KP-LAB Knowledge Practices Laboratory – Specification of end-user applications

Olli Alm, Patrick Ausderau, Merja Bauters, Markus Holi, Antti Hämäläinen, Hannu Markkanen, Eini Saarivesi, Benoit Rigolleau, Michal Racek, Ali Rantakari, et al.

27490

KP-LAB

Knowledge Practices Laboratory

Integrated Project

Information Society Technologies

DII.8 M46 specification of end-user applications

Due date of deliverable: 30/11/2009
Actual submission date:

Start date of project: 1.2.2006 Duration: 60 Months

Organisation name of lead contractor for this deliverable: Metropolia

Revision [0.5]

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Contributor(s):	**Olli Alm**	**Metropolia**	**olli.alm@metropolia.fi**
	Patrick Ausderau	**Metropolia**	**patricka@metropolia.fi**
	Merja Bauters	**Metropolia**	**merja.bauters@metropolia.fi**
	Markus Holi	**Metropolia**	**markus.holi@metropolia.fi**
	Antti-Ville Hämäläinen	**Metropolia**	**antti-ville.hamalainen@metropolia.f**
	Hannu Markkanen	**Metropolia**	**hannu.markkanen@metropolia.fi**
	Eini Saarivesi	**Metropolia**	**eini.saarivesi@metropolia.fi**
	Benoit Rigolleau	**AKKA**	**b.rigolleau@akka.eu**
	Michal Račeк	**PÖYRY**	**michal.racek@poyry.com**
	Ali Rantakari	**PÖYRY**	**ali.rantakari@poyry.com**
	Vassiliy Tchoumatchenko	**TUS**	**vpt@tu-sofia.bg**
	Tania Vasileva	**TUS**	**tkv@tu-sofia.bg**
	Angela Locoro	**DIBE**	**angela.locoro@unige.it**
	Anna Marina Scapolla	**DIBE**	**scapolla@unige.it**
	František Babič	**TUK**	**frantisek.babic@tuke.sk**
	Peter Bednar	**TUK**	**peter.bednar@tuke.sk**
	Jan Paralic	**TUK**	**jan.paralic@tuke.sk**
	Jozef Wagner	**TUK**	**jozef.wagner@gmail.com**
	Ekaterina Simonenko	**UPS**	**ekaterina.simonenko@lri.fr**
	Tsuyoshi Sugibuchi	**UPS**	**buchi@lri.fr**

Editor:	Hannu Markkanen, Olli Alm
Partner(s):	2. Metropolia, 4. AKKA, 5. POYRY, 7. TUS, 12. DIBE, 16. TUK, 20. UPS
Work Package:	WPII – Knowledge Practices Environment
Nature of the deliverable:	Report

Version history (of the main document)

Version	Date	Author(s)	Description
0.1	24.11.2009	Markkanen	First draft with chapter 1-4
0.2	04.12.2009	Alm	KPE architecture
0.3.	20.12.2009	Markkanen	Overview of tool specifications
0.4	06.01.2010	Markkanen	Proof-reading and final edits.
0.5	08.01.2010	Markkanen	Compilation of annexes.
1.0	12.01.2010	Markkanen	Final edits.

Executive summary

The present deliverable provides a high-level view on the new specifications of end user applications defined in the WPII during the M37-M46 period of the KP-Lab project. This is the last in the series of four deliverables that cover all the tools developed in the project, the previous ones being D6.1, D6.4 and D6.6. This deliverable presents specifications for the new functionalities for supporting the dedicated research studies defined in the latest revision of the KP-Lab research strategy. The tools addressed are: the analytic tools (Data export, Time-line-based analyser, Visual analyser), Clipboard, Search, Versioning of uploadable content items, Visual Model Editor (VME) and Visual Modeling Language Editor (VMLE).

The main part of the deliverable provides the summary of tool specifications and the description of the Knowledge Practices Environment architecture, as well as an overview of the revised technical design process, of the tools' relationship with the research studies, and of the driving objectives and the high-level requirements relevant for the present specifications. The full specifications of tools are provided in the annexes 1-9.

Table of Contents

Table of Tables

Table of Figures

List of Abbreviations

API	Application Programming Interface
ASDT	Activity System Design Tools
CASS	Contextual Activity Sampling System
DO	Driving Objective
DoW	Description of Work
EMF	Eclipse Modeling Framework
FMS	Flash Media Server
GUI	Graphical User Interface
HLR	High-Level Requirement
HPA	History and Participation Awareness
HTTP	Hypertext Transfer Protocol
IMS	IMS Global Learning Consortium (IMS = Instructional Management Systems)
KAS	Knowledge Analysis Service
KPE	Knowledge Practices Environment
M2T	Meeting Management Tool
SCORM	Shareable Content Object Reference Model
SMAT	Semantic Multimedia Annotation Tools
ReST	Representational State Transfer
RTMP	Real Time Messaging Protocol
RTMPT	Real Time Messaging Protocol Tunneling (via http)
SKOS	Simple Knowledge Organisation Systems
SWKM	Semantic Web Knowledge Middleware
SSp	Shared Space
TLBA	Time-line Based Analyser
TLO	Trialogical Learning Ontology
UC	Use Case
UCI	Uploadable Content Item
UM	User Management
US	Usage Scenario
UT	User Task
VME	Visual Model Editor
VMLE	Visual Model Language Editor
WP	Work Package
XML	Extensible Mark-up Language

1 Introduction

This deliverable presents the new specifications of end user applications defined in the WPII during the M37-M46 period of the KP-Lab project. It covers new tools as well as the significant enhancements and extensions to the earlier tool releases. The specifications do not, however, include tools that will have only minor improvements e.g. in usability. The deliverable follows the same way of description as the previous tool specifications deliverable D6.6 [6].

The specifications of single tools are provided in the annexes 1-9. These and the possible later revisions of specifications are also available in the DII.8 folder of the KP-Lab intranet (http://www.kp-lab.org/intranet/work-packages/wp6/result/dii-8-m46-specification-of-end-user-applications) in the folder "Specifications of single tools", see Figure 1.

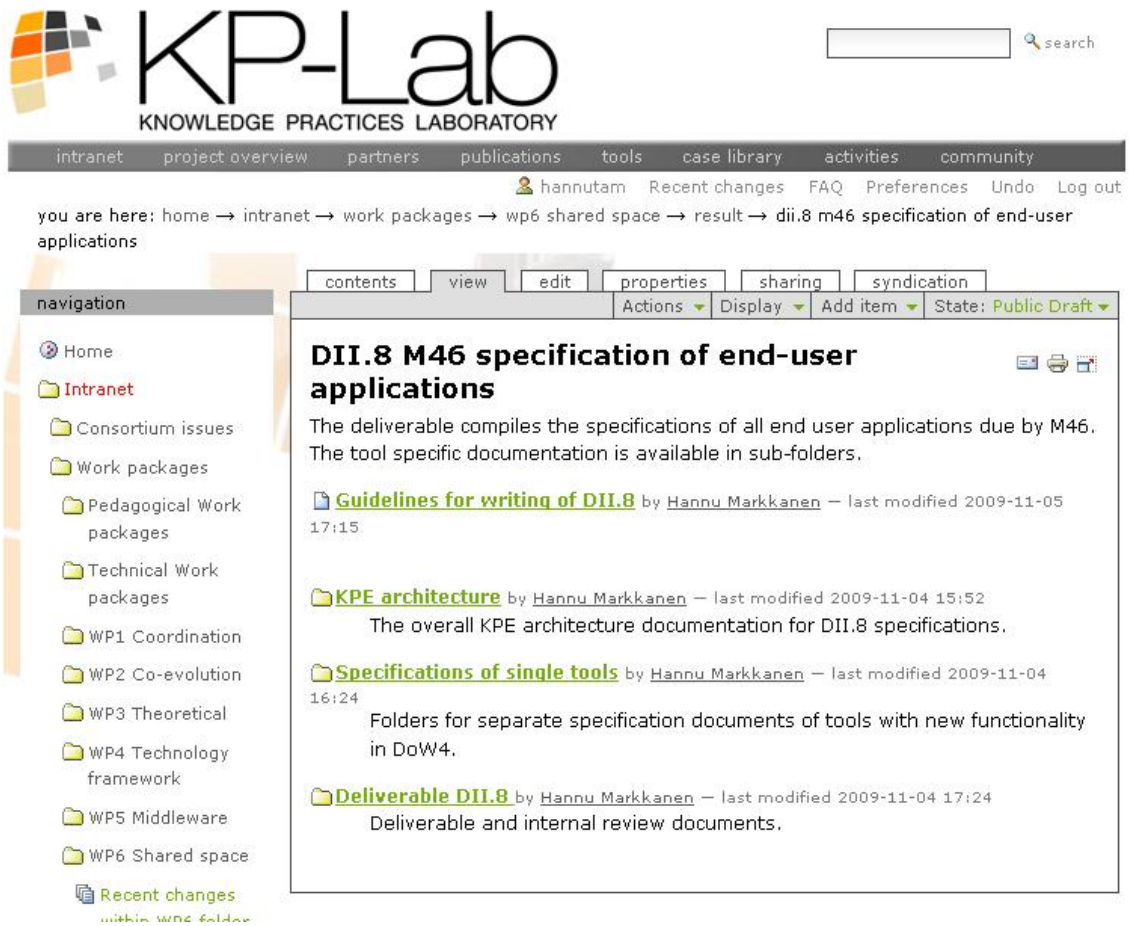


Figure 1: Part of the project intranet's section for DII.8 deliverable.

2 Overview of the KPE and the present specifications

This chapter provides an overview of the tools with specifications in the present deliverable. The chapter is organised according to the categorisation of tools in the WPII

description of DoW4.1. Further details on the tool specifications can be found in chapter 5 and in the annexes 1-9.

KPE provides a set of integrated tools and functionalities for sustained collaborative working with shared epistemic objects (artefacts, processes, practices). KPE provides virtual working spaces, called shared spaces, for the collaborative work, enables viewing the knowledge objects and their relations from different perspectives and supports object-bound development of all items in a shared space. Various tools and functionalities are highly integrated in the basic views to enable versatile and flexible connection, organization and reflection of all information related to the knowledge objects, processes and people concerned. The end user view of the KPE is depicted in the Figure 2.

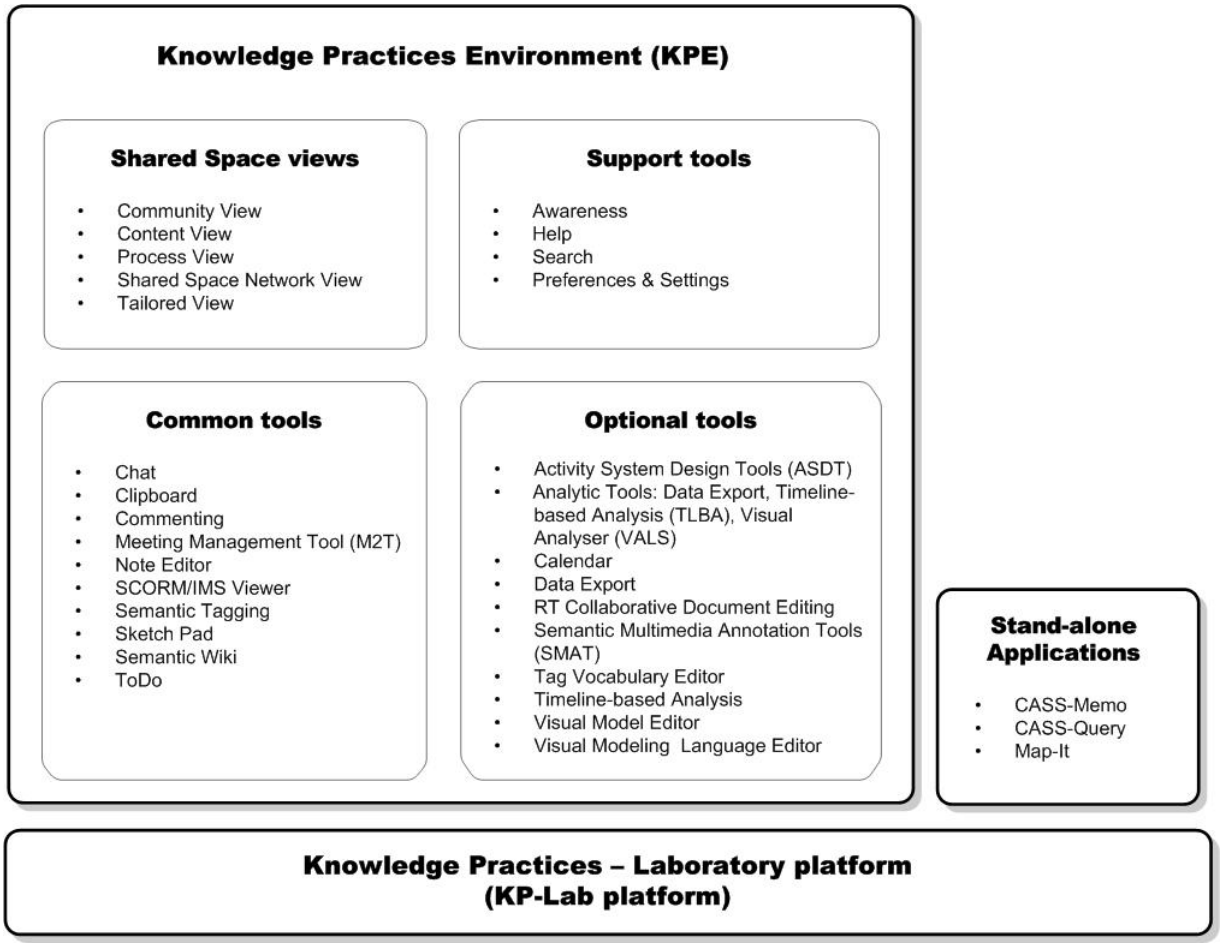


Figure 2: End user view of the KPE components.

KPE is composed of the following views and sets of tools:

- *Shared Space views* allow for viewing and accessing the information contained in a shared space in flexible manner (network view, content view, process view, community view and tailored view).
- *Common tools* refer to the tightly integrated tools of KPE, available inside a shared space, for working with knowledge artefacts. Examples of common tools are chat, commenting, and semantic tagging.
- *Optional tools* are loosely integrated applications that can be selected by the user to be available in a shared space, such as ASDT, calendar, and data export.

- *Support tools* provide generic supplementary functionality used in/by the Shared Space views and the other tools, such as awareness, help, and search.

In addition, there are three *stand-alone applications* that are used separately from the KPE, namely Map-It, and CASS Query and CASS Memo. The development of these tools has been stopped as requested after the 3rd project review.

The earlier specification deliverables produced in the WP6 (i.e. d6.1, D6.4 and D6.6) cover most of the KPE functionality. The present deliverable provides specifications for the remaining new functionalities, which focus on supporting dedicated research studies defined in the latest KP-Lab research strategy. These functionalities are as follows:

- Alternative process view
- Analytic tools: Data export, Time-line-based analyser, Visual analyser
- Clipboard for re-using of shared spaces and artefacts
- Search/saving searches
- Versioning of the uploadable content items
- Visual Model Editor (VME) and Visual Modeling Language Editor (VMLE)

Alternative process view introduces a new way for visualising knowledge creation processes in the KPE. The process view of KPE provides the user interface for managing and visualizing of the knowledge processes within shared spaces. Process planning through defining tasks and drafting visual process representations allows users to explicate their process composition and promotes responsibility and ownership over their decisions and actions. The current process view of KPE is realised as a Gantt chart. In the alternative visualization users can, however, visualize the process structure and workflow by using images and pre-defined shapes and arrows. The rationale for development of the alternative process view is that the knowledge practices and processes need to be very flexible, enabling innovative styles of working and learning, and they are dynamic in nature. Spatial representation and emphasis on relationships between tasks as well as tasks and contents is especially useful in educational settings, where the chronology of the work is not essential, but there is a requirement to see connections, associations and causal relations between the various elements of the process.

Analytic tools provide tools for two main categories of analytic facilities: 1) *Data export tool* for automatic data collection for the analysis in third party tools and 2) *Timeline-based analyser* and *Visual analyser* for integrated reflection on knowledge creation processes and their analysis. Research in related areas has produced several existing approaches to analysis of processes but to our knowledge there is no particular approach focused on knowledge creation processes, i.e. trying to analyze the employed knowledge practices, which we see as major advancement with respect to the state-of-the-art processes analysis tools.

Data export tool allows researchers and teachers to extract summary tables of user activities from the KPE for on-line investigation, and to export them for elaborations with statistical analysis packages. It gathers data about activities taking place in the KPE and extracts data from three repositories of the KP-Lab system, namely from the awareness repository, the user database and the knowledge repository. Data export tool has been part of the KPE since M24 release with subsequent extensions in later releases (see [2]). The new features specified in the present deliverable include adding graphical views to the

"Social Networks Analysis" function and exporting of data compatible with Reference model.

