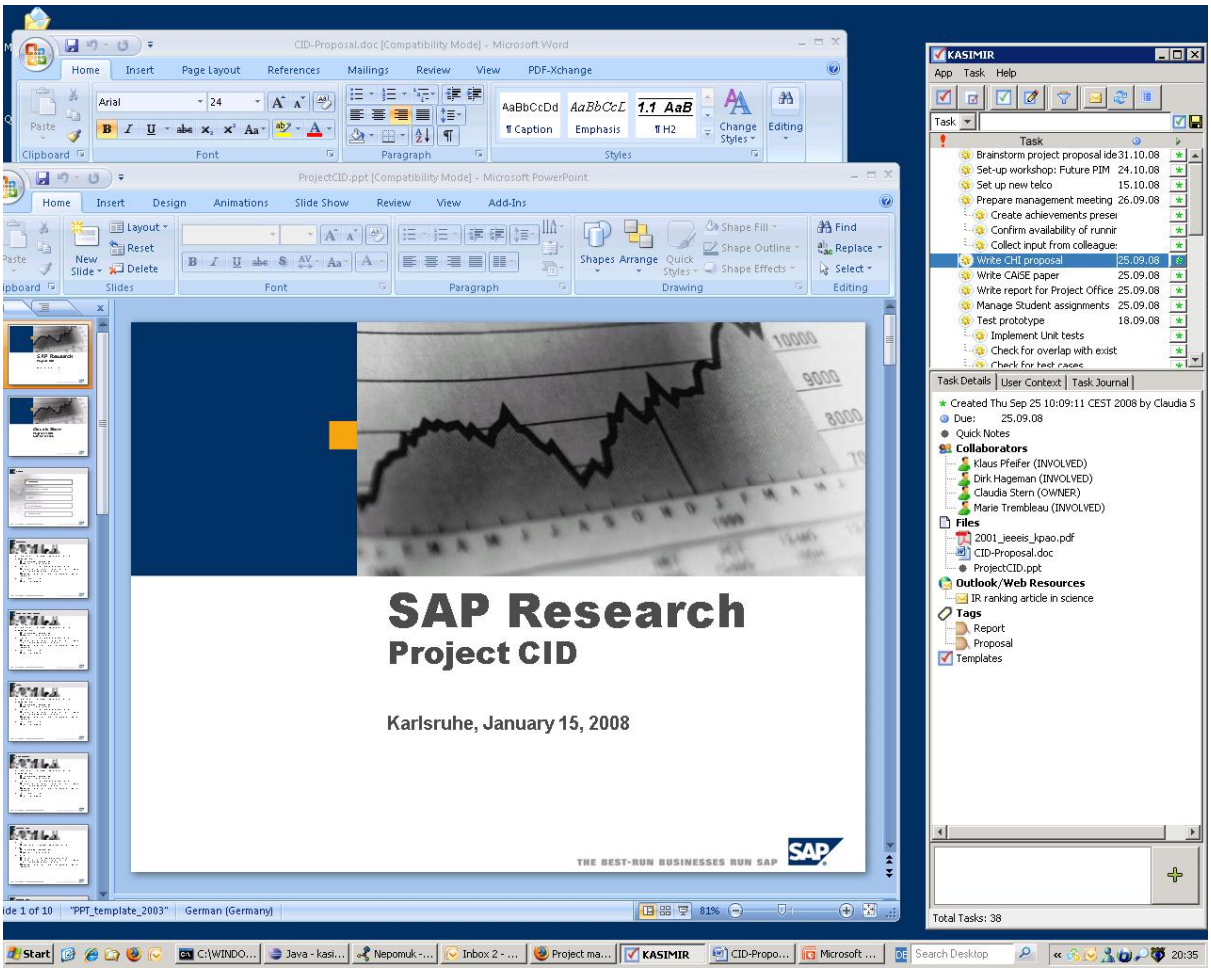


Figure 33: Kasimir Task Sidebar besides desktop applications.

Kasimir as task sidebar is visible beneath open desktop applications or can be made visible beneath applications without distracting the KWer from working within the respective application. In case that the task side is hidden behind other applications the KWer can quickly recover it and bring it back into the focus using a hotkey or by switching with the ALT + TAB window switch hotkey to the Kasimir application.

The Kasimir *task sidebar* serves as the primary front-end of the personal task management application. It enables efficient handling of existing tasks and displays detailed task information. It consists of five parts as shown in Figure 34.

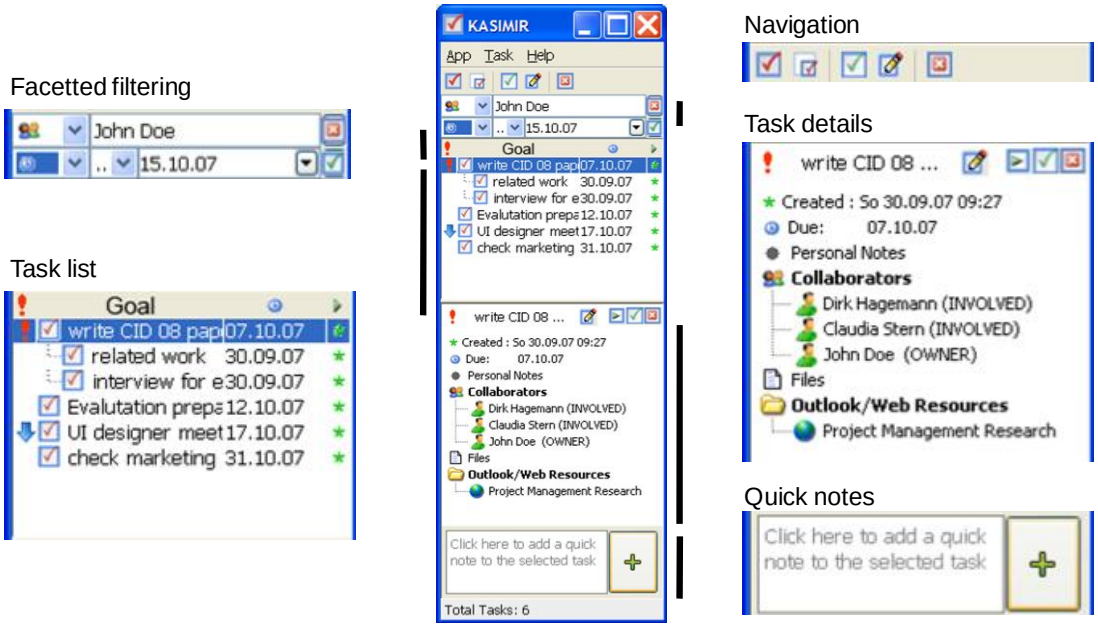


Figure 34: Kasimir Task Sidebar - Details.

All depicted Kasimir parts contribute to one of the mentioned perspectives and the respectively detailed use cases. Figure 35 presents the mapping between the Kasimir parts and the covered task management perspective and use cases.

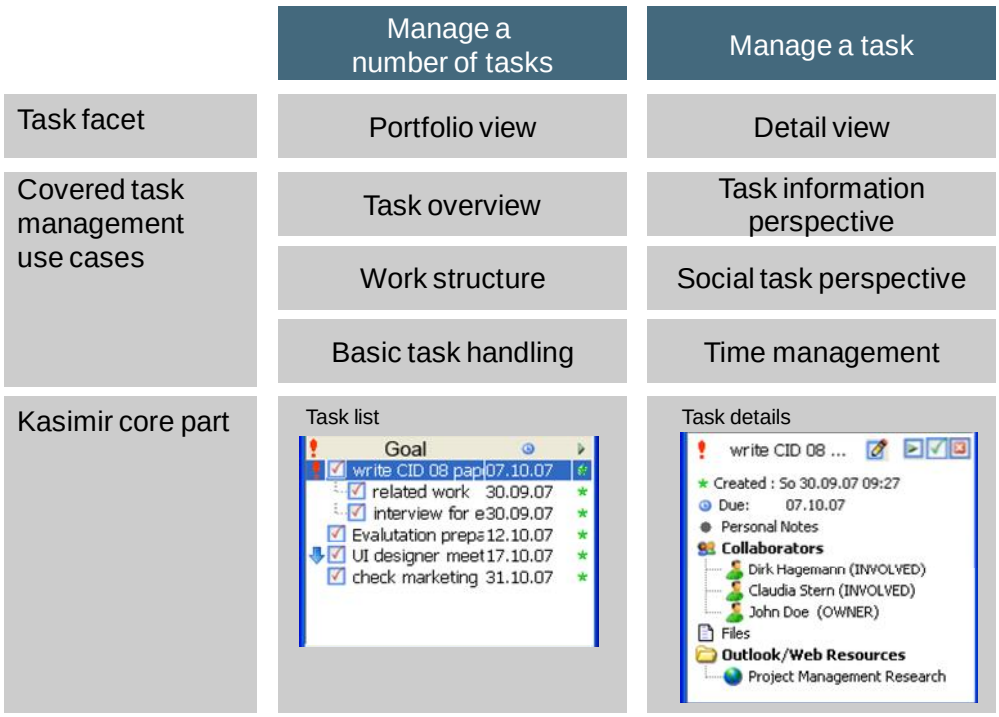


Figure 35: Mapping Kasimir parts and covered personal task management perspectives.

The KWer uses the task list to gather an overview of the individual tasks, i.e., uses the task perspective. The task list displays tasks and subtasks in several levels of detail as well as indicates their hierarchical relationship. Filtering options enable the KWer to display task subsets.

Beneath the task list, the task details view shows a single task's properties, including the task information perspective and the social task perspective. Basic properties of a task, such as task due

date and importance level are shown as well as the task details, which is a listing of associated information. This listing also displays the content of the associated information objects. For example, calendar items open in their default application in order to change them there. Information objects such as websites, documents and emails are also displayed using the default applications. There are several interaction possibilities for task detail items so that KWs can delete items or see more details.

The details of the presented Kasimir task sidebar are explained along the covered perspectives below.

5.2 Personal Task Management Perspective

Kasimir supports the KWer's personal task management and covers several use cases. These are basic task handling, consolidated task overview and task list management, task priority management, task time management and task planning.

5.2.1 Basic Task Handling

Basic task handling deals with organizing the KWer's task representations. The KWer can use a number of basic lifecycle actions to *handle a task*. This includes creating a task, populating a task with information, updating and modifying a task as well as deleting a task.

The KWer can *create a task* with just a task name to create a task representation with the least possible effort and then extend it to suit it for more sophisticated task management support. Kasimir was designed for an incremental definition of a task representation to adapt the effort needed to create and maintain a task representation to the benefit the KWer gains from the task representation. The KWer can incrementally specify the task. Starting with just the task's name the KWer can enrich the task's representation along with gained insights through the task execution. The more information the KWer enters the higher the benefit the KWer gains in form of task management support through Kasimir. The same holds true for creating subtasks where the KWer in addition needs to select a task which then becomes the subtask's parent task.

In Kasimir, the KWer can use for this a *simple task creation dialog window*, see Figure 36. This dialog window was designed for a quick task entry and besides the task name the KWer optionally can enter further basic task attributes like the task due date and the task priority.

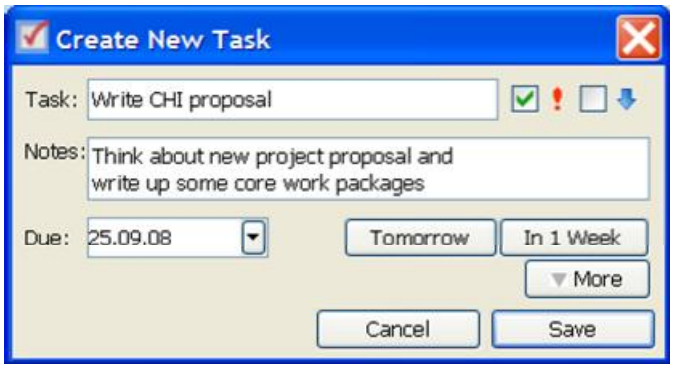


Figure 36: Simple new task dialog window in Kasimir.

When the KWer wants to specify further task details, clicking the 'More' button extends the simple form into the *extended new task dialog window*. This dialog window allows the KWer to add additional information to the task, see Figure 37.

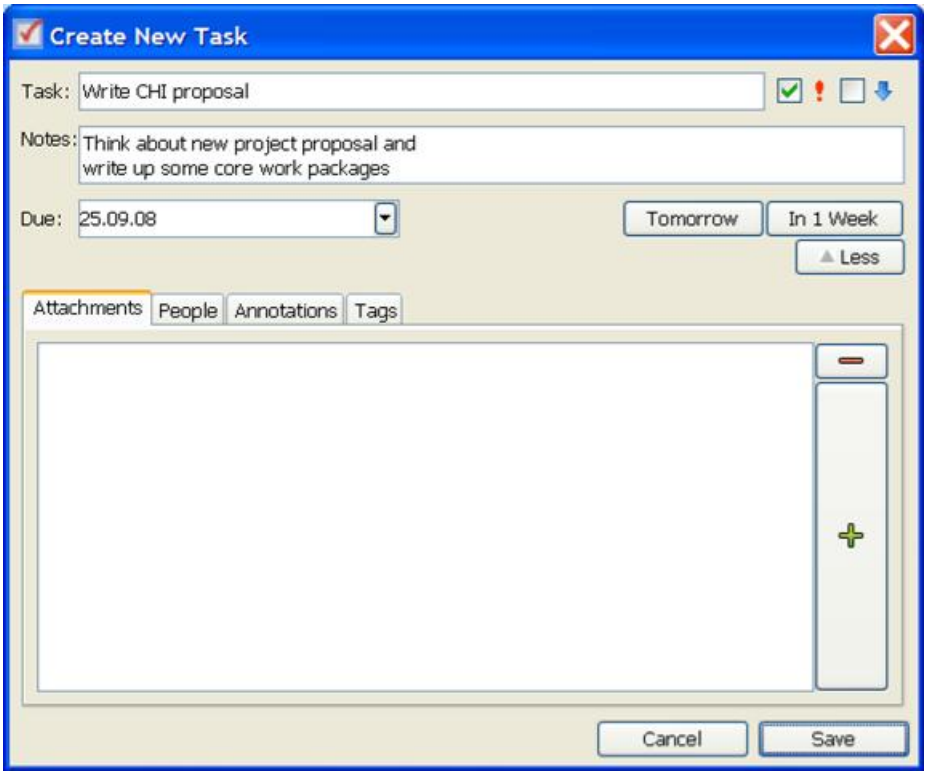


Figure 37: Extended new task dialog window in Kasimir.

In particular, the dialog window enables the KWer to attach information objects like, e.g., files, websites or emails needed for the task to the task representation, to specify the social perspective by defining involved persons and their roles for the task and to categorize the task with tags. All these functions are explained in detail in the following sections.

The KWer can *update and edit the task* using the same dialog window to streamline the KWer's user experience and reduce learning effort. The task editing dialog window shows the already entered task information and the KWer can modify all task attributes.

The KWer can *set a task state* for each task to manage active, running and completed tasks. The KWer can select a particular task state, i.e., new, running or completed in the task list, see Figure 38. This function has been positioned in the task list as the task state usually is set from an overview perspective on all available tasks.

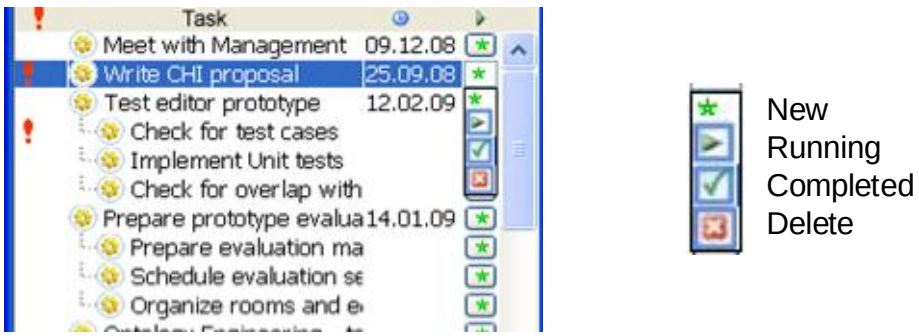


Figure 38: Setting a task state in Kasimir.

The KWer can *delete a task* as well in the task list, see Figure 38. The task is immediately removed from the task list and possibly existing subtasks are removed as well. However, referenced information objects and tags are not deleted themselves. Deleting such referenced items would

remove these items from other tasks and applications. This would lead to astonishing side effects for the KWer as she doesn't expect information items to disappear in applications just because of deleting a particular task.

5.2.2 Consolidated Task Overview and Task List Management

Kasimir presents a consolidated overview on the KWer's tasks in a task list and provides sophisticated task list management functionality. The KWer can sort the task list according to several criteria, can define subset of tasks in the task list by creating filters and can save and re-use each created filter.

The Kasimir task list presents a *consolidated overview on the KWer's tasks*, see Figure 39. The underlying rationale of having all of the KWer's tasks available in a single list is to provide the KWer with a comprehensive overview on the personal workload, independent of their source, i.e., independent of where the task was created or in which application the task emanated originally.

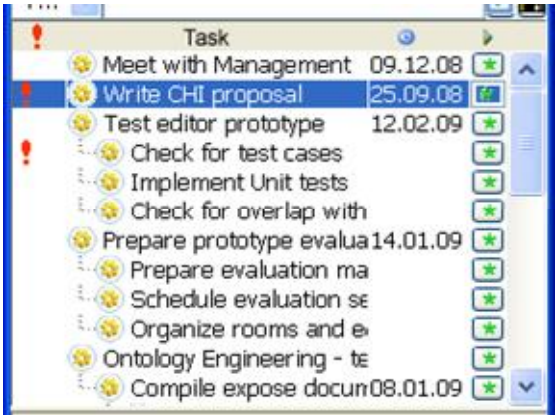


Figure 39: Task list in Kasimir shows tasks independent of their source.

The Kasimir task list presents every KWer's tasks, i.e., tasks that target the KWer or which the KWer needs to pursue. These tasks either are created by the KWer herself, in Kasimir or any other application that is based on the unified personal information model, or are imported from different applications and sources[9].

The KWer can *sort the task list* according to several criteria to be able to obtain different viewpoints on the tasks. Depending on the KWer's interest, the task list can be sorted by task priority, task name, task due date and task state, i.e., the tasks in the task list are displayed in the order of the selected criterion. The current ordering including the sorting direction, i.e., upward or downward, is indicated by a triangle in the header of the column of the selected criterion. When sorting by priority, the task list alternatively presents the task with highest or lowest priorities on top enabling the KWer to focus on the highest ranking tasks or respectively dealing with lowest ranking tasks. Sorting tasks by name allows the KWer for quickly identifying a particular task based on its name. In a time-ordered sorting, the task list can display the most recently due tasks on top or display the oldest due tasks on top which enables the KWer to assess the workload with regard to a particular deadline. Sorting tasks by their state enables the KWer to, e.g., quickly find out pending tasks which need to be finished or to archive completed tasks.

[9] Other sources for tasks include for example applications with task management functionality like, e.g., Microsoft Outlook [Microsoft Corporation, 2009a].

The KWer can *display a subset of tasks* in the task list by defining filter criteria on the tasks. The KWer can filter tasks according to several criteria using Kasimir's *facetted filtering* [Yee et al., 2003] functionality on tasks. Facetted filtering enables the KWer to filter tasks by defining multiple criteria that can be combined which drill the task list down to the intended subset of tasks. A filter consists of one or more filter items which can be arbitrarily combined, where each filter item represents one filter criterion. In Kasimir, the KWer can use the following filter criteria on tasks to build a filter: Task name, task due date, involved persons, involved documents and task tags. To construct the filter, the KWer selects a filter item, defines its value and adds it to the filter, Figure 40 for example shows the task satisfying the two selected filter items of the tag 'Proposal' and the person 'Dirk'. For certain filter items, additional dialog windows help the KWer to select existing filter item values. For example, for setting the filter item for task tag, the KWer can choose a tag value from a tag cloud which shows the used tags and their frequency for the so-far filtered task list, see Figure 40.

Figure 40: Facetted Filtering in Kasimir (left) and Tag Cloud for Task Tag Filter Item in Kasimir (right).

For example, the KWer can build the following filters. Creating a filter which shows all tasks a particular person is involved in enables the KWer to, e.g., prepare a meeting with a person and thereby quickly oversee what tasks are still open and tasks are already completed. Creating a filter which shows all tasks that have been tagged with "next" and that are not yet completed enables the KWer to maintain a daily to-do list which only contains the few tasks that need to be tackled. Creating a filter which shows all tasks which have a due date in the past and which have a task status other than completed to find out the personal backlog of overdue but not yet finished tasks.

The KWer can *save each created filter* to re-use it without investing effort into building it again. After having created a filter, the KWer simply presses the 'Save' button besides the filter, see Figure 41, to save the filter. To re-use a filter, the KWer can select a filter from a list of saved filters in the filter selection dialog window, see Figure 41. After selecting a filter, the filter is applied and the filtered task list shows up.

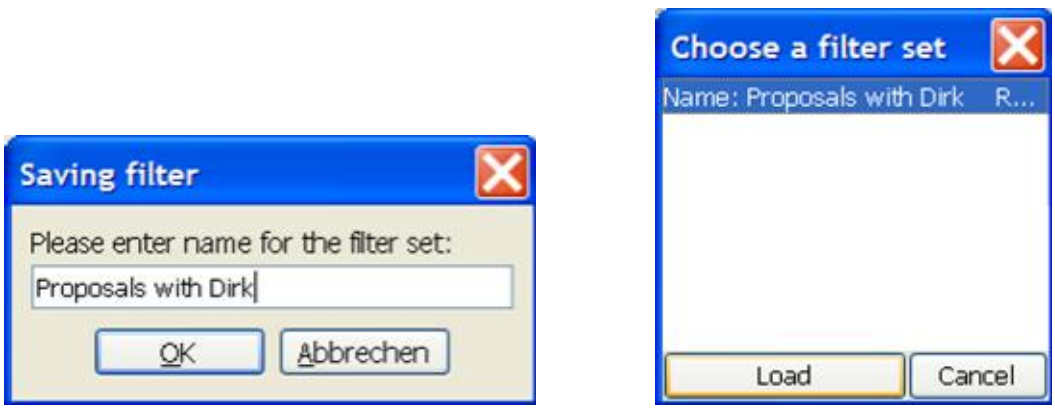


Figure 41: Save filter dialog window (left) and Selection dialog window for saved filters (right).

Task priority management helps the KWer to *maintain the priorities* coming with a task. The KWer can assign for each task a priority. Within the task list, the KWer can adjust the priorities from an overview perspective.

Task time management focuses on the *time needed to execute a task* and the KWer can assign a task's due date to manage the time-related aspects of work.

5.2.3 Task Planning – Structure the Workload

Task planning helps the KWer to structure the workload and perform work decomposition, i.e., breaking down and categorizing tasks.

The Kasimir task list *shows tasks and subtasks in a hierarchical structure* representing the broken down tasks and contributing subtasks. A subtask is itself a task which has an own goal etc. and whose execution contributes to the goal of its super-task. The task-subtask relationship is graphically indicated for a task and its subtasks, see Figure 42.

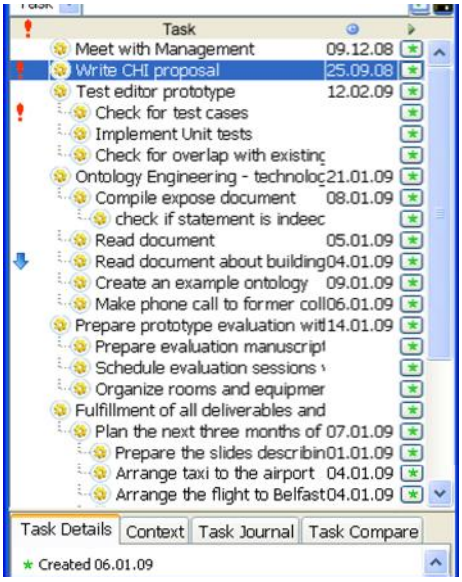


Figure 42: Task list in Kasimir showing task and subtask hierarchies.

Of note is the non-mandatory task hierarchy, i.e., tasks not necessarily need to be a subtask of a particular root task. The KWer can create a task with as little effort as just typing a name and the task shows up in the unfiltered task list.

The KWer can *create tasks and subtasks and their relations* to hierarchically represent the KWer's workload and thus Kasimir supports the KWer's task planning effort. The KWer can plan her workday by breaking the workload down into manageable pieces and to be able to logically process the work activities. This planning continues for each piece of work, i.e., a task, by thinking on how to best execute it and by defining the next steps, i.e., subtasks.

In addition to task/subtask relationships, the KWer can *assign tags to a task* in Kasimir to categorize the tasks according to generic categories defined by the KWer. Categorizing a task with tags enables the KWer to express similarity with regard to a certain topic and this relationship is orthogonal to the hierarchical task/subtask relationship. Adding multiple tags to a task is possible.

For example, a tag ‘Nepomuk’ enables a KWer to categorize all tasks contributing to the Nepomuk project. Or the tag ‘business’ and ‘private’ can define categories that indicate whether the task is for private or business purposes.

To assign a tag to a task, the KWer enters the tag in the task editing dialog window, see Figure 43. The tag entry is supported by an auto-completion feature which proposes existing tags which contain the so-far typed characters. The KWer can then select a tag from the list of proposed tags or continue typing to create a new tag.

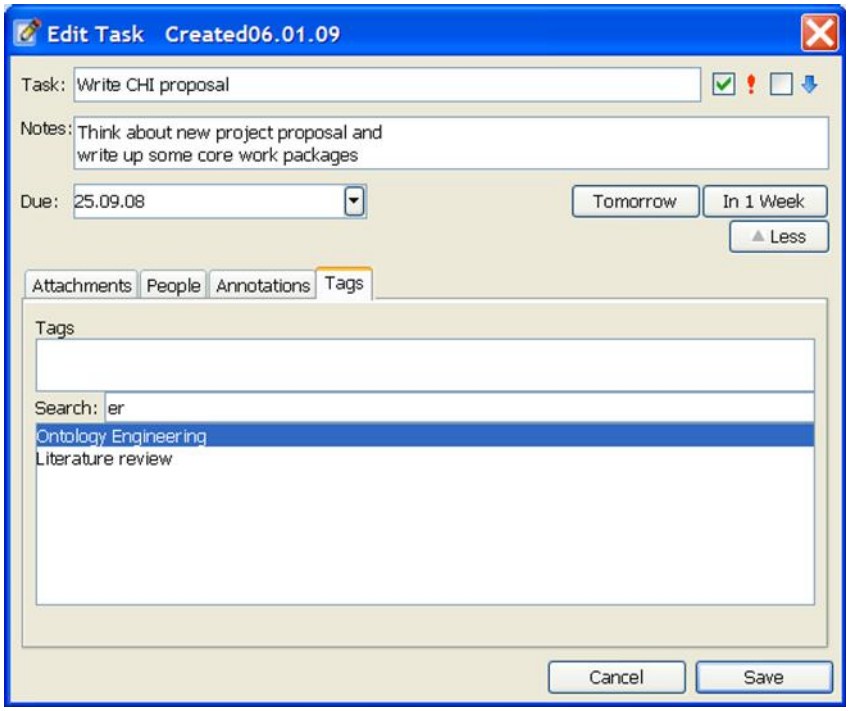


Figure 43: Add task tags in the Task Edit Dialog window in Kasimir.

Planning the KWer’s tasks happens in two different situations which are both supported by Kasimir. First, the KWer *initially breaks down the workload*. The KWer plans the proceeding of the typically new task and has no existing task structure in place. This means that the KWer creates tasks and subtasks from scratch to represent the intended tasks and execution structure. Creating a subtask in Kasimir is identical to creating a task with an identical task creation dialog window. The only exception is that before opening the new subtask dialog window the KWer needs to select a task which then becomes the subtask’s parent task.

Second, the KWer can *adapt the task structure during task execution* to the gained, evolving knowledge. The KWer develops further the task structure using the insights gained throughout the task execution. Besides changing the content each task, this requires the KWer to modify in particular the task subtask hierarchy. Kasimir enables for this the KWer to move tasks in the task hierarchy, i.e., the KWer can shift a task up in the task hierarchy. To detail a task, the KWer can create new subtasks.

5.3 Task Information Perspective

Task Information Management helps the KWer to collect and associate information needed for executing a task. The KWer can gather all task-relevant information in one place with Kasimir to be able to look up this information without needing to invest effort into searching for it. On the one hand, the KWer deals with a number of information objects while working within the workspace or

particular applications. These information objects are usually scattered across the whole workspace and across several applications. And on the other hand, the KWer creates information related to the task while, e.g.., developing thoughts, thinking about the problem at hand or to record the recent progress.

The KWer can attach all needed information objects to the task that the KWer needs to execute the task. In particular the KWer can attach files, i.e., any file system based documents, spreadsheets or the like, emails and bookmarks, i.e., links to websites.

All this task information is shown in one place with the task representation within Kasimir, see Figure 34. The KWer directly gets an overview on the attached information. The task-related information shown in Kasimir is grouped by the type of the information objects, i.e., there are 'Files' for files and 'Outlook / Web resources' for information objects related to the email client and internet. The KWer can directly access the attached information objects and open each attached information object in the respective application.

The KWer can *add and delete information objects* directly within Kasimir. To add an information object either directly from within Kasimir the KWer selects a particular information object in a selection dialog for files and emails and an input dialogue for websites, see Figure 44 and Figure 45. The selection dialogue for files and emails allows the KWer to search and filter files and emails on the desktop in order to drill down to the needed file or email. In the input dialogue for websites the KWer can enter the URL of the web-based resource.

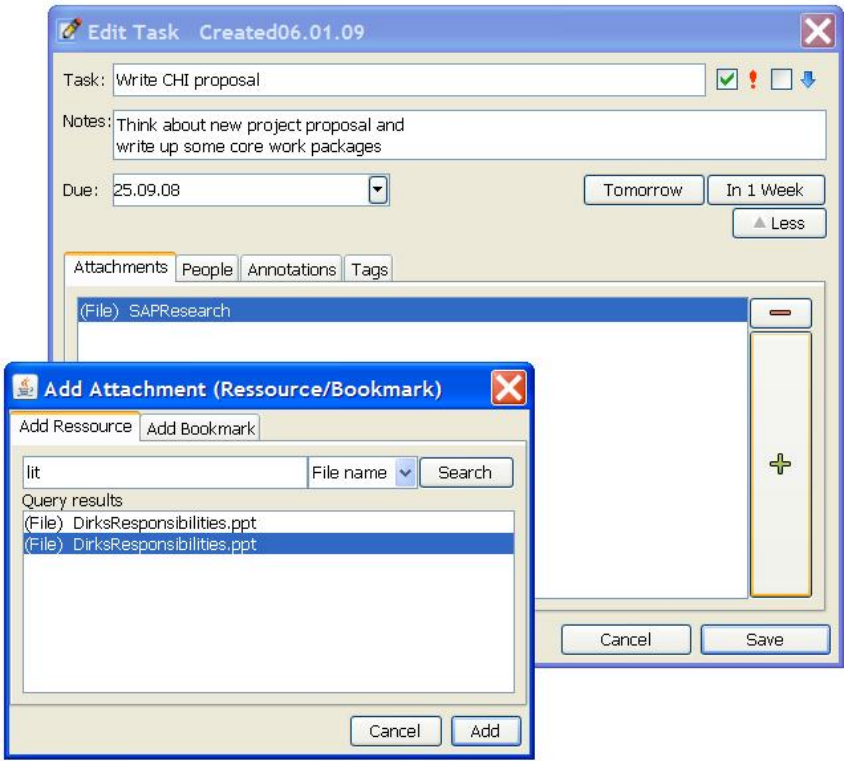


Figure 44: Selection dialogue for adding task attachments (files and emails) in Kasimir.

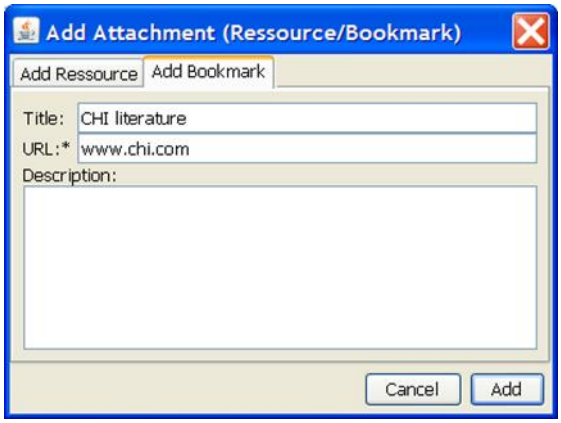


Figure 45: Input dialogue for adding task attachments (websites) in Kasimir.

Alternatively, the KWer as well can *add information objects to a task in other applications than Kasimir*. These applications need to be able to access the unified personal information model on which Kasimir is based, i.e., the PIMO [Sauermann et al., 2007]. There are, for example, a set of task plug-ins that are embedded into common office applications like, e.g., the SAP Outlook Task Plug-in, a task plug-into the Microsoft Outlook email client and the SAP Firefox Task Plug-in, a task plug-into the Mozilla Firefox web browser, see section 6 for details.

The KWer can *create personal notes for each task* in Kasimir to record information snippets relevant for the task. The KWer can create and collect a number of information snippets with the task using the 'Quick text' entry form, see the 'Quick text' section in Figure 34. With the 'Quick text' entry form, the KWer can quickly type some lines of text and attach this text to the selected task.

The content of these information snippets can be multifaceted and is fully up to the KWer on what to write down for the task. For example the KWer could record the results of developing thoughts, possible structuring options for a particular problem at hand or collect notes of gathered experience throughout the task execution.

5.4 Social Task Perspective

The KWer can *access a task-specific social perspective* in Kasimir, i.e., the KWer is able to see and interact with all persons relevant for a task in one place, see Figure 34. In addition, it focuses on collaboration in the task domain. This means, that KWers can delegate tasks to each other, can perform task tracking and conduct information sharing.

The KWer can *attach people to the task* where she feels that they are relevant for the task. Collecting all needed persons with a task enables the KWer to quickly get an overview on the involved people, identify contact persons for answering specific questions or to simply interact with one of the involved persons to execute the task. The KWer sees all relevant people at a glance and thus doesn't need to invest effort into searching all or particular persons when needed.

To *add a person to a task*, the KWer first identifies or creates a person and attaches this person to the task in the person selection dialog window in Kasimir, see Figure 46. To *identify a person*, the KWer can just start typing the name of the person. The auto-completion functionality displays a list of persons whose first, middle or last name contains the typed name. This list is continually updated and narrowed down as the KWer types further characters of the name. In case that the KWer found the needed person, she selects the person from this list and adds the person to the task by double-clicking it. The KWer can *create a new person representation* in case that the needed person doesn't

show up in the list of persons. Therefore, the KWer completes the name and additionally can enter an email address. With the confirmation for creating the new person, Kasimir creates the person representation transparently for the KWer in the background. After creating the person representation, this person is directly added to the task.

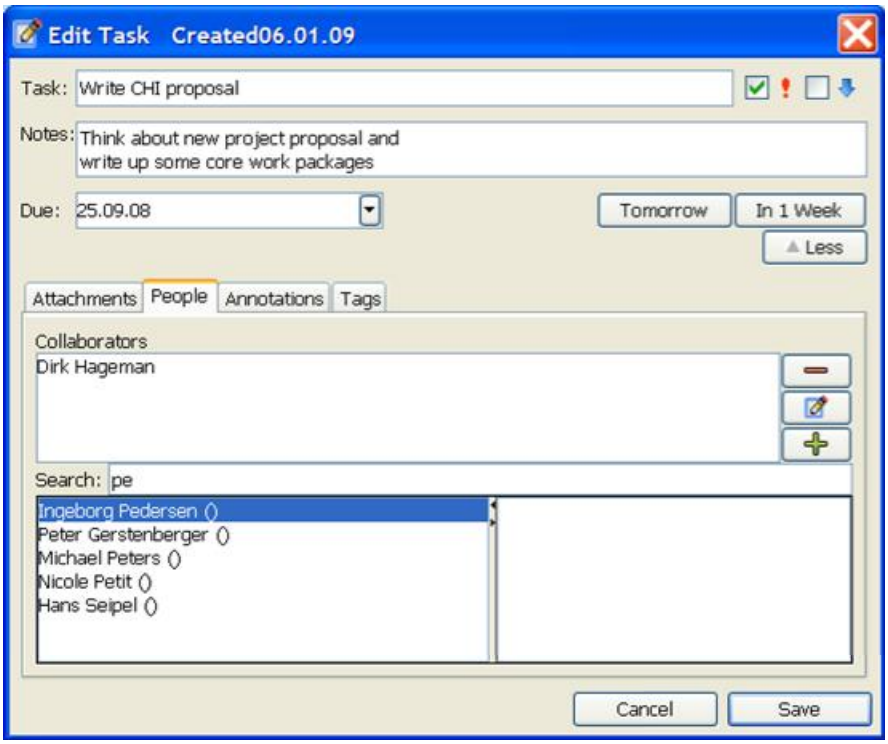


Figure 46: Person selection dialog window in Kasimir.

The KWer can *assign roles to attached people*, i.e., specify in which function they contribute to the task, see section 3.1 for possible roles. For example, a task owner has other options and responsibilities than a person who just wants to be informed on the task result. With the role concept in place for each person attached to a task, Kasimir can treat each role appropriately. This means that Kasimir, e.g., doesn't send information on the task's progress to people who are not interested in the task result like in the case of a submitted task. Currently, the role management is intentionally kept simple in Kasimir. When a task is created, the task creator, i.e., the KWer, is automatically added as the owner of the task. Each additionally attached person gets assigned the role "Involved". However, Kasimir is prepared for more fine-grained role assignments, like e.g., roles like "Collaborator", "Delegate" and "Reviewer", see the TMO information model for further possible roles [Grebner&Brunzel, 2008a].

Adding a person to a task in Kasimir *enables a number of functions for the KWer*. To invoke one of these functions, the KWer first select one or more persons and then triggers an action by right-clicking on the selection. Figure 47 shows the context menu showing the available actions for one person.

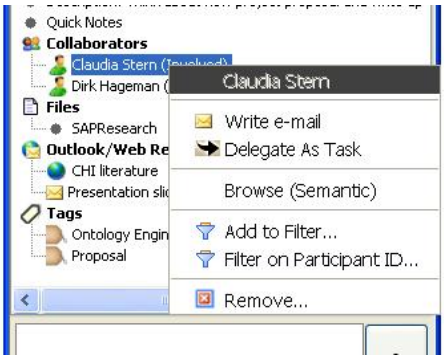


Figure 47: Interaction options in the context menu for an involved person.

The KWer can *obtain detailed information on a person* by triggering the 'Open in semantic browser' action. This launches and opens the person detail view within the PimoEditor embedded into the Nepomuk P2P Semantic Eclipse Workbench (PSEW) application [Sauermann, 2009] [Nepomuk Consortium, 2009g], see Figure 48. The person detail view shows all information related to this person which is available in the unified personal information model, like for example other related people, tags on the person, available skills or information objects like files that are assigned to the persons.

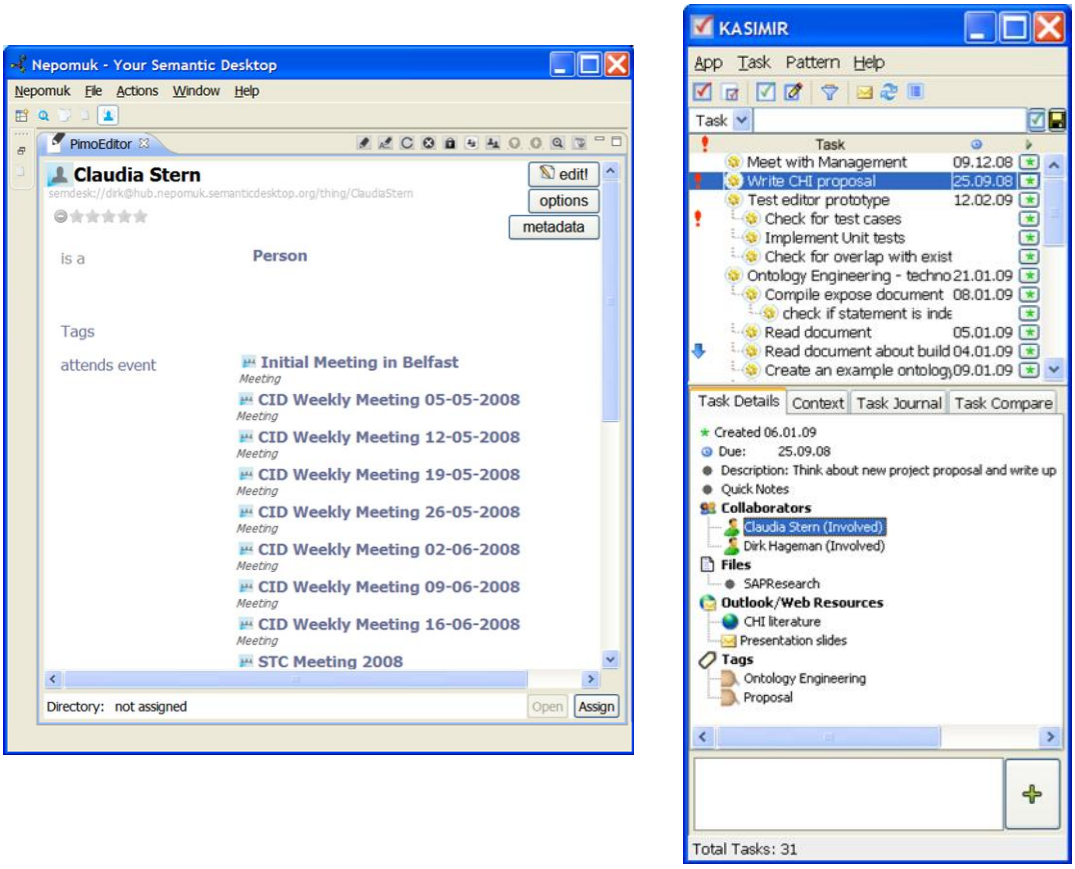


Figure 48: Person details within PimoEditor

The KWer can *interact with the person* through Kasimir by triggering the 'Send email' or 'Start instant messenger conversation' action. This either launches a 'new email' window of the primary email client with pre-populated information, i.e., the person's email as recipient and the task name as subject, or launches an instant messenger window with the person and the KWer can start a conversation.

The KWer can *collaboratively share and edit the task with a person* by triggering the 'Share task' action. In the upcoming window, the KWer can select what information should be shared with the selected KWer. After conforming this, Kasimir compiles this information, creates a collaborative task and sends this information to the receiving KWer. In the course of the delegation the KWer can synchronize the local task's information in Kasimir with the information of the collaborative task in both directions, i.e., committing and updating the task information.

The KWer can *delegate the task to the person* by triggering the 'Delegate task' action. Delegating a task implies a handover of task responsibilities and an assertion for acting in comparison to task sharing, where only the task information is shared among a number of people and they collaboratively work on it. In the upcoming task delegation dialog window, the KWer can select the task information that should be sent to the other KWer who receives the task and specify a message that receiving KWer receives together with the delegated task. Kasimir then compiles the selected task information into an email message and sends this message off to the receiving KWer. After having delegated the task, Kasimir enables *task tracking*, i.e., the task status and the task information gets updated in case that the receiving KWer delivers results back to the task sender.

5.5 Architecture and Implementation

The Kasimir application supports a KWer's personal task management activities. Its implementation focuses on the user interface, a Java Swing-based [Eckstein et al., 1998] application, whereas the invoked Task Management Service, see section 10.1, prevalently provides the business logic. This keeps the implementation of the Kasimir application lean. Figure 49 shows the principle of the Kasimir application invoking the Task Management Service.

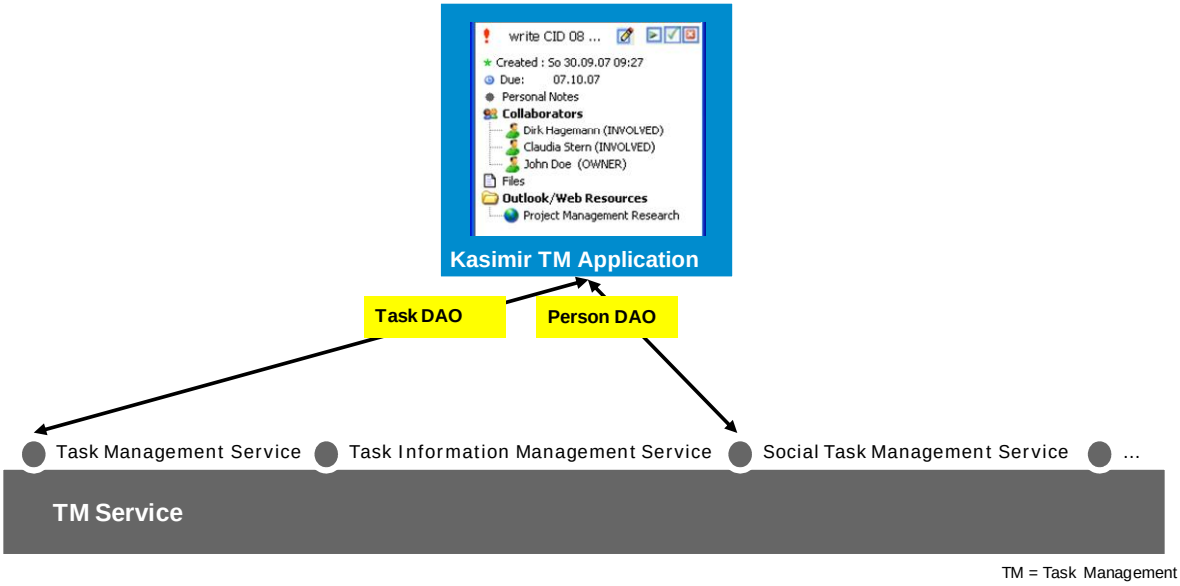


Figure 49: Kasimir implements the TM Service for personal task management functionality.

Kasimir *offers individual task management application features to other applications*. Other applications can invoke Kasimir application features through the Task Management Service interface, see section 10.1.3.3. This interface allows them to remotely trigger the same application features that a user could invoke on the screen with a manual action and in addition it transfer needed information as parameter. For example, the "add task" dialog window of the Kasimir sidebar helps an application to present this dialog window in its application context. This takes place in a unified way across all participating applications and the KWer is relieved from manually transferring the

information across applications, e.g., by copy-and-paste. Other applications can this way outsource any task-related functionality to Kasimir and thus re-use needed application features. This enables these applications to focus on their core competency instead of re-implementing task management. For example, the SAP Task Plug-ins leverage this functionality to provide a consistent task creation dialog window in applications like Microsoft Outlook and Mozilla Firefox. The Kasimir sidebar features complement the functionality of the SAP Task Plug-ins.

5.6 KWer Benefits of Unified Personal Information

The KWer benefits from that the Kasimir personal task management application operates on the representation of the KWer's personal information cloud. On the one hand, the KWer can contribute information about tasks and their related information to the personal information cloud. On the other hand, the KWer can use the information from the personal information cloud within Kasimir, see section 8.3. The KWer doesn't need to take care about keeping potentially redundant sets of information.

The Kasimir application bases on a separation of concerns for managing personal information. It focuses on managing the 'own' task information, whereas it leverages information maintained by other applications. This enables the KWer to use for each use case the favorite and best suited application.

5.6.1 Contribute Tasks to Personal Information Cloud

The KWer uses Kasimir to primarily manage the personal tasks. Thus the Kasimir application primarily manages task representations, see Figure 50 for a simplified overview. Kasimir thereby operates on the personal information cloud, i.e., the managed task representations are available in the personal information cloud.

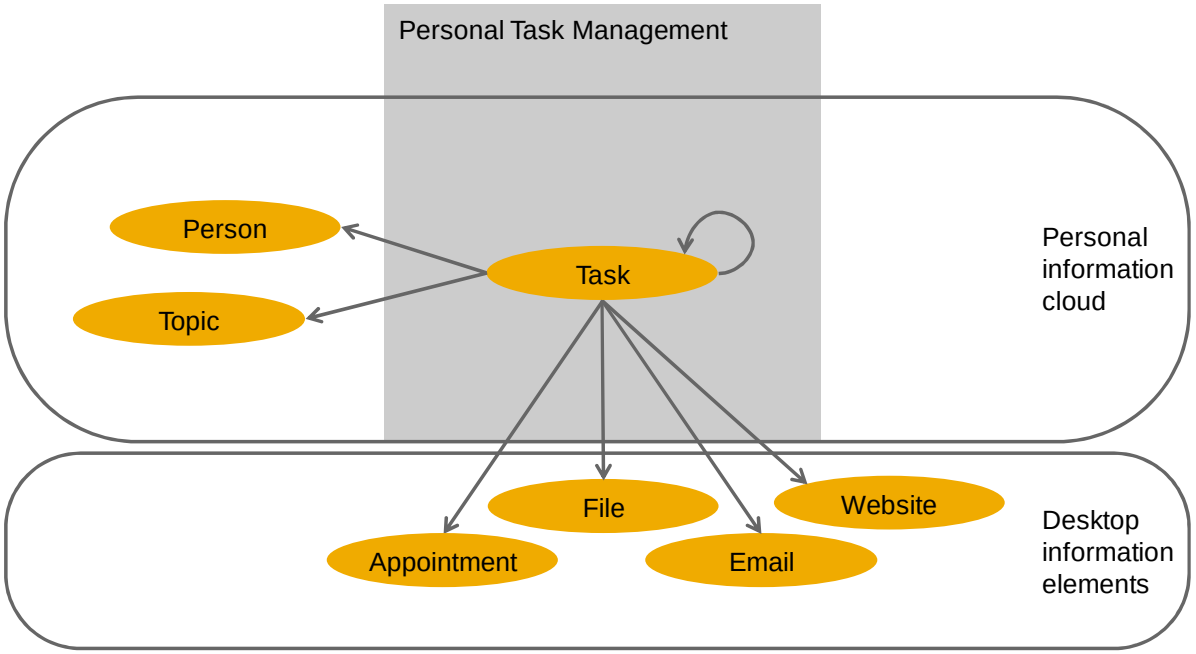


Figure 50: Personal Information Concepts managed by Kasimir.

The KWer can relate tasks among each other, i.e., hierarchically relate tasks as sub- and super-tasks. The KWer can associate with each task the representations of the involved persons. Furthermore, the KWer can associate topics to categorize the task to enable re-finding.

5.6.2 Use Personal Information Cloud for Tasks

The KWer can use through the Kasimir application functionality the information available in both the KWer's personal information cloud representation and the desktop information elements.

The KWer can categorize a task by associating a *topic*, shown as tag in Kasimir, e.g., to be able to re-find the task again. The KWer can enter a tag by typing in the name in the add task tag section of the Task Edit Dialog window in Kasimir. Thereby, Kasimir's auto-completion feature offers recommendations on possible tags based on the KWer input, see section 5.3. These recommended tags are matching topic representations from the personal information cloud. When the KWer selects one of the offered tags, i.e., a topic from the personal information cloud, Kasimir associates the task representation with the selected topic and thus associates the task representation with the remaining personal information cloud.

All *person* representations showing up in Kasimir correspond to person representations in the personal information cloud. The KWer can associate *persons* involved into a task with the task itself, see section 5.4. To attach a person to a task, the KWer can select a person and its role for the task in the person section of the Task Edit Dialog window in Kasimir, see section 5.4. There, the KWer can enter person's name. Thereby, Kasimir's auto-completion feature offers recommendations on possible matching persons based on the KWer input, see section 5.4. These recommended persons are matching person representations from the personal information cloud. When the KWer selected a particular person, Kasimir associates the person representation with the task representation in the personal information cloud and records the role of the involved person in the task.

The KWer can associate *desktop information elements* to a task representation using Kasimir's functionality for attaching needed information to a task. For example, task-relevant information elements on the desktop are files, websites and emails. When the KWer selected a particular information element in the corresponding selection dialog window in Kasimir to attach it to a task, Kasimir associates the information element with the corresponding task entity. For example, when the KWer selected to attach a file to a task in the corresponding resource selection dialog, see section 5.3, Kasimir associates the file representation with the task representation in the personal information cloud. The KWer can perform similar actions with emails and websites, see section 5.3.

6 Task Application Plug-Ins – Workspace-Level Personal Task Management

To integrate task management activities into a KWer's daily work, we have introduced a *task plug-in approach into desktop applications* which integrate task management functionality into desktop applications that the KWer uses in her daily work.

The desktop *application task plug-ins* shall enable the handling of tasks in the context of a desktop application, including its corresponding information and information objects, in which the KWer already implicitly handles tasks or would like to handle tasks but the application doesn't allow it. The goal is that a KWer can perform task management using their well-known environment of existing applications.

First, to enable systematic task management across application boundaries, the desktop application plug-ins transfer as much information as possible from information objects to tasks. The transferred information requires the KWer to supplement minimal additional information. Ideally the task creation should work without any additional effort for the KWer. For example, the email plug-in extracts some email information and fills task properties accordingly, for example the subject of the email results in the task name and selected email content results in a task description. The KWer can additionally provide other information, e.g., a due date, but is not required to do so for creating a task.

Second, to reduce interruptions, task information is displayed within the context of the respective PIM application and the information objects therein. The KWer can also interact with the displayed task information. For example, a KWer is able to see in the email inbox whether a task is attached to an email and what information it carries.

Optionally these task plug-ins can be *combined with a structured personal task management application*, e.g., in the form of a sidebar like the Kasimir application, see section 5. In combination with the plug-ins, the personal task management sidebar application enables efficient handling of existing tasks and provides detailed task information. It shows tasks that emanate from different information objects, i.e. it provides a consolidated overview on all available tasks on the desktop. This is needed since it does not make sense to overload the plug-ins with task management functionality that the KWers would not need in their particular work situation. The task management sidebar closely interacts with the plug-ins, i.e., the plug-ins can control the task management sidebar remotely. For example, when a KWer reads an email related to a task, the task management sidebar shows the details of the particular task.

In the following we present the implementation of task plug-ins for *email clients, calendar and internet browser* applications.

6.1 Email Client Task Plug-In – Link Tasks With Emails

The email client task plug-in, called SAP Task Plug-in for email, integrates itself into the KWer's email client and has two main functionalities. First, it brings tasks into the email application's context, i.e., the KWer can *create a task from within an email client* based on an email. Second, it connects emails with tasks, i.e., the SAP Task Plug-in presents *task information within the context of the email client* and the emails therein. We developed the task plug-in for the email client Microsoft Outlook.

6.1.1 Connect Emails with Tasks

First, the KWer can *create a task from within an email client* based on an email. In case that the KWer finds out - while working with an email - that this email contains a task or represents a task for her, the KWer can directly create a task representation from within the email client using the SAP Task plug-in. In the course of this task creation process, the email is linked as information object to the task.

The SAP task plug-in offers two ways to create a task representation from within the email client. On the one hand, the KWer can *mark an email as task* by using the in-built categorization mechanism of Microsoft Outlook for emails, appointments and contacts, see Figure 51. Within the email inbox the KWer can use the particular color category purple, i.e., "Kasimir Task", to flag the email. Instead of the email inbox any other overview perspective of an email folder within the email client is possible. This flag can be quickly set using a right mouse click on the email. Using this category the SAP Task plug-in creates a task representation with the email information. The task created receives the name of the email's subject field and the task's involved people are the people in the email's sender and CC field.

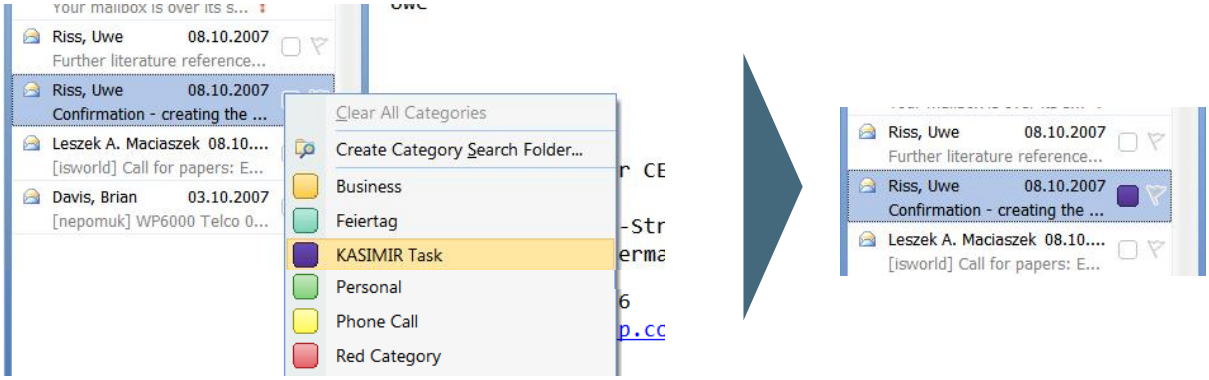


Figure 51: Mark an email as task using the Microsoft Outlook color categories.

On the other hand, the KWer can *press a "New Task" button* within an email window to create a task representation from the email which the KWer just reads, see Figure 52. As well the KWer can invoke the same function by right-clicking on the email in the email inbox, or any other email folder within the email client, and select "New Task" from the email's context menu.

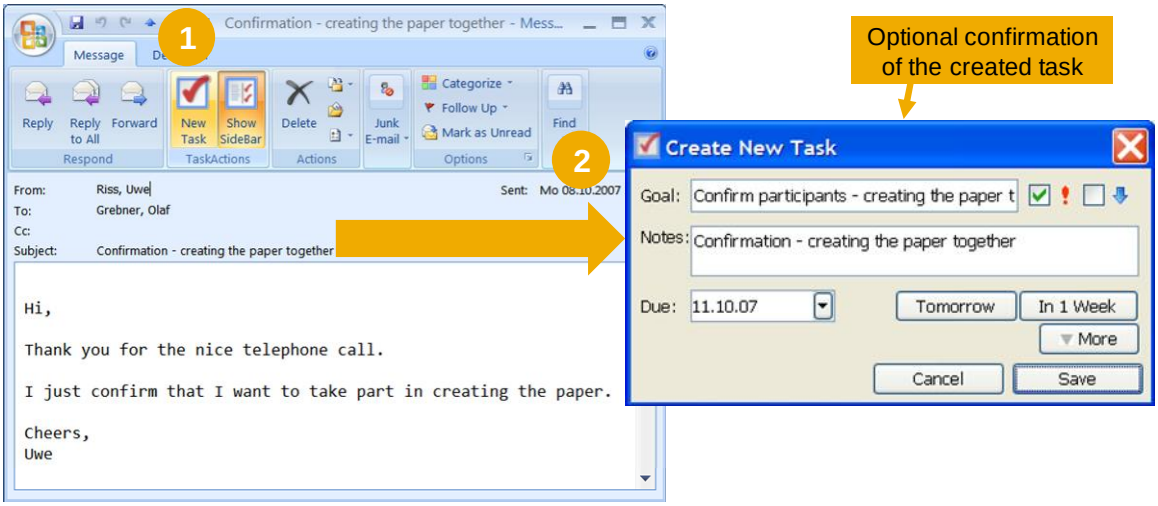


Figure 52: Task creation from an email.