Timeline-based analyser (TLBA) allows users to display chronologically events that were recorded by the KPE tools, to design and store possible external events which could not have been recorded by the KPE tools and to define 'patterns' of actions that can be identified in the historical data. In contrast to Visual Analyser and Data Export tools, TLBA brings chronological overview of user actions into the user interface, which enables users to see and explore what kind of activities were performed on certain objects in the shared space of interest. Time-line view in its initial form has been already released in M36 as ASDT Analysis Application (see [2]). The new functionalities in the present deliverable are the possibility to add external events, the possibility of commenting on user actions, defining of action 'patterns', searching the history of stored actions for defined patterns and export functionalities.

Visual analyser is a completely new tool that allows users to analyze participation and activities within past or ongoing knowledge creation processes, by visually representing them based on information stored in the produced logs. More precisely, it visualizes frequencies of object-related activities in KPE and provides detailed information on the nature of the activities performed on particular knowledge objects. These visualizations stimulate teachers and students to reflect on object progression, and on their teaching and learning practices.

Clipboard tool allows users to re-use shared spaces and artefacts within KPE by providing the copy-paste functionality of shared spaces, tasks, vocabularies and most types of content items. The rationale is to enable users to systematically build on previous work as well as to disseminate and share effective knowledge practices. The clipboard will be available in all shared space views of KPE, The clipboard tool offers to the research cases the bare bones minimum functionality of the re-use library tool envisaged in the D6.6 [6]. The cut down of the re-use functionality was done to reduce the technical development work as requested after the third project review.

Search tool will be extended with the functionality to save search results and criteria, which was planned in iteration 2 of the tool (see [6], p.31). Due to the cut-down of the resources after the third project review, this is the only new functionality of the search tool necessary for the research cases.

Versioning tool allows users to create new versions and replace latest version of the uploadable content items, which are files that can be uploaded into the KP-Lab content repository (such as office documents, pictures, SCORM/IMS packages). The user is also able to see detailed information on specific versions. Versioning will support teachers and students to review and analyze their collaborative editing activities by enabling the tracing of changes in their knowledge artefacts. The possibility to keep track of changes and revert to previous versions of content items supports the exploratory creation and use of artefacts.

Visual Model Editor (VME) and *Visual Modelling Language Editor (VMLE)* tools draw on the recent ideas for a pragmatic semantic web and consider modeling as an inherently epistemic activity that goes beyond the mere representation of what is already known and what can be agreed upon. The vision behind these tools is to provide users with the possibility to create their own conceptual tools and thereby to advance pre-existing

perspectives. The two tools provide users with a flexible and easy to use but still semantically powerful tool for the creation of visual models and their underlying modeling languages. The VME tool will be extended with the functionality for retrieving and analysing information on activities performed on a visual model. The first prototype of the VMLE (released in M36) tool will be redesigned in order to achieve the functional completeness and acceptable performance required in the research cases. This includes designing of a new data model as well as revision of the architecture and implementation of the front end service. From the functional point of view, a new history analysis and export capabilities are needed.

3 Technical design process

This chapter gives an overview of how the tool design is organised as part of the KP-Lab co-design framework and of the structure of the tool specification.

3.1 Tool design within the KP-Lab co-design framework

The software development process in WPII has been organised during the current project period as described in the D2.3 [7]. From the tool development perspective, there are no significant changes compared with the process described in the previous specifications deliverable D6.6 [6].

Figure 3 provides an overall view of the KP-Lab R&D activities from the co-design perspective. The software specifications presented in this deliverable are the result of work within the User Requirements Specification and the Technical development activities. The software specification documents are high-level design documents that are composed of the use cases and the overall technical design in terms of the architecture, user interface and front-end services of the tool.

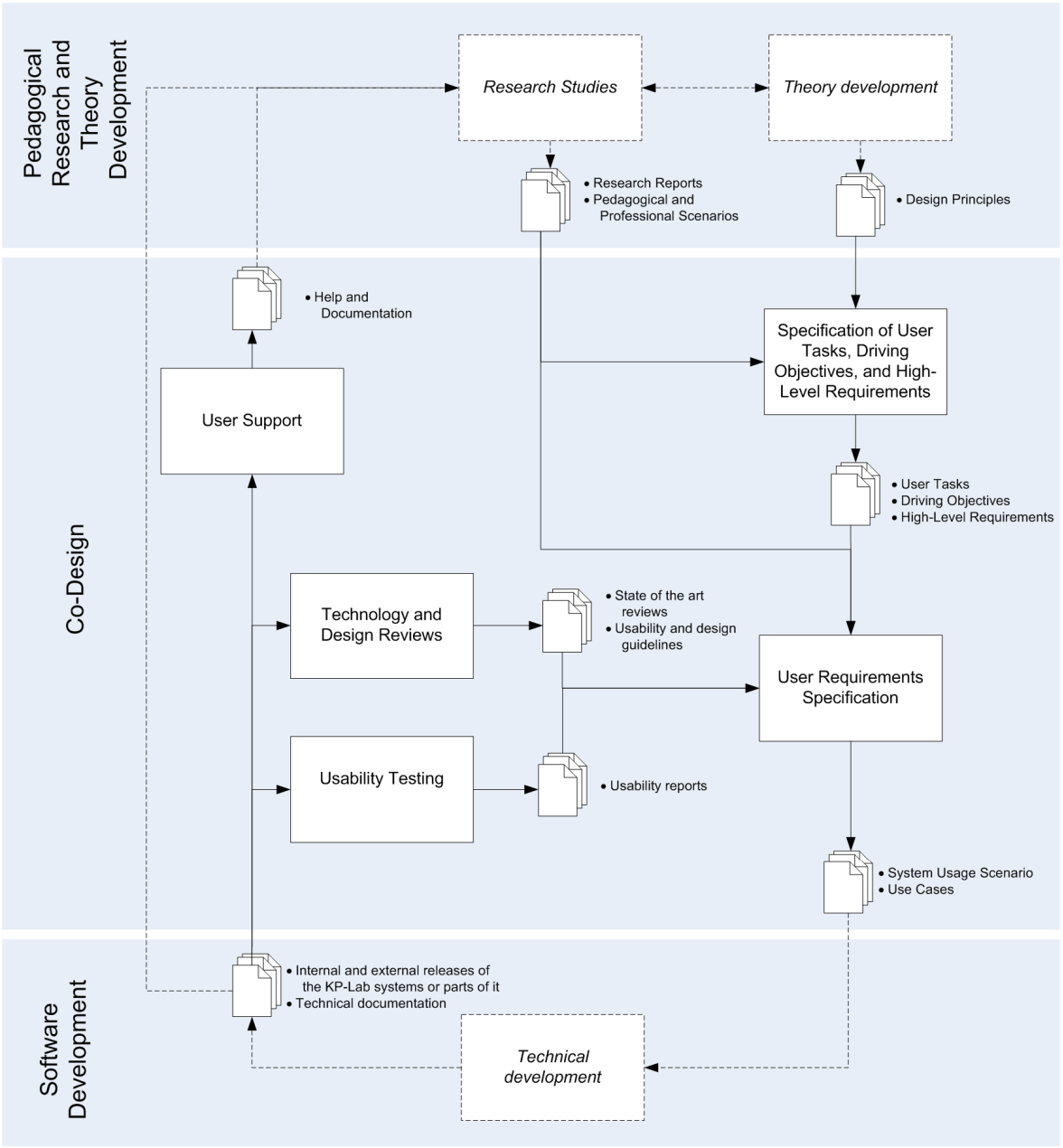


Figure 3: Overview of the KP-Lab co-design process model [7].

The system usage scenarios and use cases define the user requirements for the software. A **system usage scenario** illustrates how the user will interact with the system/tools in order to accomplish certain tasks. The system usage scenario describes typical sequences of interaction as envisioned by the designers and gives an idea of the added value for the user. The scenarios are written in plain English and as such are an informal way of writing use cases. The aim of this intermediate artefact is to facilitate the collaboration between pedagogical and technical partners when analyzing requirements. The **use cases** are derived from the system usage scenarios and describe the interactions between a user and the tool as a sequence of operations and events. Each system usage scenario results in writing one or several use cases. Chapter 5 provides the summary of usage scenarios and use cases for the specifications in present deliverable. Further details can be found in the tool specification documents available in the annexes 1-9.

Based on the software requirements specifications the WPII partners produce the various tools and integrate them, in collaboration with WPIII partners, into the KP-Lab system according to the tool development roadmap described in the chapter 5 of present deliverable. Each tool has its own iterative development roadmap and release schedule, and there is one common production release in M48 for all tools. Before the tools are used as part of the research studies or other pilots, the prototypes of new tools/ functionality are checked for bugs and usability problems by the pedagogical partners using the KP-Lab system available in the development environment[1].

WPIII provides the technical and the semantic knowledge services required by the tools. KP-Lab system is built using continuous integration server in order to reduce integration problems and allowing developers to provide (upgrades and bug fixes of) tools more rapidly.

3.2 Specification documentation

The specifications in this deliverable have been written in a common template that was published in the project intranet. The template is structured into four major chapters as follows:

1. *Introduction* describes the purpose of the document, the scope of the software, the tool roadmap in terms of milestones and functionality of the releases, the summary of new functionality covered by the present specifications and the definitions needed to understand the specifications.
2. *User requirements* outline first tool's relationship with KP-Lab research studies, Driving Objectives and High-Level Requirements. An overview of the system usage scenarios for the tool and a reference to the related requirements document is given next. These leads way to the description of the use cases which are the other major content of the specifications document.
3. *Technical design* describes tool's software architecture in the context of KPE application architecture described in the chapter 6 of the present deliverable, the design of tool's GUI and the front-end services (including their use of the KP-Lab platform services), and the collaboration with other tools.
4. *References* conclude the specification document.

4 Tools' relationship with the empirical research cases, driving objectives and the high-level requirements

This chapter describes the rationale for tool development by providing a summary of the tools' relationship with the empirical research cases and the conceptual tools of co-design, i.e. the driving objectives (DOs) and high-level requirements (HLRs). For detailed description of DOs and HLRs, please refer to [4].

Table 1 summarizes the relationship between the tool specifications available in the present deliverable and the DOs, HLRs and the empirical research cases. Short description of the tool's main purpose is provided as well.

[1] Login at http://mielikki.mobile.evtek.fi/shared-space/

Table 1: KP-Lab tools included in DII.8 specifications and their relationship to the driving objectives, high-level requirements and empirical research cases.

Tool	Description of the new functionality	Related research cases	DOs and HLRs
Alternative process view	Visualisation of the process structure and workflow by using images and pre-defined shapes and arrows.	Case 2	DO8: HLR8.8
Clipboard	Copy-pasting of shared spaces and content items.	Case 1, 2, 3, 4	DO3: HLR3.1
Data export	Graphical views for the "Social Networks Analysis" function, export of data compatible with Reference model.	Case 1,2,3,4	DO9: HLR9.2 DO13: HLR13.5, HLR13.6, HLR13.7
Search	Saving criteria and results of semantic searches.	Case 1,2,4	DO1: HLR1.4 DO4: HLR4.1 DO8: HLR8.6
Time-line based analyser (TLBA)	Chronological overview of user actions on KPE objects in a shared space.	Case 1, 2, 3, 4	DO8: HLR8.7 DO9: HLR9.1, HLR9.2 DO13; HLR13.5, HLR13.6
Versioning	Versioning of content items uploaded in the KPE.	Case 1, 2, 3, 4	DO1: HLR1.1, HLR1.2 DO9: HLR9.1, HLR9.2
Visual analyser	Visualisation of the frequencies of object-related activities in KPE.	Case 1,2,4	DO9: HLR9.2 DO13: HLR13.5. HLR13.6
Visual Model Editor (VME)	Creation of visual models based on the languages created with the VMLE tool.	Case 3	DO9: HLR9.1, HLR9.2 DO13: HLR13.5, HLR13.6
Visual Modelling Language Editor (VMLE)	Creation of modelling languages for use in VME tool.	Case 3	DO2: HLR2.2, HLR2.5 DO4: HLR4.3, HLR4.4 DO6: HLR6.1, HLR6.2, HLR6.3 DO9: HLR9.1, HLR9.2 DO13: HLR13.5, HLR13.6

By related research cases we refer to those empirical research cases that investigate (among other things) the use of tools that target the high-level requirements relevant to the tool. There are four research cases entitled in D3.2 [3] as follows:

Case 1: Trialogical work in higher education; creation and use of knowledge objects in knowledge creation practices.

Case 2: Knowledge creation practices in customer projects in higher education.

Case 3: Conceptual modelling in Design Projects.

Case 4: Producing document management solutions and practices for distributed work settings – Perspectives of KPE usage in two work organizations.

As a response to the request in the third project review, Table 2 provides a cross-reference between the research cases and the tools included in the present specifications.

Table 2: Empirical research cases' need for new tool functionality.

Empirical research case	Requires new functionality in tool
Case 1: Trialogical work in higher education; creation and use of knowledge objects in knowledge creation practices.	Clipboard, Data export, Search, TLBA, Versioning, Visual analyser
Case 2: Knowledge creation practices in customer projects in higher education.	Alternative process view, Clipboard, Data export, Search, TLBA, Versioning, Visual analyser
Case 3: Conceptual modelling in Design Projects	Clipboard, Data export, TLBA, Versioning, VME, VMLE
Case 4: Producing document management solutions and practices for distributed work settings	Clipboard, Data export, Search, TLBA, Versioning, Visual analyser

The overview of the DOs and HLRs referred to in the Table 1 is given in the Table 3. The description of the DOs and HLRs is provided in the D2.4 [4] and the more detailed account of the relationships between the tools and DOs and HLRs can be obtained from the tool specifications available in the annexes 1-9 of the present deliverable and from project's internal working documentation that the annexes refer to.

Table 3: Overview of the driving objectives and high-level requirements that are relevant for the tool functionalities specified in DII.8.

DO1	Users are provided with a collaborative environment where they can work on shared artefacts
HLR1.1	Users can create structure and share various artefacts (e.g. sketches, various kinds of texts, video and audio-files, models as well as ontologies) in one place.
HLR1.2	Users are able to view the artefacts and their relations from different perspectives.
HLR1.4	Users can search artefacts within and outside the shared environment using full text, metadata or domain ontologies.
DO2	Users are provided with a customizable tool suite for working on shared artefacts
HLR2.2	The tools are generic enough to allow users to use them in various ways of working, e.g., personalization of the tools and adaptation to different domain knowledge.
HLR2.5	Users can import and export artefacts and access data across the tools integrated or connected to the KP Environment.
DO3	Users are provided with support for the re-use of shared artefacts and structures
HLR3.1	Users can re-use process structures and artefacts.
DO4	Users can describe the semantics of artefacts and their relations
HLR4.1	Users can categorise, classify and cluster artefacts in different manners.
HLR4.3	Users can work with multiple conceptual models (vocabularies or visual modelling languages) in parallel.
HLR4.4	Users are able to save and share conceptual models (e.g. vocabularies and visual models)
DO6	Users can develop and use their own conceptual models
HLR6.1	Users can modify existing or create new visual modelling languages, ontologies and vocabularies.
HLR6.2	Users can refine, make proposals and give objections on modelling languages, ontologies and vocabularies while working on the respective knowledge representations and visual models.
HLR6.3	Users can share and integrate different visual modelling languages, ontologies and vocabularies.
DO8	Users can plan, organize and manage tasks collaboratively
HLR8.6	Users can search the content and metadata using full text or semantic metadata search for planning and reflecting on activities or both.
HLR8.7	Users are provided with a customized analysis of groups' working processes (e.g. identification of typical sequences of actions or interesting rules).
HLR8.8	Users can choose between different types of process structure visualizations (e.g. project based or progressive inquiry).
DO9	Users are provided with history on content development and work process advancement
HLR9.1	Users can track the evolution and changes of knowledge objects and find out their authors and contributors (sequences of performed steps in time, incl. versioning)
HLR9.2	Users are provided with customized summaries about the knowledge objects available within the shared environment (e.g. users might request an overview of the tasks completed within the last 2 weeks or the interactions of people within a shared workspace)
DO13	Users can collect data about activities and interactions systematically and over longer periods of time
HLR13.5	Users can customise the way they want to retrieve wanted data for analysis.
HLR13.6	Users are provided with summative information on performed actions (e.g. added comments, created tasks, modifications in metadata, background materials for decisions, etc.).

5 Overview of tools specifications

This chapter provides an overview of the specifications included in the present deliverable. For each tool, a short description of its purpose is given, followed by the summaries of the functionality covered by the present specification, of the requirements (as a list of system usage scenarios and use cases), as well as of the development roadmap

The description and rationale of the new functionalities was presented in the chapter 2. Table 4 provides the summary of the milestones in the tool roadmaps for DoW4 period of the project, as well as short description of the major functionality of the releases. Short overview of the specifications is presented in the rest of this chapter and the details in the tool specifications documents available in the annexes 1-9 and in the KP-Lab project intranet (direct web links for each tool are provided below).

Table 4 Main functionality and planned releases of tools with specifications in DII.8.

Tools	Milestones	Release functionality
Alternative process view	v1.0: M45 Specification, M46 Prototype, M48 Production release	Full functionality for creating and modifying alternative process view, and for using alternative process view elements in content and tailored views.
	v1.1: M56 Final release	Improvements and bug fixes of v1.0 based on the user trials.
Clipboard	v1.0 M44 Specifications, M46 Prototype, M48 Production release	Copying and pasting a shared space, copying and pasting elements from a shared space to another.
	v1.1: M56 Final release	Improvements and bug fixes of v1.0 based on the user trials.
Data Export	v3.0: M40 prototype, M42 Production release	Extensions of the export functionalities according to the users' priorities on the requirements already collected in M33 specifications and integration of graphical view for "Social Networks Analysis"
	v3.1 M47 Prototype, M48 Final release	Exporting data in a format compatible with Reference Model. Improvements and bug fixes of v3.0 based on the user trials.
Search	v1.1: M46 Maintenance release.	Improvements and bug fixes of v1.0 based on the user trials.
	v2.0: M46 Specification, M47 Prototype, M48 Production release	Saving of search results.
	v2.1: M56 Final release.	Improvements and bug fixes of v2.0 based on the user trials.
Time-line based analyser (TLBA)	v0.1: M44 Prototype release	Basic functionalities for launching application from KPE, for displaying user actions and for getting specific information about particular action.
	v1.0: M45 technical specification, M47 Prototype, M48 Production release	Stable release with extensions for adding external events, for commenting on user actions, for defining action 'patterns', for searching the history of stored actions for defined patterns and for export functionalities.
Versioning	v1.0 M41 Requirements, M45 Specifications, M47 Prototype, M48 Production release	Creating new versions of the uploadable content items, seeing detailed information of specific versions.
	v1.1: M56 Final release	Improvements and bug fixes of v1.0 based on the user trials.
Visual analyser	v1.0: M40: Specification, M42 Prototype, M44 Release	Basic functionality for request, summary visualization in the form of bar chart or line chart, as well as for the first integration into KPE (as a plug-in).
	v2.0: M44: Specification, M46 Prototype, M48 Release	Improved request formulation, improved user interface with new analysis attributes, interactive summary

Tools	Milestones	Release functionality
		visualization, more flexible filtering of events, possibility to export (save) the resulting summaries and their visualization in a file, final integration into KPE.
	v2.1: M56 Final release	Improvements and bug fixes of v2.0 based on the user trials.
Visual Model Editor (VME)	v2.2: M46 Specification, M47 Prototype, M48 Production release.	Retrieving information on activities performed on a visual model (history), export of visual models.
	v2.3: M56 Final release.	Improvements and bug fixes of v2.2 based on the user trials.
Visual Modelling Language Editor (VMLE)	v2.0: Updated requirements, M39 Prototype, M40 Production release	Basic VMLE functions for browsing, creating, copying, commenting and editing VMLs. Adding, editing, commenting and deleting concepts and relations to a VML
	v3.0: M40 Specification, M42 Prototype, M44 Production release	Advanced VMLE functions for comparing visual modelling languages, updating a visual model according to changes in the VML.
	v3.2: : M46 Specifications, M48 Production release	Optimized data model and front-end services. New functionality for history analysis and export capabilities.
	v3.3: M56 Final release.	Improvements and bug fixes of v3.2 based on user trials.

5.1 Alternative process view

Process View and Alternative Process View provide the user interfaces for managing and visualizing knowledge processes within KPE shared spaces. The Process View visualizes the process by using a Gantt chart, in a time-dependent manner. Alternative Process View provides a support for other models and forms of planning processes: it supports visualization and coupling of different activities. Users can visualize the process structure and workflow by utilizing predefined bitmap images and custom visual shapes to associate the activities to different Process Entities.

The Alternative Process View has an UI to support following tasks in the KPE system:

- Gives user the possibility to draw a non-temporal visualization of the process
- Provides support for enhancing the process visualization with an existing bitmap picture
- Associates and organizes existing tasks to a specific process phase.

The present deliverable describes specifications for the following functionality of the tool:

- Creating and modifying an Alternative Process View
- Using Alternative Process View elements in the Content View
- Focusing on a certain process element in a Tailored View
- Displaying the process elements of the Alternative Process View in the GANTT Chart

The requirements for the above functionalities are described by five system usage scenarios and eight use cases:
System usage scenarios

- US-APV-1-2008-11: Creating an Alternative Process View
- US-APV-2-2008-11: Modification of an Alternative Process View
- US-APV-3-2008-11: Using Alternative Process View elements in the Content View

- US-APV-4-2008-11: Focusing on a certain process element (Tailored View)
- US-APV-5-2008-11: Displaying the Process Elements in the GANTT Chart

Use cases

- UC-APV-01-2009-11: Uploading the background image for the process
- UC-APV-02-2009-11: Creating the process workflow
- UC-APV-03-2009-11: Associating tasks to process entities
- UC-APV-04-2009-11: Modifying process visualization
- UC-APV-05-2009-11: Removing process objects
- UC-APV-06-2009-11: Showing the detailed information of the process object
- UC-APV-07-2009-11: Showing the process entities in the Process View
- UC-APV-08-2009-11: Showing the process entities in the Content Item View

The current roadmap of the Alternative Process View has two iterations as follows:

- Iteration 1: First version of the full functionality. Release v1.0
  Milestones: M45 Specification, M46 Prototype, M48
- Iteration 2: Final version including improvements and bug fixes based on the user trials. Release v1.1.
  Milestones: M56 Production release

The specifications document for the Alternative Process View is available in annex 1, and in the project intranet at: http://www.kp-lab.org/intranet/work-packages/wp6/result/dii-8-m46-specification-of-end-user-applications/specifications-of-single-tools/process-view

5.2 Clipboard

Clipboard tool provides new functionality for re-using KPE objects by allowing users to create new objects starting from existing ones by copy-pasting. The following objects are supported:

- Shared Spaces
- Tasks
- Vocabularies
- Content Items
  - Uploadable File, Web link, Note, Sketch, SCORM/IMS file, Wiki pages.

The clipboard will be available in all shared space views of KPE, The clipboard tool offers to the research cases the bare bones minimum functionality of the re-use library tool envisaged in the D6.6 [6].

The requirements for the clipboard functionalities are described by two system usage scenarios and two use cases:

System usage scenarios

- US-Clipboard-1-2009-11: Copying a Shared Space and pasting it as a new Shared Space
- US-Clipboard-2-2009-11: Copying an element or elements (e.g. Content Items, vocabularies) and pasting them in another Shared Space.

Use cases

- UC-Clipboard-1-2009-11: Copying and pasting a Shared Space

- UC-Clipboard-2-2009-11: Copying and pasting elements from a Shared Space to another.

The current roadmap of the Clipboard tool has two iterations as follows:

- Iteration 1: First version of the full functionality. Release 1.0.
  Milestones: M44 Specifications, M46 Prototype Release, M48 Production release
- Iteration 2: Final version with improvements and bug fixes based on the user trials. Release 1.1.
  Milestones: M56 Production release.

The specifications document for the Clipboard tool is available in annex 2 and in the project intranet at: http://www.kp-lab.org/intranet/work-packages/wp6/result/dii-8-m46-specification-of-end-user-applications/specifications-of-single-tools/clipboard

5.3 Data Export

Data export tool allows researchers and teachers to extract summary tables of user activities from the KPE for on-line investigation, and to export them for elaborations with statistical analysis packages. It gathers data about activities taking place in the KPE and extracts data from three repositories of the KP-Lab system, namely from the awareness repository, the user database and the knowledge repository. Data export tool has been part of the KPE since M24 release with subsequent extensions in later releases (see [2]).

The present deliverable describes specifications of the Data Export tool which aims at improving the M36 release on the basis of the feedback from tests and evaluation and extending the tool in terms of graphical views to the “Social Networks Analysis” function and exporting of data compatible with Reference model. The requirements for these functionalities are described by six system usage scenarios and three use cases introduced in D6.6 [6] and updated in the present deliverable:

System usage scenarios

- US-DataExport-1-2008-11: Retrieving data of user activity in a shared space
- US-DataExport-2-2008-11: Retrieving data of content items
- US-DataExport-3-2008-11: Retrieving data of tasks
- US-DataExport-4-2008-11: Social networks content item analysis and task analysis
- US-DataExport-5-2008-11: Retrieving "User Action Log" summary for analysis
- US-DataExport-6-2008-11: Retrieving "User Action Log" of detailed analysis of users actions

Use cases

- UC-DataExport-1-2008-11: User opens the data export GUI from within shared space.
- UC-DataExport-2-2008-11: User selects the type of Data Export Analysis.
- UC-DataExport-3-2008-11: User looks at the results and can download them.

The current roadmap of the Data Export Tool has one finished and two planned iterations as follows:

- Iteration 1: Feasibility study and prototyping. Release v1.0 (M24)

- Iteration 2: Facilitating the reading of the data exported by the tool and extending the analysis tables. Improvements of the GUI and extensions of “Social Network Analysis” and “User Action Log" functionalities. Release v2.1.
  - Milestones: M32 specifications, M33 Prototype v2.0, M36 Production release.
- Iteration 3: Extensions of the export functionalities according to the users’ priorities on the requirements already collected in M33 specifications and integration of graphical view for “Social Networks Analysis”. Release v3.0
  - Milestones: M40 Prototype, M42 Production release.
- Iteration 4: Final release with the possibility of exporting data in a format compatible with Reference Model. Release v3.1.
  - Milestones: M47 Prototype, M48 Production release.

The specifications document for the Data Export Tool is available in annex 3 and in the project intranet at: http://www.kp-lab.org/intranet/work-packages/wp6/result/dii-8-m46-specification-of-end-user-applications/specifications-of-single-tools/data-export-tool

5.4 Search

The Search tool enables advanced faceted search based on the metadata and content of artefacts (content items). Search tool will be extended with the functionality to save search results and criteria, which was planned for the iteration 2 of the tool (see [6], p.31). Due to the cut-down of the resources after the third project review, this is the only new functionality of the search tool necessary for the research cases. The requirements for this functionality are described by one system usage scenarios and two use cases:

System usage scenarios

- SUS-Search-4-2009-11: Saving search criteria and results.

Use cases

- UC-Search-04-2009-11: User saves search criteria and results.
- UC-Search-05-2009-11: User opens the saved search object (content item organizer).

The current roadmap of the Search Tool has two finished and two planned iterations as follows:

Iteration 1: Basic free-term and faceted semantic search. Release v1.0

- Milestones: M33 Specification, M34 Prototype, M36 Release.

Iteration 2: Improvements and bug fixes based on the user trials. Release v1.1

- Milestones: M46 Release.

Iteration 3: Saving of search results. Release v2.0

- Milestones: M46 Specification, M47 Prototype, M48 Production release.

Iteration 4: Final version with improvements and bug fixes based on user trials. Release 2.1.

- Milestones: M56 Production release.

The specifications document for the new functionalities in iteration 3 of the Search Tool is available in annex 4 and in the project intranet at:

http://www.kp-lab.org/intranet/work-packages/wp6/result/dii-8-m46-specification-of-end-user-applications/specifications-of-single-tools/search-1

5.5 Time-line based analyser (TLBA)

Timeline-based analyser (TLBA) is a tool that allows users to display chronologically events that were recorded by the KPE tools, to design and store possible external events which could not have been recorded by the KPE tools and to define 'patterns' of actions that can be identified in the historical data.

Time-line view in its initial form has been already released in M36 as ASDT Analysis Application (see [2]), which included the basic functionalities, such as launching application from KPE, displaying user actions, and getting specific information about particular action. It was designed and implemented in such a way that it is able to visualize not only ASDT objects but all KPE objects. The additional functionalities planned for the TLBA in the present deliverable are:

- adding external events to the timeline
- commenting user actions on knowledge objects
- defining of actions 'patterns'
- searching the history of stored actions for defined patterns
- exporting of TLBA view functionalities.
- storing and loading TLBA settings.

The requirements for the clipboard functionalities are described by 10 system usage scenarios and 19 use cases:
System usage scenarios

- US-TLBA-1-2009-09: Viewing all events on the timeline that have been performed on certain objects in a shared space.
- US-TLBA-2-2009-09: Viewing all events on the timeline that a user has performed on objects in a shared space.
- US-TLBA-3-2009-09: Viewing all events on the timeline that have been performed on tasks as well as on objects that are associated with the selected task.
- US-TLBA-4-2009-09: Exploring the evolution and history of a shared space on the timeline.
- US-TLBA-5-2009-09: Exploring and comparing the evolution and history of different shared spaces on the timeline (M48).
- US-TLBA-6-2009-09: Adding external events to the timeline (M48).
- US-TLBA-7-2009-11: Adding comment to the object's action (M48).
- US-TLBA-8-2009-11: Working with patterns (M48).
- US-TLBA-9-2009-11: Exporting TLBA view (M48).
- US-TLBA-10-2009-11: Storing and loading TLBA settings (M48).

Use cases

- UC-TLBA-01-2009-11: User launches TLBA from Shared Space context-menu
- UC-TLBA-02-2009-11: User launches TLBA from within the KPE tools drop down menu
- UC-TLBA-03-2009-11: User opens-up the TLBA from within the object's context-menu

- UC-TLBA-04-2009-11: User opens-up TLBA application from within the task's context-menu
- UC-TLBA-05-2009-11: User highlights all actions performed by one person from community view
- UC-TLBA-06-2009-11: User highlights object's evolution path in TLBA
- UC-TLBA-07-2009-11: User highlights all actions performed by one person in TLBA
- UC-TLBA-08-2009-11: User explores details of an action
- UC-TLBA-09-2009-11: User launches KPE content view
- UC-TLBA-10-2009-11: User reads action's comment (M48)
- UC-TLBA-11-2009-11: User launches TLBA to compare several Shared Spaces (M48)
- UC-TLBA-12-2009-11: User adds external event into the timeline (M48)
- UC-TLBA-13-2009-11: User comments on object action (M48)
- UC-TLBA-14-2009-11: User defines actions pattern (M48)
- UC-TLBA-15-2009-11: User loads actions pattern (M48)
- UC-TLBA-16-2009-11: User searches for the pattern occurrences (M48)
- UC-TLBA-17-2009-11: User exports current TLBA view (M48)
- UC-TLBA-18-2009-11: User saves current TLBA settings (M48)
- UC-TLBA-19-2009-11: User loads TLBA settings (M48)

The planned iterations for the TLBA tool are as follows:

Iteration 1: First prototype version with basic functionalities. Release 0.1.

- Milestones: M44 Prototype release

Iteration 2: Stable release including extensions to add external events, the possibility of commenting on user actions, defining of action 'patterns', searching the history of stored actions for defined patterns and export functionalities. Release 1.0.

- Milestones: M45 Technical specification, M47 Prototype, M48 Production release

Iteration 3: Final version with improvements based on the user trials. Release 1.1

- Milestones: M56 Production release.

The specifications document for the TLBA tool is available in annex 5 and in the project intranet at: http://www.kp-lab.org/intranet/work-packages/wp6/result/dii-8-m46-specification-of-end-user-applications/specifications-of-single-tools/tlba-tool

5.6 Versioning

Versioning tool allows users to create new versions and replace latest version of the uploadable content items (UCI), which are files that can be uploaded into the KP-Lab content repository (such as office documents, pictures, SCORM/IMS packages). The following functionalities will be implemented:

- The user can create versions in uploadable Content Items.
- The user can replace latest version of the uploadable Content Item.
- The user can see certain information of the versions.

- Creating new version of the UCI
- Replacing the latest version of the UCI

- Getting information of the UCI versions (e.g. creator, date and time, comment and filename)
- Downloading of specific version of the UCI

The requirements for the versioning functionalities are described by five system usage scenarios and seven use cases:
System usage scenarios

- US-SSp&CViV-01-2009-06: Creating new version of the uploadable Content Item
- US-SSp&CViV-02-2009-06: Replacing the latest version
- US-SSp&CViV-03-2009-06: Getting information of the versions.
- US-SSp&CViV-04-2009-06: Creating new version in info tab
- US-SSp&CViV-05-2009-06: Replacing version in info tab

Use cases

- UC-VER-01-2009-10: Creating a new version of the UCI
- UC-VER-02-2009-10: Replacing the last version of the UCI
- UC-VER-03-2009-10: Getting information of the last UCI version
- UC-VER-04-2009-10: Getting information of UCI versions
- UC-VER-05-2009-10: Getting all UCI version names
- UC-VER-06-2009-10: Downloading the last version of the UCI
- UC-VER-07-2009-10: Downloading a specific UCI version.