In both cases the upcoming 'New Task' dialog window, see Figure 52, shows the information of the task to-be created to enable the KWer to confirm this information or potentially modify and extend the task information. The popping up of this confirmation dialog is optional and can be turned off by the KWer in the SAP Task Plug-in's settings menu in Microsoft Outlook. The 'New Task' dialog window is the same window as the Kasimir application uses, see section 5.2.1.

The SAP Task Plug-in uses the email information to *pre-populate the task with information*. For example, the email sender is added per default as a task observer, i.e., as a person interested in the task results, people on CC are added as involved people to the task and the subject of the email leads to the task name.

In addition to re-using the email, the SAP Task Plug-in *attaches an email reference to the task* to preserve the context of the task. This enables task applications to show a reference to the email with the task. SAP Task Plug-in first identifies the email from which the task has been created and attaches this email as information object to the newly created task.

KWers can *add further information to the task* to make the task presentation more precise. This step is optional and the KWer is not required to do so for creating a task. Ideally the task creation works without any additional effort by the KWer. When the KWer wants, she can provide additional information, e.g., add a due date, specify a task description or add tags to the task. The KWer thus can expand this window and fully edit all needed task attributes.

The KWer can *attach an email to an existing task* in case when the KWer recognizes that the email at hand contributes to an already existing task. As alternative to creating a new task, the KWer can attach an email to a task by for example right-clicking on the email and selecting the 'Associate with selected Task' action from the upcoming context menu. The KWer then selects in the upcoming dialog window the task to which the email should be attached to. This dialog window belongs to the task management application, i.e., Kasimir.

6.1.2 Bring Tasks into the Email Application's Context

Second, the SAP Task Plug-in presents *task information within the context of the email client* and the emails therein.

The SAP Task Plug-in *displays a task indicator for emails* in the inbox folder within Microsoft Outlook, see Figure 53 on the right side. With the indicator, represented by the colored category, the KWer can quickly get an overview whether an email needs to be followed up as a task or not.

The SAP Task Plug-in *lists attached tasks and basic task information within the email window* for each email with at least one attached task, see Figure 53 on the right. This provides a task context for each email and enables the KWer to see at a glance which tasks are related to this email. The basic task information shown within the email window consists of the task name, the task description, its due date and the task state, e.g., new or complete and enables the KWer to quickly oversee what state the tasks are in and what due date they have.

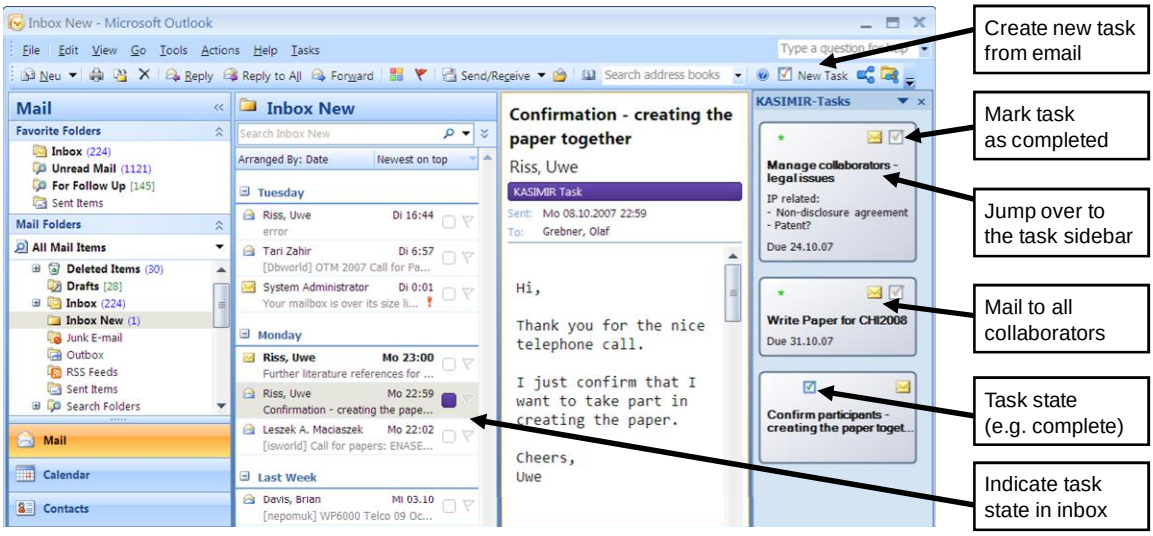


Figure 53: Task integration with email.

The KWer can decide based on this information whether she needs further task-related information or wants to take action. For this, the KWer can *interact with the displayed task information*. Looking at one of the possibly multiple attached tasks in an email, the KWer can trigger actions on the particular task, see Figure 54.

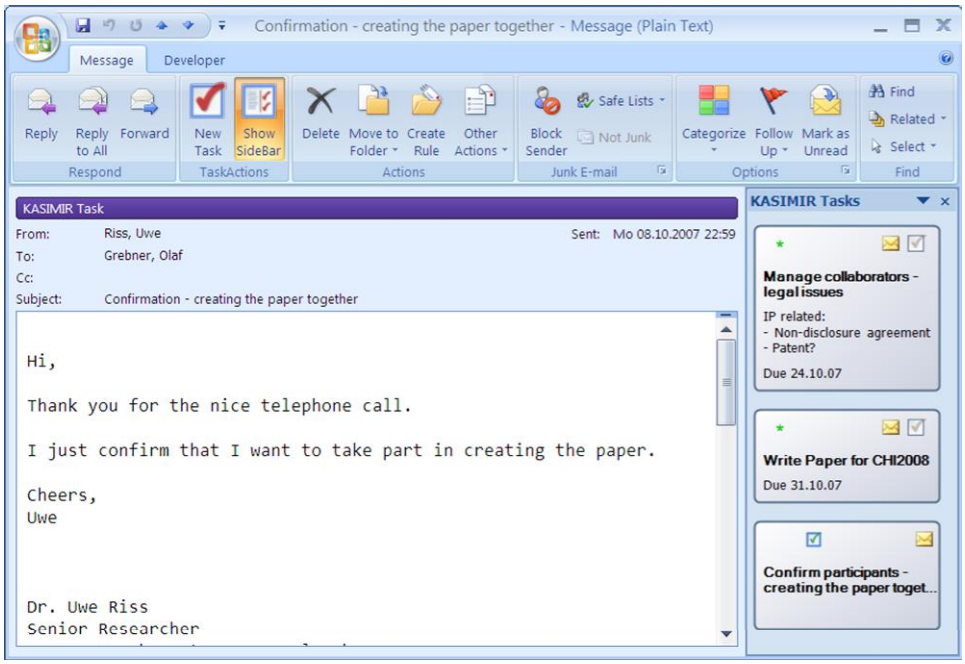


Figure 54: Task integration with email – email detail view.

The KWer can *set the task state*, e.g., the KWer can mark a task as completed when the KWer is finished with answering the email.

Furthermore, the KWer can *mail to all collaborators directly* from within the email application by clicking on the email icon of an attached task. This opens a window for sending a new email. This new email is already pre-filled with information obtained from the task. The email receiver is filled with the email addresses of the involved people in the task. The email's subject contains the task's name. The task description is filled into the email body. The KWer can modify the email message if needed and otherwise only presses a button to directly send this email off to the involved people.

Finally, by clicking on the task title, the KWer can *switch to the task into the task management application*, i.e., the Kasimir task sidebar. This enables the KWer to use the personal task management functionality. This includes a task overview which shows the task in the context of other tasks, i.e., the KWer can see in which subtasks the task is broken down and to what super-tasks the task contributes to. The KWer can inform herself about other information related to the task like, e.g., files, websites and emails and access them. Furthermore, the KWer can see from a social perspective who else is involved in the task and can use support functions to collaborate with the other involved people.

6.2 Calendar application Task Plug-In – Link tasks with appointments

The calendar application task plug-in, called SAP Task Plug-in for calendar, integrates itself into the KWer's calendar application and has two main functionalities. First, it connects appointments with tasks, i.e., the KWer can create a task from within a calendar application based on appointments. Second, it brings tasks into the calendar application's context, i.e., the SAP Task Plug-in presents task information within the context of the calendar application and the appointments therein. We developed the task plug-in for the calendar application Microsoft Outlook.

The functionality of the here presented Task Plug-in for the calendar application in Microsoft Outlook is very similar to the already above presented SAP Task Plug-in for emails in Microsoft Outlook as Microsoft Outlook handles emails and appointments with the same internal representation and in an integrated user interface. However, a task plug-in for other calendar applications would feature similar functionality embedded into the respective calendar application.

6.2.1 Connect Appointments with Tasks

Several tasks can arise when a KWer looks at an appointment. For example, the KWer wants to prepare a business meeting, needs to wrap up the results of a meeting or wants to search for further possible participants. As well, the meeting may contribute to an ongoing task about for example writing a report, preparing any other deliverable or a series of presentations about a particular topic.

First, the KWer can *create a task from within a calendar application* based on appointments. In case that the KWer finds out while reading through the appointments of the calendar that one appointment contains a task or represents a task for her, the KWer can directly create a task representation from within the calendar application using the SAP Task plug-in. In the course of this task creation process, the appointment is linked as information object to the task.

The KWer can *mark an appointment as task* by pressing a "New Task" button within the calendar application window to create a task representation from the appointment which the KWer just points to, see Figure 55. As well, the KWer can trigger the same functionality by right-clicking on the appointment and select "New Task" from the appointment's context menu, see as well Figure 55. Furthermore, the KWer can invoke the same function from within an appointment's detail view window by pressing the "New Task" button there, see Figure 56.

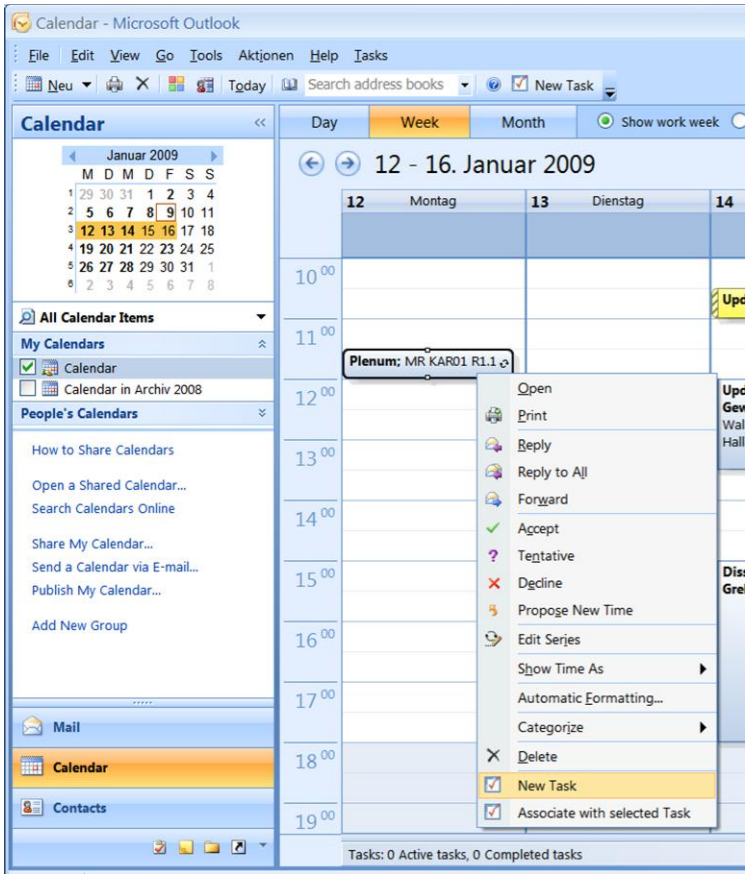


Figure 55: Task creation from a calendar appointment – within calendar overview.

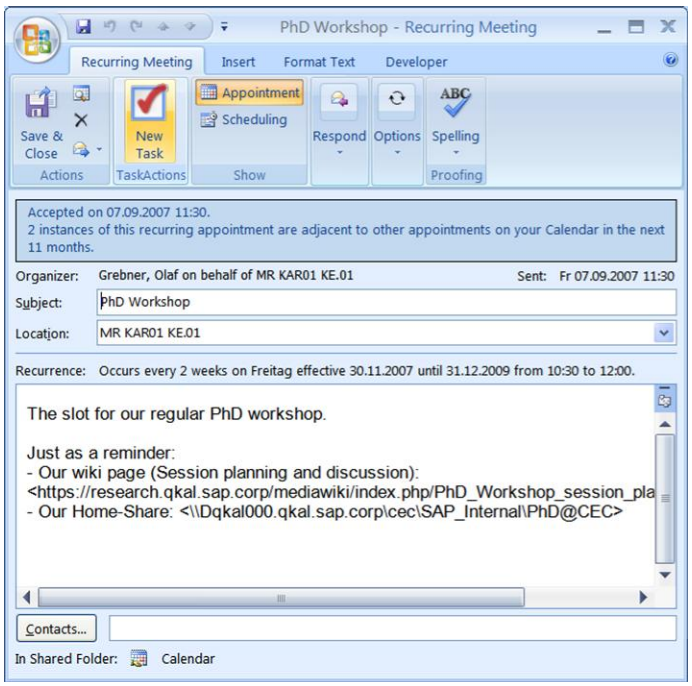


Figure 56: Task creation from a calendar appointment – within appointment detail view.

The upcoming 'New Task' dialog window, the same dialog window as for the SAP Task Plug-in for emails, shows the information of the task to-be created to enable the KWer to confirm this information or potentially modify and extend the task information. Again, the popping up of this confirmation dialog is optional and can be turned off by the KWer in the SAP Task Plug-in's settings

menu in the web browser. The 'New Task' dialog window is again the same window as the Kasimir application uses, see section 5.2.1.

The SAP Task Plug-in uses the appointment information to *pre-populate the task with information*. The appointment name is used as task name and the appointment's end date is used as due date for the task. As well, the appointment's description fills the task description.

In addition to re-using the appointment information, the SAP Task Plug-in *attaches an appointment reference to the task* to preserve the context of the task. This enables task applications to show a reference to the appointment with the task. The SAP Task Plug-in first identifies the appointment from which the task has been created and attaches this appointment as information object to the newly created task.

Again, KWers can *add further information to the task* to make the task presentation more precise. This step is again optional and the KWer is not required to do so for creating a task.

The KWer can *attach as well an appointment to an existing task* when the KWer recognizes that the appointment at hand contributes to an already existing task. As alternative to creating a new task, the KWer can attach an appointment to a task by right-clicking on the appointment and selecting the 'Associate with selected Task' action from the upcoming context menu. The KWer then selects in the upcoming dialog window the task to which the appointment should be attached to. This dialog window belongs to the task management application, i.e., Kasimir.

6.2.2 Bring Tasks into the Calendar Application's Context

Second, the SAP Task Plug-in presents *task information within the context of the calendar application* and the appointments therein. Tasks associated with appointments are shown when the appointment is displayed in the calendar application.

The SAP Task Plug-in *lists attached tasks and basic task information within the appointment window* for each appointment with at least one attached task, similar to the SAP Task Plug-in for emails in section 6.1. This provides a task context for each appointment and when looking at an appointment the KWer is able to see at a glance which tasks are related to it. The basic task information shown within the appointment window is identical to the other SAP Task Plug-ins and consists of the task name, the task description, its due date and the task state, e.g., new or complete and enables the KWer to quickly oversee what state the tasks are in and what due date they have.

The KWer can decide based on this information whether she needs further task-related information or wants to take action. For this, the KWer can *interact with the displayed task information*. Looking at one of the possibly multiple attached tasks in an appointment the KWer can trigger actions on the particular task. This again includes *setting the task state, mailing emails to all collaborators directly* from within the web browser application and the *switch to the task into the task management application* by clicking on the task title.

6.3 Web Browser Task Plug-In – Link Tasks With Websites

Internet browsers are an almost ubiquitous tool for KWers today. The importance for web browsers in the context of tasks even increases as many applications serving private and business purposes are web-based and thus live in a web browser. This includes for example enterprise intranets and document management systems.

The web browser task plug-in, called SAP Task Plug-in for browser, integrates itself into the KWer's web browser and has two main functionalities. First, it connects websites with tasks, i.e., the KWer can create a task from within a web browser based on a website. Second, it brings tasks into the web browser's context, i.e., the SAP Task Plug-in presents task information within the context of the web browser and the websites therein. We developed the task plug-in for the web browser Mozilla Firefox. Its functionality is very similar to the already above presented SAP Task Plug-ins for Microsoft Outlook.

6.3.1 Connect Websites with Tasks

Several tasks can arise when a KWer browses through a website. For example, when a KWer finds some information on a website that she wants to further consider, she can create a task at this point. Other examples are when a KWer wants to read further resources on a topic or wants to find interesting information to share with a colleague in the context of a common task.

First, the KWer can *create a task from within a web browser* based on a website. In case that the KWer finds out while browsing a website that it contains a task or represents a task for her, the KWer can directly create a task representation from within the web browser using the SAP Task plug-in. In the course of this task creation process, the website is linked as information object to the task[10].

The KWer can *mark a website as task* by pressing a "New Task" button within the browser window to create a task representation from the website which the KWer just reads, see Figure 57. As well the KWer can invoke the same function by right-clicking somewhere within the website and select "New Task" from the website's context menu.

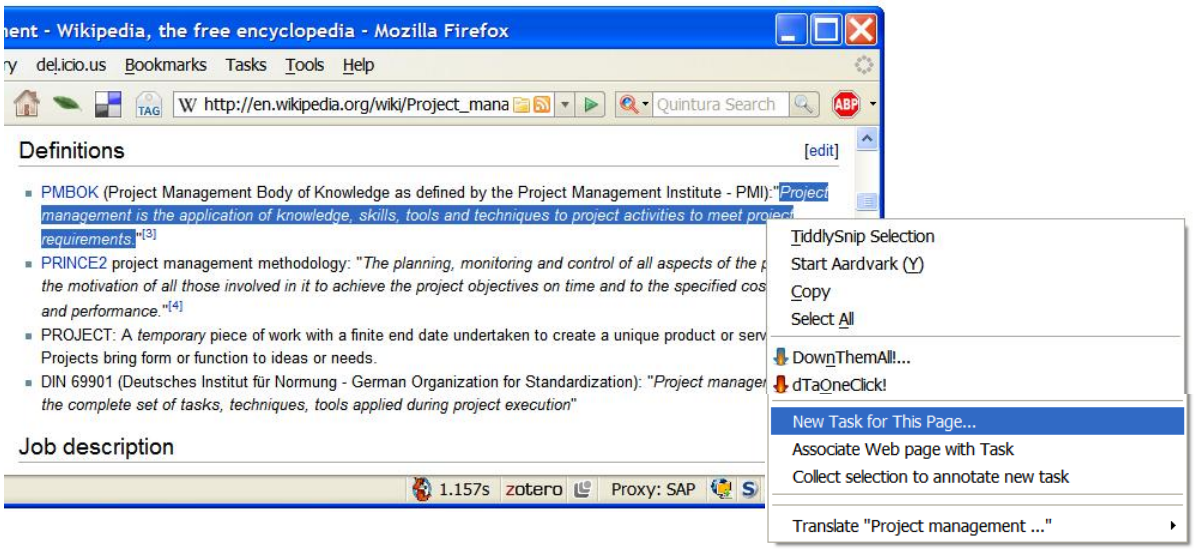


Figure 57: Task creation from a website – part one – trigger task creation.

The upcoming 'New Task' dialog window, see Figure 58, shows the information of the task to-be created to enable the KWer to confirm this information or potentially modify and extend the task information. Again, the popping up of this confirmation dialog is optional and can be turned off by the KWer in the SAP Task Plug-in's settings menu in the web browser. The 'New Task' dialog window is again the same window as the Kasimir application uses, see section 5.2.1.

[10] The current implementation references the whole website, even when the KWer marked only a part of the website, like shown in Figure 57.

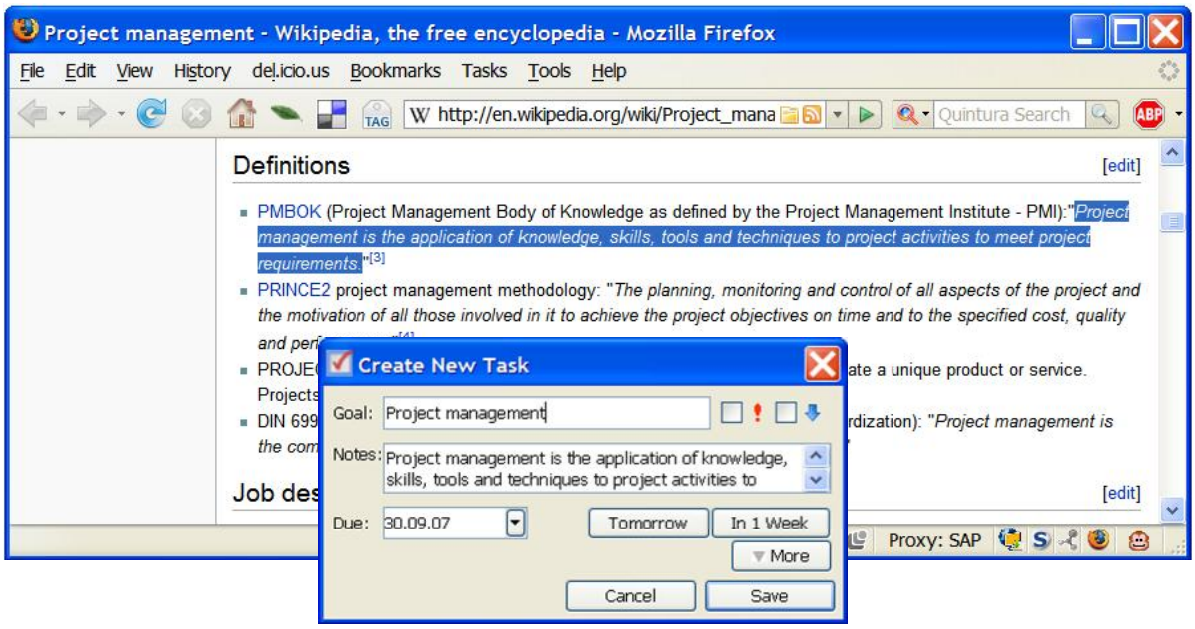


Figure 58: Task creation from a website – part two – optional confirmation.

The SAP Task Plug-in uses the website information to *pre-populate the task with information*. For example, when the KWer selected some text within the website before invoking the task creation process, this text is used as task description. Furthermore, the SAP Task Plug-in retrieves potential tags for the task from a social bookmarking service, i.e., del.icio.us [Yahoo! Inc., 2009], based on the website's URL[11].

In addition to re-using the website information, the SAP Task Plug-in *attaches a website reference to the task* to preserve the context of the task. This enables task applications to show a reference to the website with the task. SAP Task Plug-in identifies the website by its URL and attaches the website as information object to the newly created task.

Again, KWers can *add further information to the task* to make the task presentation more precise. This step is again optional and the KWer is not required to do so for creating a task.

The KWer can *attach as well a website to an existing task* in case when the KWer recognizes that the website at hand contributes to an already existing task. As alternative to creating a new task, the KWer can attach a website to a task by right-clicking on the website and selecting the 'Associate with selected Task' action from the upcoming context menu. The KWer then selects in the upcoming dialog window the task to which the website should be attached to. This dialog window belongs to the task management application, i.e., Kasimir.

6.3.2 Bring Tasks into the Web Browser's Context

Second, the SAP Task Plug-in presents *task information within the context of the web browser* and the websites therein. Tasks associated with websites are shown when the website is displayed in the web browser.

The SAP Task Plug-in *lists attached tasks and basic task information within the website window* for each website with at least one attached task, see Figure 59. This provides a task context for each

[11] Del.icio.us is a social bookmarking service. It allows KWers to assign tags to websites, i.e., create a bookmark, and to share these bookmarks with other KWers.

website and when browsing to a website the KWer is able to see at a glance which tasks are related to this website. The basic task information shown within the website window is identical to the other SAP Task Plug-ins and consists of the task name, the task description, its due date and the task state, e.g., new or complete and enables the KWer to quickly oversee what state the tasks are in and what due date they have.

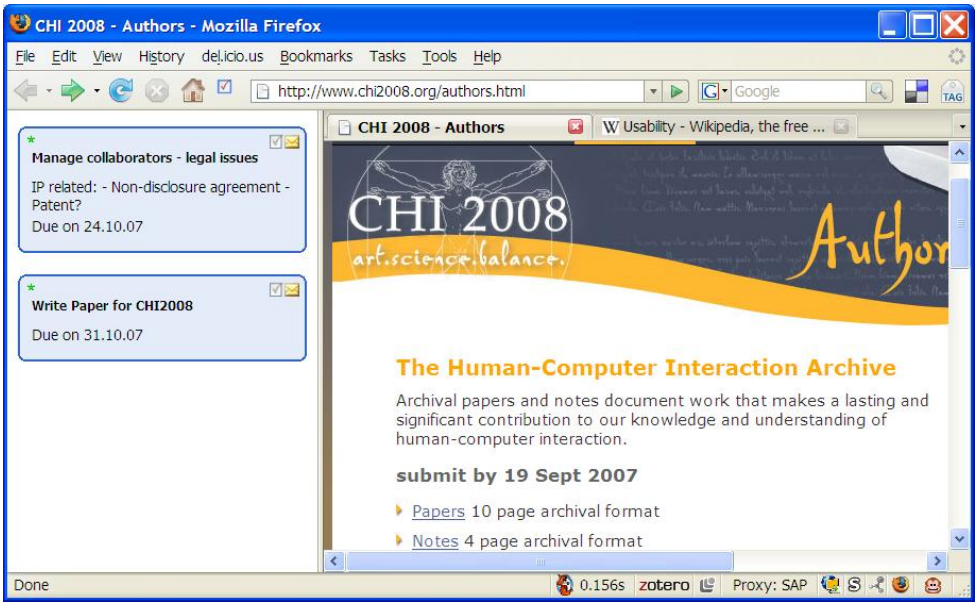


Figure 59: Tasks shown in a web browser in the context of a website.

The KWer can decide based on this information whether she needs further task-related information or wants to take action. For this, the KWer can *interact with the displayed task information*. Looking at one of the possibly multiple attached tasks in a website, the KWer can trigger actions on the particular task, see Figure 59. This again includes *setting the task state, mailing emails to all collaborators directly* from within the web browser application and the *switch to the task into the task management application* by clicking on the task title.

6.4 Architecture and Implementation

The SAP Task application plug-ins support a KWer's personal task management activities in a workspace-integrated way. They bring small pieces of application logic supporting a KWer's personal activities into existing applications. The implementation of the user interaction of each SAP Task application plug-ins depends on the application the application plug-in runs in. Contrary, at its core all SAP Task application plug-ins invoke the Task Management Service to obtain the personal task management functionality and access the personal unified information model, see section 10.1. Figure 60 shows the principle of the SAP Task application plug-ins invoking the Task Management Service.

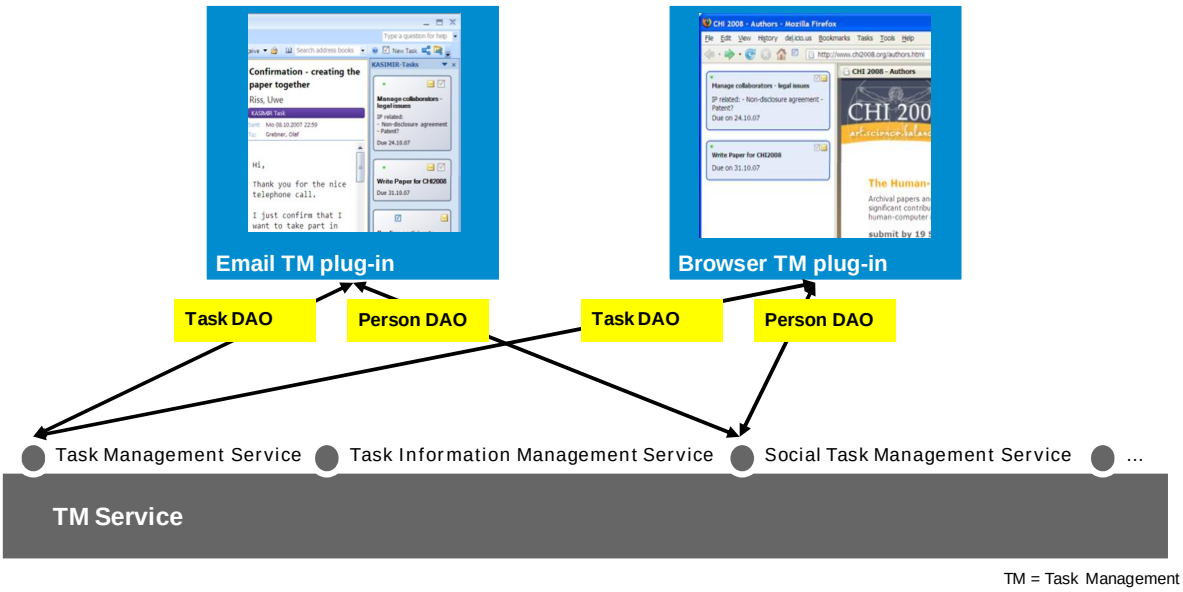


Figure 60: Kasimir implements the Task Management Service for personal task management functionality.

The application plug-ins are implemented in the technology needed for each of the respective target applications. The SAP Task Plug-in for the Web browser Firefox is implemented as Firefox Extension [Glos, 2005] in XML User Interface Language (XUL) [Mozilla Foundation, 2009a]. The SAP Task Plug-in for the email client and calendar application Microsoft Outlook is implemented using the Visual Studio Tools for Microsoft Office [Microsoft Corporation, 2009g] in C# [Wikimedia Foundation Inc., 2009d].

The SAP Task application plug-ins invoke *task management application features* of the Kasimir personal task management application through the Task Management Service. In particular the SAP Task application plug-ins invoke the Kasimir "add task" dialog window to present this dialog window in their application context. This takes place in a unified way across all participating applications and the KWer is relieved from manually transferring the information across applications, e.g., by copy-and-paste.

6.5 KWer Benefits of Unified Personal Information

The SAP Task application plug-ins enable the KWer to *conduct task management adjusted to the respective personal activity*. The SAP Task application plug-ins thus enable personal task management *across individual applications* throughout the whole workspace. The KWer can flexibly choose the applications needed to execute the work for the particular activity, see Figure 61, while at the same time the task representations in these different application contexts enable structured personal task management support.

For example, when a KWer just wants to quickly record a task to not forget a particular issue within a personal working activity, a lightweight application to capture the task aims to minimize the effort to record this. This leads to the approach of the SAP Task plug-ins to integrate lightweight task management functionality into other application's contexts. Whereas, for example, when a KWer wants to plan the next steps for the day, a dedicated task management application like, e.g., Kasimir helps to focus on this personal task management activity and to efficiently execute the task planning.

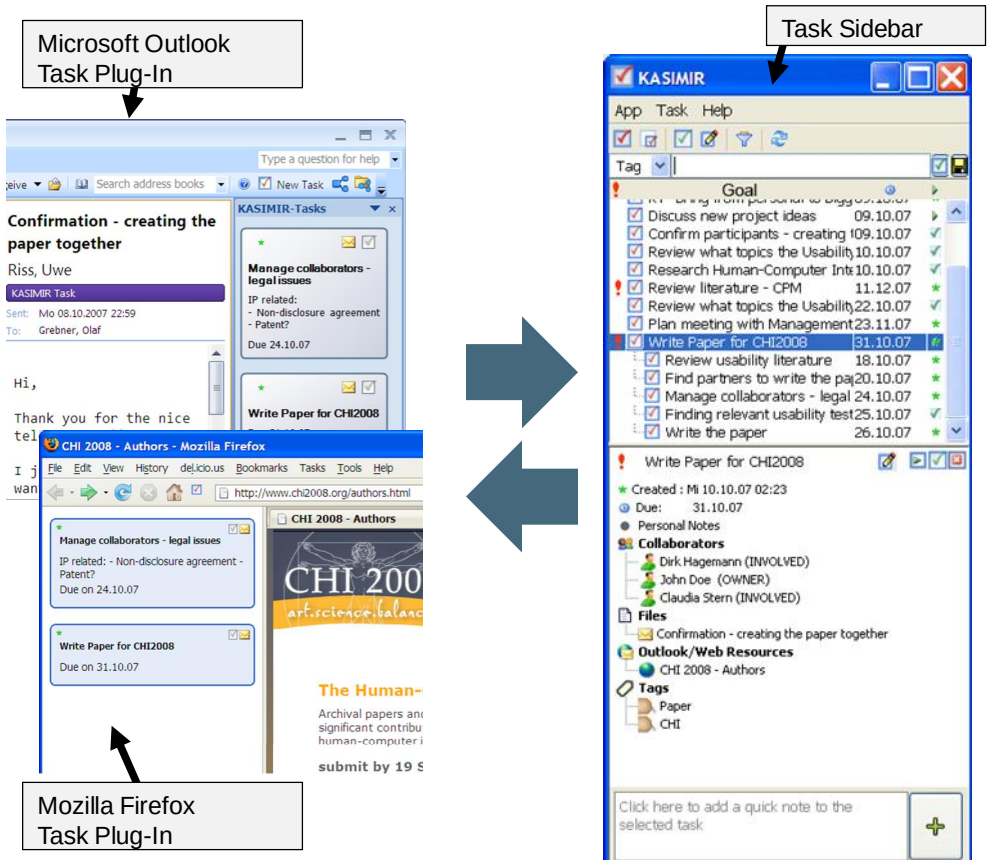


Figure 61: Task shows up in different application contexts, not only in task management application.

The KWer benefits from this approach because the SAP Task application plug-ins operate on the representation of the KWer's personal information cloud. The KWer doesn't need to take care about keeping potentially redundant sets of information. On the one hand, the KWer can contribute information about tasks and their related information to the personal information cloud directly from different application contexts. For example, while browsing the email inbox in the email client application, the KWer just sees one email she needs to follow up, but doesn't want to interrupt her ongoing activity. With the SAP Task application plug-in she can mark this email as task for follow-up and later find this task in all her task management applications, see Figure 61. On the other hand, the KWer can use the information from the personal information cloud within several application contexts. For example, the KWer sees task information created in the task management application with the related email in the email client application inbox that she's just browsing, see Figure 61.

7 Nepomuk Meeting Manager – Personal Meeting Management Application

The Nepomuk Meeting Manager supports the KWer in all facets of managing a meeting. It supports the whole meeting process, i.e., before, during and after the meeting. Before the actual meeting, it helps the KWer to prepare meetings and send invitations. During the meeting it supports taking meeting minutes. After the meeting it supports the gathering of feedback, the finalized protocol is distributed and protocol items are post-processed. The application thereby always supports the personal perspective of a KWer involved in a meeting.

The Nepomuk Meeting Manager aims to integrate all meeting-related information throughout the whole meeting lifecycle, i.e., from a meeting's inception until the completion of resulting action items. Figure 62 shows a simplified overview on the main supported meeting process activities and on the corresponding Nepomuk Meeting Manager functionalities. The focus of Nepomuk Meeting Manager's functionality is the scribing support during meetings.

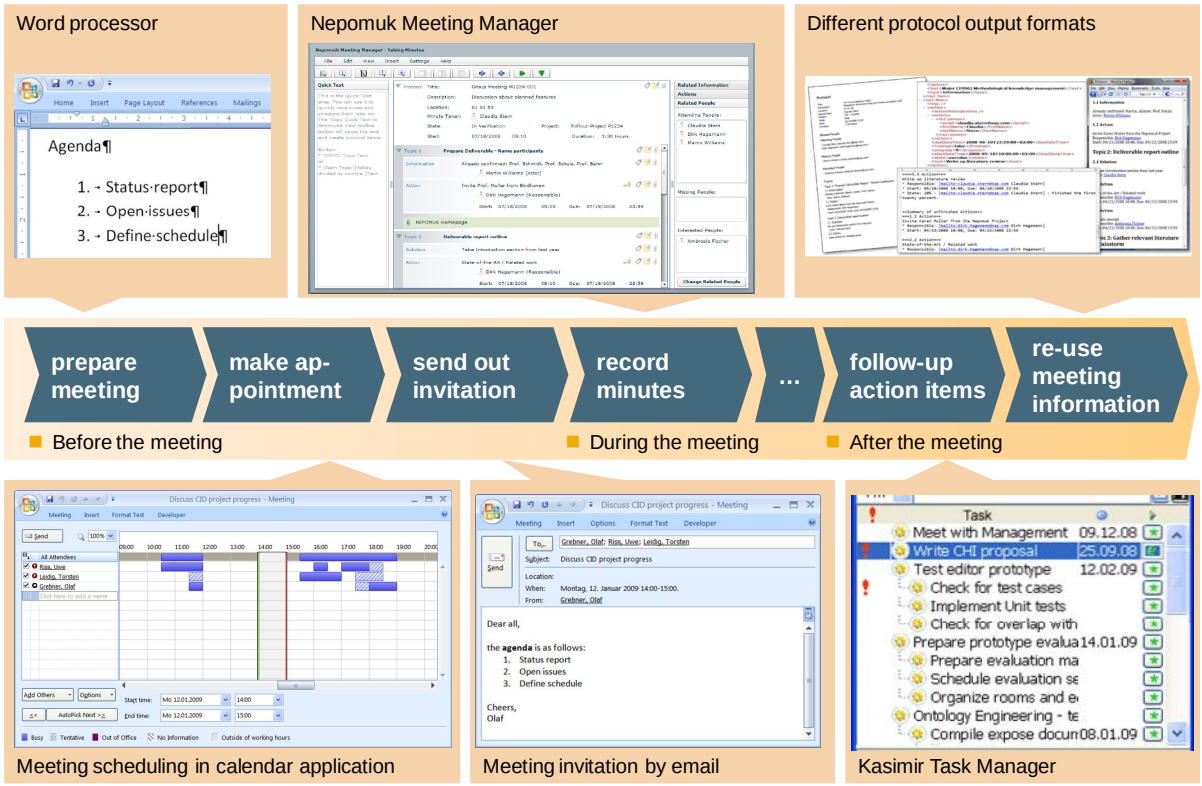


Figure 62: Nepomuk Meeting Manager Main Functions and Integrated Applications.

Thereby, the Nepomuk Meeting Manager uninterruptedly handles meeting information on the desktop, independent of their format and their maintaining application. This significantly reduces the KWer's effort to circumvent and fix such breaks.

To achieve this, the Nepomuk Meeting Manager on the one hand *integrates the meeting representation and related information* into the KWer's personal information cloud. This leads to numerous usages of the same information without investing effort by the KWer. For example, as shown in the upper half of Figure 62, Nepomuk Meeting Manager can export and open the agenda in a word processor and display the meeting protocol in both structured form as well as in different protocol output formats, e.g., as PDF file. On the other hand, Nepomuk Meeting Manager integrates

specialized applications to create one coherent flow of work for the KWer managing a meeting. For example, as shown in the lower half of Figure 62, Nepomuk Meeting Manager integrates a calendar application to schedule a meeting and can send meeting invitations using the KWer's email application. Furthermore, it can enable the KWer to follow-up a meeting's action items in the preferred personal task management application, e.g., the action items defined in a meeting show up directly as tasks in the Kasimir personal task management application.

The Nepomuk Meeting Manager *target user group* are KWers in the role of a meeting organizer or minute taker depending on the phase of the meeting process. It focuses on supporting the 'gather meeting minutes' activity for the minute taker. Thus it is optimized for a person structuring the meeting content, supporting thereby the personal perspective of that person on the meeting. It has no dedicated separate view for collaborative editing and viewing of the meeting content by multiple participants. Such a view would be, for example, to show other meeting participants a read-only variant optimized for viewing the meeting progress and content where the participants can see what is written in the protocol and instantly correct wrong information or add missing information.

7.1 Nepomuk Meeting Manager Functionality Covers the Meeting Process

The Nepomuk Meeting Manager provides support functionality for the activities of whole meeting process. Table 10 gives an overview on the Nepomuk Meeting Manager's support functionality for each meeting-related activity of the meeting process.

Phase	Activities	Nepomuk Meeting Manager Support
Pre-meeting	Write up agenda	Write up the meeting agenda by creating a new meeting representation and entering meeting topics.
	Manage participants and interested parties	Minute taker manages the people related to a meeting, i.e., enter them in attending, missing and interested person lists.
	Collect information for meeting	Collect information for the meeting preparation by associating this information with the meeting representation.
	Make appointment	Make the appointment for the meeting by integrating appointment scheduling functionality of calendar applications.
	Send invitation	Send an invitation message containing the agenda to all participants and interested persons via email.
In-meeting	Retrieve information for presentation	Browse needed meeting information in one place by accessing the meeting representation's associated information.
	Record meeting results	Quickly scribe meeting protocols in a quick text area and conveniently structure them later on using a structured view.
	Exchange information	No dedicated support functionality, browse the information gathered with the meeting representation.
	Record meeting streams	No dedicated support functionality, invokes a specialized application for recording meeting streams.
	Brainstorming	No dedicated support functionality, quickly type in text in the quick text area to capture generated ideas.
	Voting	No dedicated support functionality, invokes a specialized meeting support application to provide voting support.
	Meeting facilitation	No dedicated support functionality, invokes a specialized meeting support application to enable meeting facilitation.
	Give a presentation	No dedicated support functionality, open attached presentation files in a slide presentation application.
Post-meeting	Gather feedback on meeting minutes	Collaboratively add comments about missing or incorrect information to gather feedback on the meeting minutes.

	Finalize meeting minutes	Minute taker finalizes the protocol preventing any further change and export meeting protocol into a number of formats.
	Distribute protocol	Send out the protocol's final version to all participants as an e-mail attachment in a revision safe format like, e.g., PDF.
	Follow-up resulting action items	Provides basic task management functionalities and a task overview to enable a follow-up of meeting action items.
	Re-use meeting (protocol) information	Make meeting information available for external applications, e.g., forward problem items to risk management software.

Table 10: Overview on Nepomuk Meeting Manager's functionality by meeting-related activity.

We outline for four activities, i.e., exchange information, recording meetings streams, brainstorming and voting, how the Nepomuk Meeting Manager can support these activities by integrating specialized applications. There are specialized applications which focus on supporting these activities by providing exactly the missing functionality. This includes for example meeting audio/video capturing applications and workshop support tools, as introduced in section 4.6. These can be integrated into the Nepomuk Meeting Manager by passing relevant information to them and by invoking their functionality from within the Nepomuk Meeting Manager. For example, to use Nepomuk Meeting Manager in distributed meetings for exchanging information during a meeting, Nepomuk Meeting Manager can invoke external web conferencing, chat or instant messaging applications which than can be used in parallel to the Nepomuk Meeting Manager[12]. As well, multimedia applications are not in the focus of the Nepomuk Meeting Manager application, but can be added by integrating the above mentioned specialized applications for, e.g., audio and video streaming.

7.2 Nepomuk Meeting Manager Application Overview

The Nepomuk Meeting Manager application provides a number of functionalities in several dialog windows. The KWer can manage meetings using the application and its functionalities.

The Nepomuk Meeting Manager runs as a web-based rich-client application in a web browser, e.g., Firefox as well as it can run as standalone application on the desktop, see Figure 63.

[12] The Nepomuk Meeting Manager implementation currently integrates with an email client, a calendar application, the Kasimir personal task manager and a meeting facilitation application. The integration with the here outlined recording meetings streams, brainstorming and voting applications is not yet implemented.

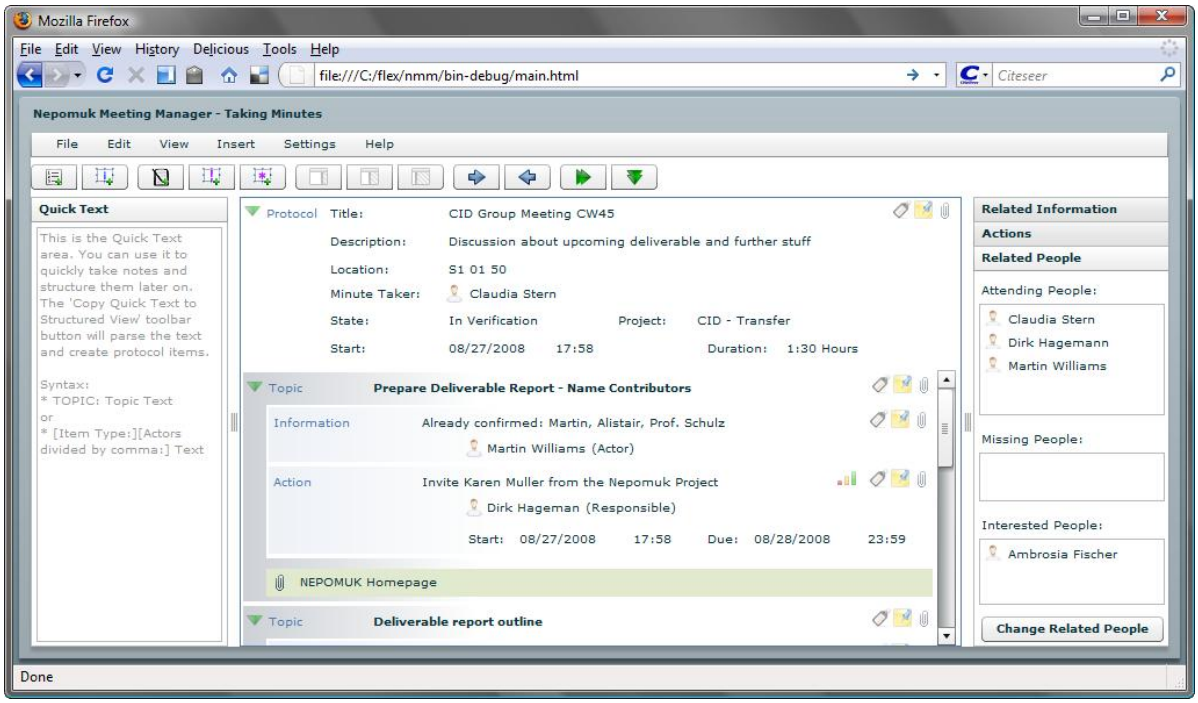


Figure 63: Nepomuk Meeting Manager running in Mozilla Firefox.

7.2.1 Meeting Process Overview Dialog

The meeting process overview dialog displays the most recent protocols categorized by their meeting process phase, see Figure 64. The KWer can choose to set up a new meeting or directly start taking minutes for a new meeting.

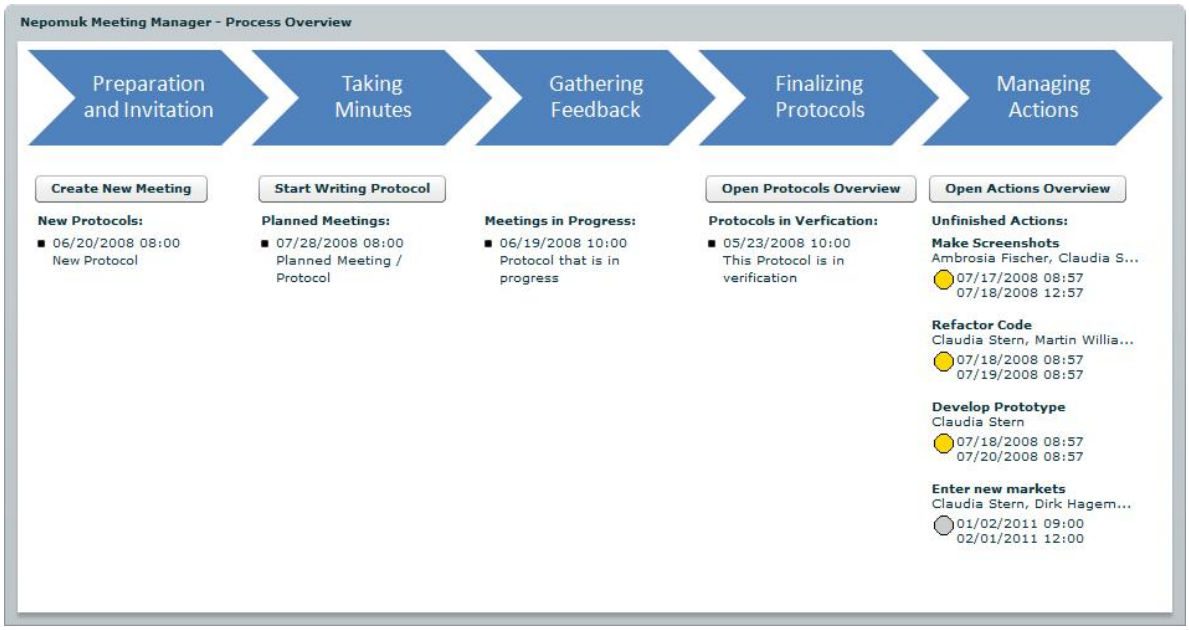


Figure 64: Meeting Process Overview.

The KWer can open existing protocols by clicking on their textual links. Then the meeting protocol edit dialog appears, see section 7.2.3.

By showing the meeting process, the KWer receives guidance to conduct a meeting according to recognized standards. This is visually indicated by moving the meeting over time from left to right through each of the meeting phases according to its progress.

The dialog also allows the KWer to switch to a more detailed overview dialog for protocols and actions to show more protocols in detail or a detailed actions overview.

7.2.2 Meeting Protocol and Task Overview Screen

The overview screen gives a detailed overview on all meeting protocols and actions. It consists of two list-focused tabs.

The first tab contains the meeting protocols list, see Figure 65. It can be filtered by the protocols' state, by period or by people. Possible periods are "Today and Upcoming", "Today", "Last 7 Days" and "All". Filtering by people makes it possible to show only protocols where the KWer is the minute taker or one of the participants.

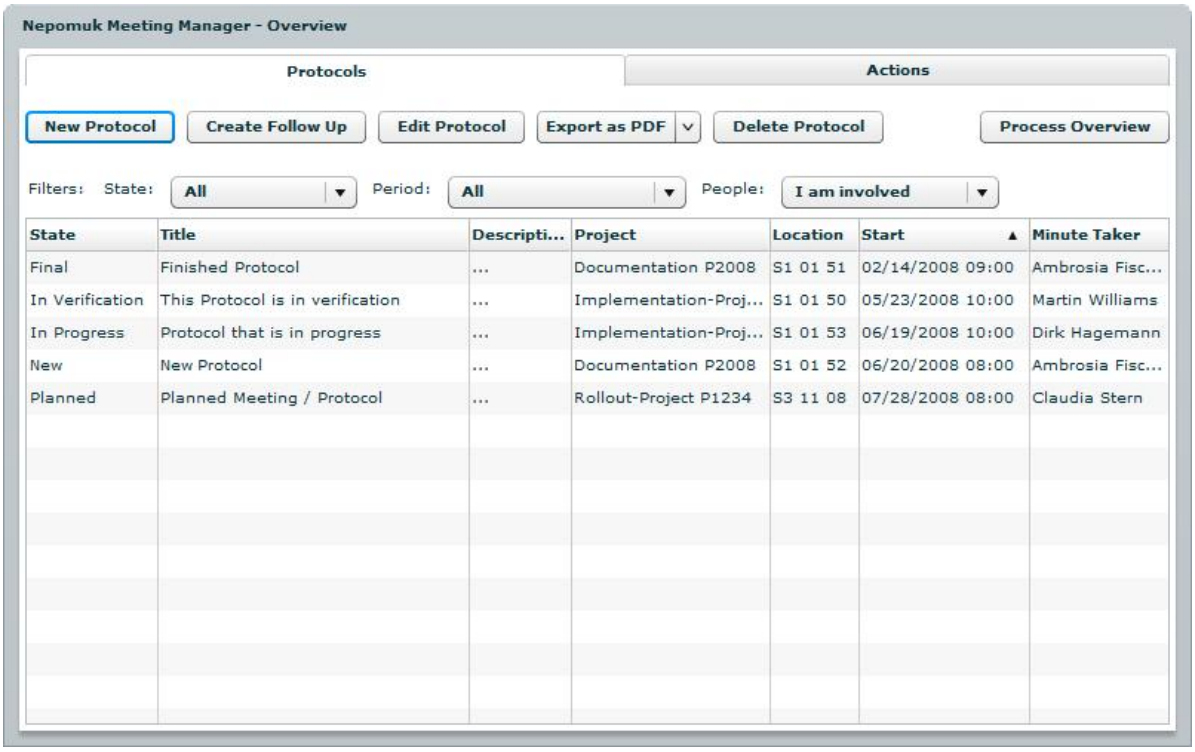


Figure 65: Meeting protocols overview.

The KWer can trigger a set of actions on the meeting protocols, similar to the meeting process overview window. Pressing the 'New protocol' button creates a new protocol and the KWer can start writing minutes. The 'Create Follow-up' button triggers the follow-up functionality for a selected existing protocol. The 'Edit Protocol' opens the meeting protocol edit window. The 'Export as...' button triggers the export functionality for the selected meeting protocols. Several options are available, e.g., an export as PDF, see section 7.4.4.2 for details. The 'Delete Protocol' button triggers the deletion of the selected meeting protocols. With pressing the 'Process Overview' button the application switches back to the meeting process overview window.

On the second tab, the actions list displays all actions which emanated from a meeting, see Figure 66. Sorting the actions by, e.g., the due date is possible by clicking on the respective column headers. It can be filtered by state and people. The state filter allows filtering for single states and for combinations like "Overdue and next 3 days", "All unfinished" and "All". Filtering by people allows to show only actions for which the KWer is responsible.

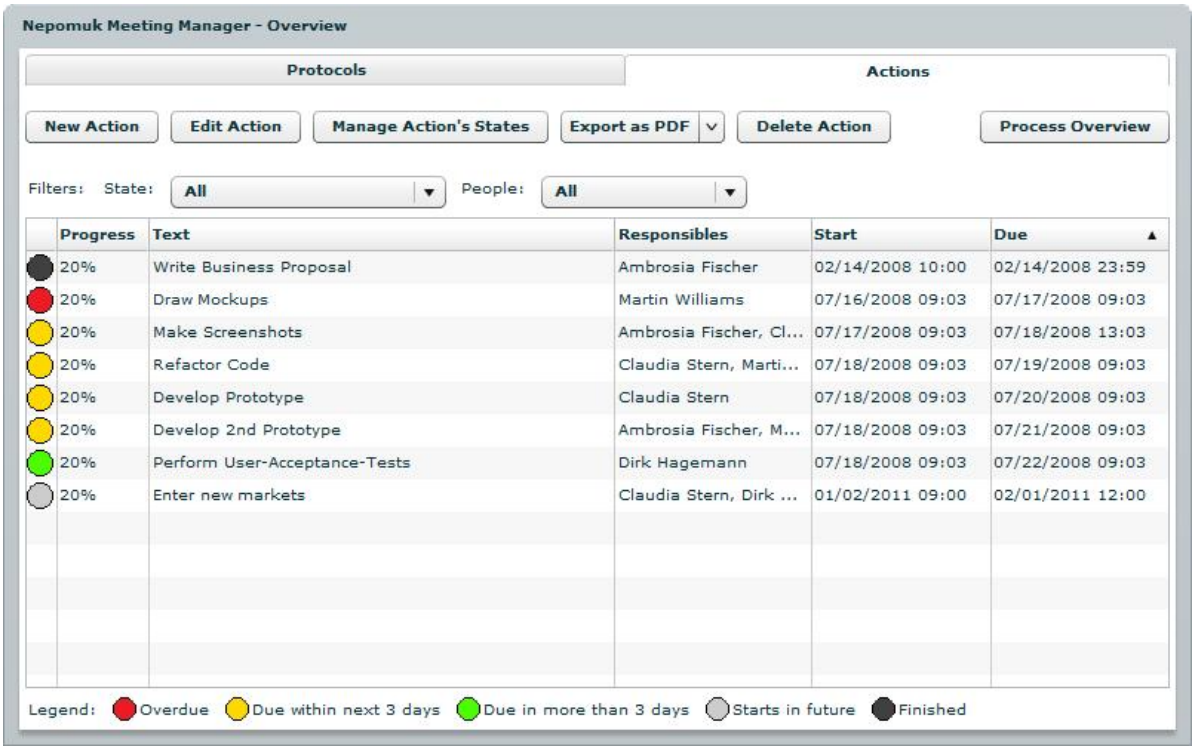


Figure 66: Actions overview.

The KWer can trigger a set of actions in the user interface on each presented actions, similar to the meeting process overview window. Pressing the 'New actions' button creates a new task manually and independent of any protocol, it can later be connected to a protocol, see section 7.2.3.5. The 'Edit Action' enables the KWer to edit the task. The 'Export as…' button triggers the export functionality for the selected tasks. Several options are available, e.g., an export as PDF, see section 7.4.4.2 for details. The 'Delete Action' button triggers the deletion of the selected tasks. With pressing the 'Process Overview' button the application switches back to the meeting process overview window.

Each action shown represents a task which is as well available in personal task management applications, like, e.g., Kasimir.

7.2.3 Edit Protocol Dialog

Protocols can be edited using the edit protocol dialog, see Figure 67. This is the application's main dialog. The dialog is divided into multiple areas.

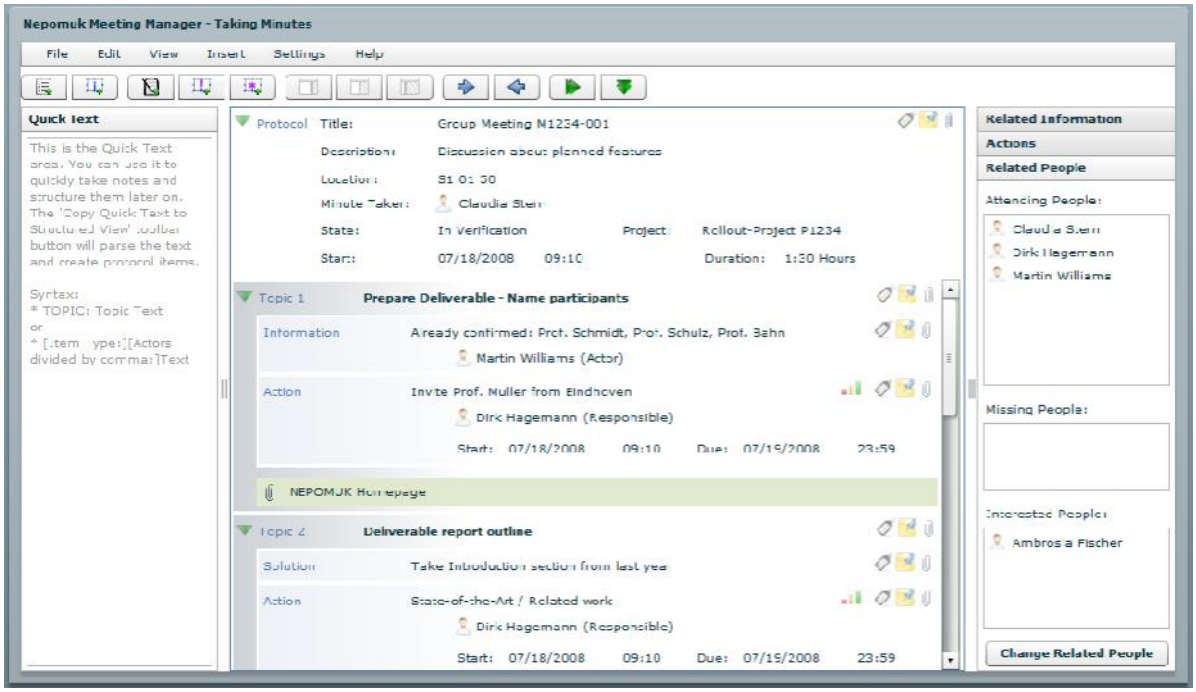


Figure 67: Edit protocol dialog overview.