The roadmap of the Versioning tool has two iterations as follows:
Iteration 1: Full functionality of the tool. Release 1.0.

- Milestones: M41 Requirements, M45 Specifications, M47 Prototype, M48 Production release

Iteration 2: Final version with improvements and bug fixes from on the user trials. Release 1.1.

- Milestones: M56 Production release.

The specifications document for the Versioning tool is available in annex 6 and in the project intranet at: http://www.kp-lab.org/intranet/work-packages/wp6/result/dii-8-m46-specification-of-end-user-applications/specifications-of-single-tools/versioning

5.7 Visual analyser

The Visual analyser as a new tool that allow users to visualize frequencies of object-related activities in KPE and provides detailed information on the nature of the activities performed on particular knowledge objects. The main functionalities of the tool are as follows:

- Formulation and refining of queries (for the data to be visualised) by defining parameters and filters.
- Creating, using and managing predefined queries
- Summary visualization in the form of bar chart or line chart with interactive summary visualization (requests sent "on the fly", any time a query parameter is modified)
- Exporting (saving) the resulting summaries and their visualization in a file

The requirements for the Visual analyzer functionalities are described by three system usage scenarios and eight use cases:
System usage scenarios

- US-VA-1-2009-09 : Analysing data based on a user defined query
- US-VA-2-2009-09: Reusing queries
- US-VA-3-2009-09: Ad hoc definition of groups.

System usage scenarios

- UC-VA-1-2009-09: Query definition and result exploration
- UC-VA-2-2009-09: Refining the query adding or removing parameters
- UC-VA-3-2009-09: Adding a parameter to obtain a stacked diagram.
- UC-VA-4-2009-09: Downloading a predefined query.
- UC-VA-5-2009-09: Saving the query as a predefined query.
- UC-VA-6-2009-09: Managing predefined queries.
- UC-VA-7-2009-09: Saving the result of the query.
- UC-VA-8-2009-09: Defining groups of users for a query

The current roadmap of the Visual analyser tool has three iterations as follows:
Iteration 1: First version of the basic functionality for query and visualisation. Release 1.0.

- Milestones: M40: Specification, M42 Prototype, M44 Release

Iteration 2: Full functionality. Release 2.0.

- Milestones: M44: Specification, M46 Prototype, M48 Release.

Iteration 3: Final version with improvements and bug fixes of v2.0 based on the user trials. Release 2.1.

- Milestones: M56 Production release.

The specifications document for the Visual analyser tool is available in annex 7 and in the project intranet at: http://www.kp-lab.org/intranet/work-packages/wp6/result/dii-8-m46-specification-of-end-user-applications/specifications-of-single-tools/visual-analyzer

5.8 Visual Model editor

The Visual Model Editor (VME) is a tool for collaborative creation of visual models based on visual modelling languages, which can be created using the Visual Model Language Editor (see the chapter 5.9). Visual model is a two-dimensional graph-structured visual representation of a certain object of interest such as flow-chart, argument-graph, organigram, decision tree, program logic model, use case diagram, conceptual map. Visual modelling language is a formal specification, which defines the syntax and visual appearance of a model and the semantics of modelling elements used.

The present deliverable describes the specifications of the new functionalities in release 2.2 of VME, which will allow the user to retrieve and analyse information on activities performed on a visual model. The requirements for these functionalities are described by two new system usage scenarios and two new use cases:

System usage scenarios

- US-VM(L)E-5-2009-09: Export of visual models
- US-VM(L)E-6-2009-09: Retrieving information on activities performed on a visual model (history)

Use cases

- UC-VME-9-2009-11: Retrieving information on activities performed on a visual model
- UC-VME-10-2009-11: Export of visual models

The current roadmap of the Visual Model Editor has three finished and two planned iterations as follows:
Iteration 1: Feasibility study, prototyping. Release v1.0
Iteration 2: Basic VME functionality (creating, editing, annotating, browsing visual models). Release v2.0

- Milestones: M21 Specification, M28 Prototype, M32 Stable release

Iteration 3: Advanced VME functions, integration with the Visual Modeling Languages Editor. Release 2.1

- Milestones: M33 Updated specifications, M35 prototype, M36 stable release.

Iteration 4: Maintenance release with the new functionality for retrieving information on activities performed on a visual model and for exporting of visual models. Release 2.2.

- Milestones: M46 Specification, M47 Prototype, M48 Production release.

Iteration 5: Final release with improvements and bug fixes of v2.2 based on user trials. Release 2.3.

- Milestones: M56 Production release.

The specifications document for the iterations 4 of the Visual Model Editor is available in annex 8 and in the project intranet at: http://www.kp-lab.org/intranet/work-packages/wp6/result/dii-8-m46-specification-of-end-user-applications/specifications-of-single-tools/visual-model-editor-vme/

5.9 Visual Modelling Language Editor

The Visual Modelling Language Editor (VMLE) is a tool for collaborative creation of visual modelling languages to be used in conjunction with the Visual Model Editor (VME). Visual modelling language is a formal specification, which defines the syntax and visual appearance of a model and the semantics of modelling elements used.

The present deliverable describes the revised specifications for the VMLE functionality. The principal difference, compared to the previous VMLE prototype, is a new data model, architecture and implementation of the front end service. The redesign is required because the previous prototypes were not able to achieve functional completeness and acceptable performance. From the functional point of view a new history analysis and export capabilities are specified.

The requirements for the above functionality are described by four system usage scenarios and 12 use cases:
System usage scenarios

- US-VM(L)E-2-2009-09: Creating and editing a visual modelling language
- US-VM(L)E-3-2009-09: Migrating to a new version of a Visual Modelling Language
- US-VM(L)E-5-2009-09: Export of visual models (extended to visual modelling languages)

- US-VM(L)E-6-2009-09: Retrieving information on activities performed on a visual model (extended to visual modelling languages)

Use cases

- UC-VMLE-1 Browsing available VMLs
- UC-VMLE-2 Creating a VML
- UC-VMLE-3 Copying a VML
- UC-VMLE-4 Opening VML for editing
- UC-VMLE-5 Adding a concept
- UC-VMLE-6 Adding an attribute
- UC-VMLE-7 Adding a link
- UC-VMLE-8 Deleting concepts, attributes and links
- UC-VMLE-9 Updating concepts and attributes
- UC-VMLE-10 Check VML consistency
- UC-VMLE-11 Retrieving information on activities performed on a VML (history)
- UC-VMLE-12 Export of VML

The current roadmap of the VMLE has five iterations as follows:
Iteration 1: Design, prototyping. Release v1.0

- Milestones: M33 Specification, M36 Prototype

Iteration 2: Basic VMLE functions - browsing available visual modeling languages (VMLs), creating, copying, commenting and editing VMLs. Adding, editing, commenting and deleting concepts and relations to a VML. Release v2.0

- Milestones: M37 Updated software requirements, M39 prototype, M40 stable release.

Iteration 3: Advanced VMLE functions (comparing visual modeling languages, updating a visual model according to changes in the VML), Release 3.0

- Milestones: M40 Specification, M42 Prototype, M44 Production release.

Iteration 4: Optimized data model and front-end services. New functionality for history analysis and export capabilities. Release 3.2.

- Milestones: M46 Specifications, M48 Production release.

Iteration 5: Final release with improvements and bug fixes of v3.2 based on user trials. Release 3.3.

- Milestones: M56 Production release.

The specifications document for the iteration 4 of VMLE is available in annex 9 and in the project intranet at: http://www.kp-lab.org/intranet/work-packages/wp6/result/dii-8-m46-specification-of-end-user-applications/specifications-of-single-tools/visual-modelling-languages-editor-vmle/

6 Knowledge Practices Environment Architecture

This chapter provides the high-level view of the Knowledge Practices Environment (KPE) architecture and the description of how the KPE exploits the middleware services provided by the KP-Lab platform.

6.1 Functional view

The functional view of the KPE architecture is illustrated in the Figure 4.**Error! Reference source not found.** It presents the KP-Lab system and the tools in particular perceived by the users. The description of the KPE and the stand-alone applications is presented in the series of four KP-Lab tool specification deliverables D6.1, D6.4, D6.6 and DII.8.

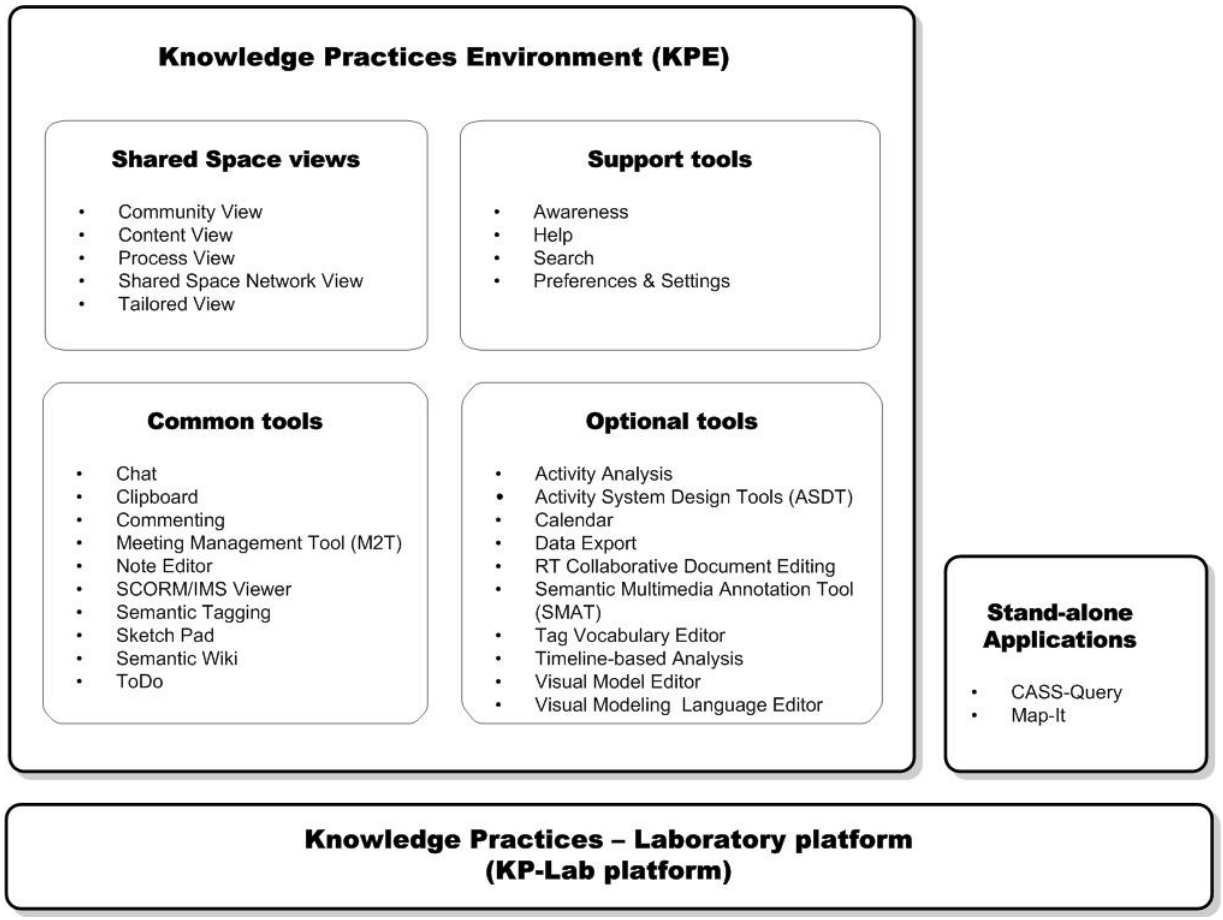


Figure 4: The functional view of the KP-Lab tools' architecture.

The components of the functional view are the following:

- **Knowledge Practices Environment** (KPE) is the application that is used to log in to the KPE, to use shared spaces by means of the various *views* and integrated *common and support tools*, as well as to launch the *optional tools*, such as the Activity System Design Tools. A Shared space (SSp) is a virtual working space for groups and communities.
- **Shared Space views** provide functionalities for viewing and accessing the information contained in a shared space. The views visualize the knowledge artefacts and their relations from different perspectives (e.g. process, community) as well as provide access to the common and optional tools of KPE.
- **Common tools** refer to the tightly integrated tools of KPE, available inside a shared space, for working with knowledge artefacts.
- **Optional tools** are loosely integrated applications that the user can make available in a shared space (using the Preferences & Settings tool). Optional tools are used directly from within the KPE.

- **Support tools** provide supplementary functionality for utilizing the Shared Space views and tools, e.g. contextual help tool and awareness information about the users presence and actions in a shared space.
- **Stand-alone applications** are used separately from the KPE due to their implementation as desktop (Map-It) or due to the focus on supporting specific pedagogical research methodology (CASS Query). The stand-alone applications are able to share their data with the KPE tools. The data of the stand-alone application is stored in the knowledge repository, provided by the Semantic Web Knowledge Middleware (SWKM) of the KP-Lab Platform.
- **KP-Lab Platform** provides common services to all KP-Lab tools, such as data persistence and authorization. The platform is described in D4.2.4 [8] and in D4.2.5 [9].

All the components mentioned above are referred to as the **Knowledge Practices – Laboratory System (KP-Lab system)**.

.

6.2 Implementation view

The major technological parts of the KPE and their connections, is illustrated in the Figure 5.

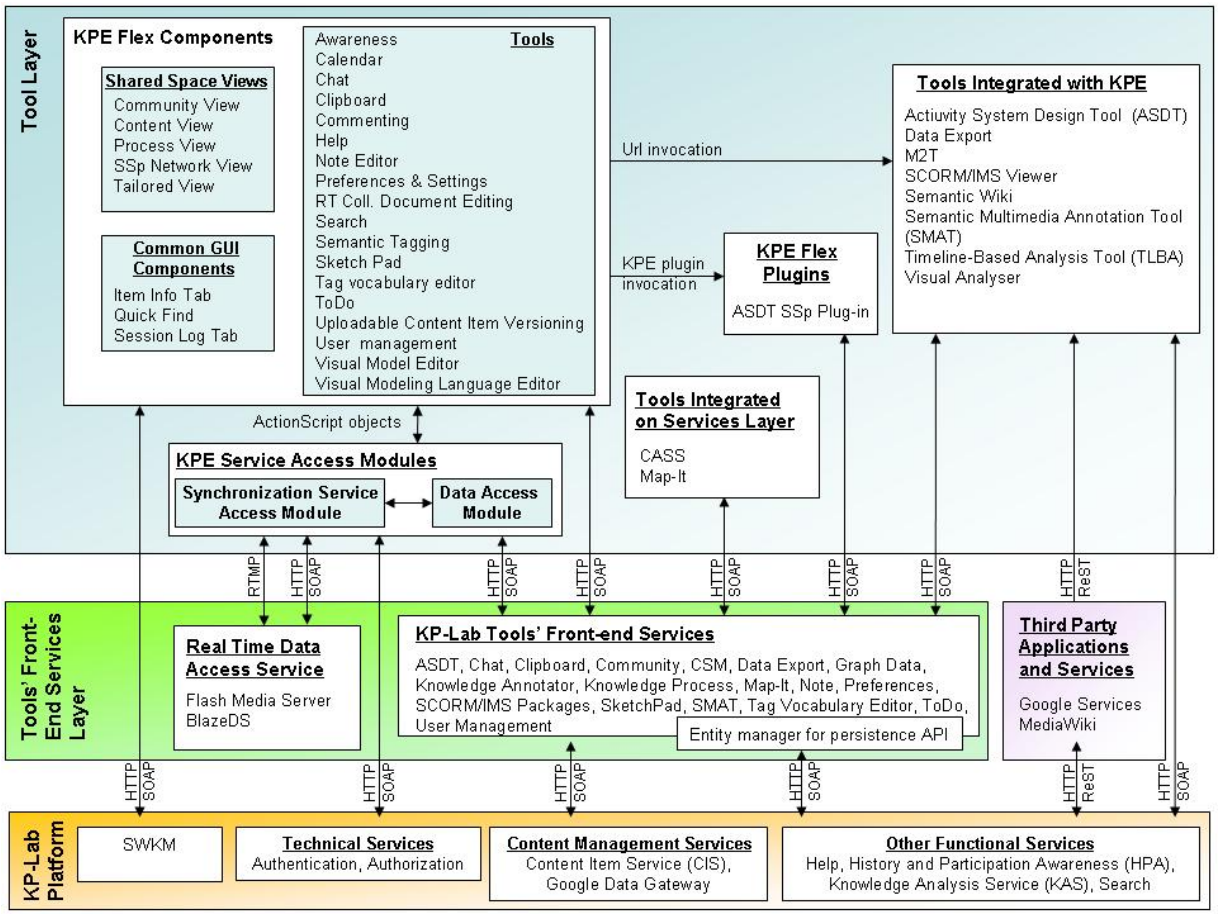


Figure 5: The implementation view of the Knowledge Practices Environment Architecture.

The architecture is divided in three separate parts: the *tool layer*, the *tools' front services layer* and the *KP-Lab Platform*.

- **Tool layer** is composed of KPE's client side components (*KPE Flex Components, KPE Service Access Modules, KPE Flex plug-ins*), of *Tools Loosely Integrated with KPE*, and of *Tools Integrated on Services Layer*.
- **Tools' front end services layer** is composed of *KP-Lab Tools' Front-end Services, Real Time Data Access Service* (provided by the Flash Media Server and BlazeDS server) and *Third Party Applications and Services.*
- **KP-Lab platform** provides common services to KPE and stand-alone tools. Examples of services provided by the platform are data persistence as well as user authentication and authorization. The KP-Lab platform is being developed within WPIII.

6.2.1 Tool layer

The tool layer of the KPE architecture consists of the following parts:

- **KPE Flex Components** offer the core functionality of the KPE. They are divided logically into three groups: *Shared Space Views*, *Tools* and *Common GUI Component*s.
- **KPE Flex plug-ins** include tools that are implemented separately from the KPE and integrated to the system dynamically during runtime.
- **Tools Integrated with KPE** are separate web applications that are invoked from the KPE using URL invocation.
- **Tools Integrated on Services Layer** are using the common underlying services but have their own, separate user interface.
- **KPE Service Access Modules** provide object libraries for accessing the underlying services from the tool layer. Two modules are available: *Data Access services* take care of invoking the data persistence services and of parsing and caching the retrieved data. *Synchronization Service* handles the real-time synchronization of tool instances on different clients and all real-time notifications about actions of KPE users.

The KPE Flex Components and the KPE Service Access Modules are implemented using the Flex technologies[2]. The Map-It Java application is based on the Eclipse platform and, more specifically, on the Eclipse Modeling Framework (EMF).

6.2.2 Tools' front-end service layer

The Tools' Front End Service Layer is divided logically into three different service groups: *1) KP-Lab Tools' Front-end Services, 2) Real Time Data Access Service* and *3) Third Party Applications and Services.*

KP-Lab Tools' Front-end Services

This group of services provides service access points for the end user tools. The KP-Lab Tools' Front-end Services are listed in the Table 5 with the description of their main functionality. For further details, please refer to the tool specifications in the annexes of

[2] http://www.adobe.com/products/flex/

D6.6 and DII.8. The Entity Manager for the Persistence API is utilized for retrieving and updating the data of the knowledge repository (SWKM).

Table 5: Overview of the KP-Lab Tools' Front-end Services.

Service	End user tools	Functionality
ASDT service	Activity System Design Tools (ASDT) clients.	Retrieving and storing data objects from/to KP-Lab System repositories (artifacts, access rights, GUI related objects, etc.).
Chat service	Chat tool	Instant messaging between the KPE users
Community service	Community View	Managing groups and personal contacts.
CSM service	VML and VMLE tools.	Creating and managing visual models and visual model languages.
Data Export service	Data export tool	Querying the Knowledge Analysis Service (KAS) for reporting and exporting data analysis.
Graph Data service	SSp Views	Retrieving and storing graph visualization, background image and object related metadata in the SWKM.
Knowledge Annotator service	SSp views, Commenting, Semantic Tagging	Annotating objects of activity with comments, linking them through hierarchical or description links, tagging objects with vocabulary terms.
Knowledge Process service	SSp Process view, SSp Content view	Management (creation, delete, modify, relations, etc.) of process elements, such as Tasks, Sub-tasks, Milestones.
Map-It server	Map-It, M2T	All Map-It APIs are exposed under Map-It server Services. HTTP/ReST invocations.
Note service	Note Editor	Creating and managing note objects in the content repository
Preferences Service	Preferences & Settings tool	Management of users' personal settings of KPE and shared spaces.
SCORM/IMS Packages service	SCORM/IMS viewer, Content item view.	Importing a content item of type SCORM/IMS package in the shared space, viewing it and exporting a content item as an IMS package.
Sketch Pad service	Sketch Pad tool	Creating and managing drawing objects in the content repository
SMAT services	SMAT	Services for loading SMAT from different usage contexts (standalone, shared space application, ASDT – Change laboratory); for structuring and composing multimedia content; for configuring of the annotation scenarios and phases; and for performing the actual semantic multimedia annotation activity.
Tag Vocabulary Editor service	Tag Vocabulary Editor tool	Creating SKOS vocabularies, management of vocabulary terms (add/modify/delete) and vocabulary metadata.

ToDo service	ToDo tool, SSp Views	Management (create, delete, modify, status) of to-do items based on To-do part of TLO.
User Management Service	User Management Tool	User registration and account management.

Front end services use the Entity Manager for the Persistence API if they use the Persistence API to persist and/or retrieve data from the Knowledge Repository.

Real Time Data Access Services
The Real Time Data Access Services provide the functionality for synchronizing user events and providing real-time awareness information between the different KPE clients. These services are implemented using the Adobe's Flash Media Server (FMS) and BlazeDS technology. Table 6 provides an overview of the existing services in the KPE.

Table 6: Overview of the Real Time Data Access Services.

Service	End user tools	Functionality
BasicSync	Awareness	Manages messaging related to moving of objects in the KPE GUI.
Chat Tool Presence Service	Chat	Keeping track of Chat tool users and synchronization between Chat tool clients.
Community FMS Application	Awareness	Manages messaging related to the session logs.
Global Presence Service	Awareness, Chat, ASDT	Keeps track of user locations and presence in KPE.
Object Locking Service	Awareness	Provides for locking a content item or only a part of a content item.
Presence FMS Application	Awareness	Keeps track of user status in a shared space.
Shared Space Sync FMS Application	Awareness	Manages messaging related to creation or deletion of objects in the KPE GUI.
Sketch Pad Synchronization service	Sketch Pad	Drawing synchronization between the users and keeping track of users currently working on the sketch.

The following **KP-Lab Platform Services** are used directly by the KP-Lab tools:

- *Semantic Web Knowledge Middleware*: Query, Update, Import, Export.
- *Content Management Services*: Content Item Service, Google Data Gateway
- *Other functional Services*: Search service, History and Participation Awareness (HPA) service, Knowledge Analysis Service (KAS), Help service
- *Technical Services*: Authentication service, Authorization service.

Third Party Applications and Services currently in use are the following:

- MediaWiki[3]
- Google Services (Calendar and Docs)[4]

[3] http://www.mediawiki.org/wiki/MediaWiki
[4] http://code.google.com/apis/gdata/overview.html

7 Conclusions and Future Work

The high-level view on the new specifications of end user applications defined in the WPII during the M37-M46 period of the KP-Lab project are presented in this deliverable. The document is divided into two parts: summary of the specifications and the individual tool specifications in the Annexes. The software design and coding will be done within the tool developers teams according to the iteration roadmaps presented in the chapter 5 of this deliverable.

This is the last specification deliverable of the KP-Lab tools and the functionalities specified here will be released by M48. After that, a final maintenance release is planned for the last period of the project, at M56.

8 References

1. KP-Lab project: *Developing Knowledge-Practices Laboratory. Description of Work 3.1* (M25-42). 2008.
2. KP-Lab project: *Developing Knowledge-Practices Laboratory. Description of Work 4.1.1 for Months 37-54*. 2009.
3. KP-Lab project: *A comprehensive research strategy*. Deliverable 3.2, revised version. 2008.
4. KP-Lab project: *Driving Objectives and High-level Requirements for KP-Lab Technologies (revised version)*. Deliverable 2.4. 2009.
5. KP-Lab project: *M21 specification of end-user applications.* Deliverable 6.4. 2007.
6. KP-Lab project: *M33 specification of end-user applications.* Deliverable 6.6. 2008.
7. KP-Lab project: *Guidelines and models on implementing design principles of KP-Lab, application scenarios and best practices, v.3*. Deliverable 2.3. 2009.
8. KP-Lab project: *KP-Lab Platform Architecture Dossier - Release 4.* Deliverable 4.2.4. 2008.
9. KP-Lab project: *Update of the KP-Lab Platform Architecture and Design Dossier (M54)*. Deliverable 4.2.5. To be submitted in January 2010.

Annex 1. Alternative Process View specifications

KP-LAB

Knowledge Practices Laboratory

Integrated Project

Information Society Technologies

DII.8 M46 specification of end-user applications – Alternative Process View

Due date of deliverable: 31/11/2009
Actual submission date:

Start date of project: 1.2.2006 Duration: 60 Months

Organisation name of lead contractor for this deliverable: Metropolia

Revision [1.0]

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission	
RE	Restricted to a group specified by the consortium (including the	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Contributors:

Name	Partner organisation	Email
Antti-Ville Hämäläinen	Metropolia	antti-ville.hamalainen@metropolia.fi
Olli Alm	Metropolia	olli.alm@metropolia.fi
Eini Saarivesi	Metropolia	eini.saarivesi@metropolia.fi
Hannu Markkanen	Metropolia	hannu.markkanen@metroopolia.fi

Version history:

Version	Date	Author(s)	Description
1.0	24.11.2009	Olli Alm	First draft
1.1	01.12.2009	Hannu Markkanen	Update of the roadmap
1.2	01.12.2009	Olli Alm	Use cases, gui, architecture and updated

Table of Contents

1 Introduction

1.1 Purpose

Process View and Alternative Process View provide the user interfaces for managing and visualizing knowledge processes within the shared spaces. The Process View visualizes the process by using a Gantt chart, in a timely-dependent manner. Alternative Process View provides a support for other models and forms of planning processes: it supports visualization and coupling of different activities. Users can visualize the process structure by utilizing predefined bitmap images and custom visual shapes to associate the activities to different Process Entities.

1.2 Scope of the software

The **Alternative Process View** has an UI to support following tasks in the KPE system:

- Gives user the possibility to draw a non-temporal visualization of the process
- Provides support for enhancing the process visualization with an existing bitmap picture
- Associates and organizes existing tasks to a specific process phase

1.3 Roadmap of the tool

The planned iterations for the tool are as follows:

Iteration 1: First version of the full functionality.
Milestones: M45 Specification, M46 Prototype, M48 Release v1.0

Iteration 2: Final version including improvements and bug fixes based on the user trials.
Milestones: M56 Release v1.1

1.4 What is new in this version?

This is the first version of the tool.

1.5 Definitions

- **Alternative Process View –** Alternative visualization of the specific tasks within a Shared Space. This view can be utilized for categorizing the tasks.
- **Content View –** a view where all the knowledge artefacts of a Shared Space is shown in two dimensional space. A content view also shows the relations between the objects.
- **Knowledge Practices Environment (KPE) –** KPE is the application that is used to log in to/out of the KP-Lab system, to use shared spaces with the integrated tools, and to launch other loosely integrated KP-Lab tools, such as the Activity System

Design Tool. The KPE covers all the (end-user) tools under development by Working Knots 1-5 (cp. DoW3.1, p. 27).

- **Knowledge Process –** A set of activities that describe a plan or a process to be followed in a certain collaborative activity. Knowledge Processes are described in the form of Tasks in the different views of a Shared Space.
- **No-man's land –** A part of the Alternative Process View which contains all the Tasks that are not assigned to any Process Entity.
- **Process Entity –** an object which categorizes the tasks of a Shared Space to have something in common. A Process Entity is shown in the Alternative Process View as a visual object in a screen, containing coupled task icons. The outlook of a Process Entity is described by the user by using predefined shapes (circle, rectangle, triangle), colours and size.
- **Process view –** a Gantt chart –like visualization of the tasks of a Shared Space. Process View visualizes the tasks in a timeline based on the start time, end time and the duration of the tasks.
- **a shared space –** a view of the KPE which consists of 1) a group of users, 2) the content items that are edited by the group members and 3) the member's activities that focus to the items. An end user can belong to multiple shared spaces. A content item belongs always to a shared space.
- **Tailored View –** a view where a set of knowledge artefacts and their relations can be inspected similarly as in a Content View.
- **Task –** a knowledge object that defines a certain activity or plan within a Shared Space. A task is visualized as 1) an icon in the Content View, 2) an object in a timeline in the Process View and 3) an icon inside a shape depicting a Process Entity in the Alternative Process View.

2 User requirements

2.1 KP-Lab empirical research

The tool functionalities specified in this document will be used in the following KP-Lab empirical research case [2]:

Case 2: Knowledge creation practices in customer projects in higher education

2.2 Driving Objectives and High-Level Requirements

The specifications presented in this document are related to the following driving objectives (DO) and high-level requirements (HLR) [1]:

DO8	Users can plan, organize and manage tasks collaboratively
HLR8.8	Users can choose between different types of process structure visualizations (e.g. project based or progressive inquiry). (New)

2.3 System usage scenarios

The specifications in this document are based on four different system usage scenarios presented in [3]:

1. US-APV-1-2008-11: Creating an Alternative Process View
2. US-APV-2-2008-11: Modification of an Alternative Process View
3. US-APV-3-2008-11: Using Alternative Process View elements in the Content View
4. US-APV-4-2008-11: Focusing on a certain process element (Tailored View)
5. US-APV-5-2008-11: Displaying the Process Elements in the GANTT Chart

2.4 Use cases

This section presents the use cases for the Alternative Process View.

2.4.1 Actors

Users of the Knowledge Practices Environment.

2.4.2 Use cases

UC-APV-01-2009-11: Uploading the background image for the process

UC-APV-02-2009-11: Creating the process workflow

UC-APV-03-2009-11: Associating tasks to process entities

UC-APV-04-2009-11: Modifying process visualization

UC-APV-05-2009-11: Removing process objects

UC-APV-06-2009-11: Showing the detailed information of the process object

UC-APV-07-2009-11: Showing the process entities in the Process View

UC-APV-08-2009-11: Showing the process entities in the Content Item View

2.4.3 Detailed Use Cases

Title: Uploading the background image for the process

Unique identifier: UC-APV-01-2009-11

Related usage scenarios: US-APV-1-2008-11: Creating an Alternative Process View

Description: The user is in the KPE and enters to Alternative Process View. User wants to visualize her process on a basis of an existing picture showing the process skeleton. The user clicks the ‘upload image’ from the left side control tab, the select file –dialog is opened. She selects the file and the picture is uploaded. The uploaded pictures are shown in the drop-down menu below the ‘upload image’ –button. The user selects the uploaded image and clicks the ‘show’ button below the menu. The image is rendered to the view’s background area.

Pre-conditions: The user has successfully logged in the KPE.

Post-conditions: -

Nominal scenario:

- User logs in to KPE.
- User enters the Alternative Process View
- User uploads the picture
- User selects a picture and it is rendered as a background image in the Alternative Process View

Non-functional requirements: The user should be informed that the quality and the size of the picture should be adequate in resolution.

Screen mock-ups:

Comments and notes: Should it be possible to resize the picture in the Alternative Process View?

Title: Creating the process workflow

Unique identifier: UC-APV-02-2009-11

Related usage scenarios: US-APV-1-2008-11: Creating an Alternative Process View

Description: The user is in the KPE and has entered to Alternative Process View. User wants to illustrate an iterative process about trialogical learning. She hasn't uploaded a background picture. The user draws a *process entity* to the screen by 1) selecting a circle shape from the bottom of the screen and 2) dragging it to the screen. The user fills the form to define the *name, description, colour, location and size* for the shape. The user then adds more *process elements* to screen, defined with specific shape (rectangle, triangle) and colour. Then user defines the flow between the elements by drawing arrows between the elements. The shapes and the arrows now illustrate a cyclic process flow.

Pre-conditions: The user has successfully logged in the KPE.

Post-conditions: -

Nominal scenario:

- User logs in to KPE.
- User has entered the Alternative Process View
- User draws *process entities* on the screen and fills the detailed information about them.
- User depicts the flow between the *entities* by drawing arrows between them.

Non-functional requirements: The drawing tool should be intuitive and easy to use. The location of the entities should be able to move without breaking the associations.