The top of the dialog consists of the menu and the toolbar. The dialog's rest is divided into the quick text area (at the left), the structured view and the sidebar (at the right). The sidebar itself contains three panels, which can be switched by clicking on the panel's header in the sidebar. The KWer can use the sidebar to display the related information panel, the actions panel or the people panel. The quick text area and the sidebars can be collapsed so that only the structured view is visible to focus on this visualization.

The KWer can use a symbol pane to centrally manage the meeting protocol content, see Figure 68. Its functions are explained below following Figure 68 from left to right.

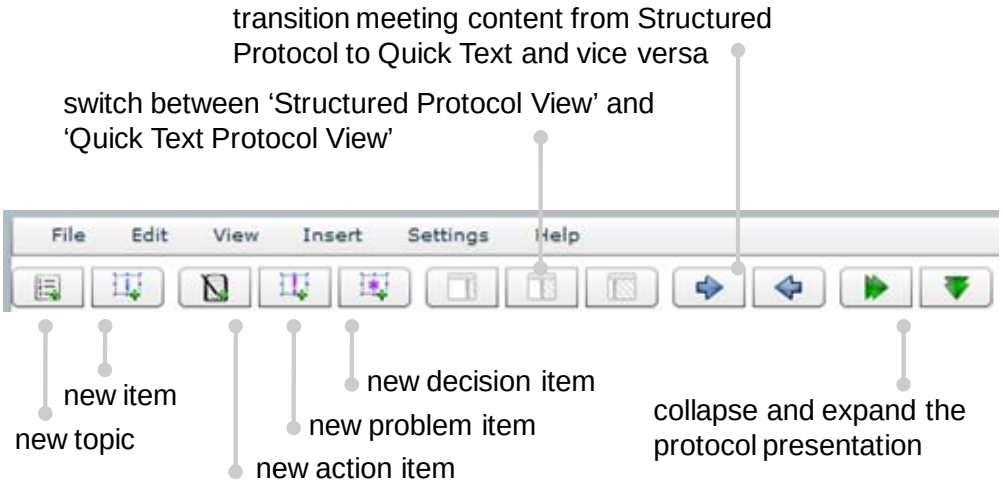


Figure 68: Symbol pane to manage meeting protocol content.

The first two buttons enable the KWer to create a new meeting topic and a new item, which is initialized with the type of an information item. The next three buttons enable the KWer to create specially typed items, i.e., to create new action item, a new problem item and a new decision item. Three buttons enable the KWer to switch the views between the 'Structured Protocol View' and the

'Quick Text Protocol View', see sections 7.2.3.1 and 7.2.3.2 respectively. The next two buttons enable the KWer to transition meeting content from the Structured Protocol to the Quick Text and vice versa, see section 7.2.3.3. The last two buttons enable the KWer to collapse and expand the protocol presentation, i.e., to show and hide protocol details. Figure 69 shows a meeting topic with collapsed details and Figure 70 shows the same meeting topic with expanded details.

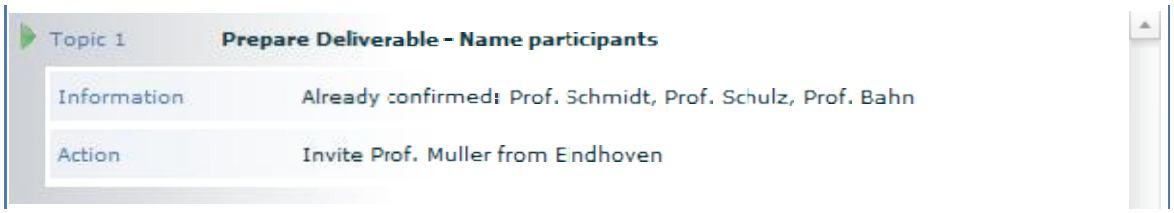


Figure 69: Meeting topic with collapsed details.

Figure 70: Meeting topic with expanded details.

7.2.3.1 Structured Protocol View – Organize Structured Protocols

The structured view is located in the center of the application's main screen where it shows the protocol in a graphically structured form. At the top of the area, there is the protocol header, which displays general information about the meeting and the protocol. Below the header, the list of topics is shown. Every topic can contain items and some items can be typed as action items. Figure 71 shows the structured view.

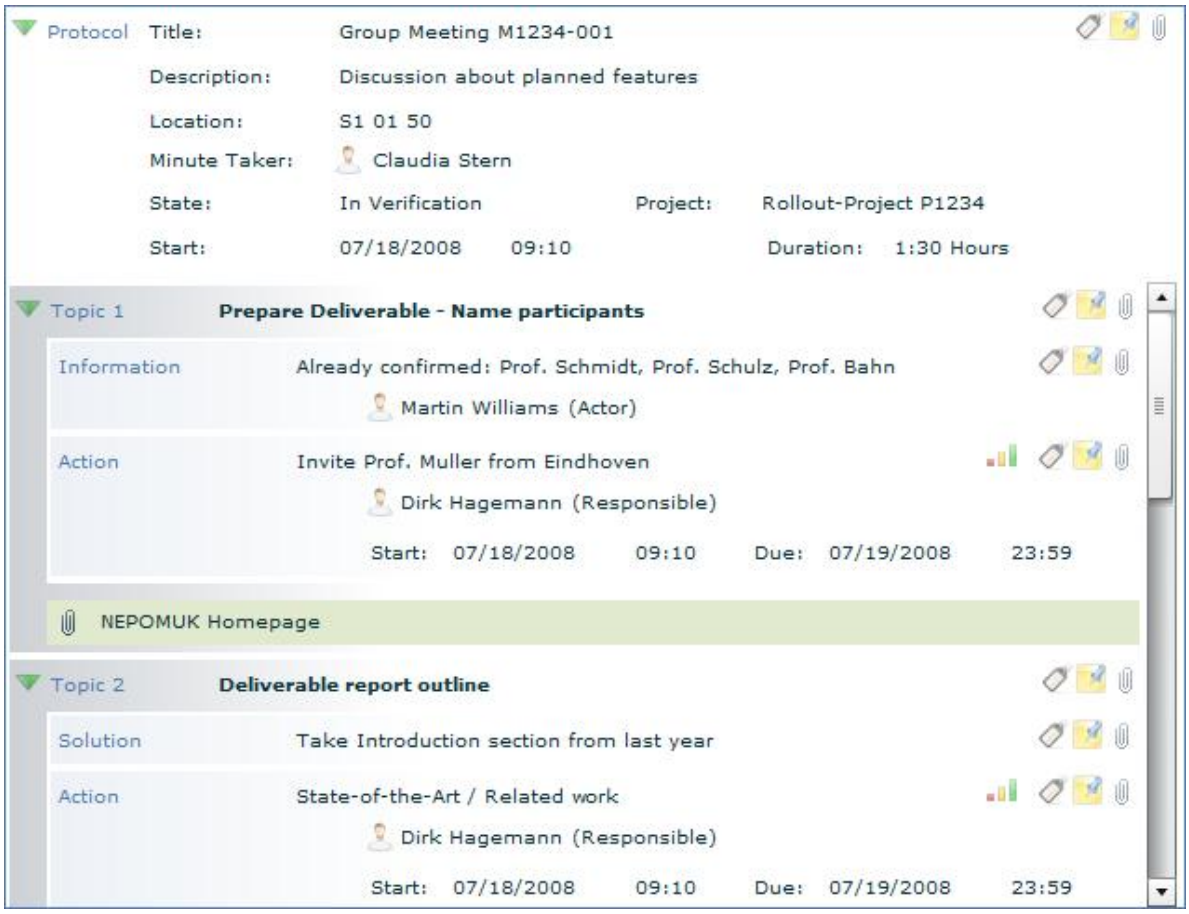


Figure 71: Overview on meeting protocol in structured protocol view.

The KWer can *refine the meeting protocol structure* without investing much effort by re-ordering information items using drag & drop operations. The KWer selects, e.g., a particular topic with a left click and moves this topic to the new position in the protocol. As well, the KWer can move information items within a topic or to them across topics. Additionally, the KWer can transform a topic into an information item when the KWer drops a topic into another topic. Vice versa, when the KWer drops an information item in-between two topics the information item is transformed into a topic and placed there.

7.2.3.2 Quick Text Protocol View – Jot down Plaintext Protocols

The KWer can quickly jot meeting minutes in plaintext in the 'Quick Text Protocol View'. It displays the protocol as editable plaintext, see Figure 72, while the 'Structured Protocol View' represents the graphical structure of the protocol. The KWer can make changes to the text and update the structured view and vice versa, see section 7.2.3.3.

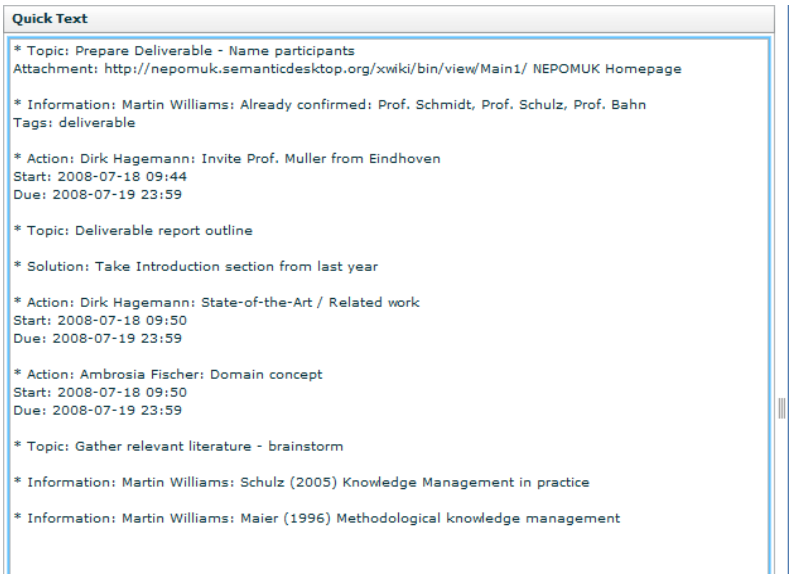


Figure 72: Overview on meeting protocol in quick text protocol view.

The quick text format represents a lightweight wiki-like syntax. This enables the Nepomuk Meeting Manager application to understand the plaintext entry of the KWer. It uses an asterisk as the first character in a new line of text to indicate that a new topic or item starts. The asterisk can be followed by the item type ("Topic" or one of the item types) and a colon. If no type is given the standard item type ("Information") is used. Additionally, it is possible to set the actors for an item by putting their names separated by commas and followed by a colon. All the text before the next asterisk, which starts a new line, is used as the item's text. Each topic and item can be annotated with tags, comments and attachments. Lines starting with "Tag:" or "Tags:" will tag the object with the text that follows. Comments begin with "Comment:". It is possible to enter a comment's author by adding a person and a colon. The rest of the line becomes the comment's text. Attachments are specified by "Attachment:" followed by the uniform resource identifier (URI) (a URI is not allowed to contain spaces) a space and the attachment's label.

The following list shows the quick text format:

* Topic: Topic Text

* [Item Type:]Actors divided by comma:] Text

Tags: tag1 tag2 tag3

Comment: [Comment Author:] Text

Attachment: URI Label

Whenever people have to be specified it is possible to enter unique first or last name or "first name last name" or their e-mail address. An example of quick text and the corresponding structured view can be seen in Figure 73.

7.2.3.3 Transition between Plaintext Minutes and Structured Meeting Content

To enable the KWer to *both quickly scribe and conveniently structure the meeting protocol*, the Nepomuk Meeting Manager "translates" plaintext following a lightweight syntax into a structured, visual representation and vice versa.

The Nepomuk Meeting Manager can target two different usage scenarios, each with a specialized view and respectively associated functionality. This is based on the same meeting minutes by synchronizing both representations through this translation step, see Figure 73.

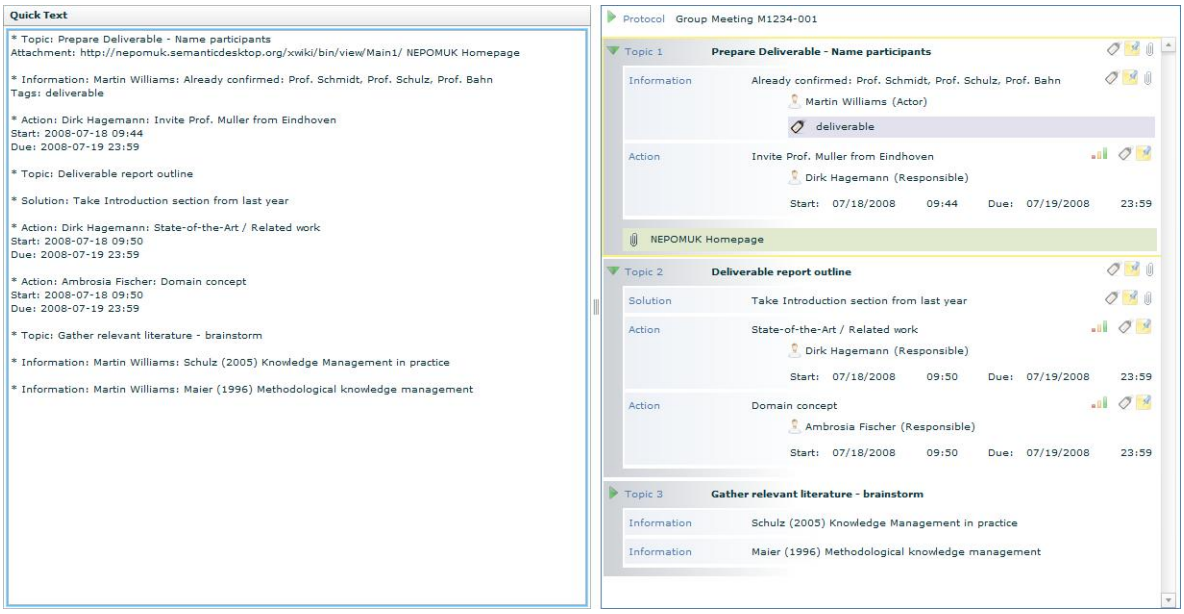


Figure 73: 'Quick Text Protocol View' and 'Structured Protocol View'.

The KWer can quickly scribe using a plaintext area. While scribing the KWer can already pre-structure the minutes by, e.g., letting action items begin with "* Action". This string represents a lightweight-wiki syntax with which the Nepomuk Meeting Manager can create the structured representation, as described in section 7.2.3.3. Vice versa, the structured meeting representation can be transformed into plaintext by applying the defined syntax and serializing it.

7.2.3.4 People Panel

The people panel is used by the meeting planner to set up the people related to a meeting, see Figure 74.

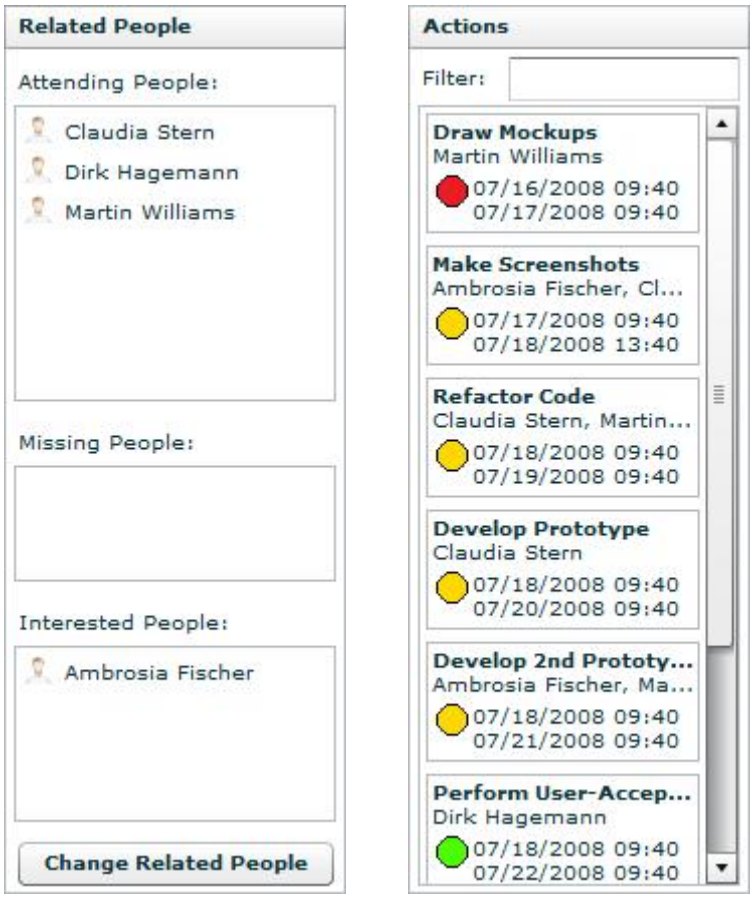


Figure 74: People panel (left), Action panel (right).

The panel supports dragging people onto the structured protocol view in order to set an item's actors or responsible persons. Multiple people can be selected and dragged at once. When the user clicks on a person's name, the related information panel displays information related to that person.

The meeting planner can add people to the protocol by opening a selection dialog for involved people, see Figure 75. This dialog opens after pressing the 'Change Related People' button. The KWer can select the related people and assign them a role in the meeting. At any time, it is possible to change the people's roles by dragging and dropping them from one list to another or by pressing a keyboard shortcut.

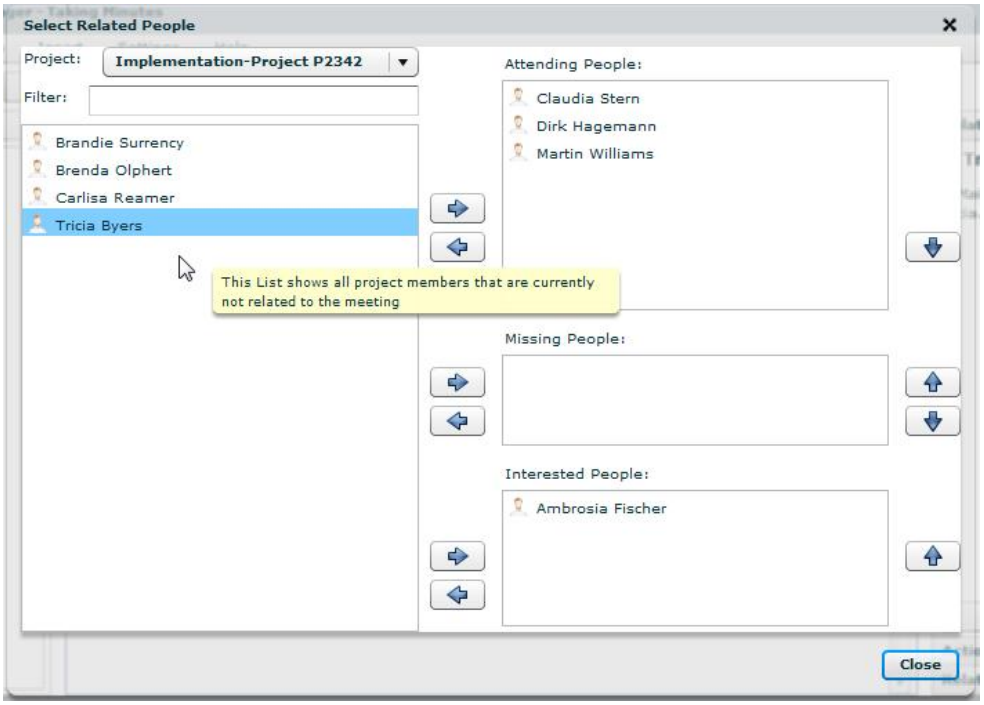


Figure 75: Selection dialog for involved people.

To identify the needed persons the KWer types in a person's name in the 'Filter' section and related people show up in the list below. The list features auto-completion functionality, i.e., the list of presented people narrows down as the KWer types in further characters in the filter field. All people registered with the KWer's person's information cloud show up in this list, see as well section 7.5.2.

7.2.3.5 Action Panel

The actions panel can be used to get an overview of all unfinished actions, see Figure 74. This includes in particular the not yet completed actions which are related to any other protocol, as shown in the action overview. The action panel list shows each action's responsible persons and start and due date. A color icon indicates whether the action starts in the future, is overdue, i.e., red, or due within the next three days.

The minute taker can *add actions to the protocol* by dragging action items from the list of actions and dropping them into the structured view. This creates an action item, which is associated with the dropped action. The panel also contains an edit field with which the user can filter the actions contained in the list.

7.2.3.6 Related Information Panel

The related information panel shows additional information associated with a particular text or information object, like, e.g., a person or tag. Its information display is triggered through the two functions text search and additional information presentation for the currently selected or hovered information entity, see below for details. The additional information shown in the panel can be dragged and dropped into the content area or the quick text area.

Figure 76 depicts three screenshots of the related information panel, which show an empty information panel and an information panel displaying related information for a person and a tag respectively.

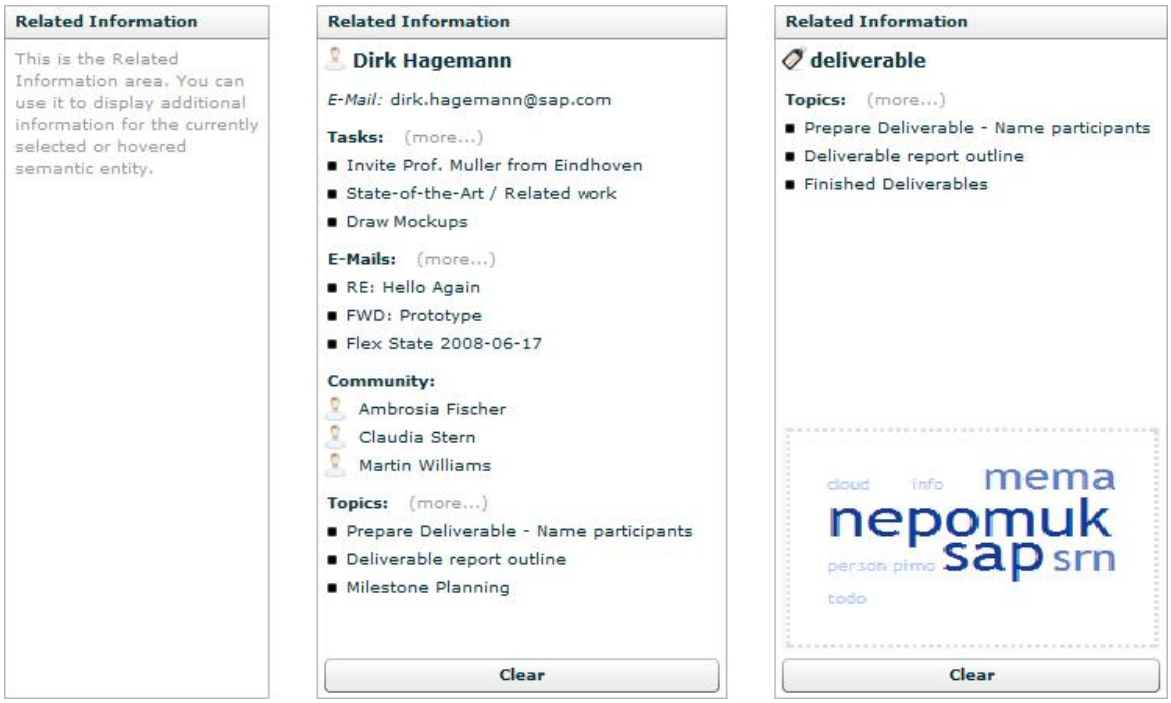


Figure 76: Related Information Panel – Examples.

When the KWer triggered a *search query for information*, the related information panel shows the search results. These search results are provided by the search facility, see section 7.2.3.8. The queries are executed in the background or when text is selected.

The related information panel displays *additional information for the currently selected or hovered semantic entity*. The search is done in background and the additional information can be used in the protocol by selecting, dragging and dropping text into the structured view.

The panel can be used to browse between known concepts. Currently, it is possible to display information for people and for tags. When people are tagged and the user clicks on a tag that is shown in the related information panel, information for the clicked tag is displayed. Links to other concepts like e-mails, tasks and topics are opened in an external viewer. The lists also do only show the three most relevant items for each group and if there are more interesting items than a "(more...)" link is displayed which also opens the external viewer.

7.2.3.7 Annotation Support

Nepomuk Meeting Manager allows the annotation of protocols, topics and items by adding tags, comments and attachments. An annotated topic can be seen in Figure 71.

Tagging is used to associate information items like single items or topics with semantic information. The user can add tags by entering the tag's name and is optionally provided with automatic tag completion.

KWers invited for reviewing the meeting protocol can *add textual comments* to a protocol, which display the author's name and the comment's text. Comments can be used in the verification phase when the related people gather feedback in order to finalize the minutes. By adding comments, the users can express unclear or missing information.

Attachments can be used to associate meeting information with information objects, i.e., their respective URIs. By doing so, the protocol can be used to point to other existing information. Attachments can link to external protocols, files, websites and emails.

7.2.3.8 Search Facility

The search facility[13] provides a central interface to query whether text is of a special kind of domain or if related information for a given text exists. The search can be triggered by selecting text in the quick text area or in the content panel. If no text is selected, typing in new words and phrases triggers a query with the search facility. The search facility itself can invoke a number of services to find out more about the queried data. Possible services are listed in Table 11.

Service	Description
Nepomuk Database, Groupware Applications	Nepomuk can provide people and company related information for people, companies, divisions and projects.
Internet Services which provide domain information	Specialized Internet catalogs provide information about companies and organizations and geographies like continents, countries, regions, provinces, states and cities. They can also supply data about people, products and key industry terms.
Wiki	The search engine embedded in a wiki can search whether a wiki entry exists with a specific title and return information about the entry.

Table 11: Search Facility Services.

7.3 Meeting Protocol Information in the Structured Protocol View

A *meeting protocol* results from a meeting. It consists of items that are grouped by topics, see Figure 71 for an overview.

Protocols store details on the meeting in the protocol header like the meeting title, description, place, date, start (time), end (time) and the date and time of the last change of the protocol, see Figure 77. The minute taker, and all related people (attending, missing and interested people) are also stored. Each protocol has a state that can be one of the following: new, planned, in progress, in verification and final. Additionally, a protocol can have multiple comments and attachments.

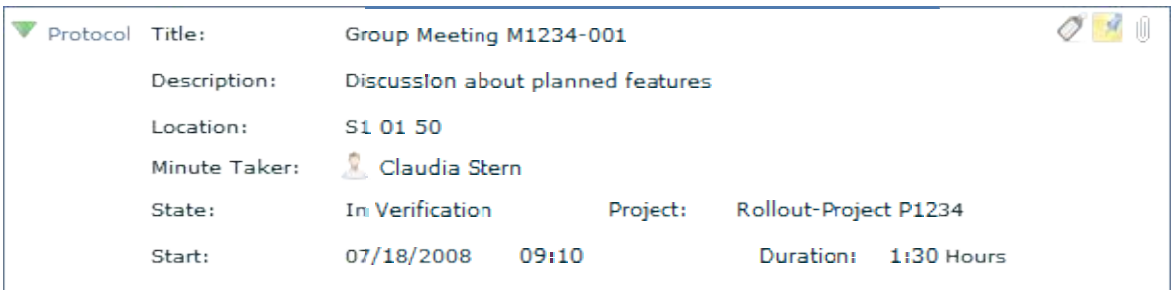


Figure 77: Protocol header – detail view.

Topics are used to group all items, see Figure 78. They are the top-level protocol elements. Each topic is defined by a title and an optional description. Topics can have multiple associated tags, comments and attachments, see section 7.2.3.7 on annotation support.

[13] This feature is mentioned for completeness, it has not yet been implemented in the depicted Nepomuk Meeting Manager prototype.

Figure 78: Protocol part - Topic overview.

Items are the most fine-granular entity in a protocol. They have a type (like information, decision, problem, state and milestone) that defines what each item stands for. Every item has one or many actors associated with it which states who proposed the item. Figure 79 displays an information item example.

Figure 79: Information item – detail view.

During the meeting, meeting participants discuss certain domain-specific types of information, take decisions or define milestones. The KWer can record these discussions by creating specific item types. Valid items types are activity, information, decision, appointment, milestone, conclusion, recommendation, comment, state and target. Special kinds of item are action items, which are related to an action. For special item types like a milestone, the actor representation can be used to define who is responsible for reaching the milestone. Items can have multiple associated comments and attachments.

Action items are items that are linked to an action, see Figure 80. *Actions* define tasks that have to be performed by a responsible person (or by multiple responsible people). Optionally an action can have a start date that states when the responsible can start working on the task and a due date that states the latest possible time to finish the task. Actions have a title and a description. An action can have multiple action states. Actions are not related to a protocol, but protocols can contain action items that are related to an action. An *action state* stores the completion status for an application for a specific date and time and for one of the responsible persons. An optional comment can be specified when the state changes.

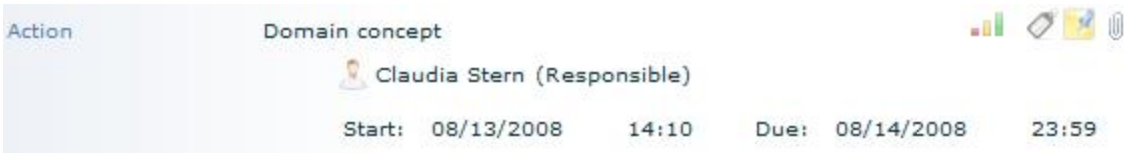


Figure 80: Action item – detail view.

It is possible that an action item is used to report about the state of an action and a certain state can be inserted in the protocol to put it down in writing. The actors of an action item are called "responsible" persons.

7.4 Nepomuk Meeting Manager Supports the Meeting Process

The Nepomuk Meeting Manager is a meeting support application which comprehensively covers the whole process of managing a meeting, i.e., the meeting process.

The meeting representation is the central information item of the Nepomuk Meeting Manager application.

- In the pre-meeting phase the meeting representation serves as central place for information related to the meeting preparation such as the agenda.
- During the meeting, i.e., in the in-meeting phase, the minute taker adds further information on the progress of the meeting as well as results like defined action items to the meeting representation. As well, the meeting participant can retrieve information needed in the meeting from the meeting representation.
- In the post-meeting phase, the KWer post-processes the meeting and re-uses the information generated in the meeting for the subsequent work.

Guiding scenarios illustrate selected parts on how the Nepomuk Meeting Manager supports described meeting process. These scenarios involve the personas Ambrosia, Claudia, Dirk and Martin and are related to the specific needs of these persons. They show how they manage their meetings, how they use the Nepomuk Meeting Manager and how the Meeting Manager makes their work easier.

All of the scenarios take place under a global "SAP Research" scenario and together they describe a meeting's typical lifetime. Claudia prepares the meeting, invites people and takes minutes. After the meeting, the related people (Ambrosia, Claudia, Dirk and Martin) gather feedback. The finalized protocol is distributed and Martin checks the status of his actions and changes one of their states.

7.4.1 Guide KWer through Meeting Process

The guidance for the meeting process has two aspects. First, the Nepomuk Meeting Manger *guides the KWer through the meeting process* using a visual process overview in the Meeting Process Overview window. This visual guidance enables less meeting experienced KWers to benefit from a proven reference process to manage the meeting. The visual process overview leads the KWer step-by-step through the meeting process. The KWer advances a meeting step-by-step through the meeting process activities and transforms thereby the meeting representation, i.e., the KWer starts with the meeting invitation, adds a protocol to the meeting and uses the resulting action items as tasks in the task management application.

Second, the KWer can *invoke all meeting-related activities and access needed functionality in one place*, the meeting process overview as shown in Figure 64. As well, meeting *information is handed-over along the meeting process activities*. This reduces the KWer's manual orchestration activities among applications to manage a meeting to a minimum. The KWer doesn't need to identify a number of applications for each meeting again, but can directly access them from the meeting process overview window. The same holds true for the meeting information, where the KWer doesn't need to manually transfer meeting information between several specialized applications.

Having all meeting-related functionality in one place and not needing to manually transfer information between each activity saves both meeting professionals and less meeting-experienced KWers time.

The Nepomuk Meeting Manager *orchestrates specialized meeting-related applications* to support the meeting process execution by the KWer from one central place, see Figure 81. The Nepomuk Meeting Manager invokes, when needed, other specialized applications to provide the requested functionality and ensures that information is passed to these applications.

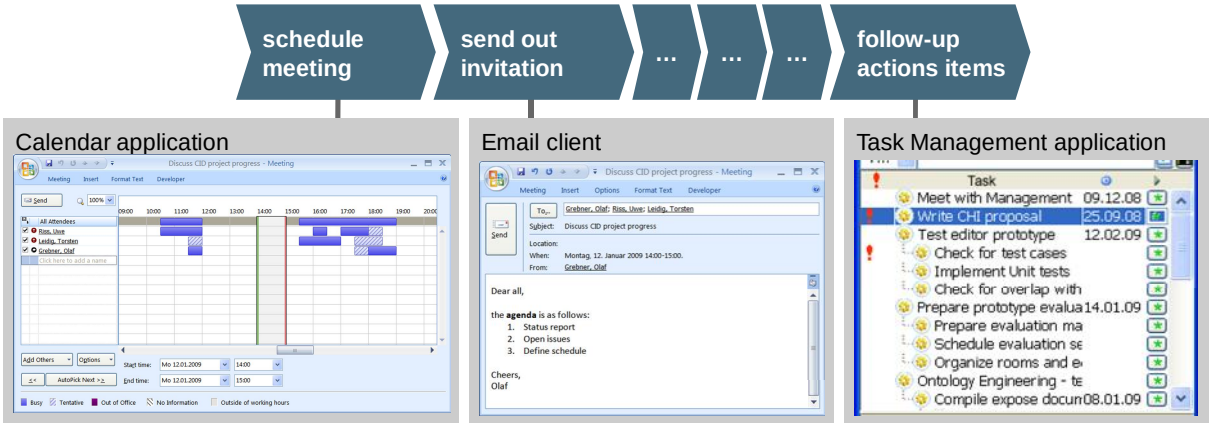


Figure 81: Meeting process orchestrates applications.

For some of the meeting process activities, specialized applications cover the needed functionality. The Nepomuk Meeting Manager in this case doesn't replicate these applications' functionality but integrates these applications into the meeting process. Integration takes place both on information and application layer. On information layer, the Nepomuk Meeting Manager passes meeting information to these specialized applications. On application layer, the Nepomuk Meeting Manager invokes particular dialog windows.

A number of *examples*[14] illustrate the integration of several applications and information sources, see as well Figure 81. For *scheduling a meeting* the KWer 'Schedule New Meeting' button opens an appointment selection dialog in the calendar application. The KWer can specify time and place of the meeting. This appointment information then shows up in the KWer's calendar and the Nepomuk Meeting Manager associates this information to the meeting. To *send out the meeting invitation*, the KWer presses the 'Send Invitation' button. The Nepomuk Meeting Manager then compiles available meeting information and creates an email which contains the appointment, the meeting agenda and addresses this to the persons attached to the meeting. The KWer just needs to press the send button in the upcoming email window of the email client to invite to the meeting. To *follow-up action items* resulting from a meeting the KWer can open the action items from within the action overview window as task in a specialized task management application, e.g., Kasimir.

7.4.2 Pre-meeting Phase

The KWer *prepares the meeting* in the pre-meeting phase. In this phase the KWer in the role of a meeting planner sets up the meeting and with it the meeting's details, like, e.g., title, date, time and location as well as the agenda and the participants list. This can be done completely from scratch or based on a past meeting and its meeting protocol.

[14] The meeting process window shown in Figure 64 doesn't show every here described feature.

A *guiding scenario* shows how Claudia prepares a meeting and invites people to it using the Nepomuk Meeting Manager:

- Every Monday, the Nepomuk Research Group uses a weekly meeting to inform the members about each person's progress. Normally, the participating people are Claudia, Dirk and Martin who joins the Karlsruhe group via web conferencing software. As Ambrosia is the research manager, she is not participating the meetings but wants to be informed about what has been said.
- Claudia is responsible for preparing the group's meetings, which she does every Thursday. She starts by logging into the application and setting up a new meeting in the meeting tool. She can save herself some time by copying the details, related people and agenda of an old progress and save it as a new meeting. She just changes the meeting date to the date of next Monday and deletes some topics that were only relevant for the last meeting. The Nepomuk Meeting Manager application automatically inserts actions contained in the old protocol, so that during the meeting the new status can be communicated. When Claudia finishes preparing the agenda, she clicks on a button and the application automatically sends invitation e-mails to all participants, which contain the meeting's details, an entry for their calendar, the agenda and a personalized list of unfinished action items.

7.4.2.1 Write Up Agenda

The KWer can write up the meeting agenda by creating a new meeting representation and typing meeting topics which altogether make up the meeting agenda. When a meeting representation does already exist, the KWer can directly edit this representation by opening the meeting.

The KWer creates a new meeting representation using the 'New Meeting' button in the Meeting Process or Meeting Overview window. In the upcoming Edit Protocol dialog window the KWer can add topics to the protocol to define the meeting's agenda. The Nepomuk Meeting Manager allows editing meeting protocols before the actual meeting takes place to organize the meeting information.

There are two options to describe the topics, as explained in section on taking meeting minutes. The KWer either types free-text topics in the Quick Text Protocol View and transforms these into structured topics, or the KWer directly creates new topics in the Structured Protocol View, e.g., by pressing the 'New Topic' button in the top symbol pane.

7.4.2.2 Manage Participants and Interested Parties

The KWer in the role of the minute taker can manage the people related to a meeting, i.e., meeting participants and interested persons in the people panel of the 'Edit Protocol' dialog window, see Figure 74.

The people panel contains three lists for attending, missing and interested persons. Using the 'Change Related People' dialog window, the KWer assign persons to these categories, see section 7.2.3.4. In the pre-meeting-phase the KWer can plan the meeting participants, i.e., attending and interested persons. In the in-meeting phase the KWer can verify the attendance and record the attending and missing persons.

7.4.2.3 Collect information for meeting

The Nepomuk Meeting Manager supports the KWer in collecting information for the meeting preparation by *associating this information with the meeting representation*. Therefore, the KWer

manually collects information with the meeting representation, takes over unfinished action items from a past meeting or imports meeting information from other applications.

The KWer can *manually collect information with the meeting representation* in the Edit Protocol Dialog by either creating meeting items or annotating the meeting items with information objects. The KWer can create information items for short, textual information snippets of, e.g., an idea to discuss by, e.g., pressing the 'New Item' button in the symbol pane. Alternatively the KWer can collect and attach information objects to a particular protocol, topic or item based on, e.g., the meeting agenda as representation for the meeting structure, see Figure 71 with an attached website to a meeting topic. The same is possible for documents or emails.

The KWer can *take over unfinished action items from a past meeting* into the current meeting to follow them up and discuss them again. The Nepomuk Meeting Manager enables the KWer to follow-up on a past meeting by pressing the "Follow-up" button in the Protocol overview, see section 7.2.2. It takes unfinished action items of this meeting protocol and shows them in the actions panel of the current meeting, see section 7.2.3.5.

The KWer can as well *import meeting information from other applications* to have this information in place with the meeting representation. For example, the KWer can import action items from other Meeting support software, e.g., action items created within DigitalModeration [Teambits GmbH, 2009], a meeting moderation support software for meetings with large audiences. Based on a file containing the moderation results from DigitalModeration, the Nepomuk Meeting Manager extracts the therein included action items and imports them into the meeting protocol as action items.

7.4.2.4 Make Appointment

The Nepomuk Meeting Manager supports the KWer in making the appointment for the meeting by *integrating appointment scheduling functionality of calendar applications*.

On the one hand, the KWer can define the date and location of the meeting manually in the protocol header of the 'Edit Protocol' Dialog to make an appointment for the upcoming meeting.

On the other hand, the KWer can *use the appointment scheduling* functionality of, e.g., Microsoft Outlook to coordinate the date, time and location among the participants and find a suitable timeslot and place where all intended participant can take part.

The KWer triggers the meeting scheduling dialog in, e.g., Microsoft Outlook by choosing the 'Create appointment' option in the 'File' menu. The participants as well as the date and location information are already included in the upcoming dialog for the Microsoft Outlook appointment when the KWer already entered a date and location into the protocol header. The KWer can now schedule the meeting as shown in Figure 82 and with saving the appointment this information is updated in the Nepomuk Meeting Manager.

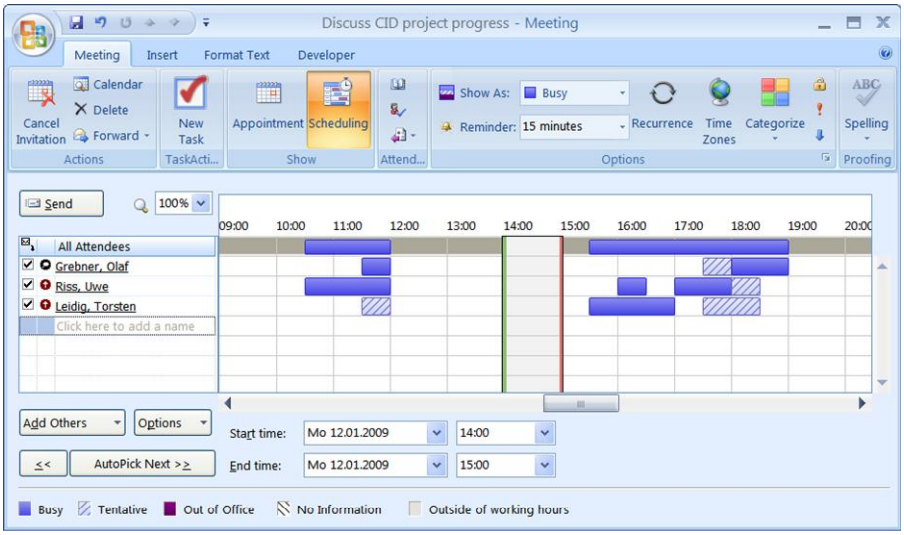


Figure 82: Scheduling a meeting in Microsoft Outlook.

As part of the invitation, the KWer can communicate the date and time to the participants and interested persons by sending an invitation message. This invitation e-mail contains an iCalendar file to create show the meeting appointment directly in a calendar application, like, e.g., Microsoft Outlook.

7.4.2.5 Send invitation

The KWer can *send an invitation message* containing the agenda to all participants and interested persons via email using the Nepomuk Meeting Manager, when she has finished the meeting preparation, i.e., successfully concluded the other pre-meeting activities.

The KWer triggers sending the invitation, e.g., with the 'Send invitation' option from the 'File' menu within the 'Edit Protocol Dialog'. An email opens in the email client, e.g., Microsoft Outlook, pre-filled with the participants and interested persons as receivers. This email contains as body the agenda, other meeting details like the location and the meeting organizer as well as all relevant attachments. The email contains additionally an iCalendar [Dawson&Stenerson, 1998] file which creates an appointment for the particular meeting time in a receiving calendaring application like, e.g., Microsoft Outlook [Microsoft Corporation, 2009a] or Mozilla Thunderbird [Mozilla Foundation, 2009b].

7.4.3 In-Meeting Phase

The KWer *conducts and participates in the meeting* in the in-meeting phase. In this phase the KWer in the role of a minute taker records the meeting protocol, a KWer in the meeting participant role contributes to the meeting.

7.4.3.1 Retrieve Information for Presentation

The KWer can browse and retrieve information needed in the meeting in one place by accessing the *meeting representation's associated information*. The KWer can open a meeting representation in the Nepomuk Meeting Manager. In the 'Edit Meeting Protocol' dialog window, the KWer sees all information relevant for the meeting in one place. Here, the KWer can browse through the meeting information, look at textual information items or other items like, e.g., action items and open the respectively attached information objects, like, e.g., emails, websites and files.

7.4.3.2 Record Meeting Results

The Nepomuk Meeting Manager enables the KWer to *quickly scribe and conveniently structure the meeting protocol*. The Nepomuk Meeting Manager's main screen allows to *quickly scribe meeting protocols* in a quick text area, the 'Quick Text Protocol View', and to *structure them later* on using a structured view, the 'Structured Protocol View'. Supporting the meeting minutes taking is the core functionality of the Nepomuk Meeting Manager.

A meeting protocol displayed in the 'Structured Protocol View' consists of topics, which contain multiple items, see section 7.2.3.1. Each item can be of a certain type, like for example decision, problem and information. An action items represents a special item type which has a responsible person and an optional due date associated. The minute taker can use the related information panel, see section 7.2.3.6, to look up related details.

During meetings, it is possible to use the 'Quick Text Protocol View' to quickly scribe minutes in plaintext, see section 7.2.3.2, and later to organize and structure them in the 'Structured Protocol View' of the 'Edit Protocol' dialog.

The KWer can *enter information on the meeting* independent of the chosen view. This includes basic information for the meeting minutes such as creating topics, creating information items or recording decided action items. In the structured view, the KWer can add information to the meeting content, e.g., set tags or annotate it with information objects, see section 7.2.3.7. As well, the KWer can drag information objects from the 'Related Information' panel, see section 7.2.3.6. From a social perspective the KWer can update the participants list or initially start it, see section7.2.3.4.

Organizing and optimizing the meeting content works best in the structured view on the meeting protocol. The KWer can re-order meeting items or categorize them by selecting tags.

To enable the KWer to *both quickly scribe and conveniently structure the meeting protocol*, the Nepomuk Meeting Manager "translates" plaintext following a lightweight syntax into a structured, visual representation and vice versa. The KWer can iteratively refine the meeting minutes by transitioning between the two views and respective representations. Starting with writing down some statements, the KWer can structure them afterwards. Vice versa, starting off with a meeting structure, the KWer can follow the meeting topics in discussion and quickly jot down the results for each topic while the meeting is ongoing.

Transforming written minutes into structured content enables the KWer to post-process the initially quickly jotted meeting minutes in various ways. For example, using the 'Structured Protocol View' the KWer can conveniently refine and structure meeting notes and organize them. The KWer can create from the meeting notes a high-quality meeting protocol with formatted meeting minutes ready for printing – all without any additional effort by the KWer. Or the KWer can follow-up the defined action items in her task management application delegate the tasks.

Vice versa, the KWer can transform the meeting minutes back from the structured form to the text form to rework the meeting minutes in the text, for example when the meeting continues after a break. As well the KWer can start with the topics in the plaintext as guidance in the hasty meeting – the KWer has fewer effort of not again typing the particular topics.

A guiding scenario shows how *Claudia takes minutes during a meeting* using the Nepomuk Meeting Manager:

- Next Monday, Dirk and Claudia meet in the room at the time when the meeting takes place. As they do not know whether Martin will come later, Claudia marks Martin's name and drags and drops it into the list of missing people. Without Martin, they start the meeting. For each subject of discussion, Claudia creates a new item, drags it to the right position and enters the item's details. 20 minutes later, Martin arrives and Claudia changes Martin's state back to "attending". While his train was stuck, he thought about another topic that they need to discuss. Claudia creates a new topic at the end of the protocol. One of the items they discuss is an action item. Martin has to search for state of the art. Claudia creates a new action item, selects Martin as the responsible person and enters a description. When the conversation is too quick during the meeting, Claudia uses the quick text area to quickly enter free text without having to deal with topics, items and details. After the meeting, when there is more time, she sits down, drags and drops the free text into new items.

Another guiding scenario shows how *Claudia uses the related information panel during a meeting* in the Nepomuk Meeting Manager:

- During the meeting, Dirk wants to give Ambrosia a call. He asks Claudia to look up Ambrosia's office telephone number. Claudia enters "Ambrosia Fischer" in the quick text area and selects the words using the mouse. The related information panel shows details for the person "Ambrosia Fischer", like her phone numbers and office location. Because she does not answer the phone, Claudia clicks on the link of Ambrosia's department to get a list of all its members, so that Dirk can try to get one of them on the phone.

A further guiding scenario shows how Claudia follows up an action item the Nepomuk Meeting Manager:

- In the previous meeting, the new action "Check for existing Web 2.0 Applications" was created and Martin has been assigned as the responsible person. An action item has automatically been created, when Claudia created the new protocol based on the older minutes. This way, the participants are reminded about the action and Martin can report about the current state. While he is speaking about his action he reminds, that some time ago Ambrosia has done something similar. Claudia opens the action panel and checks for Ambrosia's open actions. The participants can see that Ambrosia did not finish her action and so they can add this state to the protocol to remind Ambrosia of completing it.

7.4.3.3 Exchange Information

The Nepomuk Meeting Manager does not have any dedicated support functionality regarding the information exchange in meetings. The KWer can browse the information gathered with the meeting representation, see section 7.2.3.1.

As well, it is possible in a distributed meeting setting that a specialized application for sharing computer screens transmits real-time audio and video streams of the meeting. Nepomuk Meeting Manager can invoke such applications and pass information, e.g., on meeting participants to them.

7.4.3.4 Record Meeting Streams

The Nepomuk Meeting Manager does not have any dedicated support functionality regarding the recording meeting streams. It is possible that a specialized application for recording meeting streams

can capture audio and video streams of the meeting. Nepomuk Meeting Manager can invoke such applications and pass information, e.g., on meeting participants to them.

7.4.3.5 Brainstorming

The Nepomuk Meeting Manager does not have any dedicated support functionality regarding brainstorming in meetings. The KWer can quickly type in text in the 'Quick Text Protocol View' to capture generated ideas and browse information gathered with the meeting representation to get input for new ideas, see section 7.2.3.2.

As well, it is possible that a specialized application for supporting brainstorming like, e.g., a mind mapping application like MindManager [Mindjet Corporation, 2009], can stimulate the participant's creativity. Nepomuk Meeting Manager can invoke such applications and pass information, e.g., on the meeting topics and information items to them.

A *guiding scenario* shows how Claudia uses the 'Quick Text Protocol View' of the Nepomuk Meeting Manager to capture ideas:

- As Claudia, Dirk and Martin want to gather some thoughts on additional features, they do a quick brainstorming. To write down the different thoughts quickly, Claudia uses the quick text area to freely enter the text. This way, she can easily collect all topics that come to their minds. Later, she can drag and drop the different thoughts in another order to the content panel to create items.

7.4.3.6 Voting

The Nepomuk Meeting Manager does not have any dedicated support functionality regarding voting. It is possible that a specialized meeting support software application provides voting support, like, e.g., DigitalModeration [Teambits GmbH, 2009], a meeting moderation support software for meetings with large audiences. Nepomuk Meeting Manager can invoke such applications and pass information, e.g., on the meeting topic in question and the voting participants.

7.4.3.7 Meeting Facilitation

The Nepomuk Meeting Manager does not have any dedicated support functionality regarding meeting facilitation. It is possible that a specialized meeting support software application provides meeting facilitation support, like, e.g., DigitalModeration [Teambits GmbH, 2009], a meeting moderation support software for meetings with large audiences. Nepomuk Meeting Manager can invoke such applications and pass information, e.g., on the meeting topics.

7.4.3.8 Give a Presentation

The Nepomuk Meeting Manager does not have any dedicated support functionality regarding presentations. It is possible that a specialized presentation application enables the KWer to present slides, like, e.g., Microsoft PowerPoint [Microsoft Corporation, 2009f]. Nepomuk Meeting Manager can open presentation files when they are attached to a meeting item, like e.g., a topic.

7.4.4 Post-Meeting Phase

The KWer *post-processes the meeting results* in the post-meeting phase. In this phase the KWer in the role of a minute taker gathers feedback on the meeting minutes, finalizes them and distributes the finished protocol. As well, all KWers can follow-up the resulting action items and can re-use the meeting protocol information.

7.4.4.1 Gather Feedback on Meeting Minutes

To gather feedback on the meeting minutes, people related to a meeting can collaboratively add comments about missing or incorrect information. The meeting protocol is then in the verification phase.

When the protocol's first version is finished after the meeting, the KWer triggers sending an email to all participants inviting them to approve or decline the protocol. Thereby they can comment the meeting protocol, see section 7.2.3.7. Based on these received comments the minute taker can decide to change the protocol or to finalize it. When the feedback has been gathered, the protocol is finalized and no further changes are allowed.

A *guiding scenario* shows how Claudia gathers feedback from Ambrosia on her meeting protocol using the Nepomuk Meeting Manager:

- When Claudia finishes the protocol's first version, she clicks on a button to start with the feedback phase. E-mails containing the protocol's text are sent to all related people. Dirk is not satisfied with what Claudia had written down during the meeting, so he logs into the application, clicks on the according protocol and enters a comment. In this comment, he proposes a new text that should replace the old text. Ambrosia, who has not attended the meeting, reads the e-mail and is not sure what one of the items means. She logs into the application, clicks on the according protocol and enters a comment. She asks to clarify the items text so that she can understand it without having participated in the meeting.

7.4.4.2 Finalize Meeting Minutes

The minute taker can *finalize the protocol* so that no further changes are possible. Therefore, the KWer can export the meeting protocol into a number of formats depending on the particular need.

The Nepomuk Meeting Manager exports protocols into the formats RTF for Microsoft Word, PDF for revision safe archiving, HTML for web browser and wiki-syntax for copy & pasting into wiki applications. Figure 83 exemplary shows a protocol as a PDF file (left), an XML file (middle), wiki syntax (bottom) and an HTML file opened in a browser (right).

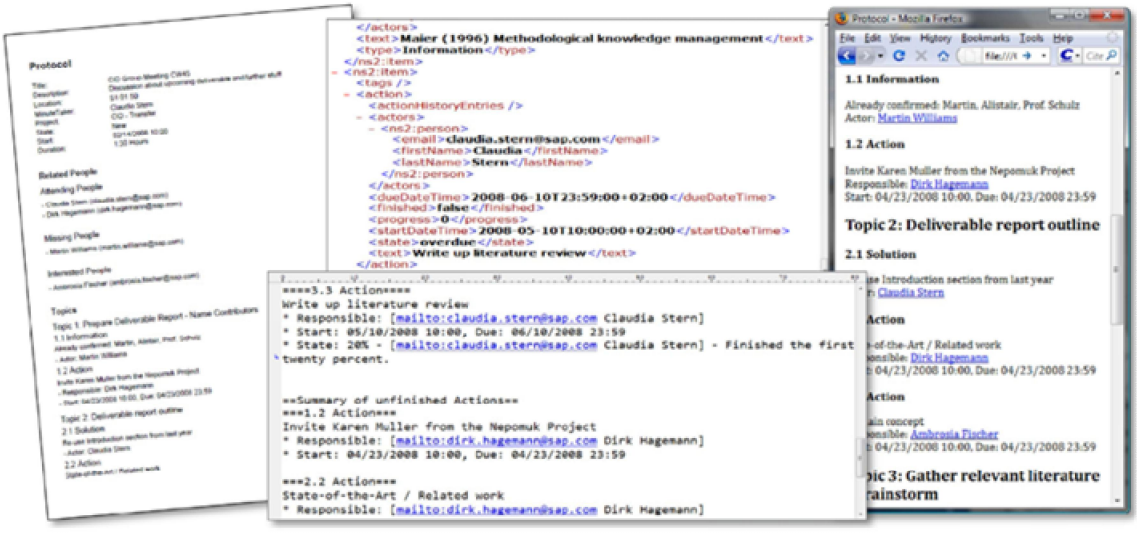


Figure 83: Examples of Exported Meeting Protocols.

A *guiding scenario* shows how Claudia finalizes her meeting using the Nepomuk Meeting Manager:

- Claudia, the minute taker, logs into the application and is presented an overview, where she can see all comments that have been made by Dirk and Ambrosia. She changes the protocol according to the comments, by going into edit mode and changing and adding text. Afterward, she clicks on a button to finalize the protocol. The application stores the protocol in a finalized state and sends e-mails to all related people, which contain the PDF and a text version of the protocol in the e-mails body. The body also contains a personalized list of unfinished actions.

7.4.4.3 Distribute Protocol

The KWer can send out the protocol's final version to all participants and interested people as an e-mail attachment in a revision safe format like, e.g., PDF. The KWer can distribute the meeting protocol by triggering the 'Send to Participants' option in the 'File' menu. The Nepomuk Meeting Manager then sends the finalized protocol to all related people as an e-mail attachment in a revision safe format like, e.g., PDF.

7.4.4.4 Follow-up Resulting Action Items

The Nepomuk Meeting Manager provides basic task management functionalities and a task overview to enable a follow-up of action items resulting from a meeting.

Nepomuk Meeting Manager allows users to browse protocols and action items, see section 7.2.3.1. Here, the Nepomuk Meeting Manager provides an overview for all existing task resulting from these action items and the KWers can see their unfinished actions. They can be filtered (only personal action items), grouped (by project, due date, state of action items) sorted and searched as each of the protocol's items is typed and action items allow setting a responsible person as well as start and due date. Existing action items can also be referred in other protocols with the actions panel, see section 7.2.3.5.

A *guiding scenario* shows how Martin tries to get an overview of his personal action items using the Nepomuk Meeting Manager:

- Martin is unsure what his unfinished actions are. So he logs into the application and Nepomuk Meeting Manager displays an overview of all upcoming meetings. As he is interested in his unfinished action items, he clicks on the actions button. In the overview, he can see the action "Check for state of the art" that he is responsible for. This is his only unfinished action. He changes the filter to display all tasks because he is unsure what his last actions were. He can see a longer list with older and already finished actions and his unfinished action at the top.

A guiding scenario shows how Martin changes the state of an action using the Nepomuk Meeting Manager:

- One of Martin's actions is to check for state-of-the-art. After he finishes the search and thinks that the list of existing software he has found is complete, he logs into the application. The displayed overview of unfinished actions contains only one item for which he is responsible. He clicks on this action titled "Check for state of the art" and is presented with a dialog where he enters a short comment and that the action is 100 percent complete. After clicking a save button, the application displays that he has no unfinished actions left and he chooses to logout.

7.4.4.5 Re-Use Meeting (Protocol) Information

The Nepomuk Meeting Manager enables the handling of particular types of protocol items by external applications to make meeting information available for these applications. For example, the Nepomuk Meeting Manager can forward problem or solution items to a risk management application. The KWer then can directly use this information within the risk management application like, e.g., SAP BusinessObjects Risk Management [SAP AG, 2009], and process it there without investing effort in the information transfer.

External applications can access the meeting content and work with this information. It is directly available to them in case they access the common personal information cloud. If needed, an export step prepares the information to be in a certain form. Table 12 gives an overview on possible applications re-using meeting information.

Application	Meeting information re-use
Nepomuk	Action items become Nepomuk tasks. Re-use a meeting's action items in Kasimir
Groupware	Action items can become groupware tasks.
Bug Tracking	Item Types "Bug" or "Feature Request" can be treated like action items as they have a state and a responsible person.
Risk Management Software	Items with type "Problem", "Solution" or "Risk" can be connected and pushed into an external database.
Project Management Software	"Milestone", "State" and "Target" are item types that can be handled by project management software.

Table 12: Possible applications re-using meeting information.

7.4.5 Architecture and Implementation

The Nepomuk Meeting Manager application supports a KWer's personal meeting management activities. Its implementation focuses on the user interface, a combined Adobe Flex [Adobe Inc., 2007] and Adobe Air [Adobe Inc., 2008] application, whereas the invoked Meeting Management Service, see section 10.2, prevalently provides the business logic. This keeps the implementation of the Kasimir application lean. Figure 84 shows the principle of the Nepomuk Meeting Manager application invoking the Meeting Management Service.