Screen mock-ups:

Title: Associating tasks to process entities

Unique identifier: UC-APV-03-2009-11

Related usage scenarios: US-APV-1-2008-11: Creating an Alternative Process View

Description: The user is in the KPE and has entered to Alternative Process View. The user has defined *process entities* in the Alternative Process View (UC-APV-02-2009-11). The user wants to associate a specific task to a process entity. All the tasks that are defined in the shared and are not associated to any process entity, lie in the *no-man's land* area at the bottom of the screen. The user associates a task by dragging and dropping it inside a shape. The *task icon* is now removed from the *no-man's land* and located inside the shape.

Pre-conditions: The user has successfully logged in the KPE. The user has existing *process entities* in the Alternative Process View.

Post-conditions: -

Nominal scenario:

- User logs in to KPE.
- User has entered the Alternative Process View which contains previously defined *process entities.*
- User associates a task to an entity by dragging it from *no-man's land* inside a shape presenting an entity.
- Task is now associated to entity and the task icon is relocated inside the shape.

Non-functional requirements:

Screen mock-ups:

Title: Modifying process visualization

Unique identifier: UC-APV-04-2009-11

Related usage scenarios: US-APV-2-2008-11: Modification of an Alternative Process View

Description: The user is in the KPE and has entered to Alternative Process View. User has previously defined process entities in the Alternative Process View. The user is not satisfied with the current visualization. User changes the location of several *process entities* by dragging them on the screen. The tasks inside the entity and the related links (arrows) are relocated at the same time. The user also changes entity's outlook and description by right-clicking the entity and selecting 'modify entity' from the context dialog: the form with entity information is shown. User changes the description and the shape of the object and proceeds. The entity's shape is now changed.

Pre-conditions: The user has successfully logged in the KPE. The user has existing *process entities* in the Alternative Process View.

Post-conditions: -

Nominal scenario:

- User logs in to KPE.
- User has entered the Alternative Process View which contains previously defined *process entities.*
- User changes the location of the *process entities* by dragging them on the screen.
- User changes the outlook and information of an entity from the modification form.

Non-functional requirements: The layout algorithm for included task has to be smooth and intelligent: if there are multiple tasks inside an entity shape and the shape size is reduced, the tasks may be difficult to separate from each other.

Screen mock-ups:

Title: Removing process objects

Unique identifier: UC-APV-05-2009-11

Related usage scenarios: US-APV-1-2008-11: Modification of an Alternative Process View

Description: The user is in the KPE and has entered to Alternative Process View. User has an existing visualization of the process, including associated tasks in the entities. User breaks the association between entity and task by right-clicking the task icon inside the process entity and selecting 'detach'. The task is removed from the entity and brought back to *no-man's land* region at the bottom of the screen. Then, user wants to remove a process entity. By right-clicking inside the shape, the context dialog is opened and user clicks 'remove'. The entity shape and the links associated to it are now removed. The associated tasks are relocated to *no-man's land*.

Pre-conditions: The user has successfully logged in the KPE. The user has existing *process entities* in the Alternative Process View.

Post-conditions: -

Nominal scenario:

- User logs in to KPE.
- User enters the Alternative Process View
- User detaches a task from the entity.
- The detached task is relocated to *no-man's land*.
- User removes a process entity.
- The entity shape and its associated links are removed, tasks inside it are relocated to *no-man's land.*

Non-functional requirements:

Screen mock-ups:

Title: Showing the detailed information of the process object

Unique identifier: UC-APV-06-2009-11

Related usage scenarios: US-APV-4-2008-11: Focusing on a certain process element (Tailored View)

Description: The user is in the KPE and has entered to Alternative Process View. User has an existing visualization of the process, including associated tasks in the entities. User wants to see objects related to a task. User selects the task and clicks 'open' from the right-click context dialog. The task view is opened and task's subtasks and content items related to task are shown in two-dimensional space as icons. User goes back to Alternative Process View. Now, user wants to view all the objects associated to a *process entity*. User right-clicks the region inside the shape and chooses 'open' from the context dialog. The *tailored view* is opened, showing 1) all the associated tasks, 2) tasks' subtasks and 3) tasks' related content items.

Pre-conditions: The user has successfully logged in the KPE. The user has existing *process entities* in the Alternative Process View.

Post-conditions: -

Nominal scenario:

- User logs in to KPE.
- User enters the Alternative Process View
- User opens a task: the related items are shown as icons in the two-dimensional space.
- User goes back to Alternative Process View
- User opens an entity: the tailored view is opened, showing all the tasks inside the entity and their related content items.

Non-functional requirements:

Screen mock-ups:

Title: Showing the process entities in the Process View

Unique identifier: UC-APV-07-2009-11

Related usage scenarios: US-APV-5-2008-11: Displaying the Process Elements in the Gantt chart

Description: User is in the KPE and enters the Process View. The tasks are visualized on a timeline. User has already defined *process entities* and tasks associated to them. A checkbox dialog is shown on the right side of the Process View. On each line of the dialog an entity name and the associated colour is shown. User selects an entity by clicking the checkbox on the specific row. The tasks associated to entity are highlighted in the Process View with the specific colour shown in the selection dialog.

Pre-conditions: The user has successfully logged in the KPE. The user has existing *process entities* in the Alternative Process View with tasks associated to them.

Post-conditions: -

Nominal scenario:

- User logs in to KPE.
- User enters the Process View
- User selects an entity from the dialog on the right-side tab
- The tasks associated to the entity are highlighted with the specific colour shown in the checkbox dialog.

Non-functional requirements:

Screen mock-ups:

Title: Showing the process entities in the Content Item View.

Unique identifier: UC-APV-08-2009-11

Related usage scenarios: US-APV-3-2008-11: Using Alternative Process View elements in the Content View

Description: User is in the KPE and enters the Content Item View. The entities defined in the Alternative Process View. User has already defined *process entities* and tasks associated to them. The *process entities* are presented with the specific icon type. The tasks associated to them are visualized with a relationship link (an arrow) pointing from task to the entity. User is able to open a process entity by double-clicking the icon or choosing 'open' from the context dialog (see use case UC-APV-06-2009-11).

Pre-conditions: The user has successfully logged in the KPE. The user has existing *process entities* and tasks associated to them.

Post-conditions: -

Nominal scenario:

- User logs in to KPE.
- User defines / has defined *process entities* and associated tasks to them in the Alternative Process View.
- User opens the Content Item View.
- Process entities are shown as icons in the two dimensional space, associated tasks are related to them with the relationship links (arrows).

Non-functional requirements:

Screen mock-ups:

3 Technical design

3.1 Software architecture

Figure 1 presents the overall architecture of the Alternative Process View. The user interface and the visualization functionalities of the Alternative Process View are implemented in the Tool layer. For storing and fetching *process entities* and tasks, the Knowledge Process Service is utilized. Knowledge Process Service stores the data inside SWKM, the semantic data repository.

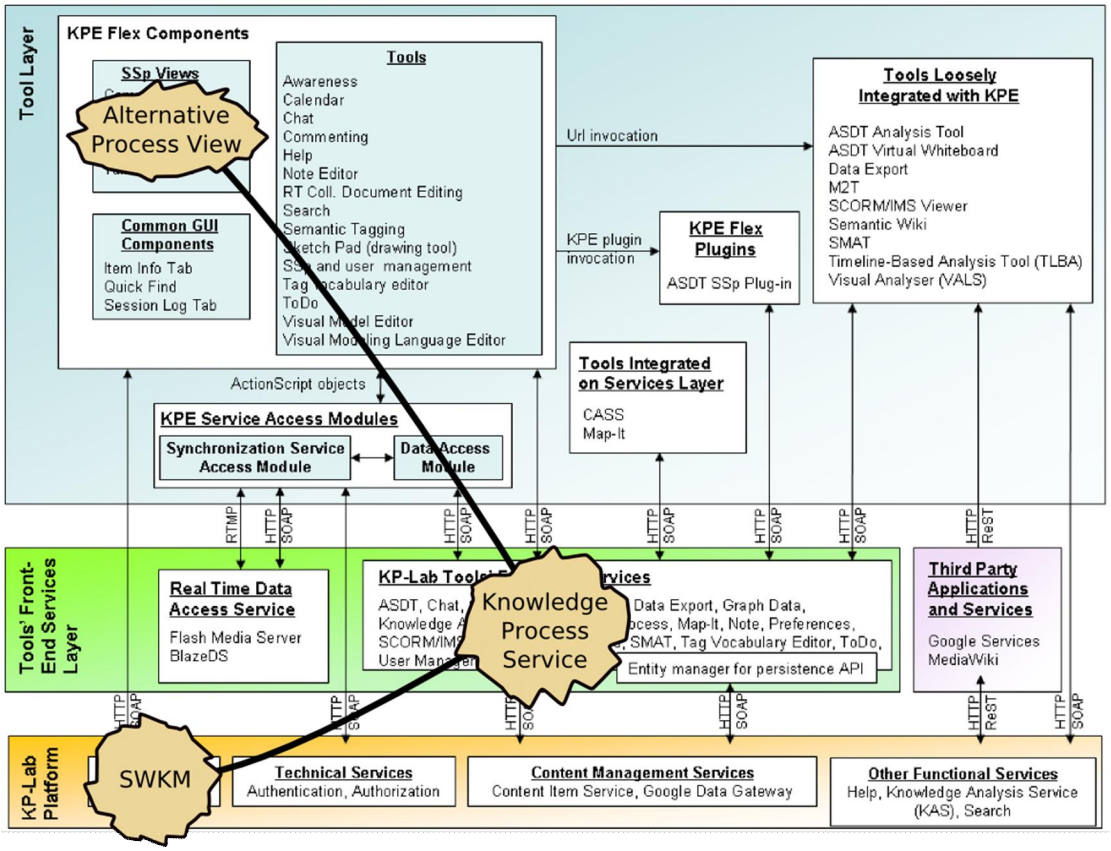


Figure 6: Alternative Process View and the overall architecture

3.2 GUI

3.2.1 Screen Design

In the figure 2, an empty Alternative Process View is presented: user has entered the view and has not defined any elements to it.

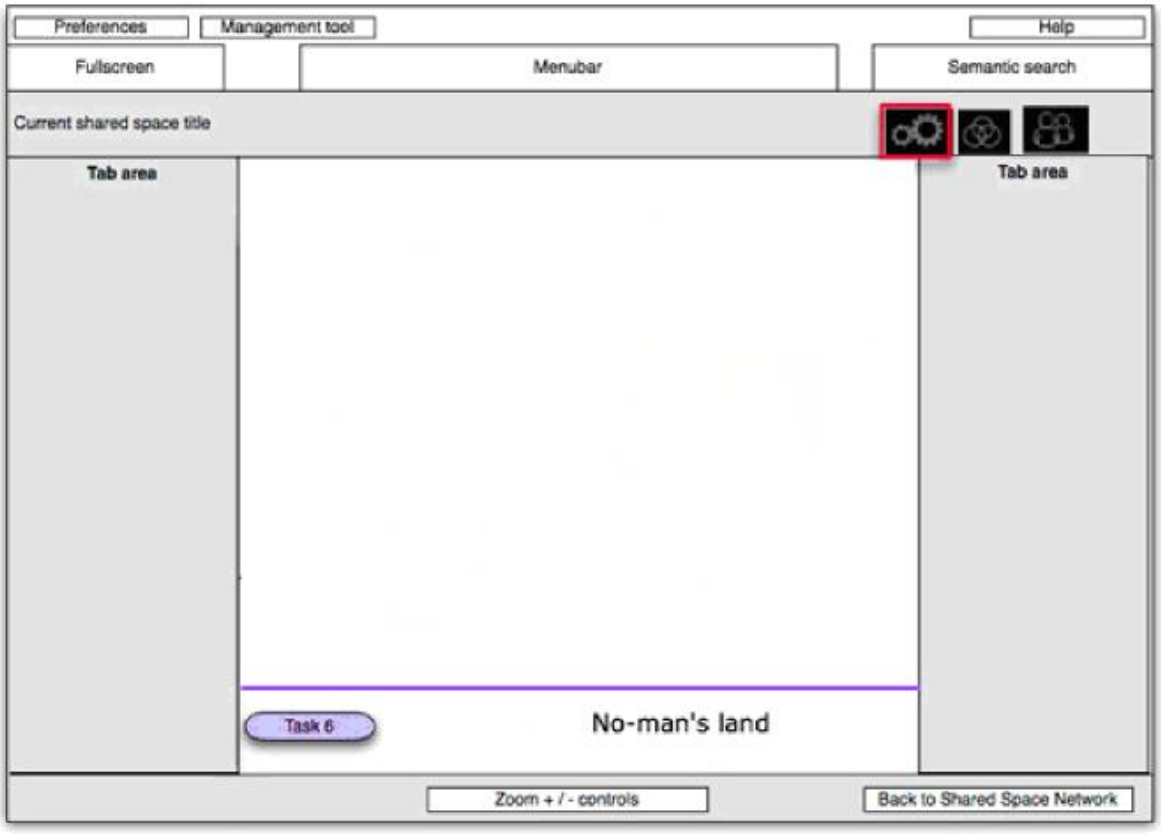


Figure 7: Empty Alternative Process View

The tasks defined in the shared space are located in the *no-man's land*, at the bottom of the screen. Currently, there is only one task ('task6').

In the figure 3, user has uploaded a background image to visualize the process to be presented. The background image presents a cyclic model of progressive inquiry. On the top of the picture, user can add shapes that define the *process entities* for the model: with the *process entities* user defines formally the borders of an activity from the picture. If the user doesn't want to cover the original picture, she can use transparent *process entities.*

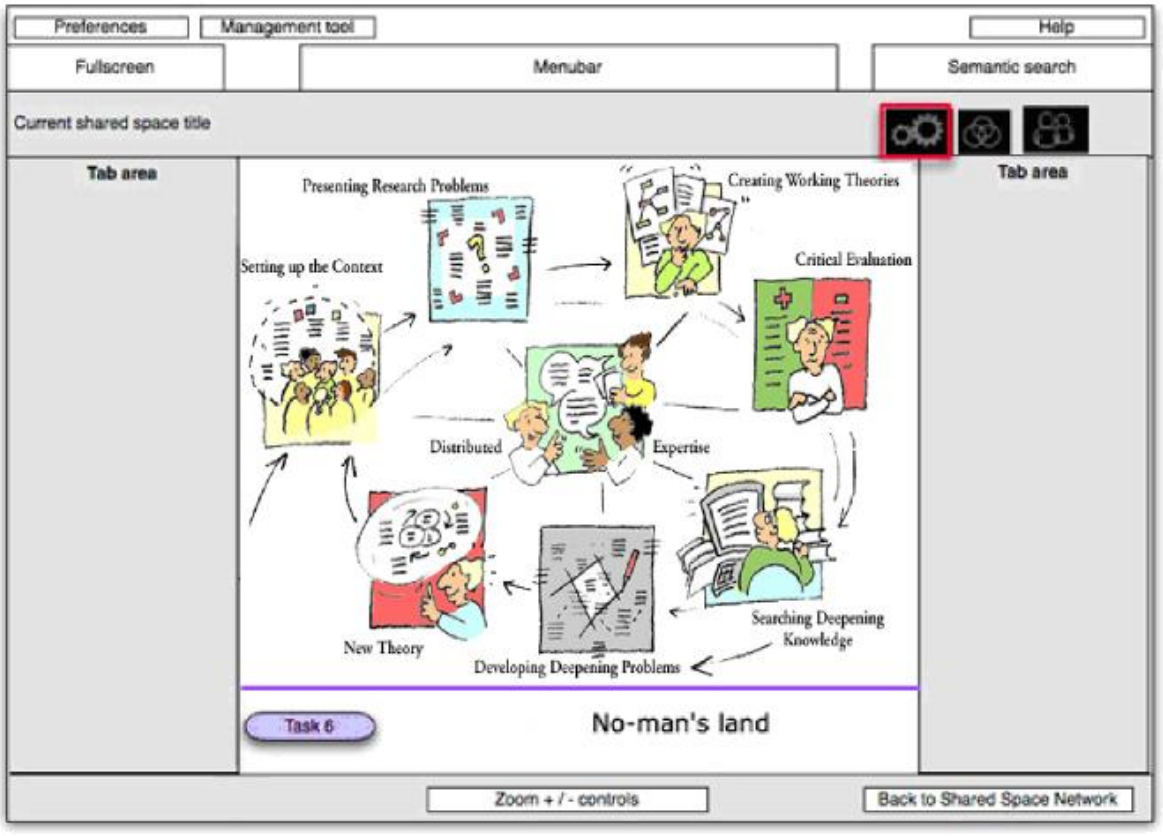


Figure 8: Alternative Process View with uploaded background image

In figure 4, user has 1) defined transparent *process entities* on the picture AND 2) associated several tasks to them. The tasks are associated by dragging and dropping them from the *No-man's land* inside the shape. The process entities are depicted in the picture with dashed lines.

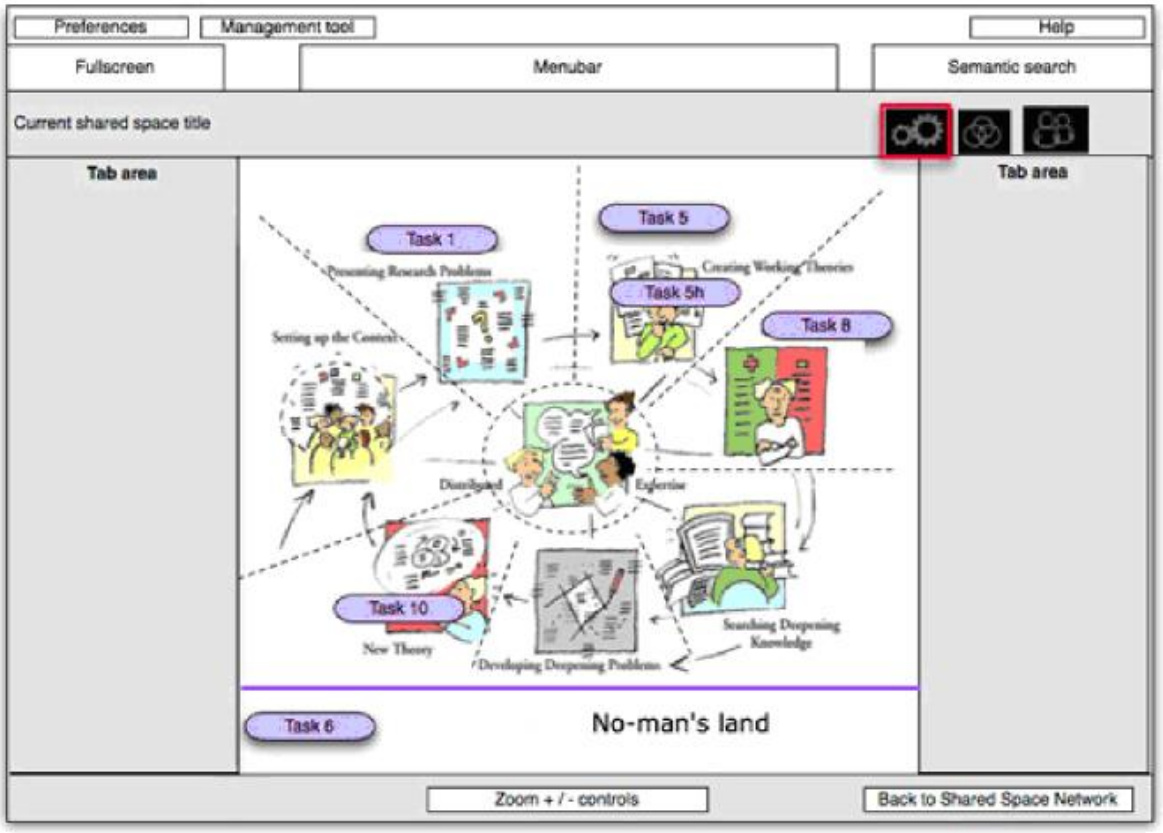


Figure 9: Process entities with associated tasks on the picture

3.2.2 Services used

The Alternative Process View utilizes Knowledge Process Service for storing and fetching the process entities, links associating the process entities and tasks.

3.2.3 Relation to Ontologies

The process model defined in the Alternative Process View is stored in the SWKM repository. The data and their relations are modeled in RDF. All the actions taken during the definition of the process are logged for further semantic analysis of the knowledge evolution.

3.3 Front end services

Alternative Process View utilizes the Knowledge Process Service for storing the task and process entity information.

3.4 Collaboration with other tools

The Alternative Process View data is synchronized with the Process View and the Content Item View. In the Alternative Process View, the association between a task and a process element is created by dragging a task inside a process element. In the Content Item View,

the association is shown with a link between the items. In the Process View, the association is shown with a colour highlighting.

References

[1] D2.4 Driving Objectives and High-level Requirements for KP-Lab Technologies (revised version). http://www.kp-lab.org/intranet/work-packages/wp2/result/D2.4_Resubmission_290809.doc

[2] D3.2 A comprehensive research strategy (an updated and revised version for years 4 and 5). http://www.kp-lab.org/intranet/work-packages/wp1/submission-to-commission-23-9.2009/D3.2-Rev-Research-Strategy_September_submitted.doc

[3] Alternative Process View System Usage Scenarios http://www.kp-lab.org/intranet/design-teams/wk-process-management-and-analysis/alternative-process-views/wk3-apv_systemusagescenarios02_TUK-FHOOE.doc/view?searchterm=alternative%20process%20view

Annex 2. Clipboard tool specifications

27490

KP-LAB

Knowledge Practices Laboratory

Integrated Project

Information Society Technologies

M44 software specification of end-user applications – <Clipboard Tool in KP Environment>

Due date of deliverable: …..
Actual submission date:

Start date of project: 1.2.2006 Duration: 60 Months

Organisation name of lead contractor for this deliverable:

Revision [1.0]

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission	
RE	Restricted to a group specified by the consortium (including the	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Contributors:

Name	Partner organisation	Email
UH	Minna Lakkala Sami Paavola	minna.lakkala@helsinki.fi , spaavola@mappi.helsinki.fi
UU	Crina .Damsa	C.I.Damsa@fss.uu.nl
METROPOLIA	Markus Holi Hannu Markkanen Eini Saarivesi Merja Bauters Patrick Auderau	markus.holi@metropolia.fi hannu.markkanen@metropolia.fi eini.saarivesi@metropolia.fi merjab@metropolia.fi patricka@metropolia.fi
TUK	Jan Paralic Jozef Wagner	Jan.Paralic@tuke.sk jozef.wagner@gmail.com
DIBE	Marine Scapolla Angela Locoro	scapolla@unige.it, angela.locoro@unige.it
FH-OÖ	Christoph Richter Eva Zoeserl	Christoph.Richter@fh-hagenberg.at , eva.zoeserl@fh-hagenberg.at

Version history:

Version	Date	Author(s)	Description
0.1	21.9.09	Marina Scapolla	First draft
0.1	21.9.09	Eini Saarivesi	User Requirements and MockUps
	21.9.09	Merja Bauters	User Requirements
0.2	26.10.09	Angela Locoro	Changes to the front end services description
	28.10.09	Marina Scapolla	Modifications to the technical part
0.3	30.10.09	Patrick Ausderau	Contributions on how to manage the copy of objects and interactions with the GUI level.
0.31	02.11.09	Merja Bauters	User requirements Updates
0.32	04.11.09	Angela Locoro Marina Scapolla	Modifications to the technical part given the suggestions from Patrick
0.4	16.11.09	Merja Bauters and Saarivesi	New SUS added
1.0	18.11.09	Marina Scapolla and Angela Locoro	Final revision

Table of Contents

1 Introduction

1.1 Purpose

This document describes the functionality of the Clipboard Tool facility inside the KP Environment (KPE). It supports the operations of copying and pasting different items in the KPE.

The document contains use cases, mock-up screens, data model and describes the web service that will be developed for this functionality.

1.2 Scope of the software

This software allows creating new items in the KPE starting from existing ones.

These items can be:

- Shared Spaces
- Tasks
- Vocabularies
- Content Items
    - Uploadable File, Web link, Note, Sketch, Scorm/IMS file, Wiki pages

1.3 Roadmap of the tool

This is the first version of the tool.

M44 Specifications
M46 Prototype Release
M48 Production release

2 User requirements

2.1 Relationship with KP-Lab research studies

The pedagogical studies related to the KPE Clipboard Tool functionality are:

Case 1: Trialogical work in higher education; creation and use of knowledge objects in knowledge creation practices

Case 2: Knowledge creation practices in customer projects in higher education

Case 3: Conceptual modelling in Design Projects

Case 4: Producing document management solutions and practices for distributed work settings – Perspectives of KPE usage in two work organizations

(See details in the D 3.2. Revised Research Strategy)

2.2 Driving Objectives and High-Level Requirements

The Driving objective (DO) relevant for the tool is:

DO 3: Users are provided with support for the re-use of shared artefacts and structures

The high-level requirement (HLR) relevant for the tool is:

HLR3.1	Users can re-use process structures and artefacts.

The DO and HLR are specified in Deliverable 2.4 Resubmitted ([1], pp 28-29).

2.3 System User Scenarios

These specifications are based on two different system usage scenarios presented in the "WK1: Shared Space and Common Tools System usage scenarios for the Clipboard Tool" document:

1. US-Clipboard-1-2009-11 Copying a Shared Space and pasting it as a new Shared Space
2. US-Clipboard-2-2009-11 Copying an element or elements (e.g. Content Items, vocabularies) and pasting them in another Shared Space.

2.4 Use cases

This section presents the two relevant use cases of the tool.

2.4.1 Actors

KPE users

2.4.2 Index of Use cases

1. UC-Clipboard-1-2009-11 Copying and pasting a Shared Space
2. UC-Clipboard-2-2009-11 Copying and pasting elements from a Shared Space to another.

2.4.3 Detailed Use Cases

1)

Title: Copying and pasting a Shared Space

Unique identifier: UC-Clipboard-1-2009-11

Related System Usage Scenarios: US-Clipboard-1-2009-11

[1] D2.4 Driving Objectives and High-level Requirements for KP-Lab Technologies, version 0.4. http://www.kp-lab.org/intranet/work-packages/wp2/deliverable-2-4-driving-objectives-and-high-level-requirements-for-kp-lab-technologies-m32/Deliverable24-fourth-draft-DK-CR.doc/view

Actors: KPE users who have right to create Shared Spaces

Description: The user wants to make a copy of a Shared Space s/he used before.

Pre-conditions: The user has entered the Shared Space Network.

Post-conditions: A new Shared Space is created. Details about its properties are in the Technical Design Section. The action is logged in the HPA archive. Pasting a Shared Space implies making a copy also of Tasks, Links between tasks (if tasks have links to other objects they will be ignored), Vocabularies and Tools.

Nominal scenario:

1. The user has logged into the KPE, is in the Network View and has rights to create a Shared Space.
2. The user selects her/his Shared Space s/he wants to copy and by the top edit menu or the right click menu, or by using shortcuts (Ctrl+C) chooses ”Copy”.
3. The user navigates to the place in the Network View where s/he wants to paste the Shared Space and selects “Paste” right clicking on the background, or from the top menu.
4. A pop-up is displayed where the user can define the title, description and starting date for the tasks of the new Shared Space. The user fills a unique title and optionally a new start date for tasks and a new description, and click OK.
5. The user receives a message confirming that the Shared Space was successfully created.

Screen mock-ups:

Mock-ups of general functions of copying and pasting are reported in the section that details the use case UC-Clipboard-2-2009-11.

The image in Figure 1 displays the pop-up form that is presented to the user to fill before the actual pasting will occur. The user can define the title, modify the description and if s/he wishes change the starting date of the tasks. The system will check if the title provided by the user is unique, i.e., that it is not already in use by some other Shared Space. If this would happen the system advises the users to change the title by stating in the red advice style “The title you suggest is already in use, please set another title”.

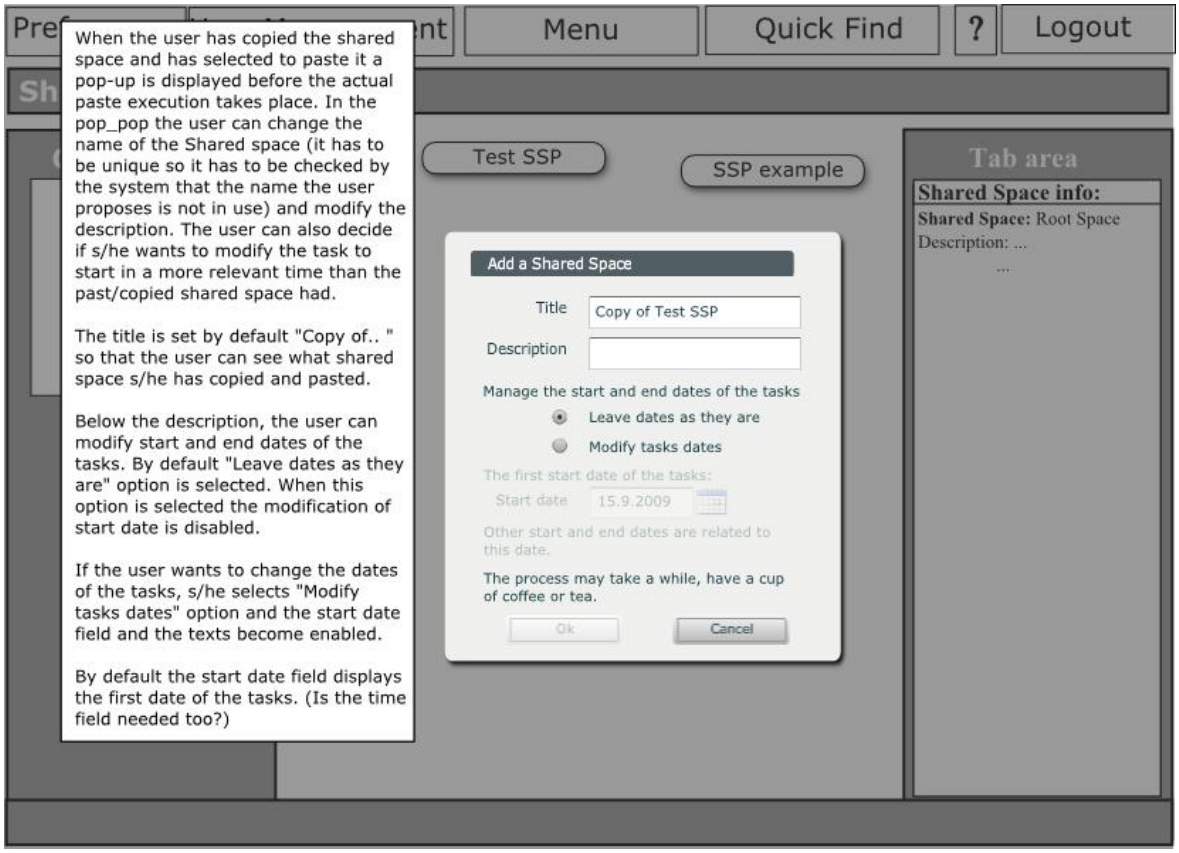


Figure 1 – Pasting a Shared Space.

2)

Title: Copying and pasting elements from a Shared Space to another

Unique identifier: UC-Clipboard-2-2009-11

Related System Usage Scenarios: US-Clipboard-2-2009-11

Actors: Users who are allowed to add items in the source and the destination Shared Spaces

Description: The user wants to reuse his/her items again in another Shared Space. The user opens the Shared Space from where s/he wants to copy the items and selects them. Then the user navigates back to the Network View and finds the Shared Space into which s/he wants to paste the items. The user goes into the Shared Space and pastes them. A lock icon appears on top of the to be pasted Content Items. In addition, a notice appears on the Content View. The notice states that "The process may take a while", since the user copied and pasted many Content Items.

The items selected to be copied can be of type: Wiki, Uploadable File, Web link, Note, Sketch, Scorm/IMS file, Wiki page, Task, Vocabulary.

Pre-conditions: The user has entered a Shared Space and his/her role allows adding items**.**

Post-conditions: New items are created, the action is logged in the HPA archive.

Nominal scenario:

1. User has logged into the KPE and navigates from the Network View into the Shared Space s/he wants to copy the items from.
2. The user selects the items to be copied.
3. The user can copy them from the "Edit menu", the right click menu or by using shortcuts (Ctrl+C) (see Fig. 2). A notice is displayed stating the copying was successful.
4. The user moves to a different Shared Space.
5. The user selects "Paste" from the Edit menu (see Fig. 3), the right click menu or by using shortcuts (Ctrl+V).
6. A lock icon and notice is displayed. The notice states: "The process may take a while" The new items are created and the user sees them in a diagonally overlapping pile.

Screen mock-ups:

The picture in Figure 2 presents how the copying and pasting occurs and how these options are displayed to the user.