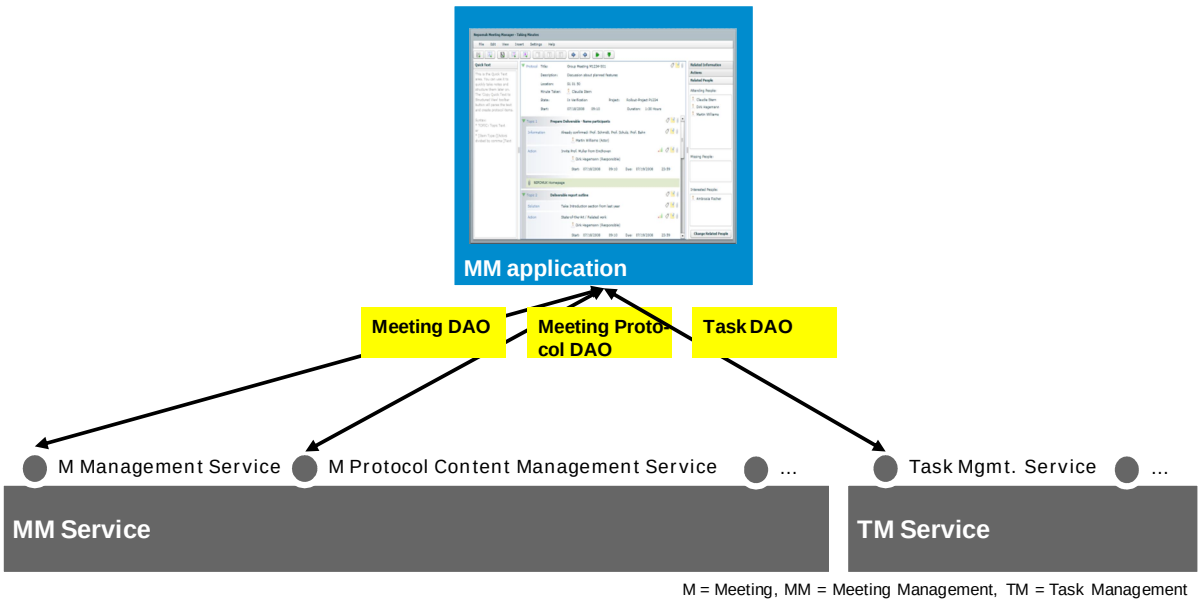


Figure 84: Nepomuk Meeting Manager implements both Meeting and Task Management Service.

The Nepomuk Meeting Manager not only invokes the Meeting Management Service, it as well invokes the Task Management Service for creating action items and reading meeting protocol-related action items.

7.5 KWer Benefits of Unified Personal Information

The Nepomuk Meeting Manager operates on the representation of the KWer's personal information cloud. On the one hand, the KWer can contribute information about meetings and their meeting protocol's content to the personal information cloud. On the other hand, the KWer can use the information from the personal information cloud within the Nepomuk Meeting Manager. The KWer doesn't need to take care about keeping potentially redundant sets of information.

The Nepomuk Meeting Manager bases on a separation of concerns for managing personal information. It focuses on managing the 'own' meeting and meeting protocol information, whereas it leverages information maintained by other applications. This enables the KWer to use for each use case the favorite and best suited application.

7.5.1 Contribute Meetings and -Protocols to Personal Information Cloud

The KWer primarily manages with the Nepomuk Meeting Manager meetings and meeting protocols as well as their representations. Thus the Nepomuk Meeting Manager primarily manages a meeting representation and the corresponding meeting protocol representation, see Figure 85 for a simplified overview. The Nepomuk Meeting Manager thereby operates on the personal information cloud, i.e., the managed tasks are available in the personal information cloud.

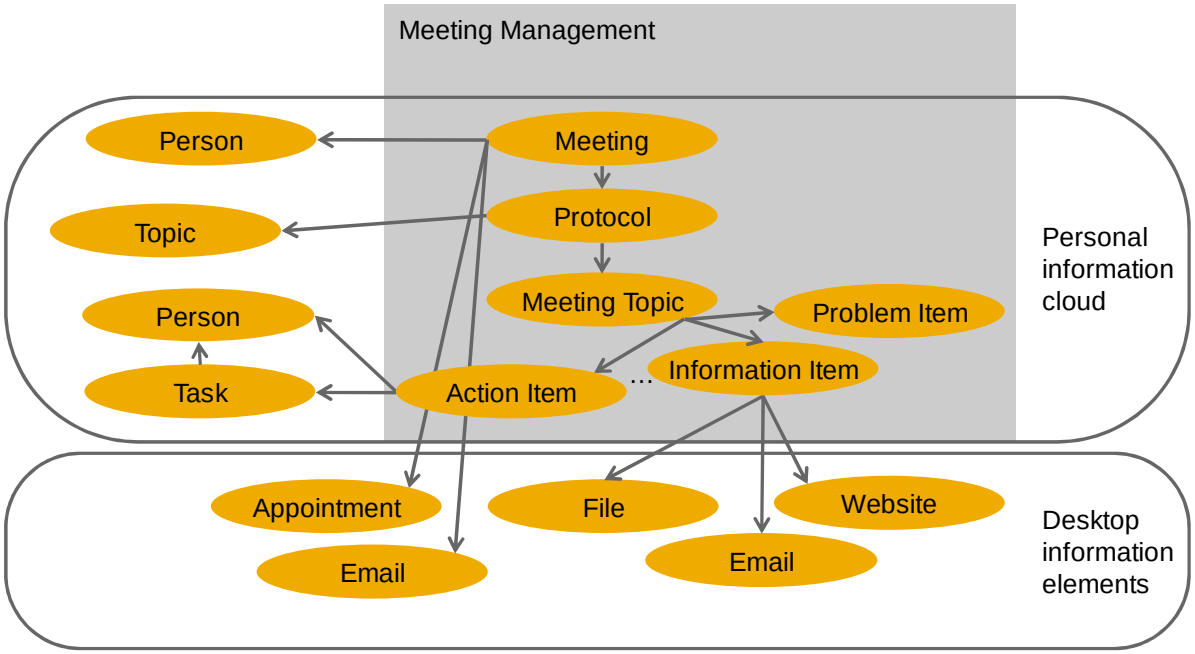


Figure 85: Personal Information Concepts managed by Nepomuk Meeting Manager.

The KWer can assign for each meeting a meeting protocol. A meeting protocol consists of multiple meeting topics which themselves contain multiple items. The meeting topics make up the agenda of the meeting. Each item can be of a certain type, like, e.g., decision, problem and information item. Additionally, each item can have one or many associated actors, e.g., in the case of an information item the actor denotes who uttered the point in the meeting. Action items represent a special item type. An action item has a mandatory responsible person associated and an optional due date.

7.5.2 Use Personal Information Cloud for Meetings and Meeting Protocols

The KWer can use through the Nepomuk Meeting Manager functionality the information available in both the KWer's personal information cloud representation and the desktop information elements.

The KWer can categorize a meeting protocol with a *topic*, shown as tag in the Nepomuk Meeting Manager, e.g., to be able to re-find the meeting protocol again. The KWer can enter a tag by typing in the name using the annotation functionality in the Nepomuk Meeting Manager, see section 7.2.3.7. Thereby, the Nepomuk Meeting Manager's auto-completion feature offers recommendations on possible tags based on the KWer input. These recommended tags are matching topic representations from the personal information cloud. When the KWer selects one of the offered tags, i.e., a topic from the personal information cloud, the Nepomuk Meeting Manager associates the meeting protocol representation with the selected topic and thus associates the meeting protocol representation with the remaining personal information cloud. As well, the KWer can tag[15] each of the below explained meeting topics and the information, action, problem etc. items.

All *person* representations showing up in the Nepomuk Meeting Manager correspond to person representations in the personal information cloud. The KWer can associate persons involved into a meeting with the meeting itself, see section 7.2.3.4, e.g., to name the meeting participants. The same holds true for persons involved in the meeting protocol, a meeting topic or item, e.g., to define a responsible person for an action item, see section 7.2.3.4. For example, to attach a person to a meeting as meeting participant, the KWer can select a person in the 'Change Related People' dialog,

[15] For simplicity reasons this is not shown in Figure 71.

see section 7.2.3.4. There, the KWer can enter person's name. Thereby, the Nepomuk Meeting Manager's auto-completion feature offers recommendations on possible matching persons based on the KWer input, see section 7.2.3.4. These recommended persons are matching person representations from the personal information cloud. When the KWer selected a particular person, the Nepomuk Meeting Manager associates the person representation with the meeting representation in the personal information cloud and records the meeting participant role of the involved person. As well, the KWer can associate persons to the meeting protocol and its meeting topic and items in the same manner.

The Nepomuk Meeting Manager creates a task representation for each *action item* that was defined in a meeting protocol and where the KWer is either responsible or involved in. The task information contains the same information as defined within the action item, e.g., it references the same responsible person as the action item. In addition, it associates the action item representation with the newly created task representation.

The KWer can associate *desktop information elements* to a meeting protocol, meeting topic or item representation using Nepomuk Meeting Manager's annotation functionality, see section 7.2.3.7. For example, meeting-relevant information elements on the desktop are files, websites, emails and appointments. When the KWer selected a particular information element in the corresponding selection dialog window in Kasimir to annotate, e.g., a meeting with needed information, the Nepomuk Meeting Manager associates the information element with the corresponding meeting entity. For example, when the KWer selected to attach a file to a meeting topic in the corresponding file selection dialog, the Nepomuk Meeting Manager associates the file representation with the meeting topic representation in the personal information cloud. The KWer can perform the same action with websites, see section 7.2.3.7. As well, the Nepomuk Meeting Manager can associate a meeting representation to an appointment representation, which itself is managed in the calendar application, when the KWer used appointments for scheduling the meeting. Furthermore, the Nepomuk Meeting Manager can associate the meeting representation to an email representation when the KWer used it as invitation for the meeting.

PART III – DESIGN – SOFTWARE REFERENCE ARCHITECTURE

ACKNOWLEDGEMENTS AND PUBLICATIONS

- The software reference architecture presented in section 8 has been published in [Grebner, 2009].
- The Task Model Ontology (TMO) presented in section 9 is publicly available in [Brunzel et al., 2008].
- The Task Management Services presented in section 10 have been published as Semantic Task Management Framework in [Ong et al., 2008]
- Tobias Ackermann contributed to section 10 with his study thesis [Ackermann, 2008] on meeting management and the Nepomuk Meeting Manager.

8 Software Reference Architecture – Workspaces on Unified Personal Information

We present a *software reference architecture* with whom developers can efficiently implement domain-specific applications based on unified personal information, see Figure 86. We use a unified representation of the KWer's personal information cloud for the common abstract information model underlying these applications. The proposed software reference architecture consists of three layers.

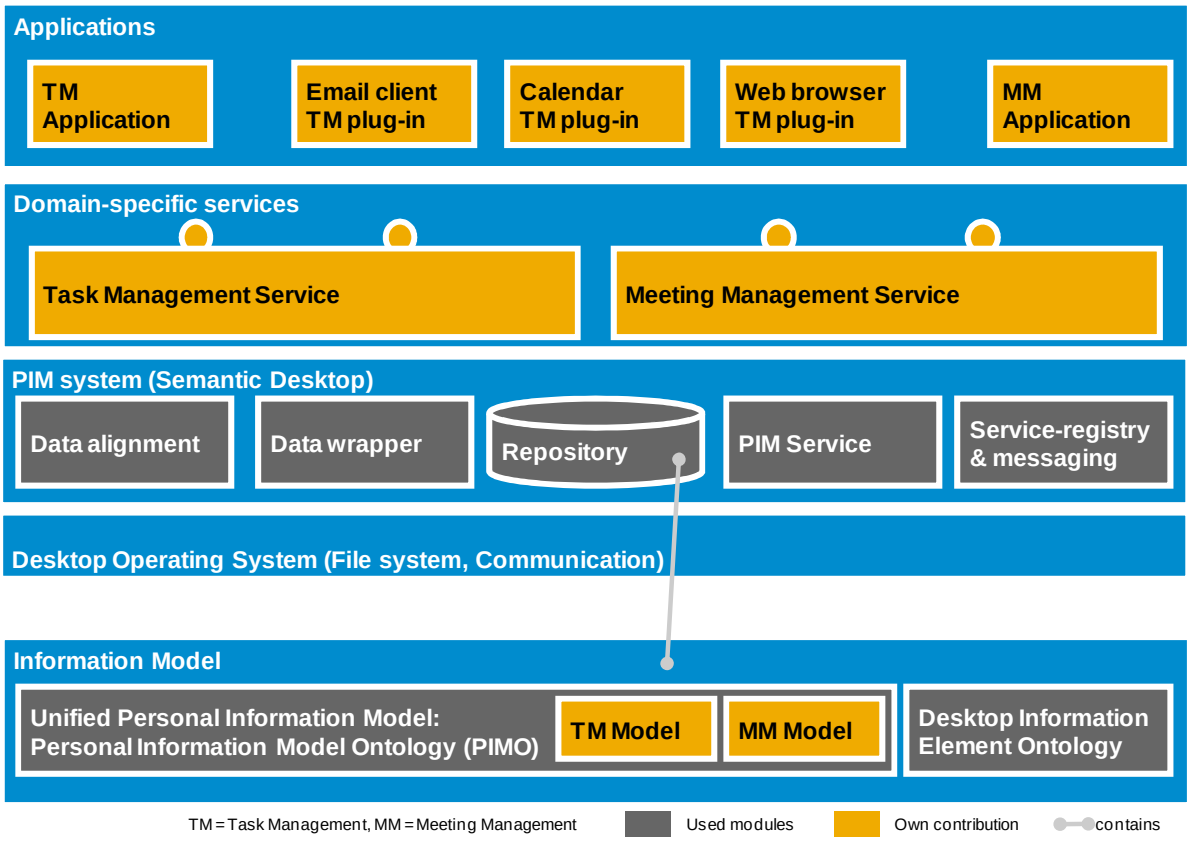


Figure 86: Software reference architecture.

First, facing the end-user, *functionally-oriented workspaces and applications* each support a particular type of a KWer's activity. They each represent a distinct perspective on the common underlying unified personal information set. These workspaces and applications can be implemented using multiple programming languages and technologies.

Second, a set of *domain-specific services* handles the access to a particular perspective on a KWer's personal information from the application layer to the unified information model. The domain-specific service's functionality provides the needed parts of the personal information to domain-knowledgeable developers. This way, the user experience of each workspace can be designed solely relying on the domain and completely independent from the underlying (operating) system and associated metaphors and paradigms.

Third, using a *basic PIM system* we can access numerous desktop services and thus re-use desktop functionality without replicating it in each workspace. Here we use the semantic desktop *Nepomuk* [Groza et al., 2007] as example for a service-oriented basic PIM system. The Nepomuk semantic

desktop provides a service-oriented desktop architecture and services. The basic PIM system manages the unified personal information model representing the KWer's personal information cloud and offers a PIM Service providing generic read and write functionality for the unified personal information model. Nevertheless, it is user-owned but formally represented using a set of ontologies. Furthermore, it encapsulates core operating functionality in services like, e.g., handling desktop information objects like for example, emails, websites and files.

Using a basic PIM system enables an *evolutionary architecture* to transition current desktop workspaces to the proposed multiple functional workspaces by step-by-step replacing their redundant proprietary stacks for managing a KWer's personal information. Offering a modularized architecture enables this evolution of existing applications. With putting this architecture in place, the today's commercial workspaces can transition into the here outlined multiple functional workspaces without dismissing existing workspaces and investments.

8.1 Functional Workspaces and Applications

Functional workspaces and applications support KWers in executing their higher-level activities throughout the whole workspace. Each functional workspace and application targets a specific type of a KWer's activities while accessing a corresponding particular perspective of the unified personal information.

We implemented functional workspaces and applications targeting the two personal supporting activities task management and meeting management, see Figure 87. In the task management [Grebner et al., 2008] activity, KWer manages to-do lists, prioritizes tasks, organizes task-related information in the context of tasks, browses task-related information and delegates tasks to other persons. In the meeting management activity, a KWer prepares and conducts a meeting, e.g., writes meeting minutes, as well as post-processes the meeting by e.g. following up defined action items.

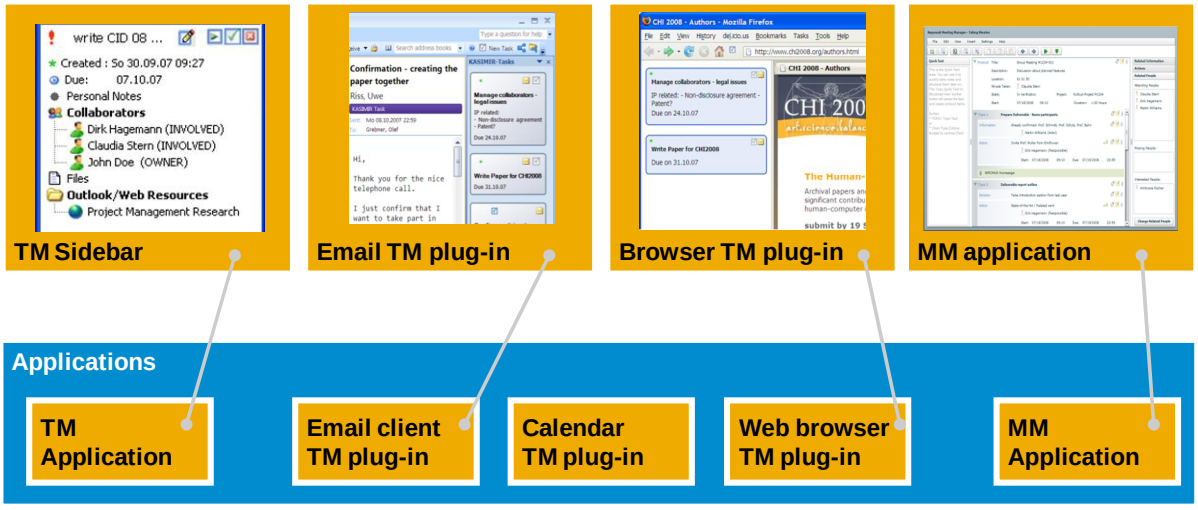


Figure 87: Multiple applications targeting task and meeting management.

Due to the unified personal information model commonly underlying the workspaces, changes in the personal information performed in one workspace perspective are immediately visible in the other workspace perspectives. For example in the task management workspace, the representations of persons to whom tasks can be delegated are the same ones as used for the persons that the KWer can manage in the personal social network management workspace. The meeting management perspective consequently accesses the same set of persons as managed in the task management

perspective. And in addition, action items defined in a meeting protocol occur as tasks in the task management perspective and can be directly delegated using this perspective's functionality.

These workspaces are positioned besides the prevailing desktop metaphor. The KWer can switch between the perspectives depending on the particular personal process or higher-level task to be executed. Each workspace targets a specific personal process or a smaller part, a higher-level task. Ideally, for each major and important personal process a specialized workspace exists. A workspace groups a set of functionalities that support the KWer working on a specific personal process or a higher-level task. Parts of a workspace, i.e., selected functionality, can be re-used in other workspaces. For example the visualization of the personal social network can be used in several workspaces to select persons although it is mainly part of the Personal Social Network workspace.

Architecture-wise these functional workspaces are independent of the underlying operating system through the use of the domain-specific services. The domain-specific services act as controller to read and manipulate the unified information model directly from the workspaces and applications consuming it.

Of note thereby is that the unified personal information model not necessarily must be the single source of information for an application. An application may use both personal information and other proprietary information to provide its functionality to the KWer. The domain-specific services cover the personal activity and personal-information related part.

8.2 Domain-Specific Services

Domain-specific services provide controlling business logic for applications on the application layer in the form of supporting functionality and commonly handling the access to a corresponding perspective of the KWer's unified information model. We implemented domain-specific services targeting the two personal supporting activities task management and meeting management, see Figure 88.

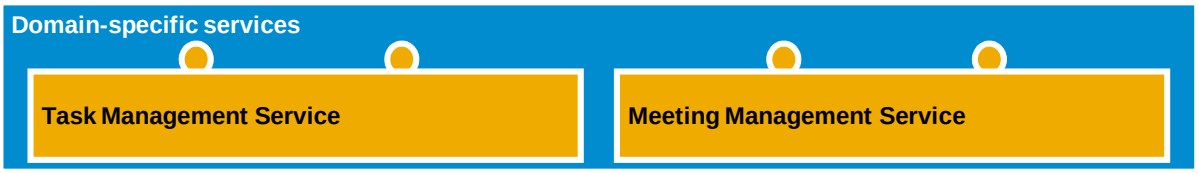


Figure 88: Domain-specific services provide activity-specific support functionality.

There are particular services for defined types of a KWer's activities. The services include functionality needed to support these activities. This includes, but is not limited to, the management of a corresponding perspective on a KWer's unified personal information. Typically focusing on a major entity of the information model, as for example tasks or people, the particular domain-specific service handles this perspective and the major entities and contains business logic needed to process these items. For example, for tasks there are service methods that create a new task, retrieve tasks by different criteria or methods to delegate a task.

The domain-specific service encapsulates the domain-specific business logic in a way that domain-knowledgeable KWers can apply these services without being experts in personal information management. As this domain-specific business logic is encapsulated in a service it can be re-used at every workspace or workspace-level application that needs to access an entity of the information model.

On the implementation side of the domain-specific services, their business logic interfaces core services provided by the basic PIM system such as, e.g., storage and communication. The domain-specific services' business logic interfaces the basic PIM system by invoking its services like, e.g., a PIM service to read and write the unified information model and a data wrapper for handling desktop information elements like files and emails. In our implementation with the Nepomuk semantic desktop as basic PIM system, they interface the core system services like for example PimoService [Sauermann&Klinkigt, 2009] to read and write the unified personal information model and the Aperture DataWrapper [Aduna B.V.&DFKI GmbH, 2005] as indexer for desktop information elements like files and emails.

8.3 Unified Personal Information Model in the PIM system

We use a unified representation of the KWer's personal information cloud as a *common abstract information model*. This model is at the heart of the reference architecture and is stored in the repository of the basic PIM system, see Figure 89.

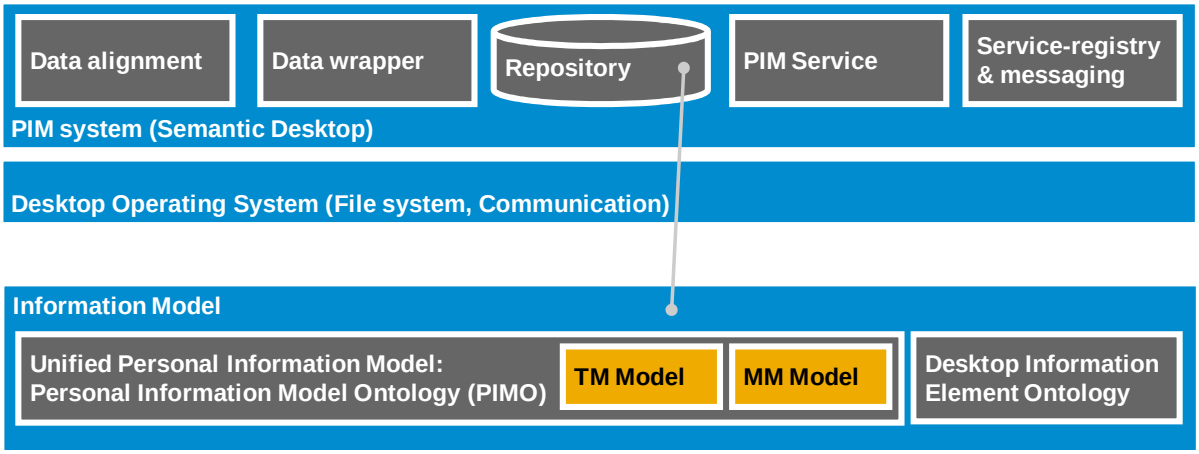


Figure 89: Unified personal information model in the PIM system.

We use a formalized representation of this model, the Personal Information Model Ontology (PIMO) [Sauermann et al., 2007] to enable machine processing on it by the domain-specific services, see section 10. It features a unified representation through the RDF representation and it is integrated along the KWer's personal things.

This represents the KWer's mental model, i.e., the personal, subjective view on the world. It is fully user-owned, i.e., only the KWer intentionally populates it with the information entities that occur in her personal, subjective view on the world.

These information entities of the PIMO are the domain-specific entities of the personal information cloud that the KWer deals with during pursuing her activities, i.e., like for example people, tasks, locations, events or topics. The KWer can explicitly interlink the entities with each other which are related in the KWer's view, e.g., because they were helpful for the same task. This way, in the end not every entity is related to every other entity. The KWer can link as well to existing desktop information elements, like, e.g., files, emails and bookmarks, to indicate a relation to this information. In addition, the KWer can extend the personal information model to for example categorize existing people into different groups.

8.4 Advantages and Limitations of the Reference Architecture

Through using the proposed reference architecture, developers gain number of benefits. First, it significantly reduces the complexity of implementing applications.

Offering a domain-specific interface reduces the complexity to build an application by opening the development of personal activity support applications to domain-knowledgeable developers while omitting the need for deep personal information management expertise. The domain-specific interface of the domain-specific services allows domain-knowledgeable developers to efficiently handle unified personal information. A domain-specific interface hides the underlying personal information management technology details. In comparison to a generic personal information management interface this saves domain-knowledgeable developers getting into the internals of personal information management and related technology.

A *common PIM system infrastructure* reduces the complexity to build a personal activity support applications by replacing the need for applications and the domain-specific services to each redundantly implement a proprietary stack for integrating desktop information objects, see Figure 90. The common PIM system infrastructure underlies the applications and the domain-specific services. The PIM system, i.e., the semantic desktop, provides services and models to manage existing desktop information elements and to integrate them with the unified information model. This means that the PIM system infrastructure is re-usable instead of each application building an own management stack. This replaces the need for applications and the domain-specific services to each redundantly implement a proprietary stack for integrating desktop information objects, significantly reducing the complexity of implementing applications.

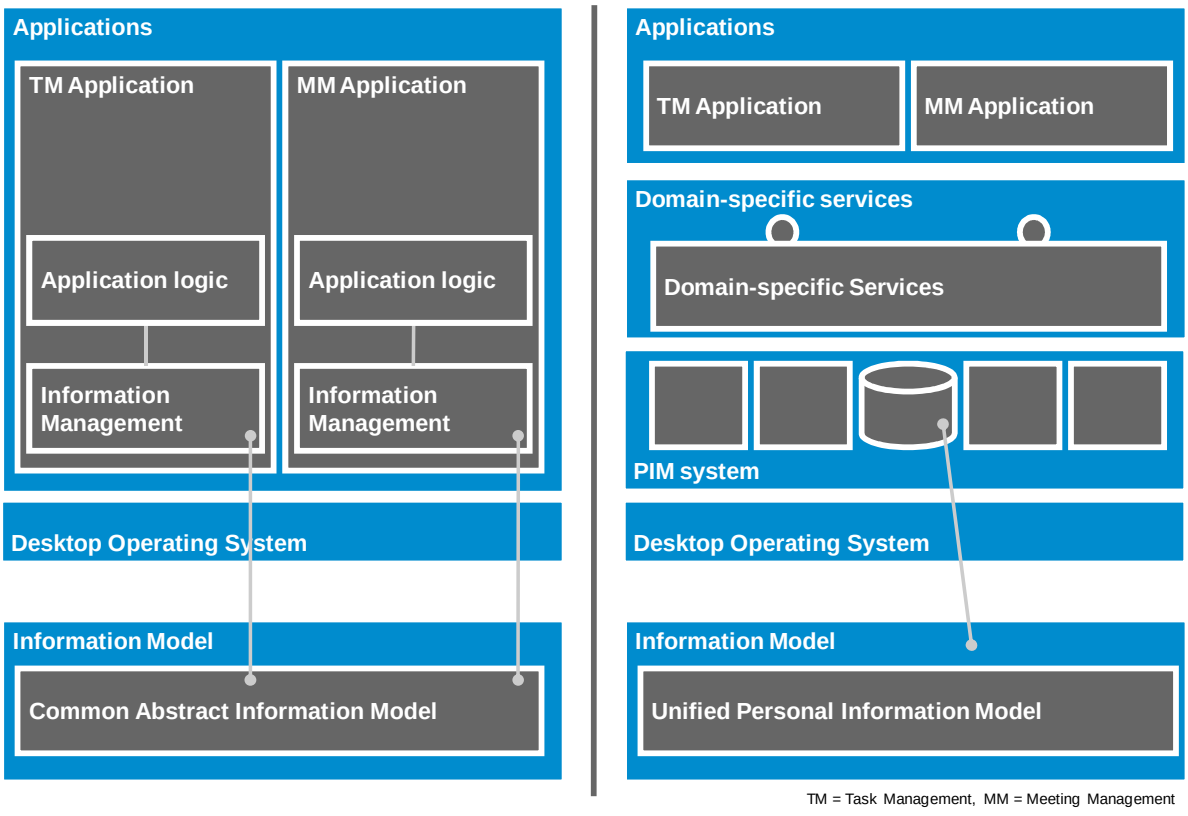


Figure 90: State-Of-The-Art Research (left), Functional Workspaces on PIM system (right).

Second, using this reference architecture significantly improves the capabilities that developers can implement into personal activity support applications.

An *increased flexibility for building applications* can be realized as existing business logic for certain KWer activities can be re-used and only the user interface needs to be re-developed and designed. The domain-specific services provide a number of services to realize the functionality needed for supporting a KWer's activities like, e.g., task delegation to send off a task to other people. As long as these services are available, developers can quicker develop new applications featuring an improved user interaction compared to when they need to re-develop the full personal information management stack. The service-oriented desktop architecture of the Nepomuk semantic desktop PIM system facilitates this re-use further as it keeps the available services in a service registry allowing other applications to discover and use these services. This enables multiple applications that are implemented in different programming languages to invoke the provided services. By invoking services, multiple applications can, e.g., query and write to the common unified personal information model.

The domain-specific services ensure a *consistent handling of personal information across applications*. When multiple applications access, i.e., query and write to the common unified personal information model, a common set of application logic in the domain-specific services facilitates the handling of a particular personal information perspective like, e.g., the domain-specific services for personal task management, see section 10. According to our experience, despite having a formalized and shared conceptualization in place, i.e., an ontology, there is still some ambiguity left when using this information programmatically. For example, some meaning of information is interpreted different by applications, e.g., how to interpret a task description. This leads to incompatibilities across applications. In this regard, using only a unified information model is not enough for efficiently using this information across applications. Using instead a common set of services, and thus using shared application logic, standardizes the usage of the information. This helps to prevent ambiguity when using the information across applications.

The presented reference architecture is *platform independent* insofar as its domain-specific services use only the PIM system services and don't directly access operating system services. Using the domain-specific services, the user experience of each workspace and application can be designed solely relying on the domain and thus completely separate from the underlying system paradigm. The PIM system provides services for a unified access to the desktop operating system for communication capabilities and desktop resources like the file system or emails. Each commercial desktop system that can implement the PIM system services can leverage the presented approach. The Nepomuk semantic desktop PIM system is implemented for example on Microsoft Windows, Apple Mac OS and Linux KDE.

The use of a *centralized personal information model can be a possible limitation* of the approach. A central information model can lead to performance problems when many applications access it. In our tests we didn't experience problems, however it can be the case in large-scale deployments. In addition, as noted before, an application doesn't need to exclusively store all information in the central information model.

9 Domain-Specific Adaptation of Unified Personal Information Model

The unified personal information model needs to be tailored to support a particular use case and related functionality, see Figure 91 for an overview. Depending on the use case, a unified personal information model needs to keep different information and cater for information needs in different levels of detail.

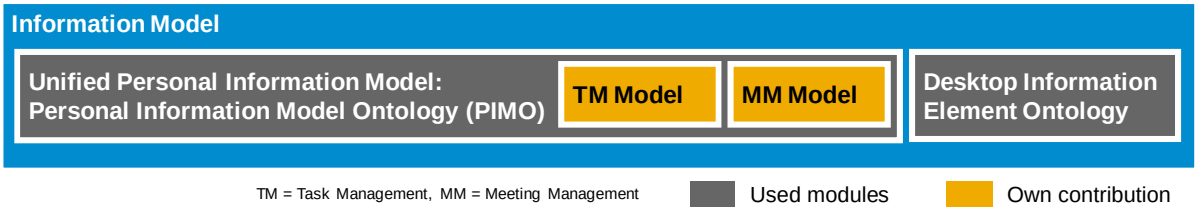


Figure 91: Unified personal information model and domain-specific extensions.

We implemented adaptations and extensions to the PIMO unified personal information model for the two use cases of personal task management and personal meeting management. Further use cases can be covered in a similar way drawing upon the here presented models and findings. In the following, we explain these models in detail including their relation to the corresponding use case.

9.1 Task Management Model – Task Model Ontology (TMO)

The Task Management Model (TMM) is a conceptual representation of tasks and related information for use in personal task management applications for KWers. It is part of the KWer's unified personal information model. The Task Model Ontology (TMO) represents an agreed, domain-specific information model for tasks and covers personal task management use cases, see section 3.2.

9.1.1 Domain-Specific Task Model Requirements

When conducting personal task management as described in the use cases above, the applications implementing task management functionality deal with tasks and further related personal information. This leads to several requirements regarding the applied information model, as shown in an overview in Table 13.

#	Situation	Requirements
Personal information-perspective		
1	Tasks are a part of this personal information cloud and require several attributes	Cover task information in task model and integrate it into the unified personal information model
2	Several applications represent different perspectives of the tasks and the personal information cloud	Applications access and use a common personal information model
3	Desktop information entities like emails and files are task-relevant information	Integrate and use desktop information objects like emails and files

Table 13: Overview on task model Requirements for Personal Task Management.

The KWer regards all personal information as a single body of information, see section 2.3.2, a personal information cloud including tasks. Information-wise, the use cases focus on an individual KWer's personal tasks and further related personal information. Tasks are a part of this personal information cloud and are used in the context of it, like, e.g., with persons and topics. Particular task attributes enable personal task management use cases, e.g., a priority attribute enables the applications to manage the task's priority. This leads to

- *Requirement 1: cover task information in task model and integrate it into the unified personal information model*

The personal task management use cases can be implemented in several applications, where each application represents with its task management functionality a different perspective on the KWer's personal information. Thus, several applications manipulate and access tasks and further personal information. This leads to

- *Requirement 2: Applications access and use a common integrated / unified personal information model*

Furthermore, tasks are used in the context of desktop information entities like emails and files. These are task-relevant information as they represent important information objects in the KWer's personal task management. This leads to

- *Requirement 3: Integrate and use desktop information objects like emails and files*

9.1.2 Task Model Ontology (TMO) – Task and Information Model

The Task Management Model Ontology (TMO) is a conceptual representation of tasks for use in personal task management applications for knowledge workers (KWers). The Task Model Ontology (TMO) represents an agreed, domain-specific information model for tasks and covers personal task management use cases.

As domain model the TMO *models the tasks a KWer deals with in the context of the KWer's other personal information* (requirement 1). It thereby represents an activity-centric view on the KWer's personal information, as it models beneath tasks as well the relations to task-relevant personal information. As tasks are an integral part of the KWer's personal information cloud, the TMO is integrated into the Personal Information Management Ontology (PIMO) that represents the KWer's personal information cloud.

The task model captured in the TMO *has a formal, shared representation as ontology* to make the task model accessible from different applications (requirement 2). The formalized representation enables the application-independent representation of the task information as the task model is explicated, shared and available with the task information. This way, several applications can access and use the TMO and the common unified personal information model.

The TMO *enables the attachment of desktop information objects to tasks* (requirement 3). Using representations of the Nepomuk Information Elements (NIE) ontologies [Nepomuk Consortium, 2009c] the relation to files, emails and bookmarks can be realized.

The TMO is released to the public as part of Nepomuk Ontology framework [Nepomuk Consortium, 2009h]. The TMO ontology itself is published [Brunzel et al., 2008] along with extensive documentation. As well, the document [Grebner et al., 2007c] presents the state-of-the art in task models and the semi-formal description of the ontology with links to the supported use cases.

The current version is considered as stable at the core, only minor attribute changes can be expected. It is in use in several personal task management applications, like, e.g., [Grebner et al., 2008], [Brunzel&Mueller, 2008].

9.1.2.1 TMO Concepts and Attributes – Overview

The TMO models a task concept and further task-related concepts needed for the use cases. Along with the task concept, attributes are modeled in the TMO.

The core task management use cases are covered in the TMO. Further use cases which target more specific task management related use case like task knowledge management are covered in a TMO Extension model, so-called extended task model (TMOE).

Figure 92 shows a simplified overview on the TMO and TMOE. Thereby, it shows first the covered use cases like, e.g., personal task management, second the modeled concepts, like e.g. a task, and third the corresponding example attributes of the modeled concepts, a task's priority.

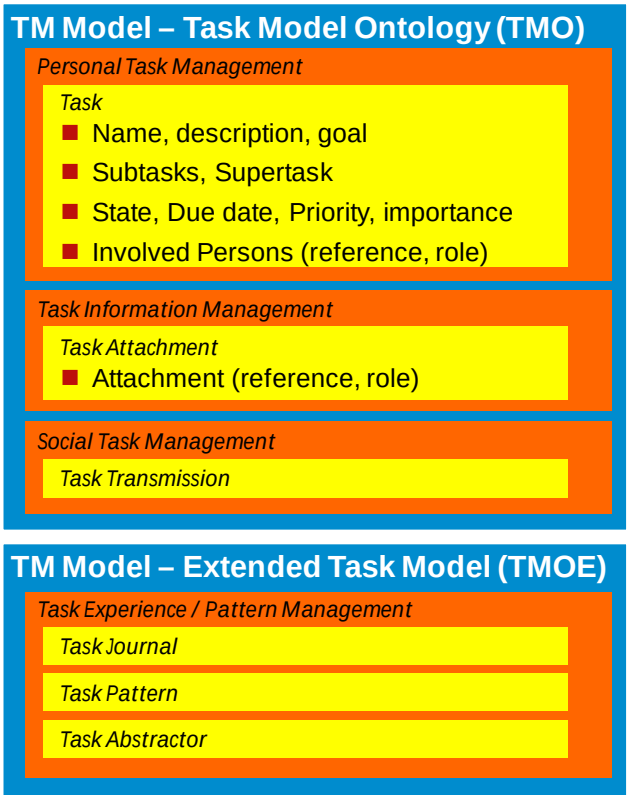


Figure 92: Overview TMO & TMOE Concepts clustered by use case, with selected attributes.

The task representation, i.e., the 'tmo:Task' class, is the most elaborated concept in the TMO. Table 14 lists its properties listed by the respectively supported use case. As well, Table 14 lists further classes included in the TMO, as well listed by their supported use case.

Use Case / Application Area	tmo:Task properties	Further TMO Classes
Basic task handling	tmo:taskName tmo:taskDescription tmo:subtask tmo:goal tmo:taskState	tmo:TaskState
Task list management	tmo:containsTask tmo:dependency	tmo:TaskContainer tmo:TaskDependency
Task priority management	tmo:priority tmo:importance tmo:urgency	tmo:Priority tmo:Importance tmo:Urgency

Task time management	tmo:dueDate tmo:actualCompletion tmo:lastReviewDate tmo:nextReviewIntervall tmo:actualEndTime tmo:actualStartTime tmo:taskEffort	tmo:TaskEffort
Task Information Management	tmo:attachment tmo:personInvolvement tmo:abilityCarrierInvolvement tmo:contextThread	tmo:Attachment tmo:AttachmentRole tmo:PersonInvolvement tmo:PersonInvolvementRole tmo:AbilityCarrierInvolvement tmo:AbilityCarrier tmo:AbilityCarrierRole tmo:Skill tmo:Role
Social Task Management	tmo:taskTransmission	tmo:TaskTransmission tmo:TransmissionType tmo:TransmissionState tmo:TaskPrivacyState tmo:Delegability
Task Knowledge Management	tmo:logEntry tmo:taskSource	

Table 14: Overview TMO concepts and tmo:Task attributes clustered by covered use cases.

The next section, section 9.1.2.2, gives further details on the classes tmo:PersonInvolvement, tmo:AbilityCarrierInvolvement, tmo:Attachment and tmo: TaskDependency.

The tmo:TaskTransmission class and the related classes tmo:TransmissionType, tmo:TransmissionState, tmo:TaskPrivacyState, tmo:Delegability enable support for task delegation and task transmission. Task delegation refers to the transfer of a task to another KWer along with handing over the responsibility for this task. Task transmission refers to the transfer of a task to another KWer without handing over the responsibility. These classes keep the necessary management information and keep a record of the transferred task in the course of a task delegation or task transmission.

9.1.2.2 TMO Concepts and Attributes – Detailed Presentation of Selected Aspects

The core class of the TMO is the class tmo:Task. The tmo:Task is a subclass of pimo:ProcessConcept. The inheritance hierarchy of the tmo:Task is shown in Figure 93 and Figure 94. Figure 94 shows in addition all tmo:Task properties. The applied notion first lists the property name, whether it is an instance including its cardinality indicated by a star and the property type.

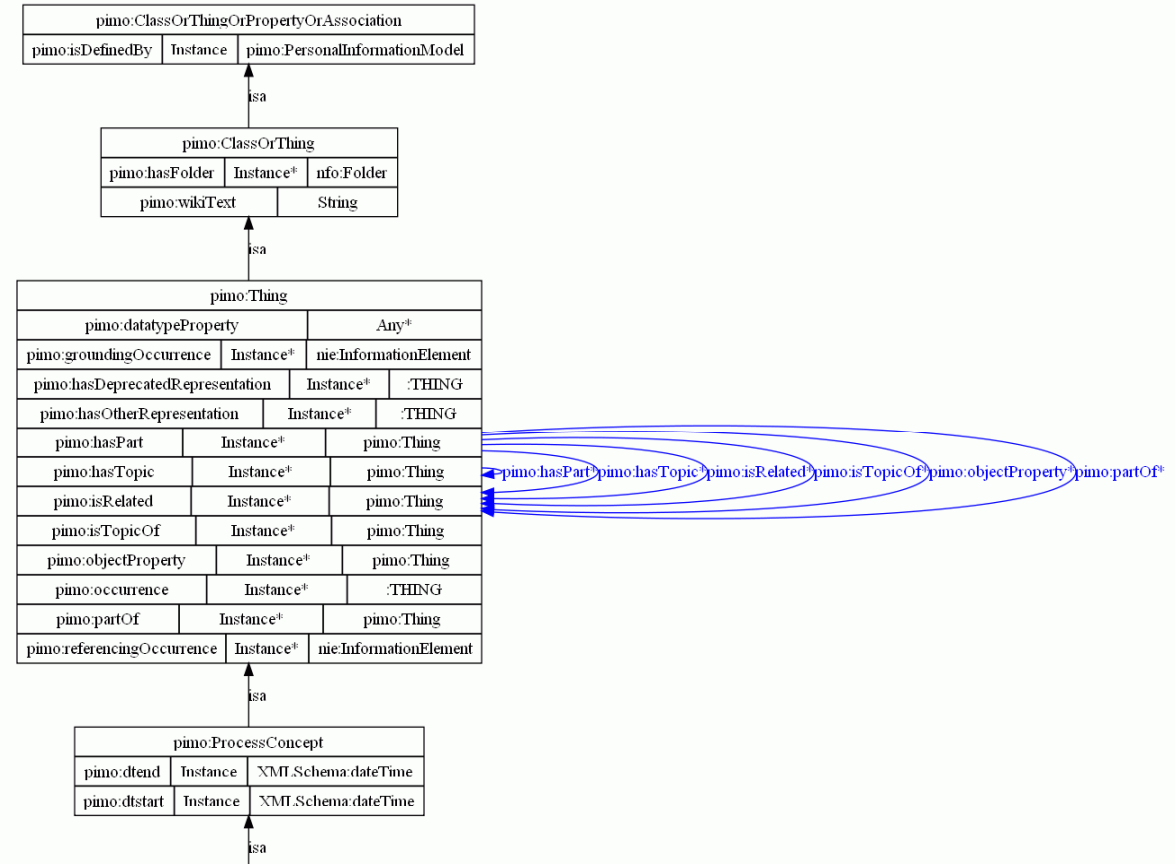


Figure 93: TMO tmo:Task inheritance hierarchy [Brunzel et al., 2008].

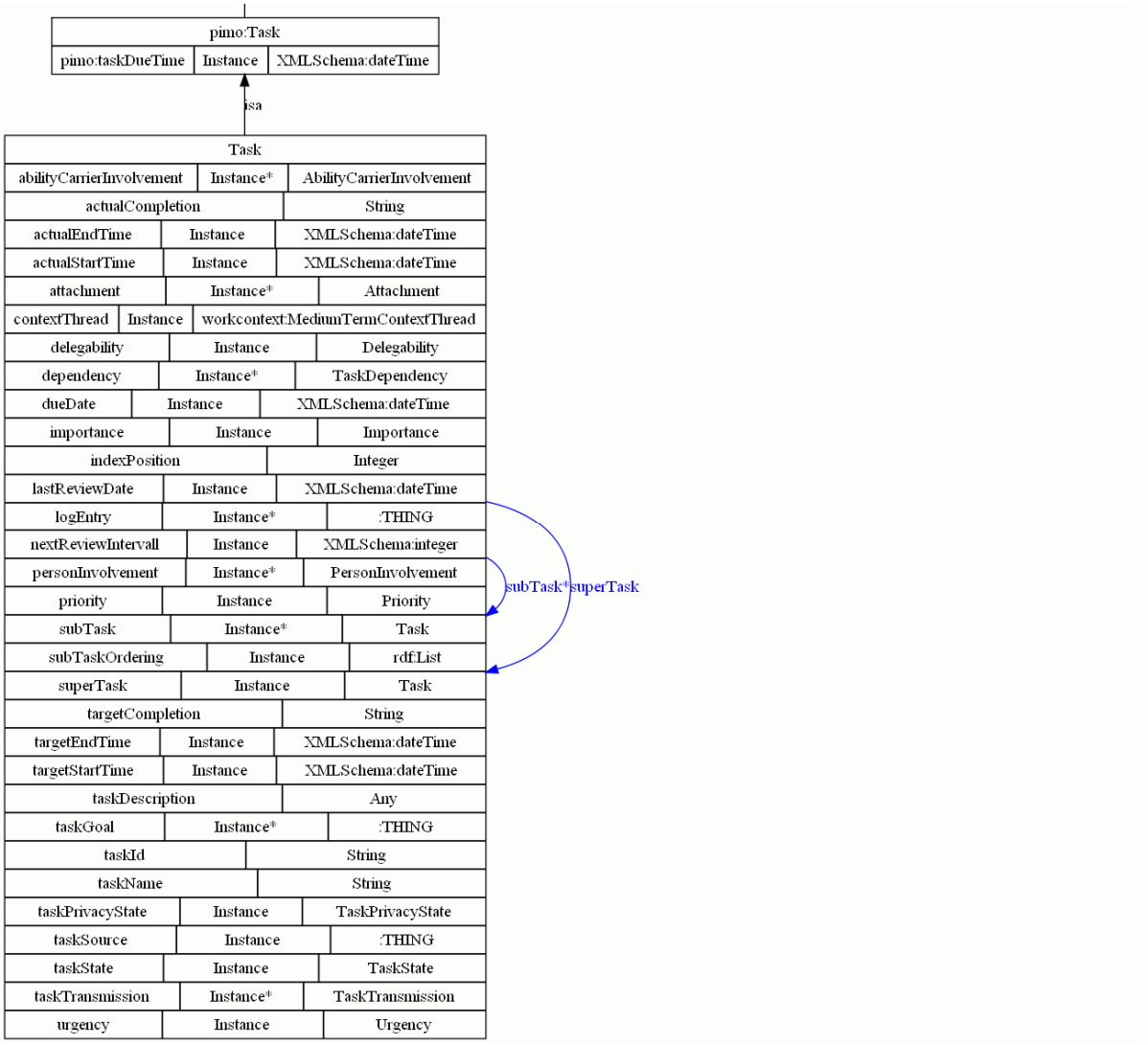


Figure 94: TMO tmo:Task properties [Brunzel et al., 2008].

Some classes have been modeled using a *role based modeling approach*. This enables the modeling of n-ary relations. This modeling approach has been applied to task attachments, the involvement of persons, the involvement of actors and resources (referred to as AbilityCarriers) as well as task dependencies (dependency between two tasks). Figure 95, Figure 96, Figure 97 and Figure 98 show the respective modeling artifacts.

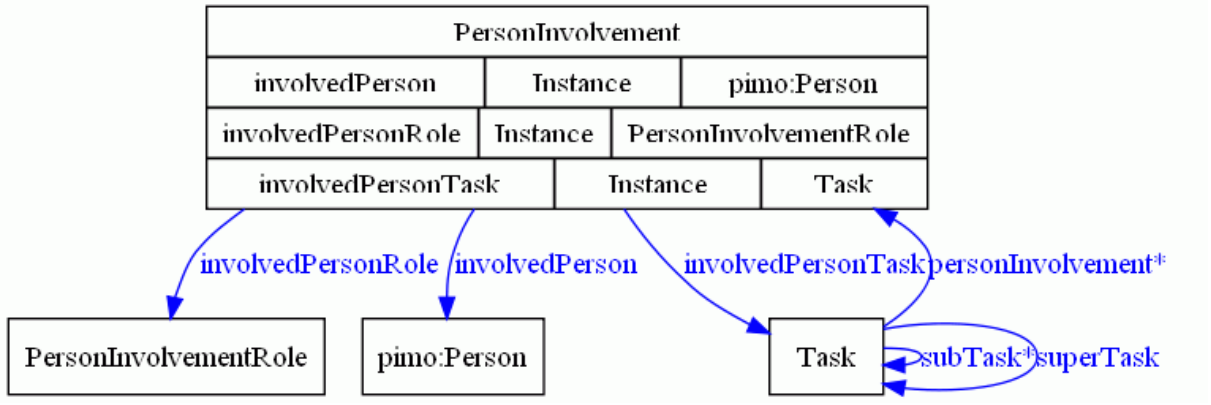


Figure 95: Modeling of tmo:PersonInvolvement [Brunzel et al., 2008].

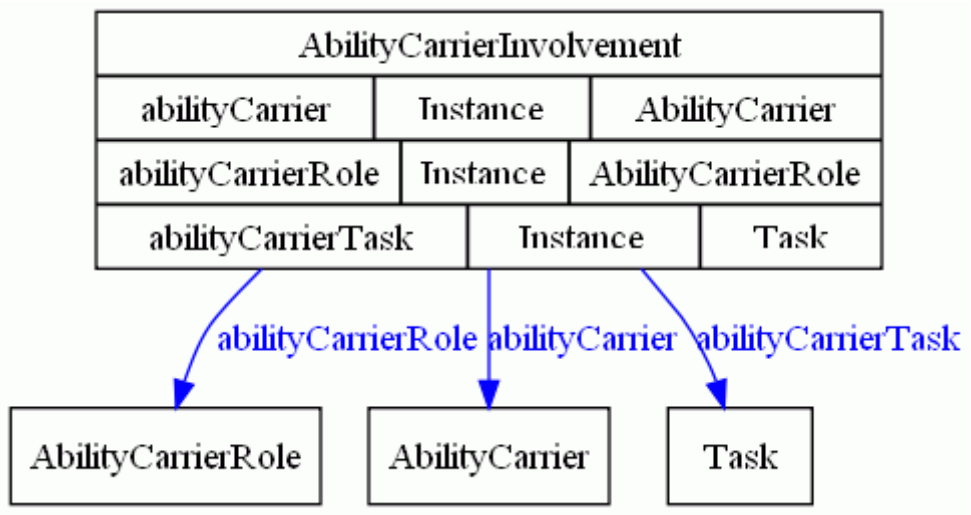


Figure 96: Modeling of tmo:AbilityCarrierInvolvement [Brunzel et al., 2008].

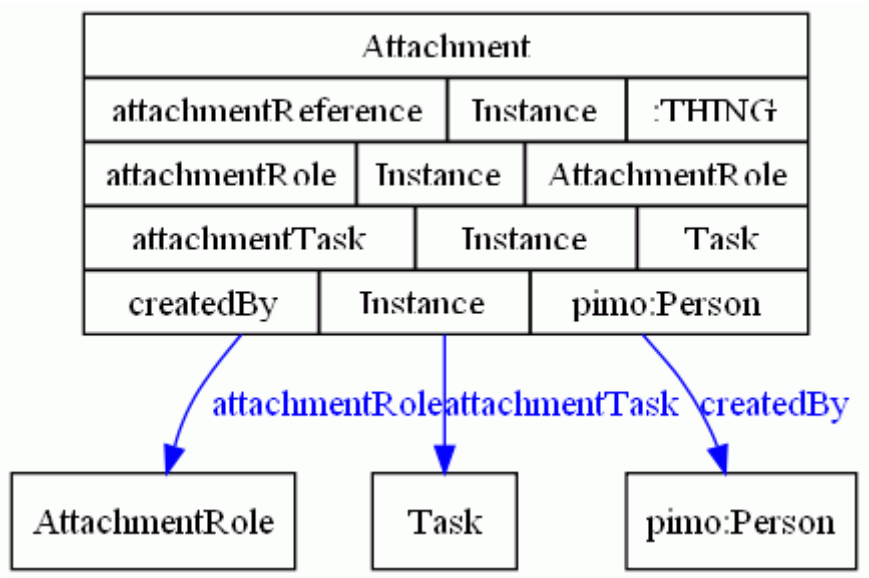


Figure 97: Modeling of tmo:Attachment [Brunzel et al., 2008].

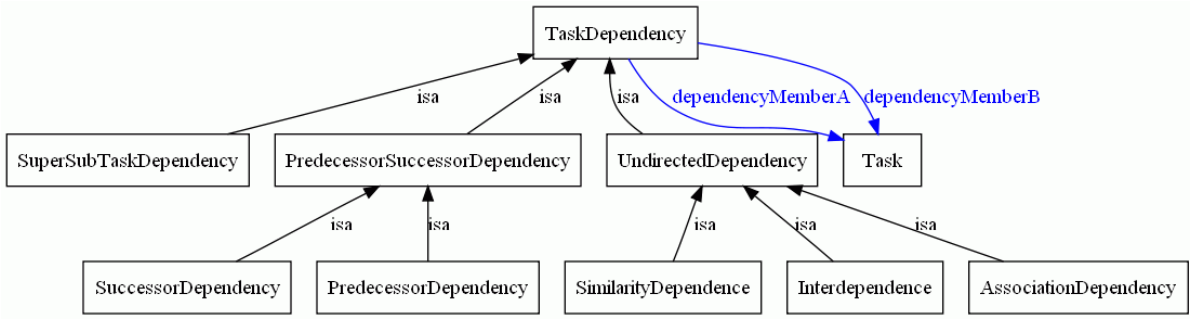


Figure 98: Modeling of tmo: TaskDependency [Brunzel et al., 2008].

9.1.3 TMO Extensions Facilitate Modeling for Further Specific Use Cases

KWers and application developers can extend the TMO to cover further use cases beyond core personal task management. When targeting specific task management use cases, this information is not directly relevant for core personal task management use cases. Capturing the specific information in a separate information model and thus a separate Ontology, i.e., TMO Extensions, allow a modular composition of the task information model according to envisioned use case.

These TMO extensions can for example support task knowledge management for tasks with task patterns [Riss et al., 2005] [Riss&Grebner, 2008]. The TMO Extension to support for Task Knowledge Management is called TMOE and covers concepts for a Task Journal [Riss&Grebner, 2008] and task patterns [Riss et al., 2005]. As well, the TMOE acts as blueprint for further TMO extensions showing developers a practical example on how to extend the TMO.

9.1.4 TMO Integrated into the Nepomuk PIM System's Ontology Framework

The TMO is designed for use as part the PIM platform Nepomuk [Nepomuk Consortium, 2009a]. Therein, the Nepomuk Ontology framework [Nepomuk Consortium, 2009h] provides ontologies for conducting personal information management in particular on the desktop. For example this includes the Personal Information Management Ontology (PIMO) [Sauermann et al., 2007] and the Nepomuk Information Elements (NIE) ontologies [Nepomuk Consortium, 2009c]. Figure 99 shows the TMO positioned in the Nepomuk Ontology Framework.

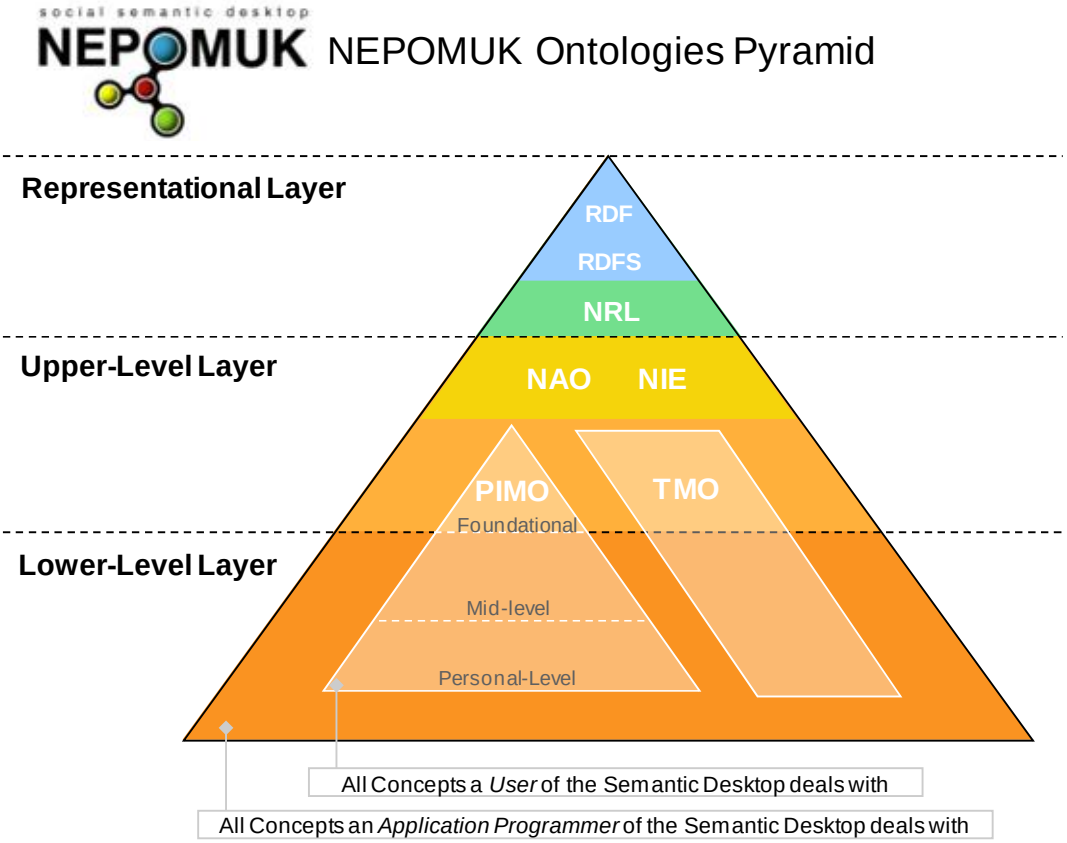


Figure 99: TMO in the Nepomuk Ontology Framework context, based on [Ong et al., 2007a, p. 12].

The TMO is an extension of the Personal Information Management Ontology (PIMO), which represents the KWer's personal information cloud. Whereas the PIMO provides the general personal information model, the TMO is a specialized extension of the PIMO ontology focusing on tasks and the support of personal task management applications.

The TMO connects tasks with desktop information elements like, e.g., emails and files using references to the Nepomuk Information Elements (NIE) ontologies [Nepomuk Consortium, 2009c]. Using NIE references the KWer and application developer gain desktop integration support for tasks, i.e., the task can reference desktop information elements like emails and files using NIE representations.

The Nepomuk Annotation Ontology (NAO) [Nepomuk Consortium, 2009i] is used for example to keep a task label, see section 9.1.6.2 below for details.

However, applications can use the TMO as well without implementing the full Nepomuk ontology framework to support personal task management, depending on the use case.

9.1.5 Methodological Ontology Engineering Resulted In a Mature TMO

The here presented TMO evolved from over 2.5 years of intense development of personal task management applications. In the accompanying ontology engineering process the TMO has undergone a continuous refinement and several beta releases have been used.

The TMO ontology engineering process bases on the OntoKnowledge methodology [On-To-Knowledge Project, 2009]. We applied the therein defined "Knowledge Meta Process" for the TMO development in the domain of the KWer's personal task management activity, see Figure 100. The Knowledge Meta Process leads to "an ontology based KM application" [Sure&Studer, 2002, p. 35]. It consists of the phases "Feasibility Study", "Kickoff", "Refinement", "Evaluation" and "Application & Evolution" each with associated outcomes and decisions.

The "Knowledge Meta Process" is one process in the development of knowledge management (KM) applications besides processes for "Human Resource Management" to get people on board for the newly developed application and "Software Engineering" for the actual development of the software application [Sure&Studer, 2002, p. 34], see Figure 100 on the right side.

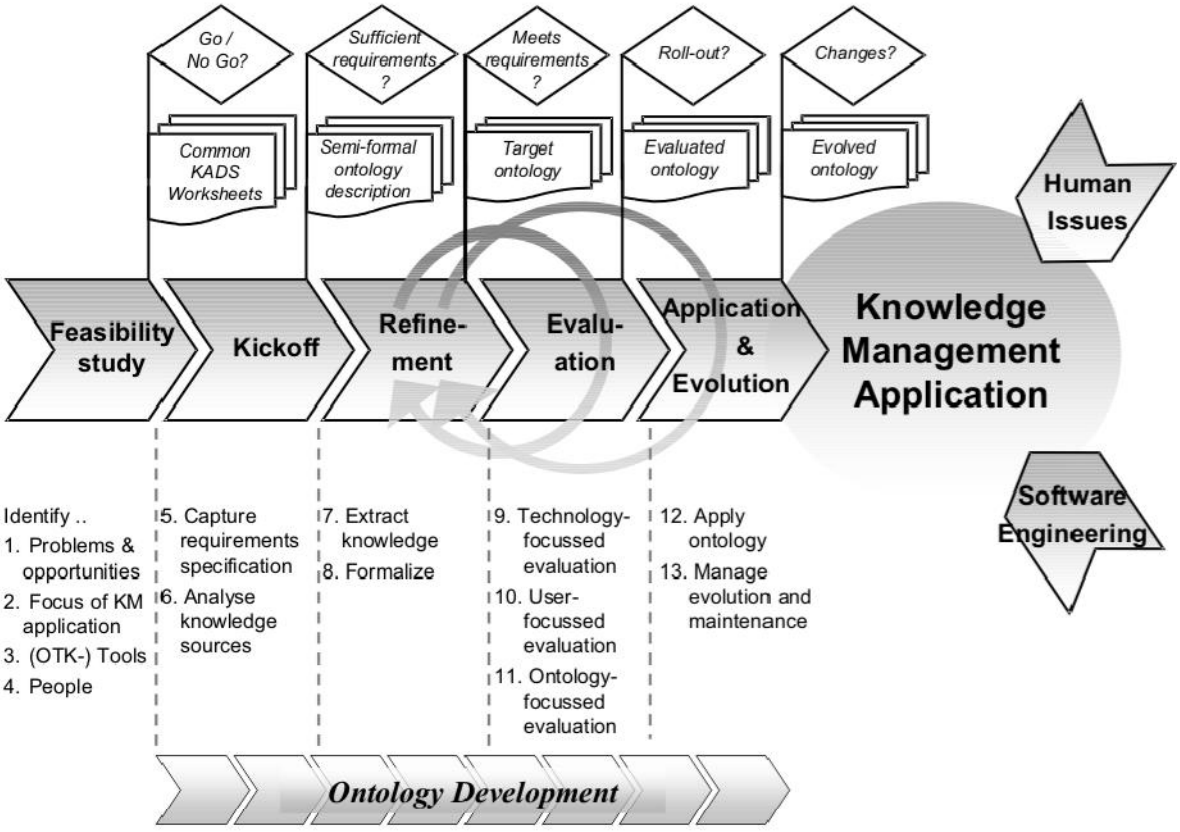


Figure 100: Knowledge Meta Process [Sure&Studer, 2002, p. 36].

The "Knowledge Meta Process" serves as blueprint for the TMO development process. In some phases we adapted the proposed process to the match the particular situation.

9.1.5.1 Feasibility Study Phase

The "feasibility study" phase aims to "*identify (i) stakeholders* related to a project divided into users of the system and supporters of the system, (ii) *use cases* describing the usage scenarios which we call user driven use cases and (iii) use cases supporting these user driven use cases which we call supporting use cases." [Sure&Studer, 2002, p. 36].

For the TMO, we analyzed a KWer's supporting activities of personal task management and related personal information management. This analysis has been conducted as an extensive user study of KWers in a research department [Grebner et al., 2006], including users, their activities and the surrounding organization as proposed by the CommonKADS methodology [Schreiber et al., 1999].

As outcome, we indentified the stakeholders as knowledge workers who work for example in a research department, see section 2.1 and [Grebner et al., 2006] for details. The usage scenarios, i.e., user driven use cases, are defined both by analyzing the KWer's personal task management activity and by showing the to-be scenarios of the anticipated tool usage, see [Grebner et al., 2006]. These usage scenarios contain as well supporting use cases.

"Given that a "GO" decision finalizes the feasibility study, the results as described above serve as input for the kick off phase of the ontology development" [Sure&Studer, 2002, p. 38].

9.1.5.2 Kickoff Phase

The "kickoff" phase aims to *specify requirements* for the ontology. Sure and Studer propose the creation of an ontology requirements specification document (ORSD). The "ORSD describes what an ontology should support, sketching the planned area of the ontology application and listing, e.g., valuable knowledge sources for the gathering of the semi-formal description of the ontology" [Sure&Studer, 2002, p. 38]. The purpose of the OSRD is to "guide an ontology engineer to decide about inclusion and exclusion of concepts and relations and the hierarchical structure of the ontology" [Sure&Studer, 2002, p. 38]. Sure and Studer further mention that in "this early stage one should look for already developed and potentially reusable ontologies" [Sure&Studer, 2002, p. 38].

As outcome, we covered the information contained in the OSRD document regarding the TMO in [Grebner et al., 2007c] and [Grebner et al., 2006]. The goal of the ontology is to enable KWers to manage their personal tasks and related information as well as involved people. Other existing Nepomuk ontologies, to which the TMO serves as extensions, represent design guidelines for the TMO. The domain and scope is personal task management. The ontology supports personal task management applications, like for example personal to-do lists and task-centered personal information management applications. Kasimir, see section 5, is one example of such an application. The used knowledge sources are the user study in the personal task management domain [Grebner et al., 2006], interviews with domain experts on task modelling and a thorough research of state-of-the art task models [Grebner et al., 2007c]. The users and usage scenarios of the TMO are knowledge workers, e.g., researchers, managing their personal tasks. In addition to the information contained in the OSRD document, [Grebner et al., 2007c] contains a semi-formal description of the ontology, i.e. a graph of named nodes and named, directed edges along descriptive text explaining the purpose of the nodes and edges.

The decision to move on to the refinement of the ontology was taken by the ontology engineers after ensuring a sophisticated coverage of the domain thorough several interviews with domain experts.

Used tool support in this phase consisted of the rapid prototyping of the TMO ontology in Protégé [Noy et al., 2003].

9.1.5.3 Refinement Phase

The "refinement" phase aims to iteratively conduct knowledge extraction and formalization to refine the ontology. This includes gathering a seed taxonomy, developing the seed ontology which is followed by the conceptualization and formalization thereof.

Sure and Studer mention that during "the kick-off and refinement phase one might distinguish in general two concurrent approaches for modeling, in particular for knowledge extraction from relevant knowledge sources: top-down and bottom-up" [Sure&Studer, 2002, p. 48]. "One starts by modeling concepts and relationships on a very generic level. Subsequently these items are refined. This approach is typically done manually and leads to a high-quality engineered ontology." [UnaMesa Association, 2009, p. 38]. For the TMO we followed a top-down-approach in modeling the domain. Starting with the core concept of a task, subsequently related concepts like, e.g., for task attachments have been developed.

"Depending on the application that has to be supported one has to choose an appropriate representation language" [Sure&Studer, 2002, p. 49]. As the TMO is part of the Nepomuk Ontology Framework, see section 9.1.4, RDF/S has been set as the representation language.

"The refinement phase is closely linked to the evaluation phase. If the analysis of the ontology in the evaluation phase shows gaps or misconceptions, the ontology engineer takes these results as an input for the refinement phase" [Sure&Studer, 2002, p. 49]. The TMO underwent a *continuous improvement* throughout a time span of two years time within a closely chained refinement and evaluation phase. In an iterative process the ontology has been revised to match the increased understanding of the personal task management domain. This mostly affected concepts in correspondence with the development of application modules and their changes due to evaluation results.

"The outcome of this phase is the "target ontology", that needs to be evaluated in the next step." [Sure&Studer, 2002, p. 50]. For the TMO the decision to move on into the next phase has been taken when the (newly) imposed requirements were met by the ontology engineers. In the subsequent phase software developers adapted the prototype applications to fit the modified TMO.

9.1.5.4 Evaluation Phase

The "evaluation" phase aims to verify the utility of the developed ontology. There are three types of evaluation, i.e., technology-focused, user-focused and ontology-focused [Sure&Studer, 2002, p. 53].

Sure and Studer mention for *technology-focused evaluation* two "main aspects: (i) the evaluation of properties of ontologies generated by development tools, (ii) the evaluation of the technology properties, i.e. tools and applications which includes the evaluation of the evaluation tool properties themselves" [Sure&Studer, 2002, p. 53]. For the TMO, we continuously checked the consistency of the TMO generated by Protégé. On the technology properties side we continually tested the TMO in the target application environment, i.e., with the Task Management Services, see section 10.1, directly interfacing the TMO. In addition, we ensured the possibility to integrate the TMO into all desktop applications interested into supporting personal task management activities, beyond the mainly operating Task Management Services, see section 10.1. For example, Distreebute [Brunzel&Mueller, 2008] accesses the TMO as well.

Sure and Studer mention for *user-focused evaluation* that ontology "engineers need to check, whether the target ontology itself suffices the ontology requirements specification document [...] analyzed in the kickoff phase of the project." [Sure&Studer, 2002, p. 55]. They continue that therefore "the ontology is tested in the target application environment. A prototype should already show core functionalities of the target system. Feedback from beta users of the prototype may be a valuable input for further refinement of the ontology" [Sure&Studer, 2002, p. 55].

For the TMO, we continuously evaluated the ontology through the applications offering personal task management support which are targeted to end-users. The mainly evaluated application using the TMO is Kasimir, see section 5. Kasimir builds on the Task Management Service functionality. The respectively conducted evaluations, see section 12, gathered user feedback, assessed it and incorporated it then in a follow-up refinement phase into the TMO.

We didn't apply formal evaluation methodologies for ontologies as proposed by Sure and Studer mention for *ontology-focused evaluation* [Sure&Studer, 2002, p. 55].

"The outcome of this phase is an evaluated ontology, ready for the roll-out into a productive system. However, based on our own experiences we expect in most cases several iterations of "Evaluation – Refinement – Evaluation" until the outcome supports the decision to roll-out the application." [Sure&Studer, 2002, p. 56].

The decision to conclude this phase is taken based on "whether the evaluated ontology fulfills all evaluation criteria relevant for the envisaged application of the ontology". For the TMO this has been the case after two years of iterative development. The TMO features a stable set of core concepts which didn't change anymore throughout the last iteration cycles. Changes are mainly to be expected for new concepts due to enlarged use cases.

9.1.5.5 Application & Evolution Phase

The "application & evolution" phase aims to apply and advance the ontology.

As *application* of the TMO, it is used on service-level by, e.g., the Task Management Services, see section 10.1. On application level, several personal task management applications like, e.g., [Grebner et al., 2008], [Brunzel&Mueller, 2008] use the TMO.

Sure and Studer "stretch that *evolution* of ontologies is primarily an *organizational process*. There have to be strict rules to the update/insert/delete processes of ontologies. We recommend, that the ontology engineer gathers changes to the ontology and initiates the switch-over to a new version of the ontology after thoroughly testing all possible effects to the application" [Sure&Studer, 2002, p. 65].

The TMO is hosted by the Nepomuk Consortium. The ontology engineers are responsible for implementing changes and corresponding TMO release management. However, any party interested in modifying the TMO can propose changes, see below at change management.

Change management for ontologies "is especially important when ontologies will be used in a decentralized and uncontrolled environment like the Web, where changes occur without co-ordination." [Sure&Studer, 2002, p. 66].

To govern the change management process, a public ticket system is in place for the TMO [Nepomuk Consortium, 2009j]. Through this ticket system, a community interested in task models for personal

task and information management can discuss the published TMO ontology, collect feedback on the TMO modeling and develop changes and possible evolvements.

Any party interested in extending the TMO can do this, e.g., through using the TMO Extension blueprints. After dissection and acceptance these extensions will be hosted on the TMO website.

9.1.6 Experiences, Design Considerations & Taken Design Decisions for TMO

This section presents some lessons learned and experiences made while developing the TMO. This includes as well the explanation of some design considerations and taken design decisions for the TMO. We gathered extensive experience in the personal information management (PIM) and personal task management domain throughout over 2.5 years of ontology development in this domain.

9.1.6.1 Ontology Development Process

In addition to continuously revising it, *yearly revision workshops* have taken place in order to completely revisit the whole ontology. The perspective-driven modifications that were incorporated throughout the year evolved the ontology but left the remaining parts mostly untouched. This lead to some inconsistencies, for example the naming conventions evolved or modeling guidelines evolved through the whole Nepomuk Ontology framework [Nepomuk Consortium, 2009h]. These workshops were held after the first and second year of development and enabled a cleaning up of the ontology.

The TMO has been *separated into core and extended concepts* to modularly cover the task management domain use cases. Over time, the planned use cases were realized step-by-step and additional use cases were added. This led to a broadened but at the same time more complex and less intuitive ontology. As a result, the ontology has been modularized and an extension system has been put in place at the second yearly revision workshops, as mentioned earlier. The TMO has been focused on core task management use cases whereas the first extension, the TMOE covers task knowledge management use cases. The TMOE thereby is the blueprint for further extensions and shows how the task model can be extended for additional task management use cases. For applications this enables a modular composition of only the TMO parts that are needed to cover the required use cases in the implementing application. This way, simple use cases don't need to deal with an overloaded ontology carrying extensive modeling content for not relevant use cases.

9.1.6.2 Coordinate and Govern a Distributed Ontology Development Process

An *ontology release management* of the TMO ensured the ordered development of task management applications. As the TMO evolved the applications using it needed to evolve too, i.e., when the TMO concepts and attributes change the implanting application has to be retested and if needed to be adapted. However, often the TMO evolved and the applications needed some time to migrate to the most recent version. As well, some applications worked intentionally stayed with a constant, older ontology version since this the implemented application functionality was tested and worked well. To mitigate risks due to the evolving ontology for applications using it, new TMO releases were introduced with each major change of the ontology. The release management is built on the versioning functionality of the SVN source code repository [CollabNet Inc., 2009] used to host the ontology. This way, applications can be built against a particular TMO release indicated by a particular SVN revision number.

The *use of the applied ontology editing toolset has been standardized* to avoid version conflicts with the ontology editor tooling, namely Protégé [Noy et al., 2003]. Throughout the TMO development process, several problems and inconsistencies occurred through using different versions of the

ontology editor Protégé. For example, saving the ontology with Protégé version 3.3 caused the ontology to not open any more in Protégé version 3.0. However, some functionality for editing the RDF based ontology was only available in Protégé version 3.0. The cause for these problems laid in general incompatibilities of Protégé's RDF plug-in which hasn't been migrated to the new plug-in framework introduced with recent versions of Protégé. As a measure to avoid these tool-induced inconsistencies, the use of the ontology editor tool Protégé has been standardized to the version 3.0. In general the learning is that that the use of the ontology editing toolset should be standardized, e.g., by organizational or community guidelines in order to minimize toolset-induced incompatibilities. At least, a list of compatible and tested ontology editor tools needs to be provided.

A *standardization and documentation of the RDF Access Framework Java class generation procedure* ensures the generation of error-free and functionally working TMO java classes for use in applications. RDFReactor [semanticweb.org, 2009a] is a RDF Access Framework that "views the RDF data model through object-oriented Java proxies" [semanticweb.org, 2009a] and helps Java developers in dealing with RDF. For each new TMO release the corresponding RDFReactor Java classes, called RDFSBeans [Nepomuk Consortium, 2009k], need to be re-generated to represent the then current TMO. The generation procedure involves several steps that first merge the involved ontologies into a temporary single RDFS file and then generates the RDFReactor java classes. In addition, both the merged ontology and the resulting RDFSBeans need to be corrected to be error-free. The reason for this is that the RDFReactor and RDF Access Frameworks in general cannot deal with multiple concept inheritance as it was used for several properties like, e.g., nao:hasTag. As workaround, the Nepomuk Annotation Ontology (NAO) [Nepomuk Consortium, 2009i] properties need to be changed manually in the ontology and additionally missing ontology concepts need to be appended in the generated RDFSBeans Java classes. To reduce errors in the generation process, a detailed procedure description explaining the ontology preparation and RDFSBeans generation has been put in place [Grebner&Brunzel, 2008b].

9.1.6.3 TMO Modeling Scope Evolved Along Use Case Understanding

The TMO modeling scope evolved along the use case understanding and the application implementation. Thereby, the TMO modeling constructs have been changed to reflect the improved understanding.

Some *planned concepts and attributes haven't been modeled* in the TMO or have been removed again after modeling them as the corresponding use cases didn't prove to be central or even were not needed.

- The concepts of Notification and Reminder were not modeled in the TMO as for notifying of new tasks and reminding of overdue tasks were not needed by the refined task management use cases. In additions notifications are handled by the Nepomuk Messaging service independent of tasks.
- The AccessRights concept with the attributes accessingPerson and accessPolicy was not modeled, as the personal information in the personal task management use case doesn't needs access right management. This is in contrast with the original assumption that parts of the personal information can be accessed by external persons.

For some other concepts the *modeling scope shifted*, i.e., the modeled detail and granularity, throughout the ontology development process.

- The 'TaskGoal' concept intended to model a KWer's goal with the task was initially modeled as complex concept keeping several goal parameters. As the use cases evolved, it turned out

that only the free-text input field for the goal was used. To reduce this over-engineering of the 'TaskGoal' concept, we introduced an informal goal description using a String property 'taskGoal' with the 'Task' concept.

- The 'Urgency' concept models the task's urgency from the KWer's perspective. Different granularity levels do exist to model the particular instances of urgency. This ranges from 'low' and 'high' to numeric values of 1-10, but as well 1-100 is possible. Looking at the use cases, we settled the urgency instances to numeric values of 1-10 as these provide a good compromise between sufficient fine granularity and too complex modeling for the use case. The same modeling considerations were elaborated for the 'Importance' concept.

To enable *TMO specific task properties to show up in generic graphical users interfaces*, task properties such as, e.g., tmo:taskName and tmo:taskDescription have been modeled as sub-properties of generic properties such as, e.g., rdfs:label and nao:description respectively. Linking specific TMO task properties and concepts to corresponding generic concepts makes TMO task instances compatible with applications that are not capable of interpreting TMO information but which are capable of interpreting RDFS and Nepomuk NAO and PIMO information. One example for such an application is the Pimo Editor [Sauermann, 2009] which shows the information of the KWer's personal information model without being aware of the detailed TMO ontology.

9.2 Meeting Management Model

The Meeting Management Model (MMM) is a conceptual representation of meetings and related information for use in personal meeting management applications for KWers. It is part of the KWer's unified personal information model. Its representation bases on the PIMO ontology and thus represents an agreed, domain-specific information model for meetings and related information and covers the personal meeting management use cases as defined in section 4.3.

9.2.1 Meeting Management Model Characteristics

As domain model the Meeting Management Model *models the meetings and meeting protocols a KWer deals with in the context of the KWer's other personal information*. It thereby represents a meeting-centric view on the KWer's personal information, as it models beneath meetings as well the relations to meeting-relevant personal information. As meetings are an integral part of the KWer's personal information cloud, the Meeting Management Model is integrated into the Personal Information Management Ontology (PIMO) that represents the KWer's personal information cloud.

The task model captured in the TMO *has a formal, shared representation as ontology* to make the task model accessible from different applications. The formalized representation enables the application-independent representation of the meeting information as the meeting management model is explicated, shared and available with the meeting information. This way, several applications can access and use the Meeting Management Model and the common unified personal information model.

The Meeting Management Model *enables the attachment of desktop information objects to meetings, meeting protocols and subordinated items*. Using representations of the Nepomuk Information Elements (NIE) ontologies [Nepomuk Consortium, 2009c] the relation to files, emails and bookmarks can be realized.

The current version is considered as stable at the core, only minor attribute changes can be expected. The Nepomuk Meeting Manager actively uses the Meeting Management Model, see section 7.

9.2.2 Meeting Management Model Concepts and Attributes – Overview

The Meeting Management Model consists at the core of the *Meeting Protocol* representation. It is the basis for all detailed, meeting-related information going beyond basic meeting attributes like date and time. It covers all detailed meeting information as required by the personal meeting management use cases, see section 4.3. This includes the meeting agenda, the meeting protocol as well as any information associated to the meeting.

A meeting protocol consists of several concepts, see section 4.5. A meeting protocol consists of one or more topics. For each topic one or more meeting items detail the information related to this topic. Each of the meeting items has a particular type, ranging from an information item, a problem item over to an action item.

The Meeting Management Model extends existing PIMO unified information model classes by creating subclasses, see Figure 101. This contrasts the Task Management Model with the TMO which models a set of new concepts besides extending the pimo:Task class. Creating subclasses is the basis for extending the PIMO to refine Elements of PIMO-upper [Sauermann et al., 2009b, p. 24]. “Creating group–level ontologies is a simple matter of defining new subclasses of PIMO-upper classes […] or sub-classing InformationElement classes” [Sauermann et al., 2009b, p. 24]. As well, “new properties can also be added, which apply to the new classes via domain or range” [Sauermann et al., 2009b, p. 24].

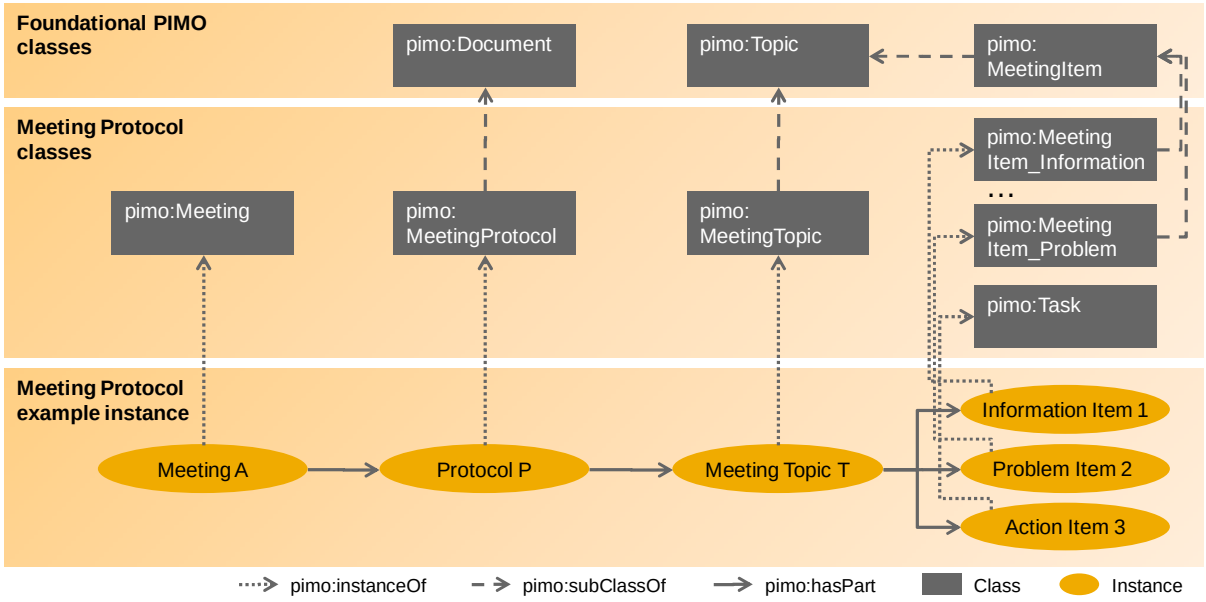


Figure 101: Meeting management model and example instances.

The pimo:Meeting class represents the Meeting concept in the PIMO unified personal information model [Sauermann et al., 2009b]. Basic properties are the meeting’s name, date and time as well as its location. The pimo:MeetingProtocol class represents the header of a meeting protocol. Its properties consist of a person as ‘minute taker’ and a ‘meeting description’. The pimo:MeetingTopic class represents the topics in a meeting, which can make up the agenda. Its properties consist of a string-valued ‘description’ and a person as ‘actor’ who is responsible for a particular topic. The pimo:MeetingItem class represents the items of a particular meeting topic. This class is an abstract class and not to be instantiated. Instead, subclasses of the pimo:MeetingItem class represent specific meeting item types. For example, there are the classes pimo:MeetingItem_Information for general information, pimo:MeetingItem_Problem for identified problems or pimo:MeetingItem_Solution for

identified solutions. A special item type is the pimo:Task class which represents a meeting's action items.

10 Domain-Specific Services

The *domain-specific services* provide a domain-specific abstraction on personal information management functionality targeted to support a particular KWer personal activity, see Figure 102 for an overview. Depending on the requirements of the supported use case, a domain-specific service provides access to a certain perspective on the KWer's unified personal information.

Technologically, a domain-specific service provides its domain-specific functionality to multiple applications and therefore uses a PIM system for managing unified personal information and accessing further operating system functionality.

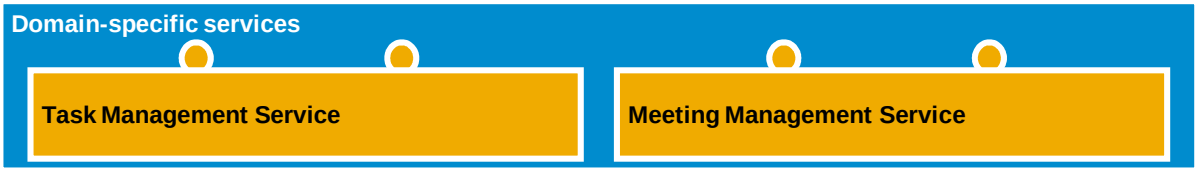


Figure 102: Domains-specific services.

We implemented domain-specific services for the two use cases of personal task management and personal meeting management. Further use cases can be covered in a similar way drawing upon the here presented architecture and implementation features. In the following, we explain these domain-specific services in detail including their relation to the corresponding use case and applications using them.

Both domain-specific services for the personal meeting management and personal task management use case, i.e., the Task Management Service and the Meeting Management Service, follow the same non-functional and technological requirements. These services just differ in their domain-wise offered functionality and operate on a different part of the same unified personal information model. We describe the non-functional and technological requirements and corresponding implementation realization for the Task Management Service in detail, but don't present the content for the Meeting Management Service in the same level of detail.

10.1 Task Management Service (TMS)

The *Task Management Service* encapsulates personal task management functionality that operates on the task management model enabling the re-use of common task management functionality. It provides task management functionality to all applications that want to conduct personal task management and handles the underlying information model, the task management business logic as well as the underlying PIM platform. Thus the Task Management Service acts as controller between the applications with user interfaces and the unified personal information model and in particular the Task Management Model. Being a service it offers task management functionality to different desktop application technologies.

It as well ensures uniform task information handling by providing developers with consistent task management services and not only with a task management model. The Task Management Service has a domain-specific interface focused on task management instead of offering a generic personal information management interface which enables the developers to focus on the domain-specific problem to solve.

The Task Management Service implementation uses a PIM System for the management of task management-relevant personal information and accesses a unified personal information model. Thereby the Task Management Service provides an abstraction layer that hides the internals of the underlying Nepomuk semantic desktop PIM system. As well, the Task Management Service interface hides used semantic technology by exposing a standard object-oriented interface to the Task Management Service.

10.1.1 Requirements for Integrated Personal Task and Information Management

When planning for the business logic that realizes the personal task and information management infrastructure, we analyzed and condensed both domain-specific functionality requirements as well as the technology requirements to make this functionality run on the desktop.

The requirements below emanate both several resources. On the one hand, we conducted an extensive literature review on state-of-the-art in commercial software and research of task management and related personal information management software [Grebner et al., 2007c]. Furthermore, we drew task management requirements from four case studies which have been compiled by ethnographic studies and user interviews [Grebner et al., 2007c]. As well, the experience gained when implementing several task management applications using semantic technologies [Grebner et al., 2008] was used.

10.1.1.1 Functional Personal Task Management Requirements – Use Cases

Functional personal task management functionality requirements emanate from the task management use cases that need to be supported. Here, we show the use cases whereas the detailed functionality is explained below.

The here mentioned use cases fully cover the domain of conducting personal task management in conjunction with personal information management (PIM). The use cases are: *Personal Task Management* consists of several parts. First, *basic task handling* deals with organizing a KWer's task, e.g., creating and populating a task. Second, *task list management* deals with the KWer handling a personal to-do list where the KWer manages a list of personal tasks that are due for execution or are already executed. Third, *task priority management* helps the KWer to maintain the priorities coming with a task. Fourth, *task time management* focuses on the time needed to execute a task and the KWer can assign a task's due date to manage the time-related aspects of work. Fifth, *task planning* helps the KWer to structure the workload and perform work decomposition, i.e., breaking down and categorizing tasks.

Task Information Management helps the KWer to collect and associate information needed for executing a task. This includes task tags to group tasks, information object attachments to connect tasks to, e.g., emails and files, and as social aspect persons involved in a task.

Social Task Management focuses on collaboration in the task domain. This means, that KWers can delegate tasks to each other, can perform and task tracking and conduct information sharing.

Personal Social Network Management deals with persons and is independent of task management. It consists of two parts. On the one hand, *person handling* allows to the KWer to manage personal acquaintances. On the other hand, the KWer manages organizational information on the personal social environment, i.e., the persons that, e.g., work in a company or a department thereof. A KWer

can use such organizational information on a person, like for example tags, skills or projects to organize personal information.

10.1.1.2 Technology-based Personal Task Management Requirements

Besides the above mentioned domain-specific requirements, the task management functionality needs to be implemented and run on desktop systems. Table 15 gives an overview on the found situations and derived requirements which are explained in detail below.

#	Situation	Requirements
Application-perspective		
1	Several applications need similar task management functionality	Re-use common business logic in several task management applications to avoid duplication
2	Applications on the desktop use several implementation technologies	Different desktop applications' technologies can invoke task management functionality
3	Several applications access and manipulate a particular task and related information	Handle task information consistently in all applications
Developer-perspective		
4	Developers are domain-knowledgeable experts	Focus on the domain-specific task management problem
5	Developers are not PIM experts	Hide underlying PIM technology, i.e., encapsulate functionality.
6	Developers are not experienced in semantic technologies	Hide underlying semantic technology, i.e., encapsulate functionality.
	Little tool support for semantic technology yet	

Table 15: Overview on technology requirements for Personal Task Management on the desktop.

Personal task management functionality occurs in a variety of personal information and desktop applications and these applications commonly use a set of task management functionality. These applications use task management aspects, i.e., work on tasks from different perspectives and with different goals. Tasks thus occur in different desktop applications dealing more or less explicitly with task management. For example, KWers use email and the corresponding email client for task management. A worklist dispenses tasks from business process management systems to the KWer. And for example for software engineers issue tracking applications contain tasks about the development activities. However, these applications commonly use a set of task management functionality. This includes for example handling a task, i.e., creating, reading, updating and deleting a task, as well as handling lists of tasks. This leads to

- *Requirement 1: Re-use common business logic in several task management applications to avoid duplication*

The *desktop applications that use task management functionality are implemented using a number of technologies*. There are the following technology types: *Native desktop applications* have full access to all desktop resources, offer a well-accepted user experience with a trained user base and are written, e.g., on the Microsoft Windows platform in Java or C#. *Semantic Nepomuk applications* are applications based on the Nepomuk Social Semantic Desktop which leverage the services thereof to manage information and communication capabilities. *Rich-client web applications* are web-based applications that feature compared to web-based applications additional functions integrating them into the desktop, for example to access the file system. *Widgets* are smaller applications, so called applets, that usually reside on a desktop screen provide where the KWer can interact with. Several

widget composition frameworks and deployment platforms allow the quick creation and distribution of a widget. *Web-based applications* reside within a browser and don't have access to desktop resources like the file system. These technology types are prevalent for applications that can use task management functionality, thus this leads to

- *Requirement 2: Different desktop applications' technologies can invoke task management functionality*

Tasks are not exclusively maintained a single application as several applications work on a common set of tasks. This means that several applications potentially access and manipulate a particular task and related information. From our development experience we saw that although a formalized task model in the form of an ontology was in place, numerous inconsistencies in the task information occurred when two applications implemented business logic for updating task information. This is due to ambiguities in the task model despite the already formalized representation. For example, inconsistencies occurred when setting a task's type, where a double-typing of the task with both the PIMO and TMO Task type is needed, but some applications just set one type. The same problem occurred for the labels of concepts, as there are again several attributes for the label ranging from the RDF label to the NAO label where some applications only set one type of label which then was missing in other applications. This leads to

- *Requirement 3: Handle task information consistently in all applications*

Application *developers using task management functionality are domain-knowledgeable experts*. Their domain knowledge enables them to use specific task management functionality within the application to be developed. As the context and the domain are known, i.e., personal task management, the developer can handle domain-specific, i.e., specialized, information and methods. Compared to this, using, e.g., generic personal information management functionality requires them to customize this to the task management domain which costs additional effort.

- *Requirement 4: Focus on the domain-specific task management problem*

However, *developers are not necessary experts for developing personal information management applications* on the desktop. In many situations, like for example with task management in email clients, task management functionality only represents a particular aspect of the application to be built or is not the focus of the development activities, e.g., when developing a quick task widget. Creating a task and showing a task list shouldn't require in-depth knowledge on personal information management on the desktop. In such cases, developers use task management functionality but don't want to be experts in building personal information management applications. This leads to

- *Requirement 5: Hide underlying PIM technology, i.e., encapsulate functionality.*

Most *developers are not yet experienced in developing applications using semantic technologies*. Semantic technologies comprise at the core of ontologies defining the data model, instances thereof and representation formats like RDF. Dealing with, e.g., RDF requires the developer to follow a resource-driven methodology to programming. The RDF data model makes statements about resources using a triple format, i.e., expressions in the form of subject-predicate-object. To deal with this, the developer needs to get experience in understanding and extending ontologies to finally handle RDF statements. Coming, e.g., with a plain Java development background to this, it takes a reasonable amount of time to get into this resource-driven description paradigm.

In addition, there is *little tool support for semantic technology* especially in the desktop resource and personal information management domain yet, as available tools are often still in research stage and thus immature for production systems.

One core simplification for dealing with RDF-based information is RDF access frameworks. RDF access frameworks like, e.g., RDF Reactor and Elmo, provide higher-level development tool-based object proxies to RDF resources. These proxy objects and methods help developers with handling triples in a well-known, object-oriented and thus development tool-specific way. However, these RDF access frameworks are only available for a small number of programming languages such as Java, whereas the remaining programming languages require the developer to work on triple level. This leaves out several convenience methods and costs the inexperienced developer lead time to get productive. Furthermore, these RDF access frameworks cannot be applied universally on ontologies as they are not yet production-ready tested on multiple ontologies and thus a constant support by the RDF access framework developer is needed. Looking at the current semantic web landscape, the community drives the development of tools, however for production-ready desktop development this situation will not change in next few years.

In another example, support for semantic querying languages like SPARQL [Prud'hommeaux&Seaborne, 2009] and SERQL [Aduna B.V., 2006] are, unlike their concept, heavily dependent on specific RDF store implementations. Specific RDF stores implement different subsets of the mentioned query languages which lead to compatibility and portability issues for applications that deal with semantic information.

Such infant state problems of semantic technology tools impose hurdles to get involved into developing with semantic technologies and lead to high costs for adoption, e.g., through needed training time. This leads to

- *Requirement 6: Hide underlying semantic technology, i.e., encapsulate functionality.*

10.1.2 Task Management Service – Offered Functionality

The Task Management Service offers task management functionality in the form of a set of services, i.e., task management services. These service interfaces are grouped by their functionality in regard to the above mentioned use cases. Figure 103 shows the Task Management Service interfaces and their sub-interfaces.

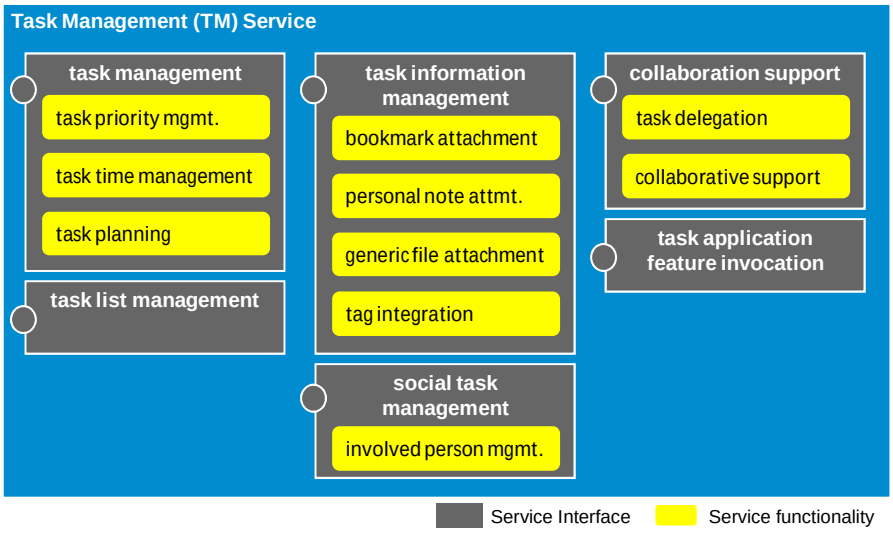


Figure 103: Overview on the Task Management Service functionality.

Overall, there are task management services for personal task management, task information management, task knowledge management and social task management. In addition, a personal social network management service offers functionality to manage persons and their relations. In the following sections, we present in more detail the offered functionality of each task management service.

The *task management* service offers functionality for handling task representations to support the KWer in conducting personal task management. It features methods for the *basic task management* of a particular task representation, i.e., to create, read, update and delete a task representation. Furthermore, the developer can specify the task's attributes, like for example the task's priority and its due date. For example there are methods for getting and setting the overall task priority level to the priorities high, medium and low like it is known to KWers from email priorities. For certain use cases a more elaborated *task priority management* is needed. Besides setting a basic task priority, methods offer setting a more fine granular priority by specifying urgency and importance values. *Task time management* methods enable the developer a dedicated monitoring and planning of a KWer's time spent on tasks. There are methods for determining the spent and planned time effort for a task. This deals both with actual as-is and planned values. Methods allow for keeping a detailed track record of spent time and further methods calculate the overall spent as-is time on the task based on several as-is measurements. Similar for planning the time to-be spend on a task, methods allow for defining and retrieving the target effort. For *task planning*, methods allow to (re-)structure tasks. For example to restructure a task hierarchy, a method enables the moving of tasks up and down in their hierarchy that, e.g., involves reassigning tasks as subtasks.

The *task list management* service offers the functionality to retrieve task lists specified by different criteria. Methods allow for querying tasks that are in a certain task container, e.g., in the inbox, outbox or active tasks container. Task containers represent specific piles of tasks. Further queries are on the one hand methods with predefined queries like, e.g., getting tasks by the name of an involved person. On the other hand, the developer can leverage the Task Management Service's query generator for assembling and combining several query building blocks to achieve a query for tasks. There are a number of query building blocks which each represent a particular filter criterion, like e.g., involved persons, involved documents, assigned tags or a due date. For example, to retrieve all tasks where the person Claudia is involved and which have the tag "next" symbolizing the next tasks

due for execution, two query building blocks for the person and the tag respectively are combined with the AND operator.

The *task information management* service enables a developer to associate information elements, tags and bookmarks with the representation of a task. For the KWer it is important to handle a task in the context of other relevant information. This results in attaching this information to the task. There are several methods that handle different types of attached information for tasks. A task representation can have several *task tags* attached. This means that there are methods that attach a given set of tags to a task representation, thereby check for the existence of a tag and create it in case that it didn't exist yet. Similar methods are available for deleting tags. As convenience method, updating tags works with submitting the updated set of tags and the corresponding method then handles all needed task tag attachment changes. *Information element attachments* connect a task to the existing desktop information. There are three types of information elements which each feature methods for adding and removing the attachments of a task representation. In particular, there are bookmarks containing references to websites, personal notes containing wiki-like text snippets and generic files referencing files in the files system, like, e.g., a document or presentation.

The *social task management* service enables a developer to associate people representations with the representation of a task for persons involved in a task. Methods enable the developer to associate or remove a person representation to or from a task representation. The developer can define the role of the involved person for the task, i.e., assign like for example a ‘responsible’ or an ‘involved’ role to a person associated with a task. An existing person can be identified by either their email address or their URI, otherwise the person is created with the information from the handed over parameters. Furthermore methods enable the retrieval of people associated to the task representation.

A *collaboration support* service enables the developer to manage collaborative work on tasks, i.e., when several people work on a common task. The service contains methods to enable different kinds of cooperation between KWers, e.g., task delegation. The respective methods enable the exchange of task data including associated information elements. This also encompasses the exchange of metadata that belongs to these information elements including attributes and relations. Beside task delegation there are methods to support collaborative tasks in which KWers more closely work together, e.g., by sharing a common task information space. It supports delegation protocols to control the processes of task delegation and metadata transfer in order to realize task synchronization, e.g., for mutual updates of task status information between delegating and delegated tasks.

The *task-related application feature invocation* service enables the developer to invoke task-related application features of other applications in the workspace and thereby to hand over needed information so that the end-user doesn't need to enter the information manually again. A method enables the remote invocation of the task creation dialog window of the Kasimir task management application. The developer can open the “Create new task dialog” in Kasimir by invoking a service method and handing over parameters. The upcoming task creation dialog window contains pre-populated information which has been handed over as parameter upon invocation. This can include the task name and description as well as people and tags to be attached to the task.

10.1.3 Task Management Service – Architecture and Implementation

This section presents the Task Management Service architecture and the technology-related considerations made for the implementation of the task management services. The implementation of the Task Management Service and its methods bases on the infrastructure of a PIM system, i.e., here the Nepomuk semantic desktop. Figure 104 shows an overview on the Task Management Service architecture and the used PIM system services that are explained below.

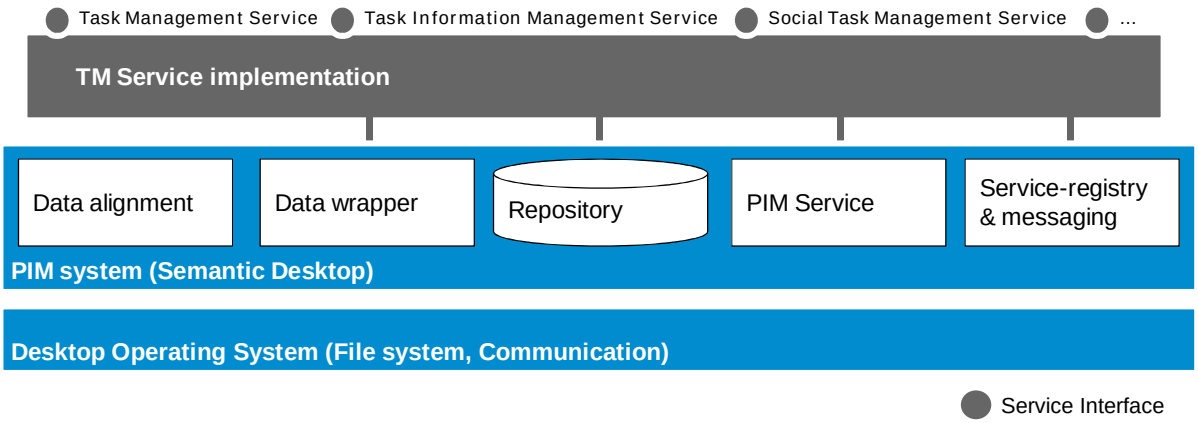


Figure 104: Task Management Service architecture and used PIM system services.

The Task Management Service implementation doesn't directly call operating system interfaces, but calls PIM system services. This way, the Task Management Service is platform independent to a high degree, as it needs only PIM system services to run, independent on which operating system the PIM system runs.

The following sections explain in detail how the Task Management Service is deployed and how it is implemented. As well, we show the role of the involved PIM system services and the semantic infrastructure.

10.1.3.1 TM Service Implements Technology-based Requirements

The presented Task Management Service implements the technology-based requirements as mentioned before.

The *Task Management Service encapsulates task management business logic* to re-use common task management functionality in several applications *(requirement 1)*. This avoids the duplication of source code and subsequent disadvantages like reduced maintainability. In addition, the Task Management Service can offer proven task management business logic, i.e., implemented task management methods that are successfully in use by several applications. Using the Task Management Service leads for the developer to a reduced complexity of the overall application, reduces the effort to develop task application logic and enables the developer to focus on domain-specific task of conducting task management.

The *Task Management Service offers task management functionality for different desktop application technologies (requirement 2)*. This enables desktop applications written in different implementation technologies to invoke the Task Management Service interfaces and use the task management functionality. For native desktop applications, the Task Management Service can be invoked using XML/RPC or by directly invoking the Task Management Service's methods in Java when the invoking application is written in Java. The Task Management Service is exposed as web service for web-based and heterogeneous environments like, e.g., widget platforms like Google Gadgets [Google Inc., 2009].

The *Task Management Service ensures uniform task information handling by providing developers with consistent task management services* and not only with a task management model *(requirement 3)*. Using the Task Management Service, developers can consistently create, read, update and delete task information. This integration on application/ business-logic level ensures the unified handling of information and helps this way to keep the task information consistent and leverage proven business logic. Having only a shared data model and description represents an information integration-level, but as mentioned before doesn't eliminate possible handling inconsistencies through different implementations and neglects existing proven business logic.

The Task Management Service has a *domain-specific interface focused on task management* instead of offering a generic personal information management interface *(requirement 4)*. With a business-level abstraction layer for task management and targeted to tasks and other involved information objects, a developer doesn't have to deal with generic information management functionality which then would need to be manually customized by the developer or even the end-user. It as well relieves the developer from interacting with a complex and generic information model where the developer first needs to get an overview on which parts of the model are potentially relevant for task management. At the same time it reduces the risk of using different modeling constructs for the same entities, which can happen in situations where potentially ambiguous constructs do exist like, e.g., relations called “is part of” vs. “is related”. This would lead to inconsistent task information where several concepts represent for a single thing from the KWer’s perspective. In addition, the Task Management Service interface defines for complex parameters like tasks and persons a data abstraction layer to offer the developer with a simple and stable set of information elements. This is realized by data access objects (DAOs). The DAOs thereby abstract from a modeling-driven representation which targets a suitable (semantic) modeling of the underlying personal information model.

The Task Management Service provides an abstraction layer that *hides the internals of the underlying PIM system (requirement 5)*. Through encapsulation, the Task Management Service reduces the complexity for developers. They don't need to dive into the particularities of handling tasks or personal information management when they, e.g., just want to quickly create a few tasks from an application. Through the Task Management Service the developers can transparently leverage a PIM system, i.e., the Nepomuk Social Semantic Desktop for task information management without caring about the details on how this is achieved. The Nepomuk PIM platform offers functionality for handling the unified personal information model and integrates desktop information elements therein, like, e.g., emails and files.

The Task Management Service interface *hides used semantic technology* by exposing a standard object-oriented interface to the Task Management Service and by encapsulating its implementation which leverages semantic technology *(requirement 6)*. Through encapsulation the developer invoking the Task Management Service doesn’t need to know how to deal with semantic information, e.g., triples in RDF. This means that no training in semantic technology and associated programming paradigms (triples vs. object-orientation) is needed and thus reduces the adoption costs. Especially, developers don’t need to deal with semantic information model particularities like, e.g., label types changed frequently during the Nepomuk PIM system development. Semantic URIs are used as identifier in the object-oriented interface, but transparently assigned by the Task Management Service for new objects.

10.1.3.2 Service Deployment and Invocation Options

With the goal of making the Task Management Service available for most of the applications that run on the desktop, it needs to be taken into account that today's desktops applications use a wide variety of technology types.

To cater for this wide variety of desktop applications, the Task Management Service needs to be deployed in several service frameworks. To achieve this, the task management services are registered with and deployed in the services-registry of the PIM system, i.e., here the Nepomuk Service Registry, a service registry for desktop services. The desktop-based service-oriented architecture of the PIM system is realized for the Nepomuk semantic desktop through the Nepomuk Middleware [Nepomuk Consortium, 2009d], an OSGI-based service management platform. Therein the Task Management Service is deployed as OSGI plug-in and additionally carries a WSDL description file. The Nepomuk Middleware acts as desktop application server and hosts the task management services both as OSGI services and as Web services. The deployment option of exposing the task management services as SOAP-based Web service leverages the Task Management Service's WSDL description and in particular caters for all web-based application types. In addition, invoking the Task Management Service via XML/RPC is possible through the Nepomuk Middleware, e.g., for invoking the Task Management Service from Java. Through these deployment options, both native desktop as well as web-based desktop application types are taken care of.

10.1.3.3 Service Interfaces and Parameters

The Task Management Service offers a set of services that applications can consume. Therefore, these task management services have interfaces where some services are technically grouped by their service's main functionality, as shown in Table 16.

Task Management Service	Task Management Service Interface (technical name)
task management service[16] task list management service task information management service social task management service	TaskManagementService
collaboration support service	TaskManagementCommunicationService
task-related application feature invocation service	TaskManagementFeatureInvocationService

Table 16: Task Management Service Functionality and Interfaces.

Each service interface exposes a number of methods that make up the Task Management Services' functionality. Each method can be invoked by passing parameters. Most methods deal with information entities of the KWer's personal information cloud, which represent complex parameters to the service's methods.

Complex parameters are represented as data access objects (DAOs). These DAOs represent a data abstraction layer for each entity of the KWer's personal information cloud. For each entity, the DAO represents the entities' attributes that are relevant from a KWer's domain-specific view. The DAOs thereby abstract from a modeling-driven representation that targets a suitable (semantic) modeling of the underlying personal information model. For example, the task DAO has a string-value attribute

[16] The task management service offers the functionality to perform task management in conjunction with other services like, e.g., the task list management service. In contrast to this, Task Management Service denotes the overall name of these services altogether.

for priority whereas the task model represents each priority value by an ontology instance. Figure 105 shows the DAOs as applied in the methods of the four presented service interfaces.

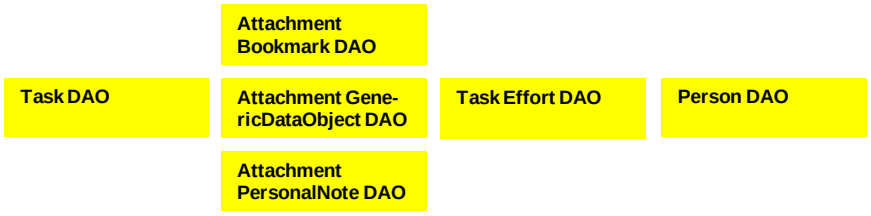


Figure 105: Overview on data access objects (DAOs) for the Task Management Service.

The DAOs and all other parameters that the Task Management Service interfaces expose are composed of simple data types like, e.g., string and integer. Avoiding complex data types helps to reduce serialization problems when exposing the service in multiple technologies like web services or XML/RPC.

When the consuming application is written in Java and runs in the same virtual machine as the Task Management Service, the DAOs are represented by Java objects and can be passed over to the Task Management Service on invocation. In scenarios where applications like, e.g., widgets, invoke the Task Management Service as web service, the task DAO are marshaled in an XML-based SOAP message.

10.1.3.4 Task Management Service Implementation Leverages PIM System

The Task Management Service implementation leverages the series of the underlying PIM system in numerous ways to provide its functionality. Here, the Task Management Service uses the Nepomuk semantic desktop PIM system.

The Task Management Service realizes a *persistent data access through a data abstraction layer*, i.e., it manages persistence for tasks and other personal information by accessing the PIM system's repository, here the Nepomuk Local Storage [Nepomuk Consortium, 2009e] service. The Task Management Service uses DAOs to mediate access to the task data in the RDF repository, i.e., the DAO classes define an adapter layer that manages the mapping between the underlying data access framework and a stable set of DAO Value Objects (VOs) used on the service interface for transfer. The *Data Access Object (DAO) layer* was introduced to provide an abstraction layer to the underlying RDF access mechanism. Here, RDF access frameworks provide Java-based object proxies to RDF resources, of which several are available, e.g., RDF Reactor [semanticweb.org, 2009a] and Elmo [Aduna B.V., 2009b]. The provided object-level abstraction of semantic resources ensures stability in the higher-level Task Management Service classes and Task Management Service clients which depend on these. It allows the different access mechanisms to be used without any impact to non-persistence classes in the Task Management Service. To achieve this, DAO factories are used to instantiate concrete DAO classes. This approach allows the use of best-of-breed technologies to manage the storage layer for each data access interface. When the Task Management Service would have been developed against a low-level RDF API, choosing different RDF access frameworks wouldn't be possible without major code changes. Extensions to the task data model can be supported by introducing new DAOs to encapsulate access to the then extended data model. The RDF access framework invokes the RDF2Go model, a *storage abstraction layer on top of RDF databases*. Using RDF2Go [semanticweb.org, 2009b], the implementation codes against a stable storage abstraction layer and the underlying RDF database can change without requiring changes in

the Task Management Service code. All persistence in the Task Management Service is ultimately managed by the PIM system repository, here the RDF database repository Sesame2 [Aduna B.V., 2009a].

The Task Management Service uses the PIM Service of the PIM system for *creating identifiers of newly created information entities* of the unified personal information model like, e.g., tasks, persons or tags. The PIM Service creates unique resource identifiers (URIs) and manages the concepts of the unified personal information model, i.e., here the Personal Information Model (PIMO).

The Task Management Service enables the developer to *flexibly query for tasks using the task query generator*. The task query generator builds a SeRQL query by combining several query building blocks, executes the SeRQL query and returns the resulting task list. The developer chooses from several pre-defined query building blocks that each represent a particular filter criterion like, e.g., involved persons, involved documents, assigned tags or a due date. The Task Management Service then compiles using the selected building blocks a SeRQL query that translates each building block into a corresponding SeRQL query clause and which takes care of dependencies and duplicate checks between the building blocks.

The Task Management Service *ensures the semantic representation of desktop information elements used for task attachments* by invoking the data wrapper of the PIM system, see section 2.7.5. Invoking the Data Wrapper for an information element makes sure that suitable RDF representations are available for, e.g., task attachments which can be interlinked with the task representation. The data wrapper furthermore offers desktop information object related functionality, like, e.g., triggering an operating system call for opening an information object in the respectively registered desktop application. For this, the data wrapper accesses the operating system API and the operating system opens the information object in a desktop application as registered for the information object's MIME type.

The Task Management Service uses a *messaging infrastructure to enable task delegation*, i.e., uses messages to send and receive tasks. Thereby two messaging services are used. On the one hand, using the PIM system's messaging service, i.e., here the Nepomuk Messaging Service [Nepomuk Consortium, 2009f], the developer can send and receive tasks from other PIM system-enabled desktops. The Nepomuk Messaging Service is a messaging infrastructure to exchange messages and receive event notifications based on the Jabber / XMPP protocol [Wikimedia Foundation Inc., 2009e]]. To make the Nepomuk Messaging Service work, a central messaging hub needs to register participants and routes the messages. On the other hand, email messages are used for delegating tasks and are sent and received by an email client, e.g., Microsoft Outlook [Microsoft Corporation, 2009a]. The task delegation emails contain both a human readable part with the task information as well as a machine-readable part with serialized RDF data containing the task representation and all related information like attachments and tags. The human readable part enables KWers who don't use any task management application or who don't have a PIM system compatible personal task management application in place to directly access the task representation's information. The machine-readable part ensures that PIM system enabled personal task management applications can access the task representation's data. The Task Management Service checks in intervals for new incoming mails and processes them so that the resulting task representations are stored in the inbox task container.

10.1.4 Applications Use the Task Management Service to Offer Task Management Functionality

From an application perspective, the Task Management Service offers personal task management functionality to support the KWer in the personal task management activity. Several applications on the desktop can invoke the Task Management Service to expose the task management functionality to the KWer as end-user. Thereby they access the underlying unified personal information model in the PIM system repository containing the task representation and related personal information. Figure 106 depicts from an overview perspective an example with three applications calling the Task Management Service. It further depicts what information these applications transmit with the Task Management Service in the case of creating a new task representation and attaching a person representation, see below for details.

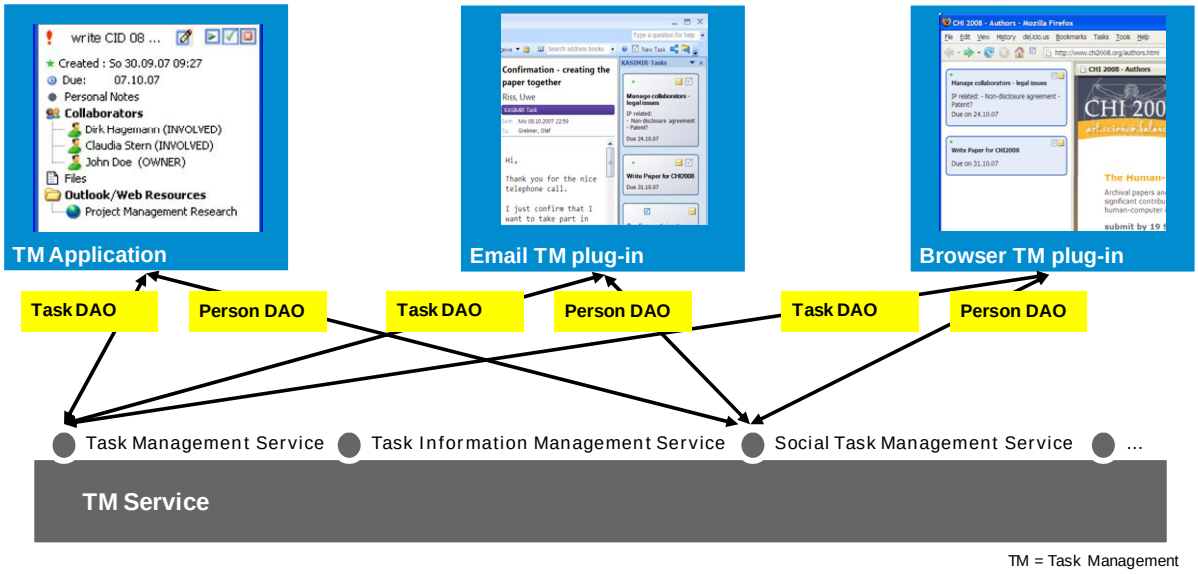


Figure 106: Applications interact with the Task Management Service.

The three depicted applications invoking the Task Management Service are on the one hand the Kasimir personal task management sidebar, see section 5, the Microsoft Outlook Task Plug-in and the Mozilla Firefox Task Plug-In, see section 6 [Grebner et al., 2008]. In particular Kasimir uses the majority of the Task Management Service as it provides full-fledged functionality to support the KWer's personal task management activities. Both the Microsoft Outlook Task Plug-in and the Mozilla Firefox Task Plug-In show tasks within the context of emails and websites, respectively, as well as enable the KWer to create new tasks from their hosting applications.

When an application invokes the Task Management Service to interact with the task representations in the underlying unified personal information model in the PIM system repository, the needed DAOs are transferred as invoking parameter. For example, when creating a new task representation and attaching a person representation, as shown in Figure 106, the invoking application creates and fills both the task and person DAO. It then passes them over as parameter when invoking a method of the Task Management Service. Correspondingly, the Task Management Service returns the same DAOs for read operations.

The Task Management Service takes care of concurrency issues that can occur when several applications try to access the common set of personal information. When several applications

interact with the Task Management Service, it ensures consistency by queuing the read and write operations in the incoming sequence.

10.2 Meeting Management Service (MMS)

The *Meeting Management Service* encapsulates personal meeting management functionality that operates on the meeting management model enabling the re-use of common meeting management functionality. It provides meeting management functionality to all applications that want to conduct personal meeting management and handles the underlying information model, the meeting management business logic as well as the underlying PIM platform. Thus the Meeting Management Service acts as controller between the applications with user interfaces and the unified personal information model and in particular the Meeting Management Model. Being a service it offers meeting management functionality to different desktop application technologies.

It as well ensures uniform meeting information handling by providing developers with consistent meeting management services and not only with a meeting management model. The Meeting Management Service has a domain-specific interface focused on task management instead of offering a generic personal information management interface which enables the developers to focus on the domain-specific problem to solve.

The Meeting Management Service implementation uses a PIM System for the management of meeting management-relevant personal information and accesses a unified personal information model. Thereby the Meeting Management Service provides an abstraction layer that hides the internals of the underlying Nepomuk semantic desktop PIM system. As well, the Meeting Management Service interface hides used semantic technology by exposing a standard object-oriented interface to the Meeting Management Service.

10.2.1 Meeting Management Service – Offered Functionality

The Meeting Management Service offers meeting management functionality in the form of a set of services, i.e., meeting management services. The Meeting Management Service functionality thereby fully covers the domain-specific use cases for personal meeting management as defined in section 4.3. The meeting management services expose their functionality through service interfaces. These interfaces group similar functionality in regard to the above mentioned use cases for personal meeting management. Figure 107 shows an overview on the Meeting Management Service functionality by its offered service interfaces and their sub-interfaces.

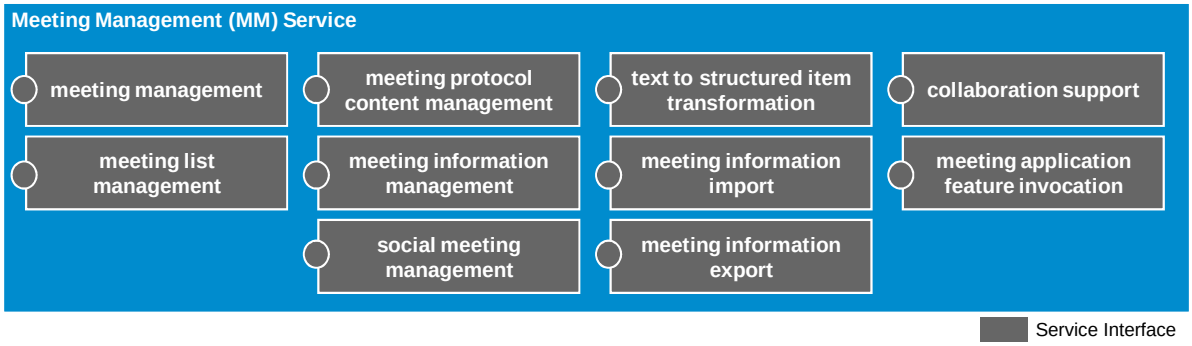


Figure 107: Meeting Management (MM) Service functionality overview.

The *meeting management* service offers functionality for handling meeting representations to support the KWer in conducting personal meeting management. It features methods for the *basic*

meeting management of a particular meeting representation, i.e., to create, read, update and delete a meeting representation. Furthermore, the developer can specify the meeting's attributes, like for example the meeting's date, time and location.

Meeting list management service offers the functionality to retrieve meeting lists by different criteria. Upon invocation a set of methods uses the parameters handed over by the developer to execute queries. The developer can request meeting lists where the meetings and related information complies with certain criteria. For example, return meetings within a certain date range or meetings where certain people are involved. As well, it is possible to retrieve meetings where a certain type of meeting item involved, e.g., a problem item. Similar to this, it is possible to retrieve meetings that have action items, i.e., tasks, or even specify the task's state, e.g., to show meetings with open action items.

The *meeting protocol content management* service enables a developer to keep a record of the meeting structure. This includes the meeting agenda which consists of meeting topic representations and the meeting protocol itself which extends the meeting agenda with meeting items of different types. At the core the methods enable the management of the meeting protocol representation, i.e., to create, read, updated and delete it, and its relation to a selected meeting representation. Furthermore, meeting topics can be added and removed for a meeting protocol, as well as meeting items all possible types can be added and removed from a meeting topic. The corresponding attributes, like the item description and for, e.g., action items the status can be set with creating and modifying the item. Furthermore methods enable the retrieval of the meeting protocol representation's associated structured information, or parts of it.

The *meeting information management* service enables a developer to associate information elements, tags and textual comments with the representation of a meeting protocol and subordinated meeting items, see section 7.2.3.7. There are three different types of information that can be associated with the representation of a meeting protocol and subordinated meeting items. The developer can attach information elements on the desktop, assign tags and textual comments. Furthermore methods enable the retrieval of the information associated to the meeting protocol and subordinated meeting items representations.

The *social meeting management* service enables a developer to associate people representations with the representation of a meeting protocol and subordinated meeting items. An existing person can be identified by either their email address or their URI, otherwise the person is created with the information from the handed over parameters. For the meeting protocol as proxy for detailed meeting information the developer can set the role of the person for the meeting, i.e., attending, missing and interested. For dedicated subordinated items the developer can associate persons. For an action item, the person receives a "Responsible" role and for other items the person receives an "Actor" role. Furthermore methods enable the retrieval of people associated to the meeting protocol and subordinated meeting item representations.

A *text to structured item transformation* service transforms between text in wiki-like syntax and structured meeting protocol representation back and forth. The developer can input text in wiki-like syntax, see section 7.2.3.2 for the wiki-like syntax, and the method creates from this text a structured meeting protocol with meeting topics and meeting items. Vice versa, the developer can generate based on a structured meeting protocol with meeting topics and meeting items text in the same wiki-like syntax.

A *meeting information import* service enables a developer to import meeting and meeting protocol information from a number of different formats. Particularly this includes the import of action items into a meeting protocol from the digital meeting facilitation and moderation application 'DigitalModeration' [Teambits GmbH, 2009].

A *meeting information export* service enables a developer to export meeting and meeting protocol information into a number of different formats. This includes the formats RTF for Microsoft Word, PDF for revision safe archiving, HTML for web browser and wiki-syntax for copy & pasting into wiki applications.

A *collaboration support service* enables the developer to manage the distribution and commenting of a meeting protocol. To distribute the meeting protocol, a method can send out the structured meeting protocol content in an email message containing both human-readable and machine-readable versions. In case of a finalized protocol, this message contains the meeting protocol in a revision safe format like, e.g., PDF. To return comments on a meeting protocol to the minute taker, a method can send out a message containing comments based on a particular meeting protocol. To process received comments on a meeting protocol, a method can import and aggregate a number of messages which contain comments on a particular meeting protocol.

The *meeting-related application feature invocation service* enables the developer to invoke meeting-related application features of other applications in the workspace and thereby to hand over needed information so that the end-user doesn't need to enter the information manually again. In a simple form this means the opening of associated desktop information elements in their respectively with the operating system registered applications, for example to open an attached presentation file in a slide presentation application. To make an appointment for a meeting, a method can invoke the appointment scheduling functionality of a calendar application, here Microsoft Outlook. It as well transfers meeting information relevant for the appointment like, e.g., the meeting name, when it has already been entered in the meeting or meeting protocol. To send an invitation for a meeting, a method can invoke the messaging functionality of an email client application, here Microsoft Outlook. It as well transfers meeting information relevant for the message and compiles an email message. For example, it uses the meeting name as email subject, the agenda as email body and the participants and interested persons as email recipients. To enable meeting facilitation, moderation or voting support, a method can invoke a meeting facilitation application, here DigitalModeration [Teambits GmbH, 2009]. For invoking the voting support feature, e.g., a meeting topic is transferred to the meeting facilitation application upon invocation. For moderation support, the method transfers the meeting agenda to the moderation feature of the meeting facilitation application.

10.2.2 Meeting Management Service – Architecture and Implementation

The Meeting Management Service's architecture bases on the use of PIM system services. Its implementation is based on Nepomuk semantic desktop PIM system services. As its implementation is very similar to the implementation of the Task Management Service, we give here only a brief overview on its architecture.

An overview on the Meeting Management Service architecture and thereby used PIM system services is shown in Figure 108.

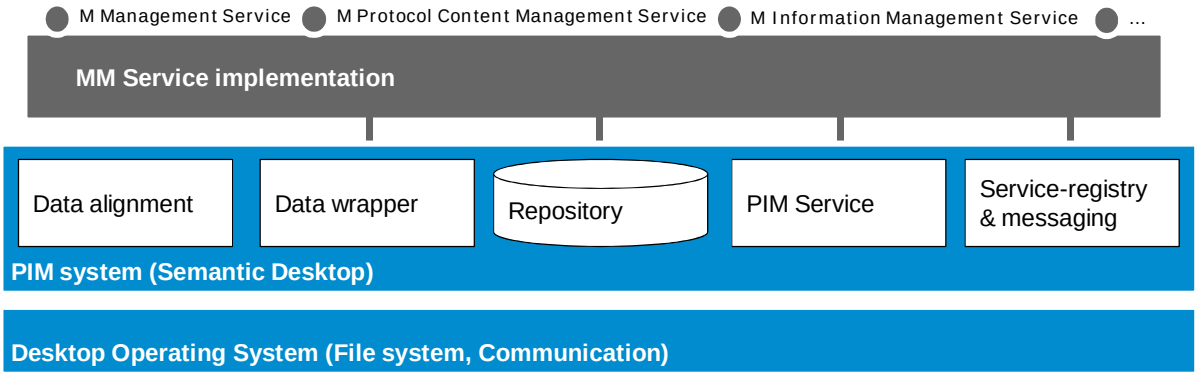


Figure 108: Task Management Framework implements Semantic Desktop functionality.

Table 17 explains the role of each used PIM system service that the Meeting Management Service uses to implement its functionality.

PIM System module	Role for the Meeting Management Service
Data Wrapper	The Meeting Management Service uses the Aperture Data Wrapper [Aduna B.V.&DFKI GmbH, 2005] to open information elements in their respective applications.
PIM Service	The Meeting Management Service invokes PIMOService [Sauermann&Klinkigt, 2009] methods to access the unified personal information model in the Local Storage.
Repository	The Nepomuk Local Storage service [Nepomuk Consortium, 2009e] is the repository for the unified personal information model PIMO including the KWer's meeting and meeting protocol representations.
Service-registry	The Meeting Management Service registers its services in the Nepomuk Service-registry [Nepomuk Consortium, 2009d] to be available and visible to other applications and services seeking it.

Table 17: PIM system services used by the Meeting Management Service.

The Meeting Management Service implementation doesn't directly call operating system interfaces, but calls semantic desktop services. This way, the Meeting Management Service is platform independent to a high degree, as it needs only PIM system services to run, independent on which operating system they themselves run.

10.2.3 Applications Use the MMS to Offer Meeting Management Functionality

Multiple desktop applications can invoke the Meeting Management Service and access the offered personal meeting management functionality to expose it to the end-user. Accessing the offered functionality includes accessing the underlying unified personal information inclusive meeting and meeting protocol representations. An example with three applications invoking the Meeting Management Service is shown in Figure 109 from a simplified overview perspective.

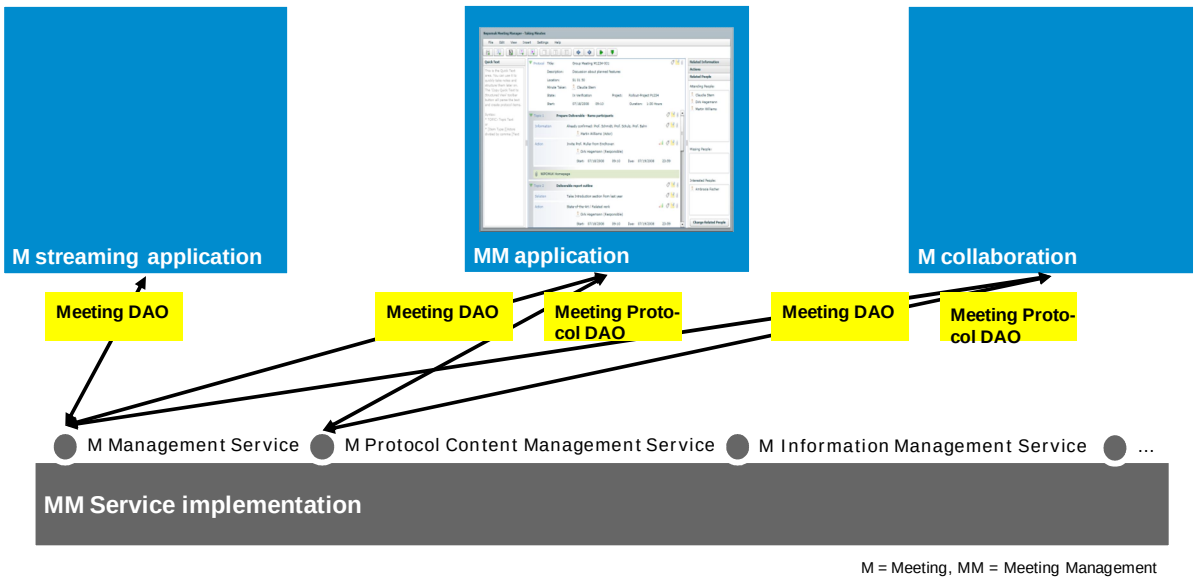


Figure 109: Applications interact with the Meeting Management Service.

The three depicted applications invoking the Meeting Management Service are a meeting stream recording application being able to record a meeting on audio and video stream, the Nepomuk Meeting Manager as described in section 7 and a meeting collaboration application allowing meeting participants to synchronously exchange messages. The Nepomuk Meeting Manager uses the majority of the services offered by the Meeting Management Service as it provides extensive personal meeting management support functionality to the KWer, see section 7. The meeting stream recording application can access meeting representations for information like date, time and location of a meeting and updates meeting representations with the actual time used for the meeting and the location of the recorded meeting stream. The meeting collaboration application can access meeting protocol representations to update information from a message exchange among meeting participants in the meeting protocol.

When an application invokes the Meeting Management Service to interact with the underlying unified personal information repository, the needed DAOs are transferred as invoking parameter. For example, when creating a new meeting and a corresponding meeting protocol, as shown in Figure 109, the invoking application creates both the meeting and meeting protocol DAO and initializes them with the respective information. It then passes them over as parameter when invoking a method of the Meeting Management Service. Correspondingly, the Meeting Management Service returns the same DAOs for read operations.

The Meeting Management Service takes care of concurrency issues that can occur when several applications try to access the common set of personal information. When several applications interact with the Meeting Management Service, it ensures consistency by queuing the read and write operations in the incoming sequence.

PART IV – RESULTS

ACKNOWLEDGEMENTS AND PUBLICATIONS

- The formative usability studies presented in section 12 have been conducted in cooperation with the HCI Group at the Royal University of Stockholm (KTH). Special thanks goes to Rósa Guðjónsdóttir, Henrik Edlund, Kristina Groth and Sinna Lindquist. Furthermore to Pär Lannerö, Bo Westerlund, Cristian Bogdan and Yngve Sundblad. These studies are published in the technical report [Grebner et al., 2007b].
- The long-term evaluation study presented in section 12 has been conducted in cooperation with Marlen Jurisch and Uwe Riss. It is published in the technical report [Riss et al., 2009].

11 Evaluating Personal Information Management Applications

In the following we present in detail the evaluation of the Kasimir personal task management application and the SAP Task application plug-ins.

These applications are prototypes demonstrating the principles of the underlying technology. They are not full-blown products ready for an end-user market, e.g., due to the fact that they are not multilingual (English as the only language) and they don't comply to product development standards as, e.g., defined by large business software manufacturer.

Common to the overall prototype development process is the user-centered design approach. This means that the development process for the prototype applications consists of an iterative refinement and development driven by the user feedback. Starting with user evaluations in the early design phase and continuing with continuous feedback gathering prevents costly developments that miss the goal or prevents re-developments to adjust the prototypes.

11.1 Evaluating PIM Applications: Formative Usability and Long-Term Evaluation Studies

There are the two approaches of formative usability studies and long-term evaluation studies to evaluate PIM software applications, see Figure 110 for an overview and comparison. The two different types of evaluation studies cater for different stages of the prototype development.

Characteristics	Formative usability study (Lab-setting)	Long-term evaluation study (Field experiment)
Duration & Methodology	■ 1 hour sessions, around 8 participants ■ One-on-one evaluation sessions, KWer conducts defined tasks with the prototype	■ 2-4 weeks, 10+ participants ■ KWer works with the prototype in real-life, uses prototype in the daily work
Evaluation goals & scope	■ Validate and gather feedback on idea (prototype aspects), identify usability flaws ■ Similar digital work environment	■ Validate and gather feedback on overall application ■ KWer's personal digital work environment
Environment & data set	■ Office-like room ■ Similar digital work environment ■ Evaluation data set	■ Own office ■ Personal digital work environment ■ Personal data set
Potential risks for evaluation quality	■ Unfamiliarity with physical and digital work environment as well as with the 'personal' test data set may bias the KWer	■ KWer cannot be observed while working, other measures like, e.g., workshops, needed to gather feedback and observe behavior
Evaluation costs & requirements	■ Moderate costs. Conducting an evaluation with 8 participants takes one day. ■ Lo-fi prototype possible, elaborate features	■ Costly. Hard participant recruitment, considerable resources needed for evaluation support ■ Hi-fi prototype needed, near-product quality
	early design & refinement stage	mature design stage

Figure 110: Comparison overview on formative usability and long-term evaluation study.

On the one hand, a *formative usability study* enables the validation of ideas or particular prototype aspects. As well, it helps application designers to identify usability flaws. Suitable for the beginning and refinement phase of the application development, formative usability studies help engage early on with a limited number of end-users. In each one-on-one evaluation session lasting one hour, the KWer conducts defined tasks with the prototype. Through this it is possible to identify prototype

flaws before investing extensive implementation effort in potentially unsuccessful prototype features. For this purpose, lo-fi prototypes like, e.g., paper prototypes, serve as means in these studies.

On the other hand, a *long-term evaluation study* enables the validation and the gathering of feedback on the overall application. Long-term evaluation studies are suitable for the late design phase of the application development, where already a mature application does exist. A number of KWers participate over a defined period of two to four weeks in the evaluation study. They install the prototype in their respective digital work environment and work with the prototype. For this purpose, hi-fi prototypes like, e.g., elaborated and tested application prototypes, serve as means in these studies.

In the following we present their characteristics in detail and present the characteristics of both approaches besides each other to enable comparisons and point out the different capabilities.

Table 18 shows the e*valuation goals and scope* for both evaluation types.

	Formative usability study (Lab-like setting)	Long-term usage study (Field experiment)
Evaluation goal	Validate and gather feedback on idea, identify usability flaws	Validate and gather feedback on overall application
Evaluation scope	Evaluate prototype aspects, target defined application feature sets	Evaluate whole application, target overall usage and utility of the prototype application, hard to limit evaluation to a particular feature set
Expected Results	Understand degree of usefulness of idea and get design and implementation guidance	Understand degree of usefulness of application
Comparability of results	High comparability due to constant work environment, physical environment and available data set	Medium comparability due to individual differences in the KWer's personal work environment, physical environment and available data set
Suitability in development stage	Early design and refinement stages	Mature design stage

Table 18: Evaluation goals and scope, comparing field and long-term evaluation study.

A formative usability study has to some extent a character of a laboratory experiment, but for our conducted evaluations we kept the environment very close to the KWer's real environment to minimize the effects of laboratory environment. For example, evaluation sessions were held in offices in the same office building as the test participants usually work to keep the environment as familiar as possible. As well, in our evaluation the used digital work environment was identical with KWer's production digital work environment, as all KWer of the organization are equipped with the same images of digital work environments.

Table 19 shows the *evaluation study set-up* for both evaluation types.

	Formative usability study (Lab-like setting)	Long-term evaluation study (Field experiment)
Evaluation duration	1 hour sessions	2-4 weeks
Evaluation participants	Around 8	10+, the more the better
Evaluation methodology	One-on-one evaluation sessions, KWer conducts defined tasks with the prototype, ask follow up questions, use think-aloud-technique	KWer works with the prototype in real-life, uses prototype in the daily work
Physical environment	Office room, similar to the KWer's workplace	Own office, at the KWer's workplace
Work environment	Evaluation system, identical setup of work environment	KWer's own personal digital work environment (desktop)
Available data set	Evaluation data, common situation to all evaluation sessions	Personal data set, real data of the KWer

Table 19: Evaluation study set-up, comparing formative usability and long-term evaluation study.

Table 20 shows *influencing factors on the evaluation process and quality* for both evaluation types.

	Formative usability study (Lab-like setting)	Long-term usage study (Field experiment)
Environment influence	Controlled test environment, few disturbing variables	Heterogeneous and uncontrolled environment, possible disturbing variables
Evaluation process control	Full control over the evaluation process due to the controlled environment and defined time frame	Few control on the evaluation process as KWer works individually with the prototype over a longer term
Evaluation observation possibilities	KWer can be observed during evaluation session, ask questions to deeper understand particular aspects	KWer cannot be observed while working, other measures like, e.g., workshops, needed to gather feedback and observe behavior.
Potential KWer disturbances	Unfamiliarity with physical and digital work environment may disturb and bias the KWer, as well as potential unfamiliarity with the 'personal' test data set, may lead to disturbances or different behaviors	Few disturbances and bias on KWer side due to familiar physical and digital work environment, KWer is familiar with the personal data set.
Ability to discover usefulness	KWer may not grasp all aspects of the offered solution in the short evaluation session, e.g., benefits that come over time	KWer can get acquainted to the offered solution and figure out the usefulness over a longer period of time.

Table 20: Factors influencing evaluation process and quality, comparing formative usability and long-term evaluation study.

Due to the potential unfamiliarity with physical and digital work environment the KWer taking part in a formative usability study may "not be able to rely on individual information management strategies (e.g., folder hierarchies, naming conventions)" [Franz, 2008] and thus "may employ different strategies for solving tasks on the test systems compared to their behavior on their own desktop environment" [Franz, 2008].

Table 21 shows *evaluation costs and requirements* for both evaluation types.

	Formative usability study (Lab-like setting)	Long-term usage study (Field experiment)
Evaluation process costs	Moderate costs. Conducting an evaluation with 8 participants takes one day.	Costly. Hard participant recruitment, considerable resources needed for supporting evaluation over long time.
Prototype requirements	Lo-fi prototype possible, elaborated hi-fi prototype only needed for the feature set subject to evaluation	Hi-fi prototype needed, near-product quality for the application, all features need to be in a well elaborated shape

Table 21: Evaluation costs and requirements, comparing formative usability and long-term evaluation study.

11.2 Combining Formative Usability Studies and Long-term Evaluation

By combining both the formative usability study and the long-term evaluation study, the evaluation gains the benefits of both approaches. Both the formative usability study and the long-term evaluation study feature unique advantages and lead to different insights. In combination the scope of the insights gained throughout the whole prototype development process significantly increases over the development process time.

By conducting formative usability studies the application designers can iteratively evolve the prototype. By gaining feedback from end-users on initially the idea and then the evolving prototype, the application designers can prepare the prototype stepwise for the long-term evaluation study. The long-term evaluation study requires a well elaborated prototype in order to provide insights in the usefulness of the application. When usability flaws and other non-functionalities disturb the end-user to grasp the prototype idea and functionality the long-term evaluation study fails.

Figure 111 depicts a sample evaluation plan for a prototype. Several formative usability studies in a laboratory-like session are followed by a long-term evaluation study. Finally, the prototype is further refined towards a product to be used in a wide-scale.

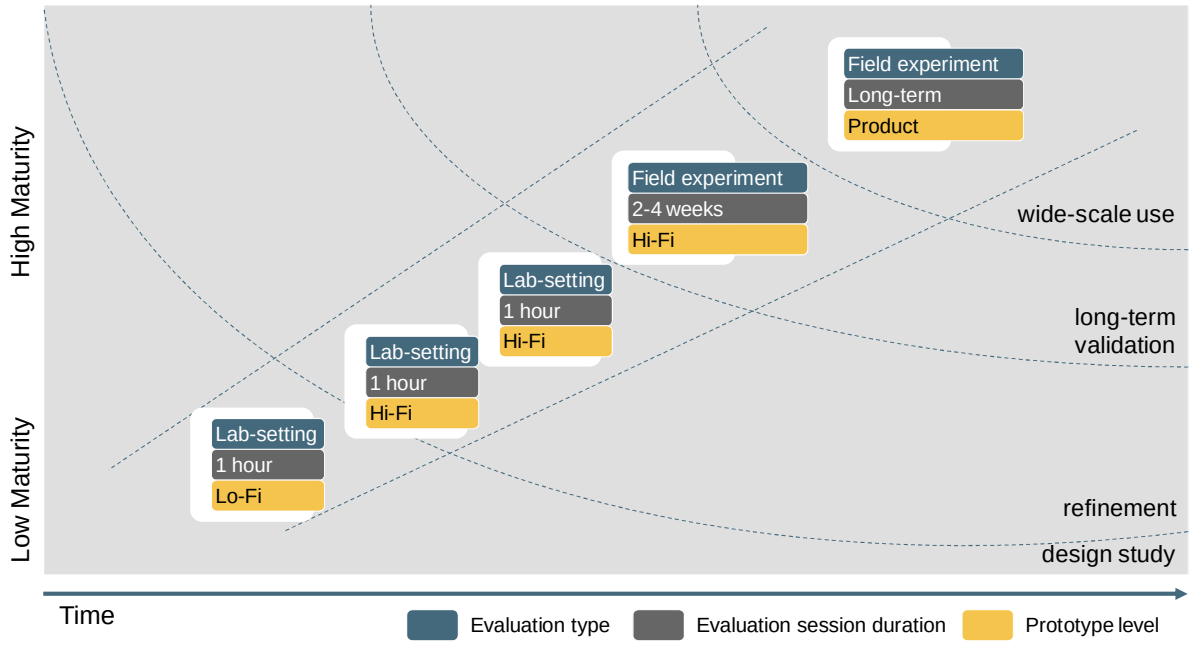


Figure 111: Gather user feedback by multiple evaluations throughout the development process.

12 Evaluating the Kasimir Personal Task Management Application

We combined both several formative usability studies and a long-term evaluation study to evaluate the Kasimir personal task management application. These evaluations have been conducted between May 2007 and December 2008, see Figure 112.

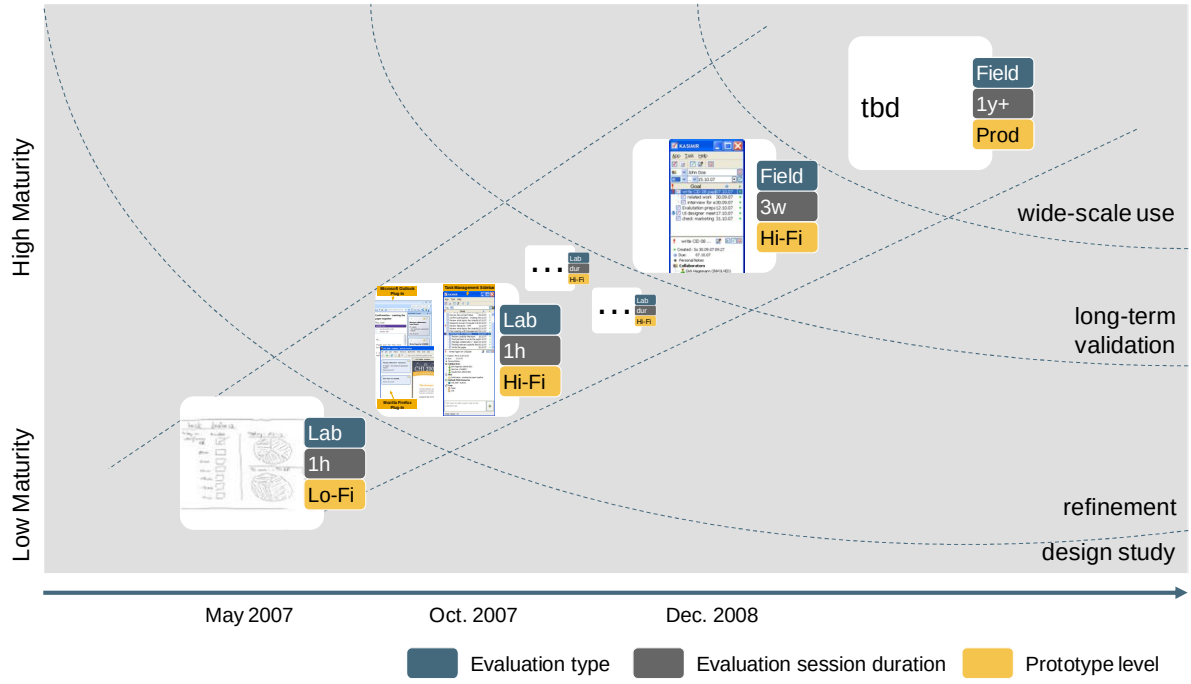


Figure 112: Kasimir personal task management evaluation.

In the following we report on the three evaluation studies. First, in May 2007 we conducted a formative usability study in an embedded office environment using a lo-fi paper prototype for a personal task management application. Second, in October 2007 we conducted a formative usability study in an embedded office environment using a hi-fi paper prototype with the then implemented Kasimir first version of the personal task management application. Third, in December 2008 we conducted a long-term usage evaluation in the KWer's real office environment using a hi-fi paper prototype with the refined Kasimir personal task management application. In-between October 2007 and December 2008 we conducted further formative usability evaluations on several Kasimir aspects, however the here presented evaluations are the most fundamental ones. As a next step it is planned to further refine the Kasimir prototype into an application suitable to be delivered as product to customers[17].

12.1 Formative Usability Evaluation of the KASIMIR Prototype

This section reports on two formative usability evaluations performed on two variations of the Kasimir prototype. The section begins with a description of the methodology chosen for the evaluation, followed by a detailed description of how we conducted the evaluations on two different occasions. The section ends with a description of the results of the evaluation and the implications they have for the design of the next iteration of the Kasimir personal task management application prototype.

[17] Delivering a product is beyond the scope of this thesis.

12.1.1 Methodology

The design of the evaluation is important in order to determine what the usability evaluation is going to accomplish. Questions that should be answered are “What does the design team want to know about the system?” and “What is going to be done with the results?”

A typical usability evaluation session consists of a pre-interview, a task performance phase and a post-interview [Rubin, 1994]. The pre-interview is performed to collect demographic data about the informant and to discover her expectations of the system being evaluated. During the pre-interview the moderator informs the informant of her rights to abort the evaluation if she feels like it, explains how the evaluation results will be analyzed and used and how the results and the video material will be treated to protect the informant’s anonymity. The moderator emphasizes during the introduction that the evaluation is of the system and not the informant’s ability to use the system. The pre-interview also functions as a way for the informant and the moderator to get to know each other before moving on with the evaluation itself.

The task performance phase consists of the moderator observing the informant while she performs pre-defined tasks with the system being evaluated. While the informant is performing the tasks the moderator collects data on how the informant is doing, which path she uses to accomplish the tasks, whether she takes long to perform the tasks, or how many errors she makes. All informants perform the same tasks, which make usability evaluations of this sort powerful in identifying usability problems. It allows the moderator to compare the different informants on an equal scale and gives the moderator an in-depth view of the tasks that were performed.

The tasks chosen for the evaluation should be functionalities that allow the target audience to fulfill their needs and requirements and that are important for the target audience. They should be tasks that the target audience normally would perform if they were using the system. The evaluation is not the normal circumstance when the informant is interacting with the system. Therefore, when introducing the tasks to the informant the moderator sets up a scenario that is relevant so that the informant understands in what kind of circumstance she should imagine herself being in. It is also important to specify what signifies a completed task so that the informant can judge herself when the task is accomplished. The moderator only helps the informant if she asks for help or when she has given up the task.

The post-interview is performed after the tasks are finished and allows the moderator to discuss the experience with the informant. They discuss what worked well and what did not. They also discuss how the system could be changed or improved to become more usable. The post-interview gives the moderator a chance to discuss how the system has met with the informants’ expectations discussed in the pre-interview.

Usually there is only one informant participating during an evaluation session, but some evaluation approaches require two informants cooperating during a session [Dumas&Redish, 1999]. An evaluation needs one moderator and one note taker and/or cameraperson.

There are several usability evaluation approaches to choose from, but the most common approach is a formative usability evaluation using the think-aloud protocol [Dumas&Redish, 1999] [Nielsen et al., 2002].

Formative evaluation is an evaluation with the goal of learning about the design of the system in order to improve it for the next iteration. A formative evaluation is a method that focuses on finding usability problems before a system is completed with the purpose of making it more successful and better adapted to the target audience. Formative evaluation can be contrasted with summative evaluation, which focuses on a final judgment of a product's usability, often a comparison between completed or competing products [Redish et al., 2002].

The think-aloud protocol is a popular technique used during usability evaluations. During the course of an evaluation, where the informant is performing a task, the informant is asked by the moderator to vocalize her thoughts, feelings, and opinions while interacting with the system. Some informants are more vocal than others and in the case where the informant finds it difficult to instinctively share their thoughts the moderator asks the informant questions throughout the session to prompt more discussion.

The think-aloud protocol allows the moderator to understand how the informant approaches the system and what considerations the informant keeps in mind when using the system. If the informant expresses that the sequence of steps she has to take to perform her tasks are different from what she expected, perhaps the interface is too complicated. Although the main benefit of the think-aloud protocol is a better understanding of the informant's mental model and interaction with the system, there are other benefits as well. The terminology the informant uses to express an idea or function should be picked up and incorporated into the design of the system or at least in its documentation.

Usability evaluations are often performed in a specially equipped usability laboratory. But in recent years most usability professionals have understood the advantage of performing the evaluation in the informants' own work or leisure context. This approach gives the evaluation team a deeper understanding of the informants' needs and requirements on the system under development since they are observing the informant in her own context [Rowley, 1994].

Recruiting participants for a usability evaluation is also a key to a successful evaluation. The informants need to be good representatives of the target audience. Other important aspects to decide are whether the informants should be novices or experts, male or female, young or old or a mix of all the above.

The test situation also needs to be designed and there are several characteristics that affect a successful setup. The moderator needs to know where the evaluation is taking place and what kind of equipment that is needed; video cameras to record the user's actions, scan converters to record the on-screen action, audio recorders to record verbal protocols, one-way mirrors to help the experimenter stay out of the informant's way, and so on. The moderator also needs to know the system intimately before performing the test in order to be able to do a good quality evaluation. The moderator needs to make sure that test apparatus is functioning and ready; the test apparatus for a usability evaluation includes the computer and system being evaluated. The moderator needs to make sure that, for example, all the tasks to be performed during the evaluation are functioning in the prototype.

Finally the evaluation results need to be analyzed. The major problems are easy to find since they are evident through the observation notes. Analyzing the video material and the observation notes can identify more detailed problems. The analysis is most effective if the evaluation team performs it

together. The final challenge is a successful communication of the evaluation results to the project team responsible for further development of the system.

12.1.2 Evaluation Design of the Kasimir Prototype (First and Second Version)

The methodology chosen for the evaluation of the Kasimir prototype in the first and second version was a formative usability evaluation using the think-aloud protocol as described above. We performed the evaluations using two different versions of the Kasimir prototype. The first version, a paper based low-fidelity prototype, was evaluated in May 2007 and the second version, a computer based high-fidelity prototype, was evaluated in October 2007. We chose to perform the evaluations in the informant's work context since it gives an in-depth understanding of their work situation and how the prototype fits into their daily work. Testing in the informant's work context also gives an opportunity to confirm conclusions drawn in the previous user research phase of the Nepomuk project.

All the evaluations were performed in a conference room or an office room at SAP Research in Karlsruhe. In collaboration Nepomuk project partners KTH and SAP provided the required personnel to perform the evaluations and all necessary equipment; i.e., computer, video camera and manuscript.

We designed the evaluations to meet the requirements imposed by the research questions, see section 1.2, and tested the second version of the prototype using a Windows PC equipped with a digital work environment as used by the KWer of the SAP Research organization. During the evaluation of the first version of the prototype we also evaluated other low-fidelity prototypes created by KTH which we don't focus on here. The evaluation sessions consisted of a pre-interview, a task phase and a post-interview. The sessions had one moderator and one or more persons observing. The evaluations were videotaped for further analysis. All evaluation sessions were individual.

12.1.3 First Kasimir Evaluation

The first evaluation was performed in May 2007 at the SAP Research office in Karlsruhe, Germany. We evaluated the prototype, see Figure 113 for a picture of one view, with seven participants, six men and one woman. The medium age of the participants was 37 and it varied from 25 to 52 years old. All informants were SAP employees working as research associates, product managers, senior researchers and external freelance consultant. See Appendix A for the evaluation manuscript.

12.1.4 Second Kasimir Evaluation

The second evaluation was performed in October 2007. We evaluated the Kasimir prototype with eight participants, seven men and one woman. Each evaluation session took approximately one hour. The medium age of the participants was 33 years and it varied from 26 years to 52. All informants were SAP employees working with, e.g., project management, ontologies, semantics, security, access control, system architecture, business grids, and software development. See Appendix B for the evaluation manuscript.

The Kasimir prototype, including two extensions (kind of plug-ins, called SAP Task application plug-in) is shown in Figure 114, see as well section 5 and 6.

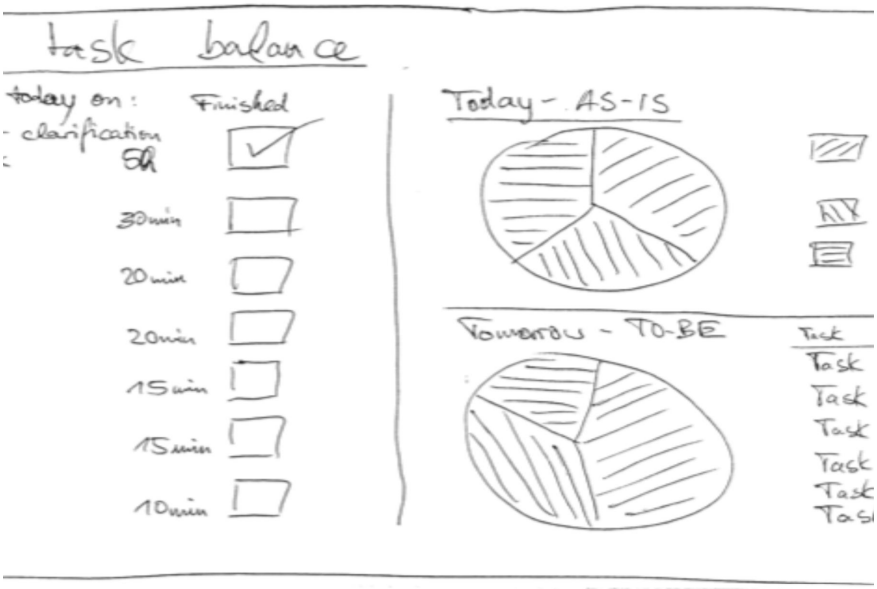


Figure 113: A picture of the first version of the prototype[18].

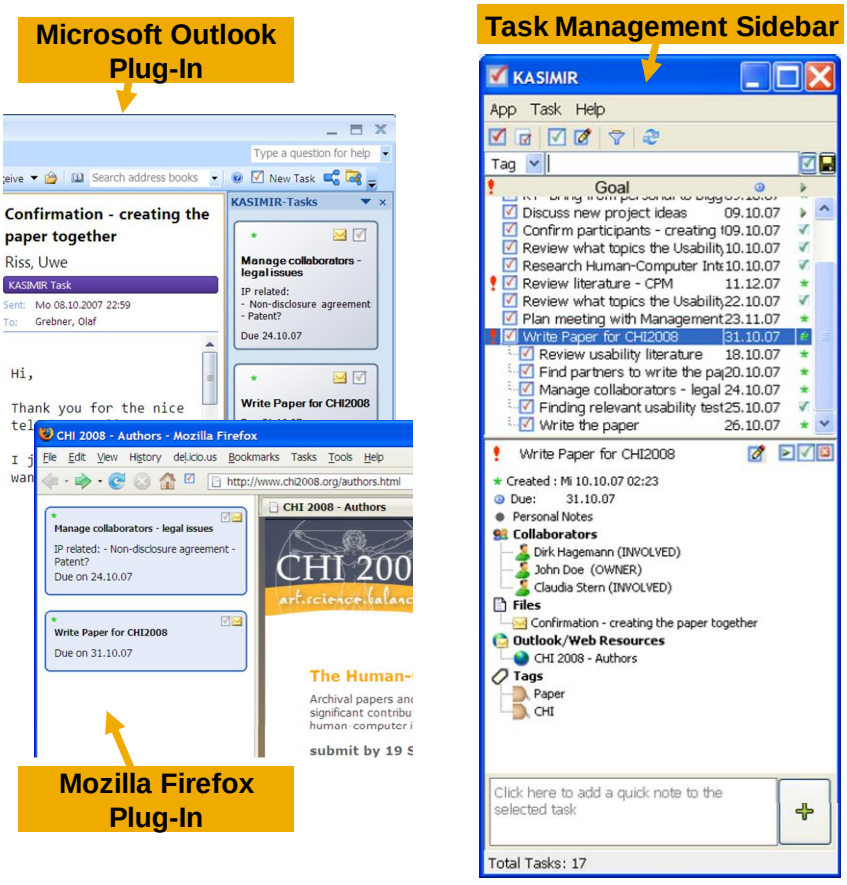


Figure 114: Screenshot of the Kasimir prototype, second version: Task sidebar window (right), Firefox extension (bottom left), and Outlook extension (top left).

Two of the informants said that they do not use any tool for task management today, one uses Microsoft tasks, one uses the task management in Outlook, one uses the calendar in Outlook, one uses Excel, and two use an internal SAP Wiki page. In Excel the informant said that he uses different

[18] This particular view is not included in the later-on evaluated Kasimir prototype aspects, but illustrates the characteristics of the used lo-fi paper prototypes.

colors for a task that is done or undone and he also adds URLs to the tasks. One of the informants who uses the Wiki for task management said that the Outlook task manager takes too much time to work with and it has too little functionality. The other informant who uses the Wiki said that he finds it difficult to manage tasks. He does not really see the benefits. Sometimes he turns to managing tasks on paper instead. He said that the task management tools annoy him, it takes too much time to plan a task. He rather uses task management as a reminder, not for planning work.

12.1.5 Results of the First Kasimir Evaluation

The first evaluation was performed on a paper prototype. The detailed interaction with the functionality was not tested during the first evaluation; the focus was more on the general usability of the application and it's functionality in order to guide us in the future development of the application. We tested different plug-ins for web and calendar, the idea of the task pattern[19] as well as starting and finishing screens.

The evaluation resulted in a positive feedback on the plug-ins. The informants liked that they didn't have to learn to use new software and that the task manager would be plugged into the other software they already use, for example Outlook and web browser.

The task pattern did not receive a very positive reaction. The informants wanted the application to reflect more that people are self-organized and that they have different ways of working with tasks. They wanted the application to allow for a different organization of tasks into categories, projects etc. They also discussed different ways of handling finished tasks; by deleting those actively or just making them disappear when they were getting too old to be visible in their tasks lists.

They also maintained that people like to personalize their tasks, and that they do not want to go totally from a template. They also wanted to organize the tasks according to their own personal needs. The informants also requested a more dynamic time frame and less detail in the task pattern.

The prototype tested had an opening and a closing window. The first window showed the overview of the day to come and the closing window showed the results of the work during the day that had gone. The participants liked the overview of both of these windows, but they found that there was a bit too much structure and detail. They also wanted to have more control over how they could use these windows to get more overview over their tasks.

12.1.6 Results of the Second Evaluation

The next sections present the results of the second evaluation grouped by tested functionality aspects.

12.1.6.1 Concepts, Icons and Menus

Two of the informants found the concept "Goals" as a title for the task list in the Kasimir sidebar confusing. Since it is a list of tasks, they also expected the title to be "Tasks".

All informants were confused about the icons used for showing priority and status, even though some informant said that they could recognize some of them from Outlook. The informants could guess some of the icons, e.g., the exclamation mark for a high priority task and the known video icons for running and pause. The blue arrow used for a low priority task as well as the status icon

[19] Task Patterns are a concept for knowledge transfer during the execution of a personal task, see [Riss et al., 2005] and [Riss&Grebner, 2008]. Task patterns are mentioned for the completeness of the evaluation report but we don't follow this topic up in this thesis.

green star needed to be explained. Several of the informants also wanted to directly interact with the icons in order to change priority or status, which was not possible. For example, one informant expected a list of icons to choose from when clicking on a status icon. Two of the informants mentioned tool-tips (an alternative text that is shown when the user moves the mouse over an, e.g., icon) as a solution for supporting the users in understanding the icons.

Several of the informants were confused about the icon in front of the task name, an icon looking like a check box. The icon is similar to the status icon showing that a task has been finished.

One of the informants mentioned that he expected a pop-up to show when clicking on the Plus button at the comment dialogue in the lower part of the Kasimir window.

12.1.6.2 Adding Colleagues to a Task

In order to add a colleague to a task the user has to find the More button in the Create New Task window. In the evaluation the informants were asked to add a new colleague to the task they had previously added. Several of the informants were confused about how to add a colleague. They either tried to interact directly with the collaborators view in the lower part of the Kasimir window, they could not find the More button in the Create New Task window, or they had problems adding a colleague. Four of the informants tried to interact directly with the collaborators view in the lower part of the Kasimir window. One of the informants suggested that sending an e-mail to colleagues "through" the task could automatically add the recipients of the e-mail as collaborators. One of the informants had trouble finding the 'More' button.

When the user has found where to add colleagues in the Kasimir prototype he or she can either create a new person to add as a collaborator, or search for already existing ones to add. These two choices were not obvious for the informants. Some of the informants objected to creating a new person when clicking on the Plus button for adding a person, they did not expect to create a person, just to add one. Others did not comment on it and simply continued to create a new person. Some of the informants explicitly looked for some kind of search or list to choose collaborators from. One informant suggested that the search field should recognize names independent of how the name is written. When finding the search function the user needs to double-click on the selected suggestion in order to add that person. Hitting the return key, which several of the informants did, closes the window without adding the selection made.

One informant did not like to type and asked for a list of persons to select from. Two informants suggested that it should be possible to change a collaborator's responsibility. Several informants liked the idea to be able to write an e-mail to the collaborators "through" the task. One informant pointed out that you can have colleagues that are not part of the project, but they may still be contributors.

12.1.6.3 Adding Keywords to a Task

In the evaluation the informants were asked to add keywords to the task that they had added. We deliberately used the word "keyword" in order to see if the informants chose tags or annotations, two features offered by the Kasimir prototype. Six of the informants started to add tags and two started to add annotations. The two informants that started with adding annotations had problems understanding what to add. Most of the informants found it difficult to explain the difference between annotations and tags, some thought they were the same, some thought there was a clear

difference, some were just confused. The informants had difficulties understanding the purpose of the annotations.

One of the informants said that tags are used for search, while annotations are more like properties and that he would not use annotations for tasks. Four informants suggested a catalogue or a list of tags to choose from when tagging information. This would minimize the risk of different kinds of spellings or using different names for the same concept. One informant suggested that a wildcard (cf. the * in the Unix operating system) could be used in order to cover several keywords in one tag.

One informant did not find the tags or annotations to be useful at all. He was very confused about the difference between tags, annotations and ordinary notes. He did not understand why he should add keywords at all when adding a task. One informant entered notes in natural language instead of keywords in the form of tags. One informant could not see the benefit of tagging tasks unless there are quite a lot of tasks to be managed. Since he usually does not manage that many tasks himself he had difficulties in seeing the benefit. The informants said that the keywords would be useful when searching and sorting text, to find tasks that are connected to each other, and when exporting forms to, for example, an HTML-page.

The view of the added annotations was also considered strange by a couple of the informants. They could not understand why it had to be in XML-format. They expected it to be presented more like tags.

Several informants had problems entering tags, i.e., if they should separate the keywords with a comma, blank or return.

12.1.6.4 Filtering Tasks

After the keywords had been added the informants were asked to search for similar tasks tagged with the word “usability”. When filtering the task list the user can select several types to filter with, e.g., goal, tag, and priority. Goal is default and when changing the filter from goal to tag the user is presented with a tag cloud. Seven of the informants spontaneously expressed that they liked the tag cloud.

The major problem the informants had with the filtering was when they started to filter with more than one concept. In this process it was not obvious how to apply the filter. Also, if not using the tag cloud to enter a keyword the informants became confused about how they should apply the filter. The Execution button was not placed in a manner that made it clear for the informants how to use it.

Two informants suggested that it should be possible to select several keywords from the tag cloud, not just one. One informant commented that having a separate row for each concept in the filter takes unnecessary space. One informant expected a pop-up to present the filtering results.

Even though filtering on different types was appreciated, some informants had problems finding out that this possibility existed.

When filtering, the user can also double-click on a tag in the lower part of the Kasimir window, but this feature was only found by one of the informants while browsing around.

12.1.6.5 Visualizing Tasks in the Extensions

In the two extensions of the Kasimir prototype the tasks are visualized on the left hand side of the Firefox browser and on the right hand side of the Outlook window.

Several informants could not see the value of visualizing the tasks in the browser. Some liked the idea to add an URL to a task, but found it a waste of space to visualize them from the specific web page. Several informants could not understand what they could do with the tasks in the web page. One informant tried to move a selected part of the text in the web page to the task, expecting that part of the text to be included in the task. One informant asked for tool-tips that can explain the icons in the tasks. One informant wanted to be able to get a more detailed view of the task. One informant asked for a context menu to be used in the task part of the web browser. Several informants had problems understanding how the tasks in the browser are connected to the tasks in Kasimir.

One informant suggested that the tasks in the browser could be color coded in order to show their priority or status. A couple of informants said that having tasks in a web browser could be useful, but that it depends on the kind of web page. It would be more useful for interactive web pages. One informant commented that it would be good to have a heading Tasks above the tasks in the web browser.

In general, the informants were more positive to visualizing tasks in Outlook. They found it interesting to see tasks that are related to a specific e-mail. Some of the informants were confused, as in the case with the web browser, about the connection between the tasks in Outlook and the tasks in Kasimir.

12.1.6.6 Create Tasks in the Extensions

In the evaluation the informants were asked to add a web page to the task they added in the beginning of the evaluation. To add a web page the user selects the 'Associate Website With Task' option from the Task menu. In the same menu there is also an option 'New Task For This Page'. The Outlook extension works in the same way but the association option was missing in the task menu.

First of all, no informant understood what happened when they associated the web page to the task. When selecting 'Associate Website With Task' from the Task menu no feedback is given on what happens. The informants could not see that the action they performed in Firefox actually resulted in a change in the Kasimir prototype. They did certainly not expect the URL of the web page to be added to the task that happened to be selected in the Kasimir prototype. They did not even know which task they had selected in the Kasimir prototype. They did not expect the tight coupling between the action performed in the Firefox and the tasks in the Kasimir prototype.

Creating a new task was not very obvious to the informants either. As in the case with associating a web page to a task, the lack of feedback confused them. Also, the name given to the created task, i.e., the name of the web page, was confusing to one of the informants. He said that he would rather copy and paste the URL to the Kasimir prototype.

The second day we explicitly helped the informants to create a new task with a selection of text from the web page. It was a complicated procedure and the selected text was added as a value in the annotation part of the task. The way this was done, the result was very confusing for the respondents. However, some of them could see a value in being able to drag-and-drop a piece of text

to a task, and to getting a context sensitive result in the task, e.g., if dragging a date, this would appear as a due date in the task.

Still, one of the informants liked the easy way of adding a URL to a task, but he wanted a more understandable connection between Firefox and the Kasimir prototype.

One informant suggested that the creation of tasks in the web browser should interact with the bookmarks. One informant expected an icon in the menu bar in Firefox for adding a task.

The informants were as confused in the Outlook extension as in the Firefox extension. One informant suggested that it might be more useful to associate an e-mail folder to a task because a task may result in quite a lot of e-mail communication, which could result in quite a lot of e-mails associated to a task. One respondent commented that maybe all recipients of an e-mail are not collaborators.

There were different opinions about how useful the extensions were. Some liked to be able to create tasks from other applications, others did not. All informants found it useful to add URL's and e-mails to tasks.

12.1.6.7 Interacting With Tasks in the Extensions

Regarding the interaction in the two extensions the informants had problems understanding how they could interact with the tasks. They were not sure what was happening when they clicked on the buttons that were shown in a task, and they also asked for more possibilities to interact with the tasks, e.g., a context menu, and more details about the task.

Some of the informants were positive towards the idea of the extensions, especially the Outlook extension, and they could see that extensions could be useful in other applications, e.g., PowerPoint and Word. However, it was confusing for the informants where their actions were applied. One informant pointed out that there were two "work-centers", one in the extension and one in the Kasimir application. He wanted one "work-center". One informant, that was more negative to the extensions, said that he would rather create a bookmark folder than tasks in the web browser.

Two informants were more negative towards the possibility to see tasks in Firefox, but they could see the benefit for some specific web pages. They would rather like to interact with the web page through the Kasimir prototype.

Another informant suggested that all tasks should be visible in the extension, where it could be possible to interact with them through drag and drop. One informant suggested another view of the tasks, where they could be sorted based on different aspects, e.g., collaborators and dates.

In the Outlook extension one informant found it useful to be able to find all e-mails that is related to a task. Several of the informants also said that they were confused about the relation between the already existing task management in Outlook and the tasks in the Kasimir Outlook extension. One informant suggested that the e-mails in Outlook could be filtered based on tasks.

12.1.6.8 Using Semantics

When asking the informants about how they thought semantics could be used in the Kasimir prototype the following comments were given (apart from what has been commented on above):

- intelligent support to filter e-mail in project folders,
- to search in general,

- to search for processes that are related to tasks,
- to suggest other topics of interest when adding a tag,
- to use an ontology to get the same concepts for tagging, think about how the ontology can be extended, e.g., with or without freedom,
- to build hierarchies,
- to make things appear as the user expects,
- to find people that have the same skills,
- for rules (to be applied in ontologies) and queries, some kind of rule based inference, who works with whom,
- to tag not only tasks, but specific parts of a task, and the tags may differ depending on what is tagged.

12.1.6.9 Expectances on the Kasimir Prototype

A number of issues concerned with what the informants expect from the Kasimir prototype were identified during the evaluations. They expect

- to be able to organize their work, to create tasks personally or together with others,
- to be able to search, select or filter among the tasks,
- to see relations between tasks and processes,
- to see what happens next in a task and who is responsible for it,
- to be able to see how much of a task that is finished and how much time that is expected to fulfill the task,
- to attach and exchange files with collaborators in a task
- to access resources in a task,
- to be able to configure the sidebar in order to reduce the space it takes,
- to be able to use the PC based right-click interaction,
- the windows to be larger, difficult to read the text in them as it is now,
- to be able to get a preview of what the task is about,
- often used interactions to be quick to perform,
- a tight coupling to the calendar,
- to get suggestions of which files, e-mails or other resources to attach to a task,
- aspect orientation, i.e., to be able to sort on different topics (cf. smart mailboxes),
- tool-tips on icons, menus, etc.,
- to find documents and other resources attached to a task,
- to be able to interact with the lower part of the window,
- more navigation possibilities, e.g., when clicking on the name of a collaborator it would show what other tasks that person is involved in, and through ontologies,
- a reduced number of clicks in order to manage the tasks,
- expired tasks to be automatically removed.
- to be able to associate folders (of files, e-mails, etc.) to a task,
- not to be forced into detailing of the tasks, which in the long run leads to too much task management.

12.1.6.10 Conclusions & Directions for the Next Iteration of the Kasimir Prototype

The here presented conclusions and directions are targeted for the next iteration of the Kasimir prototype. These suggested improvements have been considered and implemented. They have been tested in the long-term evaluation as described in section 12.2.

Both evaluations showed that the informants liked the idea with extensions. However, it is obvious from the evaluation that there were major problems with how the tasks were managed in the extensions. The informants were confused about the connection between the extensions and the Kasimir prototype sidebar. They wanted to add resources to a task, but the question is if it is necessary to integrate the tasks to those resources. The same functionality could be achieved if the list of tasks can be filtered based on the chosen resource(s).

Many informants liked the idea to connect resources to a task, and they liked the idea to interact with the tasks from other applications. However, all informants were confused about the functionality in the extensions, which has limited their general understanding about them.

It is clear that many of the informants have different needs regarding a task management tool. A future task management tool needs to be flexible and adaptable to fit as many users as possible. Today, there are almost as many tools used, or usages of the same tool, as there are people in the organization. It is important to consider how many tasks people are expected to manage and not to make the task management process more complicated than it is today. The tool should provide possibilities for task management that the users cannot access in the tools they are using today, it should be adaptable and not force a level of detail that puts an extra effort on the users.

Apart from the usability issues above we also think it is important to address the following issues:

- Possibilities to perform configurations in the Kasimir sidebar, e.g., making it possible to collapse different views to reduce the space that the sidebar takes on the desktop.
- Remove the Annotation function or clarify how it can be used.
- Provide the users with a selection of choices when appropriate, e.g., for tags and collaborators.
- Make it possible to select several keywords from the tag cloud when filtering.
- The filtering process needs to change. One suggestion is to have any easy filtering with only one type, e.g., tags, that is quickly available, and one more complex where filtering can combine different types that is available in some other way. Another suggestion that will save space in the sidebar is to keep the rows for each type, but allow several keywords on each row.
- Allow drag-and-drop functionality in places where it is applicable.
- Add tool-tips to icons and menus to explain their functionality.
- Extend the Kasimir prototype to include semantics, e.g., to suggest possible selections to the user based on the name of the task, the participants of a task, the resources added to a task etc.
- Make it possible for the user to interact directly with the view presenting the task information, i.e., the lower part of the sidebar window, as well as with the icons for priority and status.

12.1.7 Comments on the Conducted Evaluation

The evaluation of the KASIMIR prototype confirmed the need for a sophisticated task management system using extensions in other applications of the digital work environment. However, the chosen methodology led to results that gave only limited insight in the validity of the underlying ideas, i.e., the results of the evaluation couldn't fully reveal the acceptance of the concepts and design ideas implemented in the prototypes. Instead, the evaluation results mostly revealed direct usability-related issues. These issues, for example that users found hard to distinguish between actionable and non-actionable functionality or that some user interaction required a few more clicks than standard office products, had already been known.

The target group of this prototype consists of knowledge workers, i.e., researchers at SAP Research. The development and conception of the prototype is targeted towards a feasibility study of the chosen concepts within the target group. It is clear, that the KASIMIR prototype is a high-fidelity prototype that cannot fulfill product shipping standards of SAP. However it provides a working prototype featuring technology that itself is in an early research stage.

Implementing a high-fidelity prototype within an environment with a given set of resources carries implicitly a trade-off between implementation cost and implemented functionality. For example, a drag and drop feature across application boundaries has a high implementation effort. It would have come at the cost of discarding some pieces of functionally with provided in our judgment a higher value to the knowledge worker. In the concrete case, having in mind the goal for a working technology prototype, the decision has been to taken to not implement such a drag-and-drop mechanism and instead go for context menus while at the same time freeing up development resources for an extended set of functionalities.

The task is to find a healthy balance to address the presented trade-off with a given set of resources. For the obvious reasons, a high-fidelity prototype can never fulfill the standards of a shipping-ready product. Even thorough development and testing and continuous involvement of usability experts doesn't change the situation much, as many of the in the usability evaluation identified issues have been built into the prototype intentionally due to the mentioned trade-off between implementation cost and implemented functionality. In the concrete case of the KASIMIR prototype, we acknowledge, that despite an already highly usability-centered development approach, some more of the identified usability issues could have been wiped out before the evaluation by working closer with usability experts.

The conclusion of the conducted evaluations is that a refined methodology helps to focus the evaluation on the underlying concepts and design ideas of the functionally working prototype subject to the evaluation. As well, it needs to take prototype development capability constraints into account. Developing a functionally rich, working prototype which incorporates research technology leads to forced design decisions that can deviate from the optimum with regard to usability.

The next formative usability studies on the Kasimir prototype, not reported on here, took these constraints into account by adapting the evaluation methodology of the formative usability study to focus the evaluation on the understanding of the underlying ideas. After each set of clustered tasks the evaluation manuscript foresees direct follow-up questions with the participating KWers. Interviewing the KWers directly regarding the design ideas and concepts they have just seen and used in the evaluation task helped to gather feedback on the underlying ideas beyond usability issues like the ones motioned in these sections.

As well, the long-term evaluation study conducted with a matured version of the Kasimir application is another possibility to further research the underlying ideas and concepts. In this study, see section 12.2, the participating KWers used the Kasimir application for three weeks, a significantly longer period than the here applied hour. This gave more insight into the usage of Kasimir with regard to the underlying ideas and concepts.

12.2 Long-term Evaluation of the Kasimir Prototype

A long-term evaluation study of the Kasimir personal task management application was carried out over a period of 3.5 weeks in December 2008. We evaluated the Kasimir prototype with thirteen participants, ten men and three women. The age of the participants varied from 24 years to 45. The seniority level of the testers ranged from student workers to senior researchers.

The third version of the Kasimir prototype subject the long-term evaluation study is shown in Figure 115. Along with Kasimir, the Nepomuk PSEW application is shown and was available in the evaluation. Nepomuk PSEW [Nepomuk Consortium, 2009g] serves as generic user interface for the PIMO unified personal information model, see section 9. It enables a KWer to access and modify directly the unified personal information model.

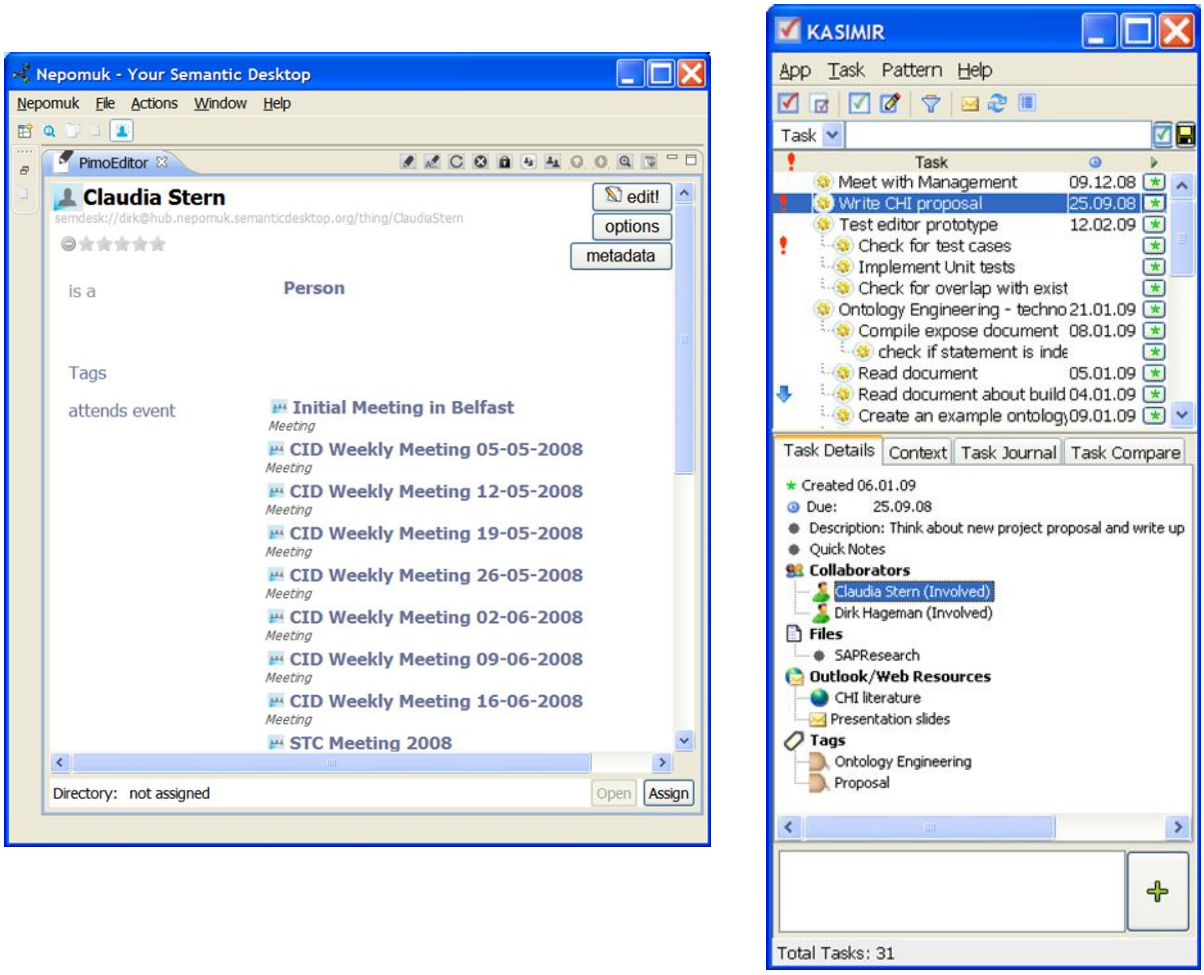


Figure 115: Third version of the Kasimir prototype.

We used activity theory for the long-term evaluation study based on the investigation reported on in [Grebner et al., 2007c] where we applied activity theory to derive the main features of a task management system.

This section describes how activity theory and its principle of contradictions guide research in information technology and its evaluation. A theoretical overview of activity theory can be found in [Grebner et al., 2007c]. It follows with an elaboration on how activity theory can be used to inform human-computer-interaction design. Next, we shortly explain the components of the Activity-Oriented Design Method since this method is employed to guide the investigation.

Throughout the last decade, activity theory has been successfully employed in a variety of ways to inform systems design. The advantage of activity theory is that the user's interests and needs become understandable on the basis of the history and the current problems of their activities [Rajkumar, 2009]. The transition from design to user activity in the development of new technologies is a critical phase in the innovation process. Past application of activity theory, where the user's history and problems with the current work activities were analyzed, helped researchers understand the advantages and the restrictions of a tool. In sum, the activity theory approach is committed to understanding and judging the usability and usefulness of a computer system from the users' points of view [Mwanza, 2001].

Stage one analyzes the situation involved. For this stage, [Mwanza, 2001] provides the first tool that is the 'eight-step-model'. This is a list of eight questions that guide the analysis of the activity and its components. Stage two concerns the modeling of the situation, using the information obtained in stage one with the activity system [Engeström, 1987]. Subsequently, stage three dissembles the activity with the goal of reducing complexity. Within this stage [Mwanza, 2001] introduces the 'activity notation' tool to assist in the decomposition of the activity. This tool consists of six 'sub-triangles' which allow the detailed analysis of the six sub-activities. In addition, a third tool is presented with the aim of supporting the analysis of stage four. It comprises six general questions, which can be used to generate a wide range of research questions to analyze the interaction and relationships within and between the components of each sub-triangle [Quek&Shah, 2004]. These questions can also elicit the presence of contradictions within and between the several components. In stage five, the aforementioned research questions are utilized for the support of data collection efforts, e.g. in interviews, questionnaires, or observation. The last and final stage entails the interpretation and communication of the research findings. For this stage a fourth tool is offered, namely the diagram for mapping operational processes. This tool presents the results of stage four in an illustrative form. In specific, the tool presents with clear visual indications the research questions and the areas of conflict that have become apparent and facilitates understanding of the process and the results [Quek&Shah, 2004].

12.2.1 Evaluation Methodology

The following section shortly discusses the methodological approach underlying the long-term evaluation study of Kasimir. First, the objectives of the study are presented. The second part highlights the steps taken from an activity-theoretical perspective. Last, the study design as well as the sampling and questionnaire method are presented.

12.2.1.1 Goal of the Study

The first and main objective of this investigation is to evaluate the overall usefulness of the Kasimir Task Management application from a user's point of view as suggested by [Dicks, 2002]. The second objective is to examine the user's usage patterns of the Kasimir Task Management application. This has led to the following research question that has been postulated to guide the investigation: How can the Kasimir Task Management help users to organize their work activities more efficiently?

In short, the aim is to observe how the users utilized Kasimir on a daily basis conducting their task management. We want to examine if there are any tools that prove to be specifically useful as working environment. In accordance to this, this study also strives to identify how Kasimir enables efficient task management and, in case, how Kasimir interrupts the user's workflow.

Furthermore, this section aims at describing how activity theory and its principle of contradictions can be applied to guide research in information technology. Contradictions point to critical conditions in a user's activity [Engeström, 1987]. As a result the second research question was formulated as follows: What are the internal and external contradictions within the activity system? Hence, we will investigate if the resolution of possible contradictions leads to useful results when analyzing the user perceptions on Kasimir.

12.2.1.2 Activity-Oriented Design Method as Methodological Conception of the Evaluation

The current study design is centered on the Activity-Oriented Design Method by [Mwanza, 2001]. Stages one and two of this study started with the interpretation of the various components of the activity in terms of the situation being examined. Within stage three we utilized the available information and methods, as proposed by [Engeström, 1987] and [Mwanza, 2001], to produce the Activity System of the situation under investigation, see Figure 116.

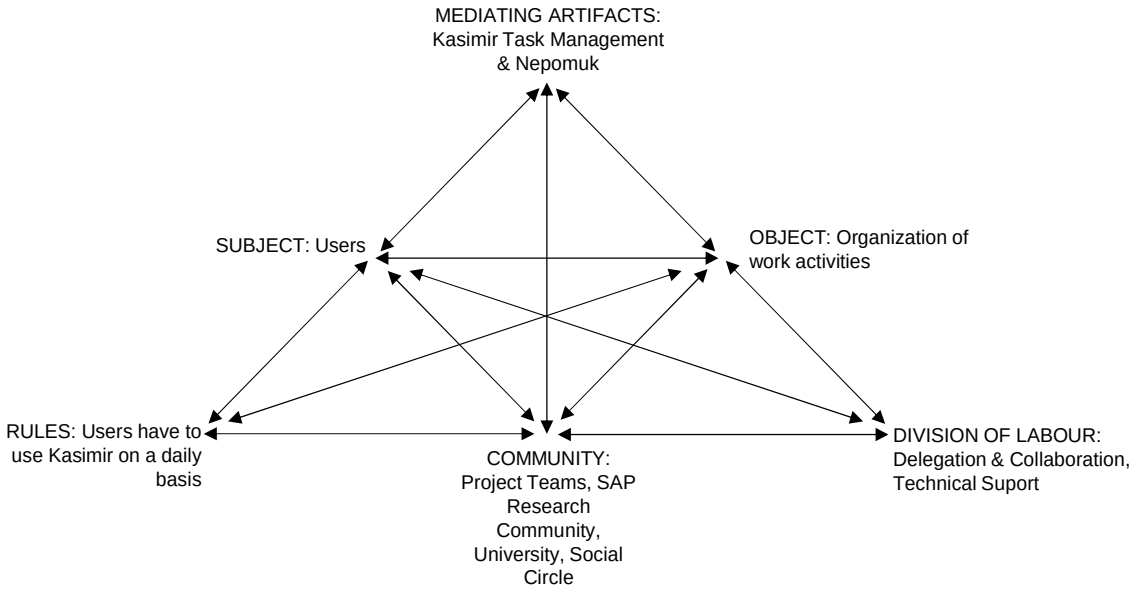


Figure 116: Activity System for the Kasimir Task Management Testing.

The next step involved the decomposition of the created Activity System with the help of the activity notation tool [Mwanza, 2001]. This process allowed simplifying the available information by breaking it down into six sub-triangles. Nonetheless, we had to alter the method to fit our testing. Since the components for collaboration and delegation of Kasimir were not functional for the testing period we had to reduce the activity notation to five sub-triangles. In the following phase the research questions specific to each particular combination within the activity notation were generated, see Figure 117.

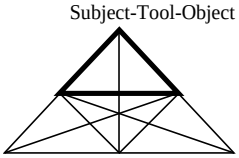
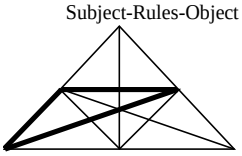
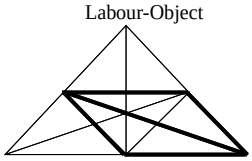
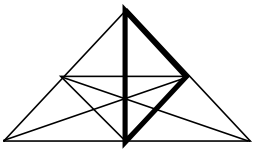
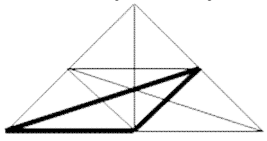
Sub-Activity Triangle focused on	Questions generated from Case Study	Identified Area of Contradiction	Resulting Contradictions/Questions (identified from field notes, demo discussion and informal interviews)
Subject-Tool-Object 	How does the Kasimir Task Management (tools) support the organization of activities (object) for the user (subject)?	Kasimir Task Management	• Benefit compared to regular task management tools? • Adding documents to a task useful? • Subtasks → actually used as a mean to structure tasks? • Task-Pattern as a mean for knowledge retrieval? • Language terms (Abstractor, Instance, etc) limit understanding & make it difficult to use • Why no reminder on tasks? • Daily vs. hourly setting? • Nepomuk as the running environment→ useful? • Semantic annotation helpful?
Subject-Rules-Object 	How does the rule of having to use the TM tool affect the way the users organize their activities?	Daily use of Kasimir	• Is it feasible to use Kasimir on a daily basis? • Does it yield benefits using Kasimir regularly? • Perceptions on Kasimir might change over time • High amount of manual effort → How to reduce manual effort?
Subject-Community-Division of Labour-Object 	How does the delegation and collaboration on tasks with team members influence the way the user organizes his/her work activities? To which extent do users work individually or achieve their goals in collaboration?	Delegation & Collaboration	• Can Kasimir support the interaction with team members? • *Delegtion & collaboration tools do not work in the deployed version* • Is collaboration on tasks desirable?
Community-Tool-Object 	How can project teams benefit from Kasimir TM and organize their activities among them more effectively?	Kasimir TM vs. Regular TM	• Compared to regular TM tools does Kasimir enable or hinder the exchange of tasks among teams? • Would Kasimir be a beneficial tool to support project coordination?
Community-Rules-Object 	Does the rule of using Kasimir on a daily basis affect the way teams coordinate and organize their tasks?	Kasimir as a communication tool	• All team members have to use Kasimir → feasible? • Exchange of tasks and documents

Figure 117: Activity Notations Table for Kasimir Task Management application.

Stage five entailed the gathering and the analysis of the data generated within stage four. Therefore, we collected the necessary data through ethnographic methods. This involved the observation of the testers in their daily work environment as well as throughout their breaks and in informal meetings. In addition, a feedback session was held throughout the testing where testers could exchange their experiences. We captured the obtained information by pencil and paper, but also by audio recording. The collected feedback was then transferred into an Excel based feedback log for further analysis. This information yielded the identification of some areas of contradictions that became noticeable during the testing or were explicitly mentioned by the testers as shown in Figure 117.

The contradictions or setbacks that the users experienced throughout the testing were selected as a starting point for a deeper investigation within the questionnaire.

12.2.1.3 Evaluation Study Design

The question of how evaluations should be approached is not an easy methodological decision. The choice of evaluation depends greatly on the role that the users are seen as taking in interactions [Kaye&Sengers, 2007]. For instance, [Kelly, 2006] noted that the difficulty of studying people performing task management is due to the longitudinal nature of this activity. Hence, it would be false to assume that the storing and retrieving of tasks can be observed in a laboratory environment within one or two hours. Thus, this study takes a different approach by performing a long-term qualitative evaluation of Kasimir Task Management. As mentioned previously, this study uses common fieldwork approaches to collecting data, such as ethnographic observation of the users in their work environment and informal interviewing. These methods helped us in gaining a profound understanding of the users experience with Kasimir. This approach is also in congruence with the methods applied by activity theorists. With regards to usability, [Mwanza, 2001] advocates a longitudinal testing in the environment where the activity takes place. She also encourages the use of ethno-methodological techniques when analyzing human activity to understand what works and what does not work.

Ethnographic field research and well constructed interviews can be improved significantly through the triangulation of methods [Berg, 2007]. Moreover, triangulation in qualitative research can be important to issues of validity. This study relies on two types of triangulation: (1) triangulation by method (ethnographic observation, interviews, etc); (2) triangulation by data-type (qualitative questionnaire responses, qualitative recordings, and quantitative survey results) [Miles&Huberman, 1994].

The long-term evaluation *study was carried out over a period of 3.5 weeks* in December 2008. Before the testing, we held an initial demo session to acquaint the user with the features of Kasimir and the Nepomuk PIM system. We provided the user with a short software manual which also featured some use cases. Since the Nepomuk Social Semantic Desktop is the running environment of Kasimir, we installed this application first on the users' computers. After approximately one week Kasimir Task Management was installed as well and tested for a remaining 3 weeks. The users were advised to use Kasimir continuously or as often as time allowed throughout the work day. During the evaluation the users shared their experiences via email with us and with each other. Additionally, we held a feedback meeting, as suggested by [Morse, 2002], at half time of the testing which allowed the users to exchange experiences and point out problems. At the end of the evaluation we administered a questionnaire to all users via email. The composition of the questionnaire will be discussed below.

Regarding the *sampling* of a study, Miles and Huberman [Miles&Huberman, 1994] assert that qualitative researchers usually rely on small samples of people, nested in their context and studied in depth. Quantitative researchers in turn aim for larger numbers of context stripped cases and seek for statistical significance. Moreover, qualitative samples tend to be purposive, rather than random [Morse, 1989]. Relying on random sampling in qualitative research can actually deal the researcher a decidedly biased hand [Miles&Huberman, 1994]. Hence, we decided to adopt a purposive convenience sampling which led to an "eat your own dog food" approach.

Prior to the evaluation, we recruited 18 participants within SAP Research Karlsruhe. All participants were approached individually and familiarized with the subject under evaluation. However,

throughout the testing five testers had to withdraw from the evaluation due to time constraints and work load issues. Nonetheless, 13 users tested Kasimir Task Management and the Nepomuk PIM system for a period of 3.5 weeks. We did not offer any financial compensation to the participants for their efforts. Participation was solely voluntarily.

The age of the participants varied from 24 years to 45. Furthermore, three participants were female and ten were male. The seniority level of the testers ranged from student workers to senior researchers. Additionally, four participants were already familiar with Kasimir and the Nepomuk PIM system since they were working on a related project. This familiarization with the tools biased the users to rate the prototype slightly better than other users did. The positive rating can, however, also be accredited to their shorter learning curve. In addition, the users familiar with the two tools were also more forgiving whenever bugs and problems occurred throughout the testing. More so, all participants were familiar with the usage of Desktop computers.

The *questionnaire* represents one part of the survey process. A poorly written questionnaire will, however, not provide the researcher with the necessary data. Nonetheless, whenever the objectives are adequately specified the researcher's task is usually more straightforward. [Brace, 2004] further asserts that when the objectives are defined in detail the questionnaire answers can be deduced easier. The objectives for this evaluation's questionnaire were clearly defined throughout the methodological conception, see section 12.2.1.2, and well documented within the Activity Notation Table through the various research questions.

The questionnaire (see Appendix C) was divided into eight sub-categories. Each section consisted of quantitative (closed) and qualitative (open-ended) questions allowing for a more thorough analysis. The quantitative (closed) questions allow for a direct comparison of the responses and also aid in supporting the qualitative results. These closed questions were either presented on a 5point Likert-type-scale or on a simple "yes or no" scale. The survey language was English. Furthermore, the questionnaire was developed in an interactive style [Brace, 2004]. Therefore, we carried out a pilot study upfront to the initial survey period where two users answered the questionnaire. This helped us to verify the correctness of the question wording and yielded feedback in respect to the overall survey design. Subsequently, some questions had to be adjusted. Subsequently, one user rechecked the questionnaire again. We administered the final questionnaire via email to the users upon the completion of the testing.

12.2.2 Evaluation Results

As aforementioned, the main objective of this study was to identify how Kasimir supports the user in the organization of his/her work activities. Therefore, we relied on an activity theoretical approach and its principle of contradictions to guide this investigation. The analysis of contradictions included those occurring between and among elements of the activity system [Murphy&Rodriguez-Manzanares, 2008]. Therefore, contradictions became the guiding principle of our empirical research. A detailed analysis of the disturbances in the activity allowed us to determine potential improvement requirements for Kasimir and the underlying Nepomuk PIM system.

The data of this questionnaire and the information collected through observation yielded six key areas of contradictions, see Figure 118. The first contradiction concerns task management in general and the overall user experience. The various features of Kasimir were identified as the second area of potential contradiction. Since Kasimir relies on the Nepomuk PIM system as the surrounding infrastructure, the third disturbance addresses the (dis-)advantages of this setup. Contradiction

number four deals with Kasimir's collaboration components which were not functional during the testing. Throughout our field work investigations we noticed that some users had difficulties to use Kasimir on a daily basis. Hence, the fifth contradiction evolves around the daily and continued use of Kasimir. The last part of the questionnaire, and therefore contradiction number six, examines if the users' perceptions of the usefulness of Kasimir changed over time. Figure 118 shows a graphical display of the six contradiction areas.

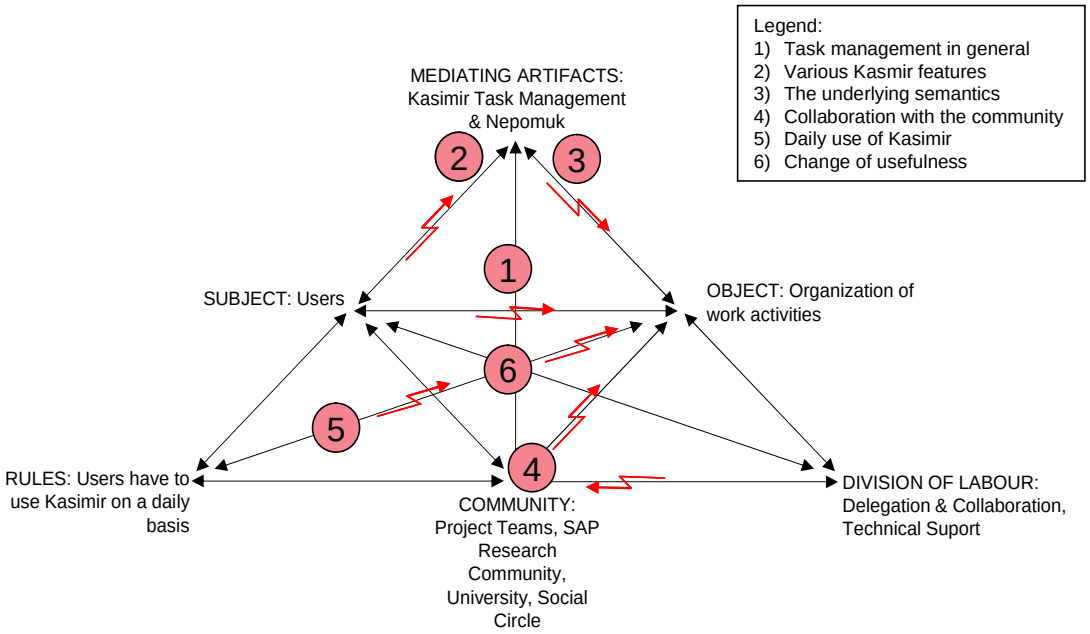


Figure 118: Contradictions & disturbances within the Activity System.

Table 22 summarizes the below presented evaluation results.

Focus of Evaluation	Observation Results
Task Management Overall Experience	• 70% of users considered Kasimir as an improvement compared to their usual mode of managing tasks. 7% considered it as a set-back. • Throughout users were content with the offered functionality on Kasimir, i.e., the opportunity to assign files, persons, subtasks to tasks. • Users would have appreciated better possibilities to prioritize their tasks. • Terminology in the UI was considered as requiring improvement. • The performance was not criticized but the users were hampered by still occurring bugs.
Kasimir Features	• 90% of the users appreciated the idea of integrating files in their tasks. • 55% of the users appreciated the ides of linking persons to their tasks (since the functional support was limited) • All users made use and appreciated the possibility to use subtasks. • 50% of the users were puzzled by the assignment of topics to task while 45% regarded this feature as helpful. • Most users liked the idea of task patterns but their handling was still considered as too complex despite the rather consequent preparation of the implementation by UI workshops.

Nepomuk PSEW [Nepomuk Consortium, 2009g] (and comparison to Kasimir)	• Most users were unsure how to use Nepomuk PSEW and considered it as too complex. • Throughout the users identified Nepomuk PSEW as a semantic application while they did not realize the semantic infrastructure of Kasimir (actually most users were puzzled when asked for the semantic nature of Kasimir).
Collaborative Aspects of Task Management	• Most users considered Task Management as a collaborative application. • Additionally collaborative task structures and chat functionality was required. • The necessity to exchange resources and metadata was not generally recognized by the users – probably due to missing communication features.
Daily and Continued use of Kasimir	• 30% of the users considered it as feasible to use Kasimir on a daily basis. • Users expressed that this was mainly due to usability deficiencies while the functionality was generally appreciated. • More integration with desktop applications was demanded. • The long-term experience increased the acceptance since users learned to find workarounds for deficiencies and became more familiar with the opportunities opened up.

Table 22: Main Evaluation Results.

Within the following paragraphs the results of each of the six contradictions will be discussed shortly.

12.2.2.1 Task Management in the Overall User Experience

The activity theoretical objective of this section of the questionnaire was to assess the users' perceptions on task management in general. We identified this area as a "primary contradiction" within the Kasimir activity system (Figure 118). According to [Engeström, 1999], primary contradiction can arise within each node of the central activity under investigation. Only users that are motivated to perform task management on a daily basis will find Kasimir beneficial. Assuming that users consider task management an unnecessary activity for the organization of their work than this would result in a principle conflict within Kasimir's activity system.

The questionnaire results showed that 70% of the participants already rely on computer based task management to efficiently structure their daily work activities. Nevertheless, the other 30% also perform task management regularly but prefer a "good hand written to-do list" over a computer based version. Hence, all participants considered task management essential in order to organize their work activities and were excited to test the Kasimir Task Management. All users voiced similar expectations of a computer based task management tool. First, it should allow them to structure, prioritize, and search their tasks easily. Second, the users expect the tool to remind them of upcoming events to keep them on track. Additionally, they expect a high degree of flexibility and compatibility with other computer based systems of any new tool.

Throughout our ethnographic research investigation we came across a user who considered his old task management ways more efficient than Kasimir. In order to verify this contradiction we asked the users to evaluate Kasimir in comparison to their former task management tools. Furthermore, we asked them to detail the benefits and limitations of Kasimir when compared to their old methods. Nine out of thirteen testers found it to be an improvement. Only one user considered it a set-back due to usability issues. The informants liked the fact that they could do almost anything task-related

in Kasimir. Additionally, the feature of relating documents, persons, and tags to tasks was generally well accepted. The users were also very fond of the possibility to decompose a task into subtasks. Overall, the users were positively surprised by the vast amount of features offered within Kasimir.

Nevertheless, there were also some features that the users missed. Kasimir, for instance, only allows the user to prioritize a task in a rather simple way, i.e., in the categories *low*, *medium*, and *high*. Also, a reminder functionality of upcoming tasks is lacking. Some participants found Outlook task management easier to handle and almost all were repelled by the complexity of the Nepomuk PIM system. Most users were additionally confused by the language terms used in Kasimir and the underlying Nepomuk PIM system (e.g. abstractor, instance, etc) and considered them a usability limitation.

In sum, the users liked many functionalities offered in Kasimir. However, according to the respondents the Nepomuk PIM system proved to be more of a limitation than a benefit for Kasimir. This can be attributed to the incomplete development of the two tools. The users were confronted with various bugs and difficulties which limited their overall experience.

These results and the more detailed issues are described in the following sections.

12.2.2.2 Kasimir Features

Each Kasimir feature can either support or limit the users in the efficient organization of their work activities. Hence, the features of Kasimir can be seen as a possible source of "secondary contradictions". Secondary contradictions arise between the constituent nodes (e.g., between the subject and the tool) of the central activity system [Engeström, 1999]. The following activity theory based research question guided the investigation of the various components: How do the Kasimir Task Management tools support the organization of work for the user, see Figure 117? Furthermore, it was inquired if it supports the users in achieving their goals. In sum, activity theory advocates a detailed analysis of the relationship between the various features offered in Kasimir (tools) and the user (subject). In the following paragraphs we will assess this relationship.

The users genuinely liked the idea of attaching documents to their tasks. Twelve out of thirteen participants thought that adding documents to their tasks helped them organize their work more efficiently. This application provided the user with an easy access to documents and it served as a reminder that a document is connected to a certain task. Nevertheless, the informants also stressed the importance of task delegation that includes document attachments.

The possibility of linking persons to tasks was only considered useful by 55% of the testers. Most users found it rather cumbersome to add a contact manually in Kasimir. They expected some kind of linkage to their Outlook address book or at least an automated transfer of their contacts stored in the Nepomuk PIM system. However, it needs to be taken into consideration that linking persons to tasks is only fully beneficial for the user if a full bunch of collaboration and delegation features generally available.

The concept of subtasks was well embraced by the Kasimir users. All users used this feature continuously throughout the testing period. Thus, all thirteen participants agreed that this feature helped them particularly in organizing their work activities more effectively. The creation of subtasks permits the user to decompose a task into smaller components. Therefore, it reduces complexity and presents a natural way of tackling complex tasks. No negative feedback was reported for this application.

Tagging in Kasimir is the annotation of tasks with a word as a descriptive piece of information. This fosters the organization of tasks via categorization. Although the concept itself seems to be rather straightforward, it caused some confusion among the users. At first, most participants had to understand the idea of tagging itself. Subsequently, more than 50% of the users found it difficult to utilize tags when performing task management. Most informants thought that tagging would only be beneficial whenever a user has to juggle a lot of tasks simultaneously. Furthermore, a respondent noticed that "one ends up with a bunch of tags which are semantically close". He suggested that a method to cluster tags into taxonomies would be even more helpful. Regardless, of the limitations, around 45% of the users found tags to be helpful when organizing their work. According to one user, tagging can be quite useful as it is "a simple method of grouping things across structures".

The majority of the users agreed that bookmarks provide a direct access to websites that are connected to tasks. Hence, it can be beneficial for the organization of their work activities. Nevertheless, most testers did not make regular use of this application. The results showed that the respondents liked this application but the regular usage seemed to depend on their individual preferences but also on the nature of the task. Some users expressed a desire for a Bookmark plug-in in their Internet browser to facilitate the utilization of this tool.

The *Task Patterns*[20] were another feature of Kasimir that was particularly examined. Since the Task Pattern approach presents a new task management paradigm, we provided the users with a short use case manual. After one week of testing we asked the participants (via email) to create a Task Pattern of a recurring task. In total, nine users followed this request and extracted a pattern of a task. The creation of this pattern proved to be rather difficult for most users. The interface concept, in particular the Task Compare view, was not as intuitive as expected and too complex for the users to understand by themselves. Therefore, we offered some participants technical assistance for the creation of their Task Pattern.

The qualitative questionnaire responses showed that the respondents liked the conceptual idea behind the Task Pattern which theoretically allows them to structure and automate reoccurring tasks in a similar manner. Nonetheless, the users criticized the insufficient usability of this tool. Subsequently, not all users found it useful for the organization of their knowledge. Regardless of the usability shortcomings, most users would like to utilize this tool in the future but only if usability can be enhanced.

12.2.2.3 Nepomuk PSEW as Generic User Interface on the Unified Personal Information Model

From an activity theory point of view the Nepomuk PSEW can become subject to a "tertiary contradiction" since its ultimate objective is the structuring and management of the users' knowledge. As we have described in section 10.1.3.4 there is a tight relation of the task management functionality offered by the Task Management Service and the semantic representation of the unified personal information model which is managed by the Nepomuk PIM system. However, the interaction with semantic-based systems is often dreary and users do not always hold the necessary skills and understanding. Hence, numerous disturbances can arise as a result of this relationship. The following paragraphs will shortly discuss the main findings in this respect.

[20] Task Patterns are a concept for knowledge transfer during the execution of a personal task, see [Riss et al., 2005] and [Riss&Grebner, 2008]. Task patterns are mentioned for the completeness of the evaluation report but we don't follow this topic up in this thesis.

The users made acquaintance with the semantic functionality of Nepomuk via PSEW which gave them access to a broad spectrum of modeling, annotation, and semantic browsing opportunities. Hence, when we talk about the perception of the Nepomuk PSEW in the following we mean the perception of PSEW.

To some extent, it is not surprising that the testers were fairly confused when being confronted with the Nepomuk PSEW for the first time. We asked the users if the Nepomuk PSEW proved to be a helpful tool for managing their personal knowledge. The results indicate that most testers were unsure on how to evaluate the Nepomuk PSEW. Seven out of ten participants answered this question with "neutral" whereas three rated it as helpful and three as unhelpful. However, it needs to be taken into consideration that during this testing most users made their first experiences with a semantic desktop environment. Subsequently, one can conclude that the Nepomuk PSEW requires some extensive training before any first time use, due to its complexity. This argument can be supported by the questionnaire results. Twelve out of thirteen users classified the Nepomuk PSEW as a complex tool. In the contrary, most testers attributed a normal or low level of complexity to Kasimir.

Based on the assumption that first time users might be particularly repelled by the semantic infrastructure of the two tools, we tried to assess the level of semantic visibility of this application. Eight out of thirteen participants rated the underlying semantic infrastructure to be evident or very evident for the Nepomuk PSEW. Five users remained neutral. Overall, no tester considered the underlying semantics of the Nepomuk PSEW to be hidden. The users were specifically bewildered when rating the semantic visibility level of Kasimir. Responses varied from evident to very hidden. In sum, first time users are likely to be confused by the semantic interface of the Nepomuk PSEW while they do not generally realize the semantic nature of Kasimir.

The *collaboration and community* features of Kasimir, in particular task delegation, allow the users to interact with the community. Nevertheless, these components of Kasimir were not generally available throughout the testing. From an activity theoretical perspective, these interactive and collaborative features play a key role for the user in achieving their objective of efficient work organization. However, since they were only partially available at the point of the testing we asked the users to name their expectations of such a tool.

Most users found Kasimir most beneficial when employed as a tool of collaboration and not solely as a task management tool. According to the results, this provides a distinguishing factor of Kasimir compared to ordinary task management tools. Concerning the expectations of this functionality, users mentioned the need for a function allowing for collaborative sharing of documents. Furthermore, the collaborative usage of attached documents should be made available. Assuming a user shares a task with a team mate it should display all task details to this person. Other users noted that they would prefer if Kasimir would be a community based tool where everyone can access it. Users also expressed the desire for a chat function in Kasimir. Finally, one user suggested that Kasimir should be connectable to heavy weight task execution planning such as Microsoft Project [Microsoft Corporation, 2009h].

We also asked the users about their perceptions of the *delegate a task* functionality within Kasimir. Most users agreed that it allows for a simple division of labor. Tasks that one would normally delegate via Email could then be delegated easily through Kasimir. Nevertheless, the users also noted that one usually delegates a task at a stage where one has not added a lot of resources to a task yet. In this case, the benefit of using the delegation function would be small for the receiver. Furthermore,

the users also expressed the need for a tracking function within this application that allows a supervisor to monitor the progress on a task.

Some users, however, also voiced concern that the usage of the *delegate a task* application depends highly on the employee's level of seniority. Most of the junior researchers among the testers indicated that they would feel uncomfortable using this component. A final solution of this feature could have a restriction allowing usage only to employees of a certain hierarchy level.

12.2.2.4 Daily and Continued Use of Kasimir

Another secondary contradiction resulted out of the rule that users had to use Kasimir on a daily basis, see Figure 118. This rule restricted the user to perform task management exclusively with Kasimir. However, from a change management perspective this can cause users' pervasive distress since it requires them to change their usual ways of task management. Within this section of the questionnaire we sought to assess the feasibility of a daily and continued use of Kasimir, and its benefits.

Only four out of thirteen users found it feasible to use Kasimir on a daily basis. This is not surprising since change takes time. One cannot expect the users to change their habits and preferences within the short timeframe of the testing. Hence, we asked them why or why not it was feasible to use Kasimir daily. Most users noted that a daily use would be possible if the interface and usability of the application could be enhanced. Currently, bugs and not functional features limited the users' experience considerably. Furthermore, most users named the overhead and resource demands of the Nepomuk PSEW as a limitation to a daily use of Kasimir. Last, most users noted the lack of integration with other Office applications as a restraint[21]. As a result, only five users expressed an interest in the continued use of Kasimir. Three out of thirteen users did not want to continue to use Kasimir after the testing due to the current usability limitations.

Change in the user perception of Kasimir

As mentioned before, users reported difficulties of adjusting to the Kasimir and Nepomuk interface. Specifically, the complexity of the semantic infrastructure with the semantic representation of the unified personal information model proved to be complicated for first time users. Hence, we anticipated a change of usefulness in the users' experience with Kasimir. This contradiction, see Figure 118, results out of the daily use of Kasimir. Assuming that Kasimir is used on a daily basis, the users might adjust to it and their perceptions of this application may change.

In fact, five out of thirteen users acknowledged a change in usefulness. The participants reported that at the beginning of the testing they had to enter a lot of data manually into Kasimir without much visible benefit at first. However, over the testing period the users learned to make use of the various functionalities of Kasimir to help them organize their work more efficiently.

Based on these results, a future evaluation of Kasimir or any other semantic related application should consider a longitudinal testing period to incorporate a likely shift in user attitudes. Additionally, extensive training will be required to familiarize future users with Kasimir and particularly the Nepomuk PIM system.

[21] The in section 6 described SAP Task application plug-ins were due to scoping reasons not subject to this evaluation.

12.2.3 Conclusions from the Evaluation

Engeström's activity system model and [Mwanza, 2001] Activity-Oriented Design Method produced meaningful insights into the Kasimir Task Management activity system. Findings from the analysis, using these two tools, revealed the weaknesses and strengths of the Kasimir Task Management. The activity notation tool [Mwanza, 2001] allowed for the thorough and stringent identification of contradictions within the activity. Paradoxically, contradictions should not be mistaken as dysfunctions, but as functions of a growing and expanding activity system. Thus, the detection of tensions and contradictions related to the mediating artifact was particularly helpful in identifying design requirements and enhancements for semantic based task management tools such as Kasimir. The main advantage of an activity theoretical based evaluation of a software application is that the user's interests and needs become understandable on the basis of the history and the current problems of their activities [Rajkumar, 2009].

Turning to the concrete conclusions regarding the usefulness it became clear that despite all efforts in improving the usability of the system it was still challenging for the users, partially due to still existing usability defects, partially due to bugs that still occurred due to the recentness of the prototype development. There are some issues, however, which have to be highlighted:

- To deal with semantic application like, e.g., Nepomuk PSEW, is definitely a considerable challenge for 'ordinary' users. In particular the manual effort of annotation and the still limited use make it difficult to persuade users to regularly work with such systems.
- Although the Kasimir application is *as semantic as* PSEW, its semantic character is rather hidden. This made it simpler for users to understand the application but nevertheless provided the benefits of a semantic application. Therefore we can consider this as a confirmation of our initial assumption that semantic technologies should not become too visible in the UI.
- Nevertheless the users' long-term perception shows that this is not an insuperable barrier. Thus, a considerable number 'discovered' benefits of the Nepomuk PIM system after some time of usage. This result indicates the general usefulness of the Nepomuk PIM system approach even if the technology and the UI are not yet fully optimized.
- The results also show that collaboration really plays a central role in the organizational setting. Support of collaboration such as delegation is crucial. However, experience also shows that this cannot be restricted to mere exchange of tasks. Thus, the workshop results have shown that users also expect further support if these functionalities are available.
- Even at the rather basic level at which they could be provided Task Patterns were considered as a useful means to support reoccurring activities. In particular the social aspect of Task Pattern handling was clearly recognized by the users, even if it could not be supported in the current version of the prototype.

A final remark concerns the methodology. As expected, we got the results that the users were disturbed by the still occurring errors and UI issues. Nevertheless the questionnaire based on activity theory was able to *look behind this veil* and discover aspects that tell us more about the proper usefulness of the develop technology. This is important since prototype development generally cannot provide the quality that characterizes product development. Nevertheless an evaluation of prototypes is required that is not obscured by superficial and often well known quality defects. The concentration on the possible contradictions that might occur in an activity diagram can help to better focus on the important and fundamental aspects.

13 Conclusions and Future Work

13.1 Summary

This thesis presents a *framework* for applying unified personal information for KWer activity-support across applications and workspaces, see Figure 119. The framework consists of the three components, i.e., applications supporting the KWer's personal activities, the software reference architecture and a unified personal information model.

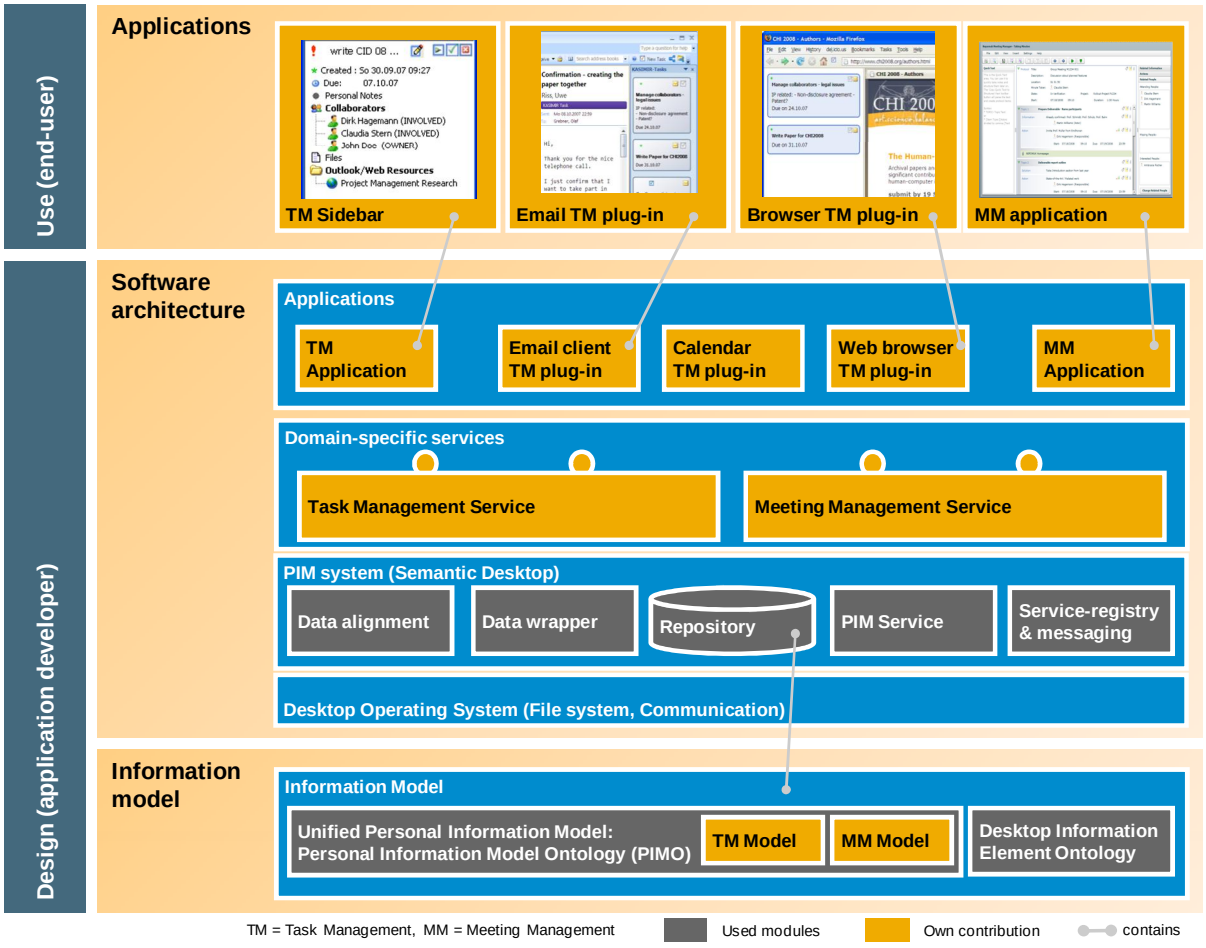


Figure 119: Framework for applying unified personal information for workspace-wide KWer activity-support.

Applications using a unified personal information model support a KWer's activities and tackle prevalent information fragmentation, both in desktop applications and workspaces. We demonstrate workspace-level application examples for personal task and meeting management, two higher-level supporting activities that a KWer can conduct. We show with a reference software architecture how application developers can efficiently implement support through the unified personal information model. Furthermore, we adapt the common, unified personal information model to the needs of the domain-specific KWer support use cases.

13.2 Contributions

The contributions of this thesis through the presented framework are twofold. Figure 120 summarizes these delivered contributions along the framework's core building blocks.

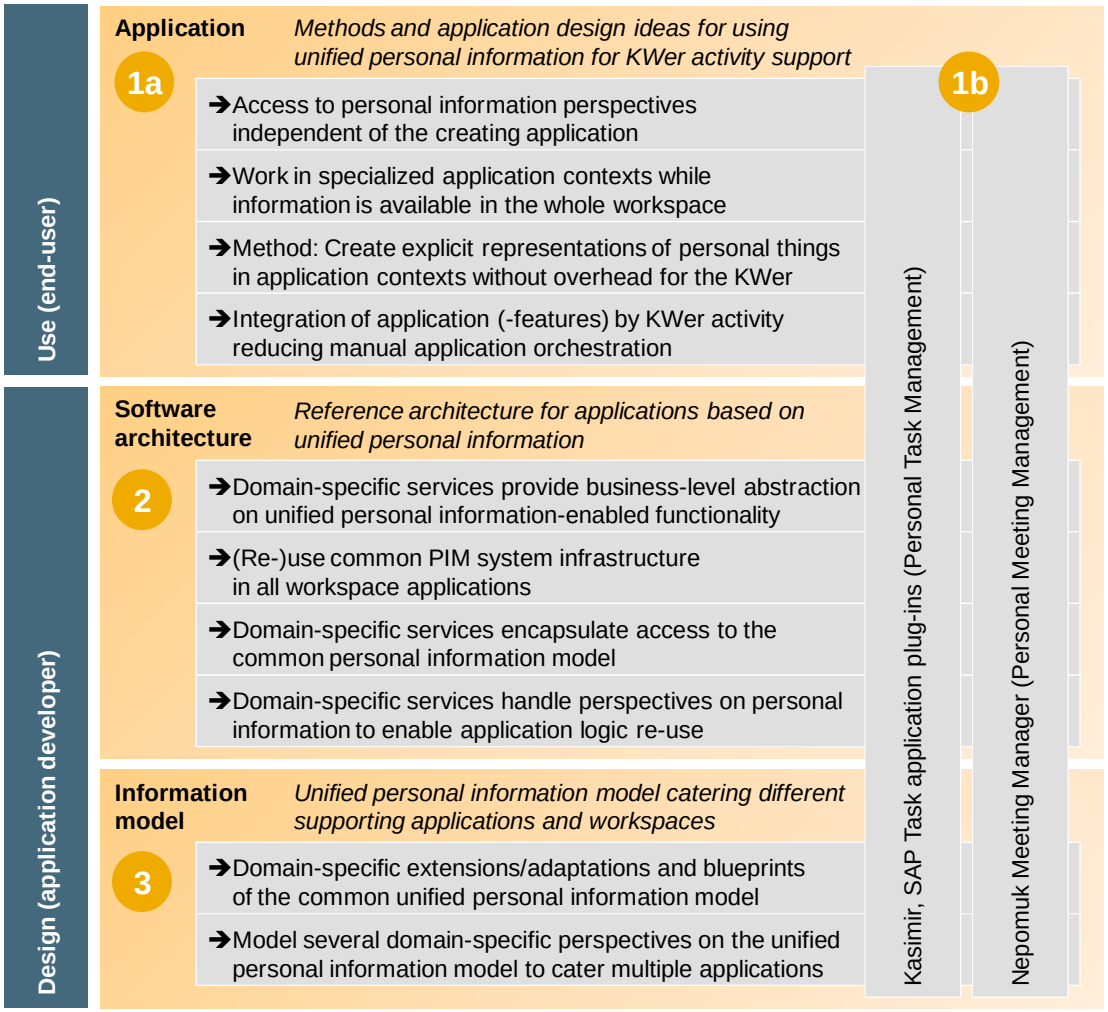


Figure 120: Summary of contributions.

13.2.1 Contributions for KWers as End-Users

Targeting end-users, we demonstrate three applications supporting a KWer's personal task management and personal meeting management activities which each tackle existing personal information fragmentation. In particular, the Kasimir application and the SAP Task application plug-ins support the KWer's personal task management activities and the Nepomuk Meeting Manager support the KWer's personal meeting management activities. These applications excel over existing state-of-the-art applications by providing the KWer with an integrated view on all information needed for the particular activity, independent of which application maintains the information on the KWer's desktop. This is achieved through using unified personal information.

The presented applications tackle the fragmentation of a KWer's personal information on multiple levels.

- On the user interface level, the KWer can see all relevant personal information needed for an personal activity integrated in one coherent user interface (Kasimir, Nepomuk Meeting Manager) or integrated in a context of an existing application (SAP Task application plug-ins). These applications enable the KWer to leverage existing personal information, contribute new personal information which can be used by all applications on the KWer's desktop, and provide an integrated user experience.

- On information level, the KWer can use one common set of personal information in an application supporting a particular KWer activity without any effort. Vice versa, personal information that the KWer created as part of the personal task management and personal meeting management activities can be used in other applications within the workspace.

An efficient implementation of the above presented applications has been achieved by using the methods, reference architecture and information models proposed by the demonstrated framework, see below for details.

In a more *general view* we contribute *methods and application design ideas to enable applications to tackle the fragmentation of personal information* induced by applications by *using unified personal information* and a set of methods. Below, we show methods and measures on how KWers in the role of *end-users* can use activity-support applications that leverage unified personal information across applications in the whole workspace. Thereby we generalize from the two application domains of personal task management and personal meeting management which are used as examples to develop and motivate these methods and application design ideas.

- *Access to personal information perspectives independent of the creating application* – We demonstrate a set of applications that present all perspective-related personal information in one application user interface to enable an integrated information supply for supporting a particular activity. Based on a unified personal information model these applications can present all personal information related to a particular perspective. For example, for personal task management activity support, an application can obtain and present all task-related information. This enables a KWer to see all task-related information in one place that is needed to manage and execute tasks. The Kasimir personal task management application enables the KWer to manage beneath a task list the related personal information, i.e., a task information perspective with respectively involved desktop information objects and a social perspective with the people involved in the task. With the Nepomuk Meeting Manager the KWer can see all personal information related to a meeting in one place. This again includes besides the meeting minutes a task information perspective with respectively involved desktop information objects and a social perspective with the people involved in the meeting.
- *Work in specialized application contexts while information is available in the whole workspace* – We present at the example of a set of applications how a KWer can manage personal information in specialized application contexts targeted to the sole support of a particular KWer activity. At the same time the KWer can make this information available beyond the individual application across the workspace to other applications. Thereby, we demonstrate this for both applications where the application developer has full control over the design as well as for established applications where only limited change possibilities do exist, e.g., due to closed source applications. The Kasimir personal task management application and the Nepomuk Meeting Manager demonstrate how an application designed from scratch can incorporate unified personal information and present it to the KWer's advantage, see section 5.6 and 7.5. The SAP Task plug-ins demonstrate the same capabilities for applications with limited programmatic modification possibilities at the example of a KWer's personal task management activities, see section 6.5. These applications thereby present the information targeted to the KWer's activity requirements in full compliance with usability standards and don't provide generic data management user interfaces which the KWer needs to adapt to each activity.

- *Create explicit representations of personal things in application contexts without overhead for the KWer (method)* – We explore and demonstrate a method to *create explicit representations of personal things without overhead for the KWer* and directly in an application context, see section 6. This enables the processing of these personal things as personal information representations and support applications can use this information. We extend existing applications with application features to capture and present personal information that fits the application context but hasn't yet been integrated in the respective application. Thereby these so-called application plug-ins provide the feature to create an explicit personal information representation directly in the application context by exploiting workaround actions already pursued by KWers. This prevents any additional effort by the KWer. We implemented these plug-ins for the personal task management activity support. These so-called SAP Task plug-ins work for the three applications Microsoft Outlook Email, Microsoft Outlook Calendar and Mozilla Firefox and demonstrate how a KWer can create tasks from emails, appointments and websites using known actions, see section 6.
- *Integration of application (-features) by KWer activity reducing manual application orchestration* – We explore and demonstrate the integration of several application features into one application context that follows a foundational information concept. We thus address the application (-feature) fragmentation regarding a particular activity in a KWer's workspace with workspace-level application integration for this KWer activity. At the example of the KWer's personal meeting management activity, we show the Nepomuk Meeting Manager as an application which integrates communication and task management features. Other specialized desktop applications, like, e.g., Microsoft Outlook for composing and sending emails or the Kasimir task management application for creating tasks, provide these features. The Nepomuk Meeting Manager provides in its application context a process-oriented overview on possible actions. When the KWer triggers one of these actions, the Nepomuk Meeting Manager invokes these features and supplies the needed information to the applications feature.

13.2.2 Contributions for Application Developers

For *application developers* we show how to efficiently implement domain-specific applications supporting a KWer's activities by leveraging perspectives of the KWer's personal information, i.e., perspectives on the available unified personal information.

We present a *reference architecture* for applications based on unified personal information. A sound software architecture is core for an efficient implementation. The presented reference architecture features the following contributions:

- *Domain-specific services provide business-level abstraction on unified personal information-enabled functionality* – The domain-specific services have a domain-specific interface focused on the particular domain, e.g., personal task management instead of offering a generic personal information management interface. With a business-level abstraction layer for, e.g., personal task management and targeted to, e.g., tasks and other involved information objects, a developer doesn't have to deal with generic information management functionality. Such generic information management functionality would need to be manually customized by the developer or even the end-user.
- *(Re-)use common PIM system infrastructure in all workspace applications* – The domain-specific services provide an abstraction layer that hides the internals of the underlying PIM

system. Through encapsulation, the domain-specific services reduce the complexity for developers. They don't need to dive into the particularities of handling tasks or personal information management when they, e.g., just want to quickly create a few tasks from an application. Through the domain-specific services the developers can transparently leverage a PIM system, e.g., the Nepomuk Social Semantic Desktop, for, e.g., task information management without caring about the details on how this is achieved.

- *Domain-specific services encapsulate access to the common personal information model* – The domain-specific services ensure uniform personal information handling by providing developers with consistent task management services and not only with the unified personal information management model. Using the domain-specific services, application developers can consistently create, read, update and delete personal information. This integration on application/ business-logic level ensures the unified handling of information and helps this way to keep the parasol information consistent and leverage proven business logic. Having only a shared data model and description represents an information integration-level, but as mentioned before doesn't eliminate possible handling inconsistencies through different implementations and neglects existing proven business logic.
- *Domain-specific services handle perspectives on personal information to enable application logic re-use* – The domain-specific services encapsulate domain-specific support business logic to re-use common domain-specific support functionality in several applications. This avoids the duplication of source code and subsequent disadvantages like reduced maintainability.

The *unified personal information model* adapted to cater different supporting applications and workspaces underlies the presented applications.

- *Domain-specific extensions/adaptations and blueprints of the common unified personal information model* – Application developers can extend the unified personal information to cover further use cases beyond the core supported functionality trough the existing unified personal information model. When targeting, e.g., specific task management use cases, this information is not directly relevant for core personal task management use cases. Capturing the specific information in a separate information model and thus a separate Ontology, e.g., TMO Extensions, allow a modular composition of the task information model according to the envisioned use cases. These TMO extensions can for example support task knowledge management for tasks with task patterns [Riss et al., 2005] [Riss&Grebner, 2008]. We presented the TMOE ontology as TMO extension to support the Task Knowledge Management use case. It acts as blueprint for further unified personal information model extensions by showing developers a practical example on how to extend the unified personal information model.
- *Model several domain-specific perspectives on the unified personal information model to cater multiple applications* – We show the adaptations to the unified personal information at hand of the two example use cases of personal task management and personal meeting management and explain how this can be reproduced for further domains. The Task Management Model (TMM) is a conceptual representation of tasks and related information for use in personal task management applications for KWers. It is part of the KWer's unified personal information model. The Task Model Ontology (TMO) represents an agreed, domain-specific information model for tasks and covers personal task management use cases, see

section 9.1. In the *Task Model Ontology (TMO)* we extend the task concept of the personal unified information model to meet the requirements for the support of the personal task management activity. The Meeting Management Model (MMM) is a conceptual representation of meetings and related information for use in personal meeting management applications for KWers. It is as well part of the KWer's unified personal information model. Its representation bases on the PIMO ontology and thus represents an agreed, domain-specific information model for meetings and related information and covers the personal meeting management use cases as defined in section 4.3.

13.3 Interdisciplinary Research

The here presented work is largely interdisciplinary. It is mainly located in the domain of *business information science*[22]. However it contributes to and leverages several further research disciplines, see Figure 121.

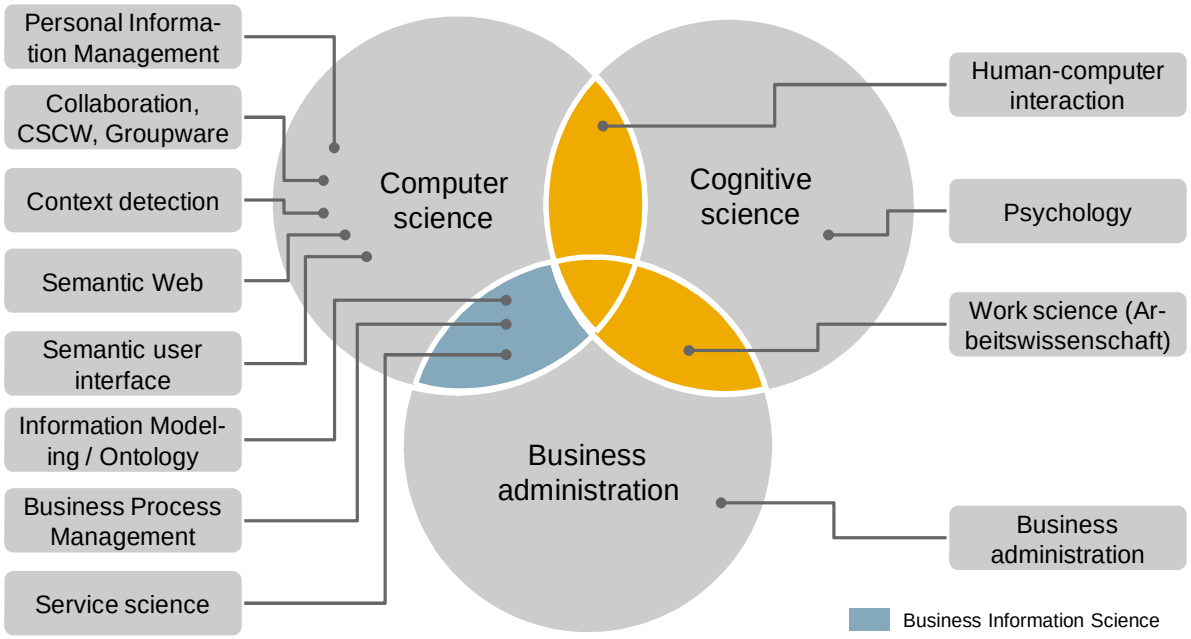


Figure 121: Interdisciplinary research.

In the domain of *computer science* the presented work contributes first and foremost to the disciplines of personal information management and semantic web with its sub-disciplines of semantic user interfaces and information modeling / ontology engineering. Furthermore, it involves the disciplines of collaboration, computer-supported cooperative work (CSCW) and groupware as well as the discipline of context detection / management. Due to the use of service-oriented architectures on the desktop this work as well touches the service science discipline.

In the domain of *cognitive science* the presented work leverages mainly the human-computer-interaction discipline. Furthermore, the psychology and work science[23] disciplines play an important role.

In the domain of *business administration* the presented work uses the core business administration discipline and draws knowledge from the business process management discipline.

[22] In German "Wirtschaftsinformatik"
[23] In German: "Arbeitswissenschaft"

13.4 Concurrent Prototype Development Process for Multiple Prototypes

The here presented work has been developed in a concurrent prototype development process with multiple prototypes involved. Multiple prototypes were in development at the same time. Each prototype passed through the complete development process. Thereby, multiple prototypes were, each their respective stage, in parallel development. For example, while the Kasimir prototype already underwent the second refinement phase, the Nepomuk Meeting Management prototype was in the "design prototype" phase. Figure 122 shows a principle sketch of the development phases of the here presented prototypes and their relation in time.

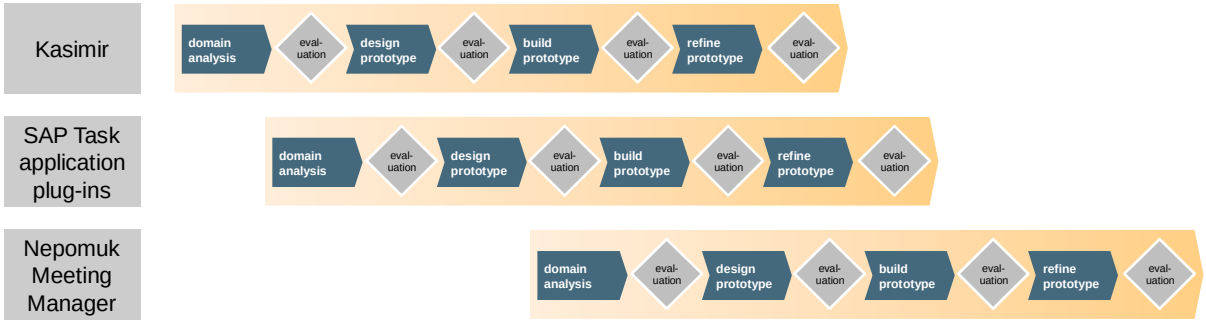


Figure 122: Development of several prototype parts in parallel.

Thereby the later positioned prototypes such as the here presented Nepomuk Meeting Manager could draw from the already gained development experience. This resulted in a significantly shorter "build prototype" phase for these prototypes. For example, the underlying PIM system and existing domain-specific services could be used and the experience of how to deal with these components reduced the development time for the new component.

13.5 Future Work

Several areas with open questions call for further research when looking beyond the scope of this thesis. Looking at overall technology trends, one can expect that the here used semantic desktop PIM system will find its way as basic system into the mainstream desktop operating systems as well as mobile phone and social network platforms. Given a then broad availability of personal information clouds for individual KWers, the question of suitable applications and efficient user interaction is still an issue. We look at these areas for further research on the one hand from an end-user's perspective on applications and on the other hand from a software developer's perspective on the architecture.

13.5.1 Future Work from an End-User Perspective

There are several personal KWer activities where supporting applications don't integrate with the KWer's personal information cloud. In this thesis we showed a way to integrate the KWer's personal information cloud into applications targeted at a KWer's personal task management and personal meeting management activities. There are further personal primary activities and personal supporting activities beyond the here targeted ones, see section 2.2.1. The open question is how the KWer's personal information cloud can be integrated into applications supporting these activities. This includes finding further use cases, where it is possible to create personal information from the KWer's personal activities, i.e., to create personal information in the personal information cloud without overhead for the KWer. For example, managing the KWer's personal social network is one of these use cases. Technology-wise, in continuing the here proposed approach of integrating unified personal information into desktop applications, there are several ways to address this question. This includes making the KWer's personal information cloud available for further applications in a

workspace, e.g., by directly integrating support for the personal information cloud into the respective application. Alternatively the development of further plug-ins is possible for applications which allow, e.g., due to proprietary implementations, only limited modifications. Another interesting approach is the development of plug-ins that extend a number of applications but are technologically application-independent. Such plug-ins provide generic access to a KWer's personal information but hook at the same time onto an application by observing operating system events triggered by the application's user interface. This relieves plug-in developers from developing a number of plug-ins in respectively application-specific technologies.

Today's KWer increasingly works with multiple computers and mobile devices at the same time. The KWer's personal information cloud should be available to all devices and applications a KWer uses. The corresponding open research question is how to make the personal information cloud available in such a distributed environment, e.g., by dealing with synchronization issues and potentially limited connectivity. Overcoming such issues allow the personal information cloud to 'follow' the KWer and thus reduce the effort in keeping and managing this personal information.

A KWer interacts and works collaboratively with other KWers and barely works isolated on her own. For example, task management in groups is collaborative work. In such a collaborative situation the open research question is how to share information captured in the personal information cloud with other collaborators. Given that the KWer wants to share a dedicated part of the personal information cloud to help other KWers, the design challenge is to make it easy for the KWer to share precisely defined parts of the KWer's personal information cloud. The core of this design challenge is to achieve this with minimal or even without overhead effort for the KWer. Furthermore, the KWer may want to benefit from the feedback on the sent out parts of her personal information. This then requires connecting the entities of two KWer's personal information clouds and delivering feedback events based on this evolving information.

13.5.2 Future Work from an Application Developer Perspective

There a number of architectural challenges from a software developer's perspective. By supporting additional personal KWer activities, additional domain-specific services emerge. The goal is to support all personal supporting activities with a set of domain-specific services. The open research question in such a situation is how to manage, update and deploy these services on each KWer's individual workspace while at the same time maintaining the KWer's personal information cloud.

Most of the here reviewed current workspace applications don't leverage contextual information. Contextual information on the one hand consists of the KWer's work context within the workspace, e.g., which applications are active and which information objects the KWer currently works on. On the other hand, sensors can detect information about the KWer's work environment outside of the workspace, such as information about the current location or people in proximity. Leveraging this contextual, personal information additionally benefits the support through the KWer's personal information cloud as it enables, e.g., on mobile devices location-based applications that take into account the KWer's personal information. The research question is thereby how to manage and derive contextual information from multiple sensors to be able to map it with the KWer's personal information cloud.

Currently the personal information cloud is stored in a central, semantic repository. A centralized storage imposes some constraints. For example, currently most applications on a KWer's workspace maintain their own information repository. Requiring all this information to be stored in a central

repository is a disruptive requirement and hard to achieve in practice. The research question is whether and how a de-centralized storage of the personal information cloud is feasible. When some applications already keep information related to the personal information cloud, is it possible to build a personal information cloud by integrating such distributed information sources. This gets even more complex in a distributed environment with multiple computers and devices.

References

3ado AG (triado) (2009). *JourFixe.* http://www.3ado.com. Accessed 2009.03.04.

Abecker, A., Bernardi, A., Elst, L. v., Lauer, A., Maus, H., Schwarz, S. & Sintek, M. (2001). *FRODO: A Framework for Distributed Organizational Memories. Milestone 1: Requirements Analysis and System Architecture - DFKI Document D-01-01.* http://www.dfki.uni-kl.de/dfkidok/publications/D/01/01/abstract.html. Accessed 2009.03.04.

Ackermann, T. (2008). *Nepomuk Semantic Meeting Support - Study thesis.* Darmstadt University of Technology.

Adar, E., Karger, D. & Stein, L. A. (1999). Haystack: per-user information environments. *CIKM '99: Proceedings of the eighth international conference on Information and knowledge management (p.* 413-422). New York, NY, USA: ACM.

Adobe Inc. (2007). *Flex 2, technical white paper.* http://www.adobe.com/products/flex/whitepapers/pdfs/flex2wp. Accessed 2009.03.03.

Adobe Inc. (2008). *Adobe Air.* http://www.adobe.com/products/air/. Accessed 2009.03.03.

Adobe Inc. (2009). *Adobe Acrobat Connect.* http://www.adobe.com/products/acrobatconnect/. Accessed 2009.03.04.

Aduna B.V. (2006). *The SeRQL query language (revision 1.2).* http://www.openrdf.org/doc/sesame/users/ch06.html. Accessed 2009.03.02.

Aduna B.V. (2009). *Elmo.* http://www.openrdf.org/doc/elmo/1.4/. Accessed 2009.03.03.

Aduna B.V. (2009). *User Guide for Sesame 2.2.* http://www.openrdf.org/doc/sesame2/2.2.4/users/index.html. Accessed 2009.03.03.

Aduna B.V.; DFKI GmbH (2005). *Aperture.* http://aperture.sourceforge.net. Accessed 2009.03.02.

Allen, D. (2003). *Getting things done: The art of stress-free productivity* Penguin Books.

Antunes, P. & Costa, C. J. (2003). Perceived Value: A Low-Cost Approach to Evaluate Meetingware. *Groupware: Design, Implementation, and Use (*Vol.2806/2003, S. 109-125). Berlin / Heidelberg: Springer.

Avente Pty Ltd (2009). *ThinkingRock.* http://www.trgtd.com.au/. Accessed 2009.03.04.

Baird, S. (2009). *MonkeyGTD.* http://monkeygtd.tiddlyspot.com/#MonkeyGTD. Accessed 2009.03.04.

Bellotti, V., Dalal Brinda Good, N., Peter, F., Bobrow G., D. & Ducheneaut, N. (2004). What a to-do: studies of task management towards the design of a personal task list manager. *Proceedings of the SIGCHI conference on Human factors in computing systems (p.* 735-742). ACM New York, NY, USA.

Bellotti, V., Ducheneaut, N., Howard, M. & Ian, S. (2003). Taking email to task: the design and evaluation of a task management centered email tool. *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems (p.* 345-352). Ft. Lauderdale, Florida, USA: ACM New York, USA.

Bellotti, V., Ducheneaut, N., Howard, M. & Smith, I. (2002). Taskmaster: recasting email as task management. *Workshop: Redesigning Email for the 21st Century, CSCW, 2002.*

Bellotti, V., Ducheneaut, N., Howard, M., Ian, S. & Grinter, R. E. (2005). Quality versus quantity: E-mail-centric task management and its relation with overload. *Human-Computer Interaction , 20*(1), 89-138.

Berg, B. L. (2007). *Qualitative Research Methods for the Social Sciences (6th Edition)* Allyn Bacon.

Bergman, O., Beyth-Marom, R. & Nachmias, R. (2006). The project fragmentation problem

in personal information management. *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems (p.* 271-274). New York, NY, USA: ACM.

Boardman, R. (2004). *Improving tool support for personal information management.* http://www.iis.ee.ic.ac.uk/~rick/thesis/boardman04-thesis.pdf. Accessed 2009.03.02.

Brace, I. (2004). *Questionnaire design: how to plan structure and write survey material for effective market research* Kogan Page.

Brunzel, M. & Mueller, R. M. (2008). Handling of Task Hierarchies on the Nepomuk Social Semantic Desktop. *Proceedings of the 10th international conference on Visual Information Systems: Web-Based Visual Information Search and Management (p.* 315-318). Springer.

Brunzel, M., Grebner, O. (2008). *Task Model Ontology (TMO).* http://www.semanticdesktop.org/ontologies/2008/05/20/tmo/. Accessed 2009.03.03.

Carnegie Mellon University – Human-Computer Interaction Institute (2007). *CALO Stardust website – Feature description.* http://www.hcii.cmu.edu/M-HCI/2007/SRI/features.html. Accessed 2009.03.04.

Carnegie Mellon University – Human-Computer Interaction Institute (2007). *CALO Stardust website.* http://www.hcii.cmu.edu/M-HCI/2007/SRI/index.html. Accessed 2009.03.04.

Central Desktop Inc. (2009). *Central Desktop quick tour: Centralize Communication.* http://www.centraldesktop.com/tour_overview?slide=6. Accessed 2009.03.04.

Central Desktop Inc. (2009). *Central Desktop tour: Milestones.* http://www.centraldesktop.com/tour?slide=5. Accessed 2009.03.04.

Central Desktop Inc. (2009). *Central Desktop.* http://www.centraldesktop.com/. Accessed 2009.03.04.

Cisco Systems Inc. (2009). *WebEx WebOffice.* http://www.weboffice.com/taskmanager/. Accessed 2009.03.04.

CollabNet Inc. (2009). *SVN. Subversion.* http://subversion.tigris.org/. Accessed 2009.03.03.

Czerwinski, M., Horvitz, E. & Wilhite, S. (2004). A diary study of task switching and interruptions. In E. Dykstra-Erickson & M. Tscheligi (Eds.), *Proceedings of ACM CHI 2004 Conference on Human Factors in Computing Systems (p.* 175-182). ACM Press.

Davenport, T. H., Jarvenpaa, S. L. & Beers, M. C. (1996). Improving Knowledge Work Processes. *Sloan Management Review , 34*(4), 53-65.

Dawson, F. & Stenerson, D. (1998). *Internet Calendaring and Scheduling Core Object Specification (iCalendar).* http://www.ietf.org/rfc/rfc2445.txt. Accessed 2009.03.03.

Dicks, R. S. (2002). Mis-usability: on the uses and misuses of usability testing. *Proceedings of the 20th annual international conference on Computer documentation (p.* 26-30). ACM New York, NY, USA.

Dourish, P., Edwards, W. K., Lamarca, A. & Salisbury, M. (1999). Presto: An Experimental Architecture for Fluid Interactive Document Spaces. *ACM Transactions on Computer-Human Interaction , 6*, 133-161.

Drucker, P. F. (1993). *Post-Capitalist Society*. Oxford: Butterworth Heinemann.

Dumas, J. S. & Redish, J. (1999). *A practical guide to usability testing* Intellect Books.

Eckstein, R., Loy, M. & Wood, D. (1998). *Java swing* O'Reilly Associates, Inc. Sebastopol, CA, USA.

Ellis, C. A., Rein, G. L. & Jarvenpaa, S. L. (1989). Nick experimentation: some selected results [meeting support]. *Vol.III: Decision Support and Knowledge Based*

Systems Track, Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences , 3, 359-369 vol.3.

Engeström, Y. (1987). *Learning by expanding: An activity-theoretical approach to developmental research* Orienta-Konsultit Oy Helsinki.

Engeström, Y. (1999). Activity theory and individual and social transformation. In Y. Engeström, R. Miettinen & R. Punamäki (Eds.), *Perspectives on Activity Theory (p.* 19-38). New York: Cambridge University Press.

Fisher, D. & Dourish, P. (2004). Social and temporal structures in everyday collaboration. *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems (p.* 551-558). New York, NY, USA: ACM.

Fisher, D. & Nardi, B. (2007). Soylent and ContactMap: Tools for Constructing the Social Workscape. In V. Kaptelinin & M. Czerwinski (Eds.), *Beyond the desktop metaphor: designing integrated digital work environments (p.* 171-190). MIT Press, Cambridge, Mass.

Franz, T. (2008). On the Evaluation of Personal Knowledge Management Solutions: Evaluating Tools of the X-COSIM Semantic Desktop. In J. Baumeister & M. Atzmüller (Eds.), *Workshop of the Knowledge Management Group (FG-WM), LWA.*

Freeman, E. & Fertig, S. (1995). Lifestreams: Organizing Your Electronic Life. *AAAI Fall Symposium: AI Applications in Knowledge Navigation and Retrieval.*

Freeman, E. & Gelernter, D. (2007). Beyond Lifestreams: The Inevitable Demise of the Desktop Metaphor. In V. Kaptelinin & M. Czerwinski (Eds.), *Beyond the desktop metaphor: designing integrated digital work environments (p.* 19-48). MIT Press, Cambridge, Mass.

Fuchs, L., Poltrock, S. & Wetzel, I. (2001). TeamSpace: an environment for team articulation work and virtual meetings. *Proceedings. 12th International Workshop on Database and Expert Systems Applications* , 527-531.

Glos, R. G. (2005). *Firefox optimal einsetzen* Franzis Verlag.

Goesmann, T. (2002). *Ein Ansatz zur Unterstützung wissensintensiver Prozesse durch Workflow-Management-Systeme [Elektronische Ressource]* Berlin, Freie Univ., Diss.

Google Inc. (2009). *Google Gadgets.* http://www.google.com/webmasters/gadgets/. Accessed 2009.03.03.

Grebner, O. & Brunzel, M. (2008). *Generate RDFSBeans with RDFReactor - How To.* http://dev.nepomuk.semanticdesktop.org/wiki/RDFReactor. Accessed 2009.03.04.

Grebner, O. & Brunzel, M. (2008). *TMO Person Involvement.* http://www.semanticdesktop.org/ontologies/2008/05/20/tmo/#PersonInvolvement. Accessed 2009.03.04.

Grebner, O. & Ebner, H. (2008). Collaborative Tasks using Metadata Conglomerates – The Social Part of the Semantic Desktop. In C. S. Calude, H. Maurer, A. Salomaa & K. Tochtermann (Eds.), *Proceedings of I-SEMANTICS '08, Graz, Austria, 3-5 September, 2008.*

Grebner, O. & Riss, U. V. (2008). Implicit Metadata Generation on the Semantic Desktop Using Task Management as Example. In S. Borgo & L. Lesmo (Eds.), *Formal Ontologies Meet Industry - Proceedings of FOMI 2008 - 3 rd Workshop on Formal Ontologies Meet Industry, 5-6 June, 2008, Torino, Italy (p.* 33).

Grebner, O. & Riss, U. V. (2008). Leveraging Personal Information in Enterprise Environments – Problem Framework and Solution Outline using the Nepomuk Social Semantic Desktop. In D. Harorimana & D. Watkins (Eds.),

Proceedings of the 9th European Conference on Knowledge Management (ECKM), Southampton Solent University, Southampton, UK, 4-5 September 2008.

Grebner, O. (2009). Architecture Framework for Higher-Level Activity Support Using Multiple Functional Workspaces on the Desktop. In K. Hinkelmann & H. Wache (Eds.), *WM2009: 5th Conference on Professional Knowledge Management, Solothurn, 25.3.-27.3.2009 (*Bonn: Gesellschaft für Informatik.

Grebner, O., Ong, E. & Riss, U. (2008). KASIMIR - Work process embedded task management leveraging the Semantic Desktop. In M. Bichler, T. Hess, H. Krcmar, U. L. F. Matthes, A. Picot, B. Speitkamp & P. Wolf (Eds.), *Multikonferenz Wirtschaftsinformatik (p.* 715-726). GITO-Verlag, Berlin.

Grebner, O., Ong, E., Riss, U. V., Brunzel, M., Bernardi, A. & Roth-Berghofer, T. (2007). *Task Management Model - NEPOMUK Deliverable D3.1*. http://nepomuk.semanticdesktop.org/xwiki/bin/download/Main1/D3-1/D3.1_v10_NEPOMUK_Task_Management_Model.pdf. Accessed 2009.03.03.

Grebner, O., Ong, E., Riss, U., Gudjonsdottir, R. & Edlund, H. (2006). *SAP Scenario Report - NEPOMUK Deliverable D10.1*. http://nepomuk.semanticdesktop.org/xwiki/bin/download/IST/WebHome/D10.1v11NEPOMUKSAPScenarioReport.pdf. Accessed 2009.03.03.

Grebner, O., Riss, U. V., Ong, E., Brunzel, M., Roth-Berghofer, T. & Bernardi, A. (2007). Task Management for the NEPOMUK Social Semantic Desktop (Poster). In N. Gronau (Eds.), *4th Conference on Professional Knowledge Management - Experiences and Visions -, March 28. - 30. 2007, Potsdam, Germany (*GITO Verlag Berlin.

Grebner, O., Riss, U., Edlund, H. & Groth, K. (2007). *First Prototype of the SAP NEPOMUK Desktop - NEPOMUK Deliverable D10.2.* http://nepomuk.semanticdesktop.org/xwiki/bin/download/IST/WebHome/D10.2_v11_NEPOMUK_1st_Prototype_of_SAP_Nepomuk_Desktop.pdf. Accessed 2009.03.03.

Groza, T., Handschuh, S., Moeller, K., Grimnes, G., Sauermann, L., Minack, E., Mesnage, C., Jazayeri, M., Reif, G. & Gudjonsdottir, R. (2007). The Nepomuk project-on the way to the social semantic desktop. *Proceedings of I-Semantics (*Vol.7, S. 201-211).

Gwizdka, J. (2002). TaskView: design and evaluation of a task-based email interface. *Proceedings of the 2002 conference of the Centre for Advanced Studies on Collaborative research (*IBM Press.

Heuser, L., Alsdorf, C. & Woods, D. (2007). *International Research Forum 2007* Evolved Technologist Press.

Jones, W. & Teevan, J. (2007). *Personal Information Management* University of Washington Press.

Jones, W. (2008). *Keeping Found Things Found* Elsevier Inc.

Kaliski, B. S. (1999). *The Gale Encyclopedia of Business and Finance* Gale Group, Farmington Hills.

Kaptelinin, V. & Czerwinski, M. (2007). Introduction: The Desktop Metaphor and New Uses of Technology. In V. Kaptelinin & M. Czerwinski (Eds.), *Beyond the desktop metaphor: designing integrated digital work environments (p.* 1-12). MIT Press, Cambridge, Mass.

Kaptelinin, V. (2003). UMEA: translating interaction histories into project contexts. *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems (p.* 353-360). New York, NY, USA: ACM.

Kaptelinin, V. I. & Boardman, R. (2007). Toward integrated work environments: Application-centric versus workspace-level design. In V. Kaptelinin & M. Czerwinski (Eds.), *Beyond the desktop metaphor: designing integrated digital work environments (p.* 295-331). MIT Press, Cambridge, Mass.

Kaptelinin, V. I. & Czerwinski, M. (2007). *Beyond the desktop metaphor: designing integrated digital work environments* MIT Press, Cambridge, Mass.

Karger, D. R. (2007). Unify Everything: It's All the Same To Me. In W. P. Jones & J. Teevan (Eds.), *Personal Information Management (p.* 127-152). University of Washington Press.

Katifori, V., Poggi, A., Scannapieco, M., Catarci, T. & Ioannidis, Y. (2005). OntoPIM: how to rely on a personal ontology for Personal Information Management. In S. Decker, J. Park, D. Quan & L. Sauermann (Eds.), *Proc. of Semantic Desktop Workshop at the ISWC, Galway, Ireland November 6 (*Vol.175,.

Kaye, J. & Sengers, P. (2007). The Evolution of Evaluation. *Proceedings of ACM CHI 2007.*

Kelly, D. (2006). Evaluating personal information management behaviors and tools. *Communications of the ACM , 49*, 84-86.

Kidd, A. (1994). The Marks are on the Knowledge Worker. In U. M. Borghoff & R. Pareschi (Eds.), *Proc. ACM CHI94: Human Factors in Computing Systems (p.* 186-191). Boston, Mass: ACM Press.

Lee, D., Hull, J. J., Erol, B. & Jamey, G. (2004). MinuteAid: multimedia note-taking in an intelligent meeting room. *ICME '04. 2004 IEEE International Conference on Multimedia and Expo , 3*, 1759-1762 Vol.3.

Leont'ev, A. N. (1978). *Activity, consciousness, and personality* Prentice-Hall Englewood Cliffs, NJ.

Leont'ev, A. N. (1981). Sign and Activity. In J. V. Wertsch (Eds.), S. 241-241). Routledge.

Li, D., Wang, Z. & Muntz, R. R. (1999). "Got COCA?" A new perspective in building electronic meeting systems. *WACC '99: Proceedings of the international joint conference on Work activities coordination and collaboration (p.* 89-98). New York, NY, USA: ACM.

Maier, R. & Sametinger, J. (2008). A Top-Level Ontology for Smart Document Access. In C. Stary, F. Barachini & S. Hawamdeh (Eds.), *Knowledge Management: Innovation, Technology and Cultures: Proceedings of the 2007 International Conference on Knowledge Management (*Vienna, Austria: World Scientific.

Merriam-Webster Inc. (2009). *Dictionary - Unification.* http://www.merriam-webster.com/dictionary/unification. Accessed 2009.03.02.

Microsoft Cooporation (2009). *Research Desktop - Microsoft Research.* http://research.microsoft.com/en-us/projects/researchdesktop/default.aspx. Accessed 2009.03.03.

Microsoft Corporation (2009). *Microsoft Exchange Server.* http://www.microsoft.com/exchange/. Accessed 2009.03.03.

Microsoft Corporation (2009). *Microsoft OneNote.* http://office.microsoft.com/onenote. Accessed 2009.03.03.

Microsoft Corporation (2009). *Microsoft Outlook.* http://www.microsoft.com/outlook/. Accessed 2009.03.03.

Microsoft Corporation (2009). *Microsoft Powerpoint.* http://office.microsoft.com/powerpoint. Accessed 2009.03.03.

Microsoft Corporation (2009). *Microsoft Project.* http://office.microsoft.com/project. Accessed 2009.03.03.

Microsoft Corporation (2009). *Microsoft Sharepoint.* http://www.microsoft.com/Sharepoint/default.mspx. Accessed 2009.03.04.

Microsoft Corporation (2009). *Microsoft Visual Studio Tools.*

http://www.microsoft.com/downloads/details.aspx?FamilyID=F5539A90-DC41-4792-8EF8-F4DE62FF1E81. Accessed 2009.03.03.

Microsoft Corporation (2009). *Microsoft Word.* http://office.microsoft.com/word/. Accessed 2009.03.04.

Miles, M. B. & Huberman, M. (1994). *Qualitative Data Analysis: An Expanded Sourcebook (2nd Edition)* Sage Publications, Inc.

Mindjet Corporation (2009). *Mindjet Mind Manager.* http://www.mindjet.com. Accessed 2009.03.04.

Mindquarry GmbH (2009). *Mindquarry.* http://www.mindquarry.com/products/scenarios/managing-tasks. Accessed 2009.03.04.

Moore, C. R. (1999). Performance Measures for Knowledge Management. In J. Liebowitz (Eds.), *Knowledge Management Handbook (*Boca Raton, FL: CRC Press.

Moran, T. P. & Zhai, S. (2007). Beyond the Desktop Metaphor in Seven Dimensions. In V. Kaptelinin & M. Czerwinski (Eds.), *Beyond the desktop metaphor: designing integrated digital work environments (p.* 335-354). MIT Press, Cambridge, Mass.

Morse, E. L. (2002). Evaluation Methodologies for Information Management Systems. *D-Lib Magazine , 8*(9).

Morse, J. M. (1989). *Qualitative nursing research: a contemporary dialogue* Aspen Publishers.

Mozilla Foundation (2009). *Thunderbird.* http://www.mozillamessaging.com/thunderbird/. Accessed 2009.03.03.

Mozilla Foundation (2009). *XML User Interface Language (XUL).* http://www.mozilla.org/projects/xul/. Accessed 2009.03.03.

Murphy, E. & Rodriguez-Manzanares, M. A. (2008). Using activity theory and its principle of contradictions to guide research in educational technology. *Australasian Journal of Educational Technology , 24(4)*, 442-457.

Mwanza, D. (2001). Where theory meets practice: A case for an Activity Theory based methodology to guide computer system design. *Proceedings of INTERACT (p.* 9-13).

Nardi, B. A., Whittaker, S. & Schwarz, H. (2002). NetWORKers and their Activity in IntensionalNetworks. Vol.11, S. 205-242). Norwell, MA, USA: Kluwer Academic Publishers.

Nardi, B. A., Whittaker, S., Isaacs, E., Creech, M., Johnson, J. & Hainsworth, J. (2002). Integrating communication and information through ContactMap. *Communications of the ACM , 45*(4), 89-95.

Nepomuk Consortium (2009). *NEPOMUK - The Social Semantic Desktop - Homepage.* http://nepomuk.semanticdesktop.org/xwiki/bin/view/Main1/. Accessed 2009.03.03.

Nepomuk Consortium (2009). *NEPOMUK - The Social Semantic Desktop.* http://nepomuk.semanticdesktop.org/. Accessed 2009.03.03.

Nepomuk Consortium (2009). *NEPOMUK Annotation Ontology (NAO).* http://www.semanticdesktop.org/ontologies/nao/. Accessed 2009.03.03.

Nepomuk Consortium (2009). *Nepomuk Developer Tickets.* http://dev.nepomuk.semanticdesktop.org/report/1. Accessed 2009.03.03.

Nepomuk Consortium (2009). *NEPOMUK Information Element (NIE) Ontology.* http://www.semanticdesktop.org/ontologies/nie/. Accessed 2009.03.03.

Nepomuk Consortium (2009). *Nepomuk Local Storage / RdfRepository.* http://dev.nepomuk.semanticdesktop.org/wiki/RdfRepository. Accessed 2009.03.03.

Nepomuk Consortium (2009). *Nepomuk Messaging.* http://dev.nepomuk.semanticdesktop.org/wiki/doc/ServiceDescription/MessagingDocumentation. Accessed 2009.03.03.

Nepomuk Consortium (2009). *NEPOMUK Ontologies.* http://www.semanticdesktop.org/ontologies/. Accessed 2009.03.03.

Nepomuk Consortium (2009). *Nepomuk Service Registry.* http://dev.nepomuk.semanticdesktop.org/wiki/doc/ServiceDescription/ServiceRegistryDocumentation. Accessed 2009.03.03.

Nepomuk Consortium (2009). *P2P Semantic Eclipse Workbench (PSEW).* http://nepomuk-eclipse.semanticdesktop.org/xwiki/bin/view/Main/PSEW. Accessed 2009.03.03.

Nepomuk Consortium (2009). *RDFSBeans.* http://dev.nepomuk.semanticdesktop.org/repos/trunk/java/org.semanticdesktop.nepomuk.comp.rdfsbeans/. Accessed 2009.03.04.

Nielsen, J., Clemmensen, T. & Yssing, C. (2002). Getting access to what goes on in people's heads?: reflections on the think-aloud technique. *Proceedings of the second Nordic conference on Human-computer interaction (p.* 101-110). ACM New York, NY, USA.

Noy, N. F., Crub´ezy, M., Fergerson, R. W., Knublauch, H., Tu, S. W., Vendetti, J. & Musen, M. A. (2003). Protégé-2000: an open-source ontology-development and knowledge-acquisition environment. *AMIA. Annual Symposium proceedings. AMIA Symposium. (p.* 953). AMIA Annu Symp Proc.

On-To-Knowledge Project (2009). *On-To-Knowledge: Project Deliverables.* http://www.ontoknowledge.org/del.shtml. Accessed 2009.03.04.

Ong, E., Grebner, O. & Riss, U. V. (2007). Pattern-Based Task Management: Pattern Lifecycle and Knowledge Management - Benefits and open issues. In N. Gronau (Eds.), *4th Conference on Professional Knowledge Management - Experiences and Visions -, March 28. - 30. 2007, Potsdam, Germany (*GITO Verlag Berlin.

Ong, E., Riss, U., Grebner, O. I. & Du, Y. (2008). Semantic Task Management Framework. In K. Tochtermann & H. Maurer (Eds.), *I-KNOW '08 Proceedings of the 8th International Conference on Knowledge Management - Special Track KS'08 on Knowledge Services*.

Ong, E., Riss, U., Grebner, O., Du, Y. & Brunzel, M. (2007). *First Task Management Prototype - NEPOMUK Deliverable D3.2.* http://nepomuk.semanticdesktop.org/xwiki/bin/download/IST/WebHome/D3.2_v10_First_Task_Management_Prototype.pdf. Accessed 2009.03.04.

Ozier, S. (2009). *Task Freak!*. http://www.taskfreak.com/. Accessed 2009.03.04.

O'Reilly, T. (2005). What is web 2.0. *Design patterns and business models for the next generation of software , 30,* 2005.

Panko, R. R. & Kinney, S. T. (1995). Meeting profiles: size, duration, and location. *HICSS '95: Proceedings of the 28th Hawaii International Conference on System Sciences (p.* 1002). Washington, DC, USA: IEEE Computer Society.

Personal Task Manager (2009). *PTM, Personal Task Manager.* http://ptm.sourceforge.net/. Accessed 2009.03.03.

Plaisant, C., Shneiderman, B., with Baker, H. R., Duarte, N. B., Haririnia, A., Klinesmith, D. E., Lee, H., Velikovich, L. A., Wanga, A. O. & Westhoff, M. J. (2007). Personal Role Management: Overview and a Design Study of Email for University Students. In V. Kaptelinin & M. Czerwinski (Eds.), *Beyond the desktop metaphor: designing integrated digital work environments (p.* 143-170). MIT Press, Cambridge, Mass.

Polanyi, M. (1967). *The Tacit Dimension* Peter Smith Publisher Inc.

Porter, M. E. (1998). *Competitive advantage: Creating and sustaining superior performance* Free Press.

Prud'hommeaux, E. & Seaborne, A. (2009). *SPARQL Query Language for RDF*. http://www.w3.org/TR/rdf-sparql-query/. Accessed 2009.03.04.

Quek, A. & Shah, H. (2004). A Comparative Survey of Activity-based Methods for Information Systems Development. *Proceedings of 6th International Conference on Enterprise Information Systems (*Vol.5, S. 221-229). ICEIS.

Rajkumar, S. (2009). *Activity Theory.* http://www.slis.indiana.edu/faculty/yrogers/act_theory/. Accessed 2009.03.04.

Ravasio, P. & Tscherter, V. (2007). Users' Theories of the Desktop Metaphor, or Why We Should Seek Metaphor-Free Interfaces. In V. Kaptelinin & M. Czerwinski (Eds.), *Beyond the desktop metaphor: designing integrated digital work environments (p.* 265-294). MIT Press, Cambridge, Mass.

Ravasio, P., Schär, S. G. & Krueger, H. (2004). In pursuit of desktop evolution: User problems and practices with modern desktop systems. *ACM Trans. Comput.-Hum. Interact. , 11*(2), 156-180.

Redish, J. G., Bias G., R., Bailey, R., Rolf., M. & Dumas Joe Spool, J. M. (2002). Usability in practice: formative usability evaluations-evolution and revolution. *Conference on Human Factors in Computing Systems (p.* 885-890). ACM New York, NY, USA.

Riss, U. V. & Grebner, O. (2008). Including temporal aspects of work in the handling and reuse of tasks and task patterns. In D. Harorimana & D. Watkins (Eds.), *Proceedings of the 9th European Conference on Knowledge Management (ECKM), Southampton Solent University, Southampton, UK, 4-5 September 2008.*

Riss, U. V., Weber, I. & Grebner, O. (2008). Business Process Modeling, Task Management, and the Semantic Link. In K. Hinkelmann (Eds.), *AAAI Spring Symposium on AI Meets Business Rules and Process Management, Stanford Univ. (p.* 99-104). American Association for Artificial Intelligence, Menlo Park, Calif.

Riss, U., Rickayzen, A., Maus, H. & van der Aalst, W. M. P. (2005). Challenges for Business Process and Task Management. *Journal of Universal Knowledge Management - Special Issue on Knowledge Infrastructures for the Support of Knowledge Intensive Business Processes , 0*(2), 77-100.

Riss, U., Schmidt, B., Herlambang, C., Jurisch, M., Du, Y., Grebner, O., Edlund, H., Groth, K. & Lindquist, S. (2009). *2nd Prototype of SAP Research NEPOMUK Desktop - NEPOMUK Deliverable D10.3.* http://nepomuk.semanticdesktop.org/xwiki/bin/download/IST/WebHome/D10.3%2D2ndPrototypeSAPResearchNEPOMUKDesktop%2Dv1.0.pdf. Accessed 2009.03.04.

Robertson, G., Smith, G., Meyers, B., Baudisch, P., Czerwinski, M., Horvitz, E., Robbins, D. & Tan, D. (2007). Explorations in Task Management on the Desktop. In V. Kaptelinin & M. Czerwinski (Eds.), *Beyond the desktop metaphor: designing integrated digital work environments (p.* 101-138). MIT Press, Cambridge, Mass.

Robertson, G., van Dantzich, M., Robbins, D., Czerwinski, M., Hinckley, K., Risden, K., Thiel, D. & Gorokhovsky, V. (2000). The Task Gallery: a 3D window manager. *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems (p.* 494-501). New York, NY, USA: ACM.

Rowley, D. E. (1994). Usability testing in the field: bringing the laboratory to the user. *Proceedings of the SIGCHI conference on Human factors in computing*

systems: Celebrating interdependence (p. 252-257). ACM New York, NY, USA.

Rubin, J. (1994). *Handbook of usability testing: How to plan, design, and conduct effective tests* Wiley.

Rura, S. (2009). *Voo2do collaboration.* http://voo2do.com/collab_index. Accessed 2009.03.04.

Rura, S. (2009). *Voo2do.* http://voo2do.com/. Accessed 2009.03.04.

SAP AG (2009). *SAP BusinessObjects Risk Management.* http://www.sap.com/solutions/sapbusinessobjects/large/governance-risk-compliance/riskmanagement/index.epx. Accessed 2009.03.04.

Sauermann, L. & Klinkigt, M. (2009). *PIMO Service.* http://dev.nepomuk.semanticdesktop.org/wiki/PimoService. Accessed 2009.03.03.

Sauermann, L. (2009). *PIMO Editor.* http://pimoeditor.opendfki.de/. Accessed 2009.03.03.

Sauermann, L., Bernardi, A. & Dengel, A. (2005). Overview and Outlook on the Semantic Desktop. *Proceedings of the 1st Workshop on The Semantic Desktop at the ISWC (*Vol.175, S. 1-19). CEUR-WS.

Sauermann, L., Elst, L. V. & Möller, K. (2009). *Personal Information Model (PIMO) Ontology Guide v1.1.* http://dev.nepomuk.semanticdesktop.org/repos/trunk/ontologies/pimo/latex/pimo.pdf. Accessed 2009.03.03.

Sauermann, L., Loannou, K., Klinkigt, M. & Trela, T. (2009). *Local Data Alignment.* http://dev.nepomuk.semanticdesktop.org/wiki/LocalDataAlignment. Accessed 2009.03.03.

Sauermann, L., Van Elst, L. & Dengel, A. (2007). Pimo-a framework for representing personal information models. *Proceedings of I-Semantics (p.* 270-277).

Schreiber, G., Akkermans, H., Anjewierden, A., Hoog, d., Shadbolt, N. R. & Velde, V. d. (1999). *Knowledge engineering and management: the CommonKADS methodology* MIT press.

Schwarz, S., Abecker, A., Maus, H. & Sintek, M. (2001). Anforderungen an die Workflow-Unterstützung für wissensintensive Geschäftsprozesse. *Workshop "Geschäftsprozessorientiertes Wissensmanagement", 1. Konferenz Professionelles Wissensmanagement (WM2001), 14. - 16. März, Baden-Baden* .

semanticweb.org (2009). *RDF Reactor.* http://semanticweb.org/wiki/RDFReactor. Accessed 2009.03.03.

semanticweb.org (2009). *RDF2Go.* http://semanticweb.org/wiki/RDF2Go. Accessed 2009.03.03.

Shneiderman, B. & Plaisant, C. (1994). The future of graphic user interfaces: personal role managers. *HCI '94: Proceedings of the conference on People and computers IX (p.* 3-8). New York, NY, USA: Cambridge University Press.

Skype Limited (2009). *Skype.* http://www.skype.com. Accessed 2009.03.04.

Spongecell L.L.C. (2009). *Spongecell Calender.* https://spongecell.com/. Accessed 2009.03.04.

Sproull, L. & Kiesler, S. (1991). *Connections: New ways of working in the networked organization* MIT press.

SRI International (2008). *CALO project - CALO: Cognitive Assistant that Learns and Organizes.* http://caloproject.sri.com/. Accessed 2009.03.04.

Streitz, N. A., Geißler, J. & Hol, J. M. H. a. J. (1994). DOLPHIN: integrated meeting support across local and remote desktop environments and LiveBoards. *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported*

cooperative work (p. 345-358). New York, NY, USA: ACM.

Sure, Y. & Studer, R. (2002). *On-To-Knowledge: Content-driven knowledge management tools through evolving ontologies - Final Version. EU-IST-1999-10132 Project Deliverable D18.* http://www.ontoknowledge.org/downl/del18.pdf.

Taylor, P., Riss, U., Grebner, O., Du, Y. & Brunzel, M. (2008). *Final NEPOMUK Task Management Prototype - NEPOMUK Deliverable D3.3.* http://nepomuk.semanticdesktop.org/xwiki/bin/download/IST/WebHome/D33%2DFinalPrototypeTaskManagement%2Dv10.pdf. Accessed 31.12.2008.

Teambits GmbH (2009). *Digital Moderation.* http://www.digital-moderation.com/. Accessed 2009.03.03.

Trio Systems L.L.C (2009). *SoniClear Enterprise Recorder.* http://soniclear.com. Accessed 2009.03.04.

UnaMesa Association (2009). *Tiddly Wiki.* http://www.tiddlywiki.com. Accessed 2009.03.04.

Voida, S. & Mynatt, E. D. (2009). It feels better than filing: everyday work experiences in an activity-based computing system. *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems (p.* 259-268). New York, NY, USA: ACM.

Voida, S., Mynatt, E. D. & MacIntyre, B. (2007). Supporting Activity in Desktop and Ubiquitous Computing. In V. Kaptelinin & M. Czerwinski (Eds.), *Beyond the desktop metaphor: designing integrated digital work environments (p.* 195-222). MIT Press, Cambridge, Mass.

Voida, S., Mynatt, E. D. & Edwards, W. K. (2008). Re-framing the desktop interface around the activities of knowledge work. *UIST '08: Proceedings of the 21st annual ACM symposium on User interface software and technology (p.* 211-220). New York, NY, USA: ACM.

Voida, S., Mynatt, E. D., MacIntyre, B. & Corso, G. M. (2002). Integrating Virtual and Physical Context to Support Knowledge Workers. *IEEE Pervasive Computing , 1*(3), 73-79.

Weber, M., Partsch, G., Scheller-Huoy, A., Schweitzer, J. & Schneider, G. (1997). Flexible Real-time Meeting Support for Workflow Management Systems. *HICSS '97: Proceedings of the 30th Hawaii International Conference on System Sciences (p.* 559-567). Washington, DC, USA: IEEE Computer Society.

Webster's (2009). *Webster's Dictionary - Integrate.* http://machaut.uchicago.edu/?&word=integrate&resource=Webster's. Accessed 2009.03.04.

Webster's (2009). *Webster's Dictionary - Integration.* http://machaut.uchicago.edu/?&word=integration&resource=Webster's. Accessed 2009.03.04.

Wikimedia Foundation Inc. (2009). *MediaWiki.* http://www.mediawiki.org. Accessed 2009.03.04.

Wikimedia Foundation Inc. (2009). *Wikipedia: The Free Encyclopedia. C Sharp (programming language).* http://en.wikipedia.org/wiki/C_Sharp_(programming_language). Accessed 2009.03.03.

Wikimedia Foundation Inc. (2009). *Wikipedia: The Free Encyclopedia. Extensible Messaging and Presence Protocol.* http://en.wikipedia.org/wiki/Extensible_Messaging_and_Presence_Protocol. Accessed 2009.03.03.

Wikimedia Foundation Inc. (2009). *Wikipedia: The Free Encyclopedia. Minutes.* http://en.wikipedia.org/wiki/Minutes. Accessed 2009.03.03.

Wikimedia Foundation Inc. (2009). *Wikipedia: The Free Encyclopedia. RACI diagram.*

http://en.wikipedia.org/wiki/RACI_diagram. Accessed 2009.03.03.

Workflow Management Coalition (WfMC) (1999). *Terminology Glossary - Document Number WFMC-TC-1011*. http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf. Accessed 2009.03.03.

World Wide Web Consortium (W3C) (2009). *Resource Description Framework (RDF).* http://www.w3.org/RDF/. Accessed 2009.03.02.

Yahoo! Inc. (2009). *Del.icio.us.* http://del.icio.us. Accessed 2009.03.03.

Yee, K., Swearingen, K., Li, K. & Hearst, M. (2003). Faceted metadata for image search and browsing. *Proceedings of the SIGCHI conference on Human factors in computing systems (p.* 401-408). ACM New York, NY, USA.

Yiu, K. S., Baecker, R., Silver, N. & Long, B. (1997). A time-based interface for electronic mail and task management. *ADVANCES IN HUMAN FACTORS ERGONOMICS , 21,* 19-22.

Appendix

Appendix A – Evaluation Manuscript – First Kasimir Evaluation

Manuscript for usability evaluations at SAP in May 2007. Authors Henrik Edlund, Kristina Groth and Rósa Guðjónsdóttir, KTH.

Introduction

This usability evaluation is a part of our research in a European project called Nepomuk. We have performed interviews and workshops with you/your colleagues at SAP last year with the purpose of understanding the work done at SAP Research. The purpose of the interviews was to get more information about the work you do here at SAP Research. The results of those interviews were that we saw that there are several problems; it is for example difficult to find time to work between meetings, and meetings are sometimes very early in the morning or late in the afternoon. It is not always easy to track the task list and the task list is very often very long. There is also a lot of collaboration between people in different offices, which could be easier to do than it is today.

What we would like to do with our computer programs is to make your work easier. We have now created four trial computer programs and we would like to ask you what you think of them – they are by no means the final computer programs, just something to test our ideas. It is important that we develop the right computer programs and the only way to know which is right is to ask you what you think of them. It is very important that you tell us what you think. And we are very grateful for your help!

We would like to videotape this session, but this is mostly for our analysis. We will perhaps use footage of this video to complement the written report we will deliver in the project, is this OK?

We will not disclose your name or your views with anyone but people involved in the project. All your information will be depersonalised to protect your privacy. You have the right to abort your participation at any time. [Give them our business cards] If you have any questions afterwards don't hesitate to contact us.

Pre-test interview

1. Name: _____
2. Age: _____
3. Gender: (Male) (Female)
4. Education: _____
5. How long have you worked for SAP? _____
6. What is your title? _____
7. What do you work with? _____

WP10 Prototype

8. We would like your input on a task manager that we are working on. It is all very preliminary as you can see. We have paper sketches that we would like your input on.
9. At first we have your start up window for each day. What do you see on this page? What is your impression of this view?
10. Here you can see a web page where you are reading about ACM. You need to become a member of ACM, can you create a task for this?
11. Here you have a task that you have created previously. What is you impression of this task? Is this the level of detail you would like to have?
12. Here you have a view of your calendar. How would you try to get an overview of your tasks when you are looking at this calendar view? How would you like that overview to look like?
13. Here you see the last window of the day. What do you see on this page? What is your impression of this view?

Post-test interview

14. What is your impression of this prototype?
15. Is there something good about it?
16. Is there something bad? What would you like to change?
17. What do you think this prototype is good for?
18. Do you think this prototype could be helpful in your work? Why? Why not?

Thank you for your help!

Appendix B – Evaluation Manuscript – Second Kasimir Evaluation

Manuscript for usability evaluations at SAP in October 2007. Authors Kristina Groth and Henrik Edlund, KTH.

Introduction

This usability evaluation of the Kasimir prototype is a part of our research in a European project Nepomuk. In May and last year we conducted interviews, workshops and paper prototype evaluations with people at SAP. [unless participated in the interviews] The purpose of the interviews was to get more information about the work you do here at SAP Research. The results of the interviews identified several problems; it is for example difficult to find time to work between meetings, and meetings are sometimes very early in the morning or late in the afternoon. It is not always easy to track the task list and the task list is very often very long. There is also a lot of collaboration between people in different offices, which could be easier to do than it is today. [unless participated in the May evaluation] The purpose of the prototype evaluation we performed here in May was to get feedback on an early version of this prototype that we will evaluate today.

What we would like to do with the Kasimir prototype is to make your work easier. This is a first prototype that has been developed here at SAP and focuses on task management. It is a running prototype that consists of three parts: a Task window, a Firefox extension and an Outlook extension. It is important that the prototype is developed for you and the only way to know which is right is to ask you and your colleagues. It is important that you tell us what you think, and we are grateful for your help!

We would like to videotape this session, but this is mostly for our analysis. We will perhaps use footage of this video to complement the written report we will deliver in the project, is this OK?

We will not disclose your name or your views with anyone but people involved in the project. All your information will be depersonalised to protect your privacy. You have the right to abort your participation at any time. If you have any questions afterwards don't hesitate to contact us.

Pre-test interview

1. Name: _____
2. Age: _____
3. Gender: (Male) (Female)
4. Education: _____
5. How long have you worked for SAP? _____
6. What is your title? _____
7. What do you work with? _____

WP10 Prototype

You and a colleague of yours that is an expert in usability testing will write a scientific paper together. There is an interesting conference in February 2008, and the paper submissions are due in the end of October (31.10.2007). Let us take a closer look at the Kasimir application.

8. This is what the Task window looks like. What do you see in this window? Can you describe the different parts and fields that you see? What do you think you can do here?
9. If you click on the task "write paper for CHI2008", what can you see? What state is the task in? Are any other tasks more prioritised? What tasks need to be finished today?
10. Now I want you to add a new task about reviewing usability literature for the paper you are going to write. You want this to be finished by October 15th. How do you do that?
11. You will do the task that you have just added together with your colleague Uwe. You should add Uwe Riss as a collaborator of this task. How do you do that?
12. You will now add some keywords (test the concepts annotations and tags) to the task. The task you have added is about finding relevant usability literature. How do you do that? What annotations/tags would you like to add? What do you find useful with the keywords? [Tags can be used to better re-find your task out of the large number of tasks.] In addition to annotations and tags what would you like to add to a task?
13. Let's assume that you remember now that you have already a task that can be similar to the task you have just added. Try to find a task that is tagged [annotated with a keyword] with "usability".
14. Do you have comments or suggestions about the Task window that you would like to add?
15. Now we will turn to the Firefox extension [show the column view at the website http://www.chi2008.org/authors.html]. This is what it looks like. What do you see in this window? What do you think you can do here? In what situations would this be helpful?
16. I now want you to go to http://en.wikipedia.org/wiki/Usability. This is very interesting for the "literature review on usability" task you previously added. You see some interesting literature references. I want you to add this webpage as an attachment to the task. How do you do that?
17. Do you have comments or suggestions about this Firefox extension that you would like to add?
18. What do you think about a similar extension in Outlook? Can similar extensions be useful in any other applications?

Post-test interview

19. What is your impression of these prototypes?
20. Is there something good about them?
21. Is there something bad? What would you like to change?
22. What do you these prototypes are good for?

23. Do you think any of these prototypes could be helpful in your work? Why? Why not?

24. Do you think semantics can be valuable in any other sense in the application other than what we have discussed? How do you think semantics could add to the possibility to suggest other people as potential collaborators (question 11)? How do you think semantics could add to the possibility to suggest keywords (annotations and tasks) (question 12)? How do you think semantics could add to the possibility to suggest to you that this webpage is relevant to your meeting task (question 15)? Examples!

Thank you for your help!

Appendix C – Evaluation Questionnaire – Third Kasimir Evaluation

Dear participants,

Once again, thank you for participating in the user testing of Kasimir & Nepomuk during the last two and a half weeks. Your feedback will help me greatly for my thesis and I am very grateful for your effort.

Regarding the survey, I would appreciate if you keep in mind the following queries. I need the survey in its entirety, so please answer all of the questions in the survey in chronological order. It would be great if your answers are formed in complete sentences. For the scaled questions, please select only one answer.

Thanks again for your time and consideration.

Kind regards,

Marlen Jurisch

Kasimir Task Management Questionnaire

Last Name: First Name:

In case you leave SAP Research by the end of 2008, please fill in your private Email address:

Private Email Address:

Age range *(please select one):*

- ☐ 18 - 24
- ☐ 25 - 34
- ☐ 35 - 45
- ☐ over 45

Sex *(please select one):*

- ☐ Male
- ☐ Female

Job title at SAP Research:

How often did you use Kasimir *(please select one):*

- ☐ Continuously throughout the work day
- ☐ Once or twice per day
- ☐ 3-4 times per week
- ☐ Once or twice per week
- ☐ Not at all

Task Management:

Question 1:

What kind of tools do you normally use when performing task management? *(Please list all tools, such as Outlook, Pencil & Paper, etc):*

-
-
-
-
-

Question 2:

Do you consider computer-based task management necessary in order to efficiently organize your work activities? *(please explain in detail)*:

☐ Yes ☐ No

Please explain in detail why, or why not, you consider task management a necessity:

Question 3: What are your expected benefits by performing task management? *(please explain in detail)*:

Kasimir Features:

Question 4: In comparison to your former task management methods, Kasimir helps you manage your work activities more efficiently. *(Please select one answer):*				
☐	☐	☐	☐	☐
Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree

Question 5: What benefits (or limitations) does Kasimir provide compared to your old task management methods? *(please explain in detail)*:

Question 6: Did the feature of adding "Documents" to a task help you in organizing your work more efficiently? *(please select one answer)*:	
☐ Yes	☐ No
Please explain in detail what you found beneficial, or hindering, about this feature:	

Question 7: Did the feature of linking “Persons” to a task help you in organizing your work more efficiently? *(please select one answer)*:
☐ Yes ☐ No
Please explain in detail what you found beneficial, or hindering, about this feature:

Question 8: Did the feature of adding “Subtasks” to a task help you in organizing your work more efficiently? *(please select one answer)*:
☐ Yes ☐ No
Please explain in detail what you found beneficial, or hindering, about this feature:

Question 9: Did the feature of adding “Tags” to a task help you in organizing your work more efficiently? *(please select one answer)*:
☐ Yes ☐ No
Please explain in detail what you found beneficial, or hindering, about this feature:

Question 10: Did the feature of adding "Bookmarks" to a task help you in organizing your work more efficiently? *(please select one answer)*:
☐ Yes ☐ No
Please explain in detail what you found beneficial, or hindering, about this feature:

Task-Patterns:

Question 11: Did you create a Task-Pattern during the testing? *(please select one answer)*:
☐ Yes ☐ No *If Not, please explain why:*
Please explain for what purpose this task-pattern was created:

Question 12: How did the Task-Pattern application help you organize your knowledge about tasks? *(please explain in detail)*:

The underlying Semantics:

Question 13:
The Nepomuk Social Semantic Desktop proved to be a helpful tool for managing your personal knowledge. *(Please select one answer)*

☐ Strongly Agree	☐ Agree	☐ Neutral	☐ Disagree	☐ Strongly Disagree

Question 14:
How do you rate the complexity of the following tools? *(please select one answer for each tool)*:

Nepomuk Social Semantic Desktop:				
☐ Highly Complex	☐ Complex	☐ Normal	☐ Simple	☐ Highly Simple
Kasimir Task Management:				
☐ Highly Complex	☐ Complex	☐ Normal	☐ Simple	☐ Highly Simple

Question 15:
Nepomuk semantic technologies allow you to define properties of resources on your desktop and relations between these resources. How evident are the underlying semantic technologies when using the following tools? *(please select one answer for each tool)*:

Nepomuk Social Semantic Desktop:				
☐ Very Evident	☐ Evident	☐ Neutral	☐ Hidden	☐ Very Hidden
Kasimir Task Management:				
☐ Very Evident	☐ Evident	☐ Neutral	☐ Hidden	☐ Very Hidden

Daily use of Kasimir:

Question 16: Was it feasible to use Kasimir on a daily basis?
☐ Yes ☐ No ☐ Sometimes
Please explain why, or why not, it was feasible to use Kasimir regularly:

Question 17: Did your perception of Kasimir's usefulness change over the testing period?
☐ Yes ☐ No
If Yes, please explain how your perception has changed:

Collaboration with the community:

Question 18: What would be the benefits and the limitations to using the "delegate a task" feature in Kasimir for the collaboration with your team members? *(please explain in detail)*:

Question 19:
Kasimir should be used as an interactive tool for collaboration and coordination within teams. Please explain which desired features should be available in order to achieve this goal? *(please explain in detail)*:

Continues use of Kasimir *(This question should help identify potential long-term testers)*:

Question 20:
Would you be interested in using Kasimir after the testing? *(Please select one answer):*
☐ Yes ☐ Not Sure ☐ No

Question 21:
Why, or why not, would you continue to use Kasimir after the testing? *(please explain in detail)*:

Further comments or remarks:
Do you have any additional comments concerning Kasimir or the Nepomuk Social Semantic Desktop?

Index