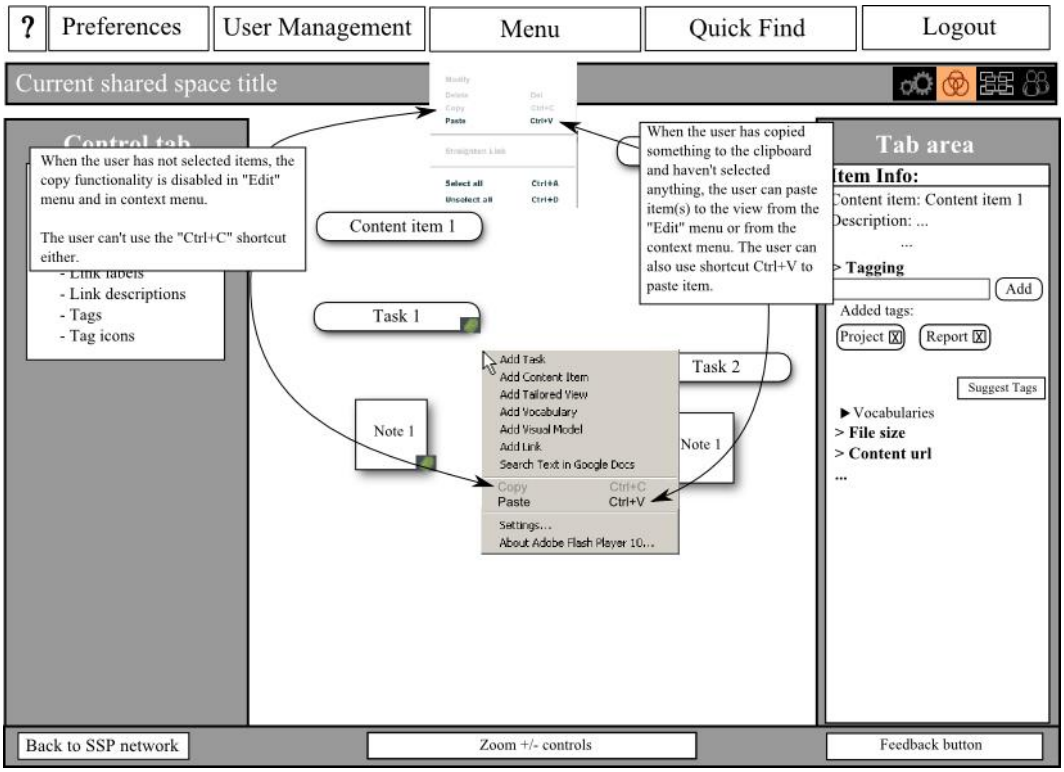


Figure 2. Copying and pasting options

Figure 3 and 4 show the process of copying and pasting item(s) or a Shared Space. The system indicates that something is happening, i.e., displaying the lock icon on the item. The item should be placed to the place in the Content View (or Network View in the case of Shared Space) where the users placed the cursor to place the item. If the user is using the menu to paste, then the item(s) is positioned in the same way as when creating a new item.

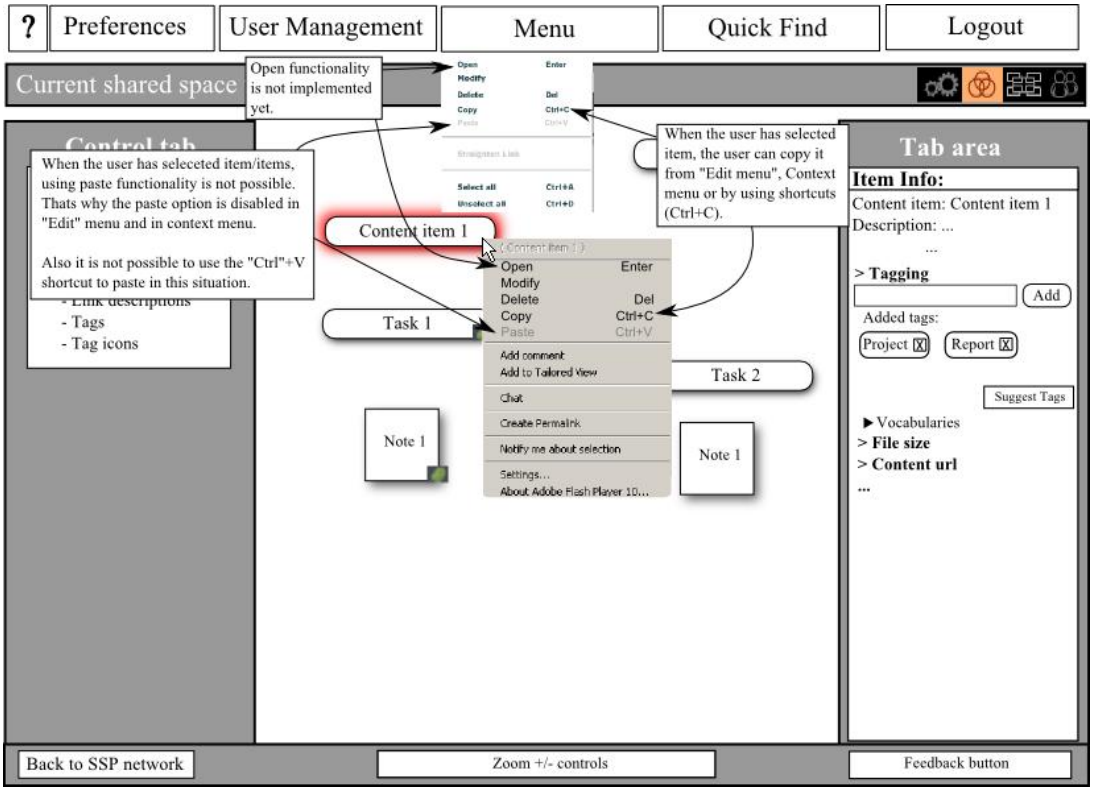


Figure 3. Copying items

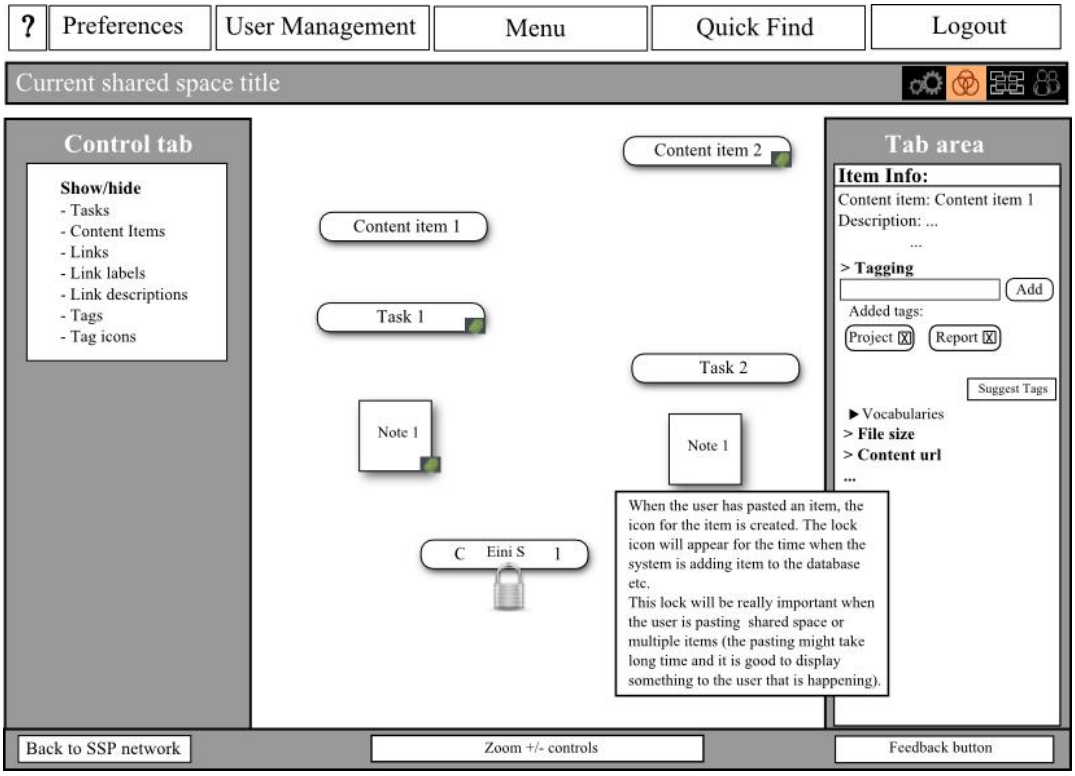


Figure 4 Pasting process

3 Technical design

3.1 Software architecture

The overall KPE architecture documentation is available in the project Plone at the URL http://www.kp-lab.org/intranet/work-packages/wp6/result/dii-8-m46-specification-of-end-user-applications/kpe-architecture/folder_listing

3.2 Front end services

The Clipboard Tool Service will be implemented as a web service. It may use either the Persistence API or the KMS Request Engine to communicate with the Knowledge Repository. The data stored on the Content Repository will be accessed via CIS Service (Content Item Web Service).

3.2.1 Web Service Interface Description

3.2.1.1 Methods provided

SharedSpacecopySSP(StringSSPURI,newTitle,newDescription,newStartDate,creatorURI)

This is the method for copying a Shared Space (see section 3.3.2.1 for details). The Shared Space creator is the user who does the copy. A "shift date" feature for the task when copying the Shared Space will be provided (e.g. I have a Shared Space for the course I hold in 2009 and I want to copy it for 2010, then I provide a new "start" date, and the service takes care of using that date for the first task starting date and calculates the durations of all the others tasks based on that).

input parameters:

Name	Value type	Description
SSPURI	String	the URI of the SSP to be copied
newTitle	String	The title for the new SSP
newDescription	String	The description for the new SSP
newStartDate	Date	start date of the earliest task of the new Shared Space
creatorURI	String	the URI of the user who does the copy and will become the creator of the new Shared Space

output parameters:

Value type	Description
SharedSpace	The new SSP just created from the copy

String copy (ObjectOfActivity objectToCopy, String destinationURI, String creatorURI)

Method for copying a single object.

input parameters:

Name	Value type	Description
ObjectToCopy	ObjectOfActivity	The wrapper of the Object to be copied
destinationURI	String	The URI of the destination SSP
creatorURI	String	the URI of the user who does the copy and will become the creator of the new object

output parameter:

Value type	Description
String	the URI of the new object just copied

The object will be copied to the destination SSP selected by the user. The creator of the new object is the user who does the copy. A new URI will be created of the same type of the

original URI, and selected properties for that type will be copied. The method returns the URI of the new object. The original type will be assigned to the new object.

The method copy is complex and consists of several operations:

- Based on the object type a different selection of properties and different copy procedures will be implemented (see details of properties and metadata for each different type in section 3.3.2).
- For any object tagged with Semantic Tags, the service will search the tagging terms in every vocabulary of the destination Shared Space (matching the PrefLabel and AltLabel fields). If a term does not exist, the term is added to the free tags vocabulary. The new URI is assigned to the hasSemanticTag property of the new object.

Uploadable Content Items (as the Notes) will have versions in the KPE M46-48; for the moment the last version of the item will be copied. When versioning will be used the method will be adjusted in order to take care of which version/versions of the item must be copied.

String[] copyMultipleItems (OOA[] objectsToCopy, String destinationURI, String creatorURI)

Same method as above, but usable when the user selects multiple items to copy. These items can be of different types. The creator of the new objects is the user who does the copy. The method will copy relationship/hierarchical links if both start and end nodes that participate in the relation are selected for the copy, as well as the link itself.

input parameters:

Name	Value type	Description
ObjectsToCopy	ObjectOfActivity[]	The wrapper of the Objects to be copied
destinationURI	String	The URI of the destination SSP
creatorURI	String	the URI of the user who do the copy and will become the creator of the new object

output parameter:

Value type	Description
String[]	the URIs of the new objects just copied in the same order as they were requested

3.2.2 Analysis of metadata and properties of the items created using templates

Here below a list of the properties that will be set for each type of object at the moment of creating the new object. Notes beside the "Y" choice are present when the value of the property is not copied from the source object but is set to a new value.

3.2.2.1 SharedSpace

Property	***Copied? Y/N***
belongsToSharedSpace	Y Property must be set to the default "RootSpaceURI"
dc_alternative	N
dc_created	Y The date is set to the date of copy
dc_creator	Y The creator is the user who does the copy
dc_description	Y The new description is given as a parameter
dc_modified	N
dc_title	Y The new title is given as a parameter
employsOntology	Y
hasCommentAnnotation	N
hasOutboundRelationship	N
hasSemanticTag	N
hasStatus	N
isProtectedTo	N
refersTo	N

A Shared Space can have tasks and they will come along (see section 3.3.2.2 for details). Links between tasks will be copied. If tasks have links to other objects they will be ignored.
A Shared Space can have vocabularies, they will be copied when copying the Shared Space (see section 3.3.2.3 for details).
A Shared Space can have tools and these tools must be associated also to the copied Shared Space. This association implies setting the following properties for each tool:

Property	*Copied? Y/N*
belongsToSharedSpace	Y
dc_created	Y the new date is the date of copy
dc_creator	Y the new creator is the user who does the copy
dc_description	Y
dc_modified	N
dc_title	Y
manifestURI	Y

3.2.2.2 Tasks

To copy Tasks we must consider three different cases:

1. When Tasks are copied as a consequence of copying a Shared Space: subTask and taskPrerequisites are copied, other items related to the tasks are not copied;
2. When a specific Task is selected by the user inside a Shared Space to be copied: the subTask and taskPrerequisites are not copied;
3. When multiple Tasks are selected to be copied then also relationship/hierarchical links, properties such as subTask, hasContentItem or hasOutboundRelationship can be copied too if the user has selected them for copy.

Property	**Copied? Y/N**
assignedMember	N
endTime	Y According to newStartDate parameter this value will be shifted properly
hasContentItem	N Unless link to the Content Item and The content itself are selected for copy
hasContentItemOrganizer	N
hasMilestone	N
startTime	Y According to newStartDate parameter this value will be modified properly
subTask	Y In case subTasks are selected for copy
taskIndex	Y This property is used for ordering purposes and must be adjusted on the basis of tasks copied
taskPrerequisites	Y This property describes relations between task, example. The next task cannot start …
taskStatus	N
belongsToSharedSpace	Y This property must be set to the proper destination Shared Space
dc_alternative	N
dc_created	Y The date is the date of copy
dc_creator	Y The creator is the user who does the copy
dc_description	Y
dc_modified	N
dc_title	Y
hasCommentAnnotation	N
hasOutboundRelationship	Y Only in case subtasks will come along with the copy
hasSemanticTag	Y
hasStatus	N
isProtectedTo	N
refersTo	N

3.2.2.3 Vocabularies

Property	***Copied? Y/N***
vocabularyURI	Y The new vocabulary will have a new URI
hasTopConcept	Y The new top concepts URIs will be associated
resource_Description	Y
belongsToSharedSpace	Y This property will be created to link the SSP where the object is copied
dc_created	Y The date is the date of copy
dc_creator	Y The creator is the user who does the copy
resource_Title	Y

A vocabulary has concepts, they will be copied in the vocabulary copy.

Property	***Copied? Y/N***
conceptURI	Y The new concept will have a new URI
inScheme	Y The new concept will be associated with new vocabulary URI
prefLabel	Y
altLabel	Y
broader	Y The new concept URIs will be associated
narrower	Y The new concept URIs will be associated
belongsToSharedSpace	Y This property will be created to link the SSp where the object is copied
dc_created	Y The date is the date of copy
dc_creator	Y The creator is the user who does the copy
resource_Title	Y

3.2.2.4 Content Items

- Uploadable File (UploadableContentItem)
- Web link (ExternalContentItem)
- Note
- Sketch
- Scorm/IMS file
- Wiki page

Property	***Copied? Y/N***
contentItemURI	Y The new content item will have a new URI
contentURI	Y A new contentURI will be created for each

	physical file in the content repository. For Web Links the property will be copied as it is
dc_format	Y
dc_hasVersion	N
belongsToSharedSpace	Y This property will be created to link the SSP where the object is copied
dc_alternative	N
dc_created	Y The date is the date of copy
dc_creator	Y The creator is the user who does the copy
dc_description	Y
dc_modified	N
dc_title	Y
hasCommentAnnotation	N
hasOutboundRelationship	Y Only if the links and the objects related to the one to be copied are also selected for copy
hasSemanticTag	Y
hasStatus	N
isProtectedTo	N
refersTo	N

3.3 Collaboration with other tools

3.3.1 Information to be logged into the HPA repository

According to HPA official specification (3) all the actions of creating new items will be logged in the HPA, with the addition, in custom property fields, of a property with key "copied" and value the URI of the original object. The log will be implemented at KPE GUI level.

The table above reports as an example the logging record related to the copy of a task.

ID	Time	Subject ID	Subject Type	Object ID	Object Type	Action	Comment	Properties		Custom Data
								Key	value	
1	Date time	User doing the action	e.g. teacher	Object URI	Task	**creation**		User	Name Surname	
								Title	Tasktitle	
								SSPID	Destination SSP URI	

								copied	URI of the original object	

4 References

1. Dow4.1.1 document: http://www.kp-lab.org/intranet/work-packages/wp1/description-of-work/dow4-1-1/?searchterm=DoW4.1
2. HPA Official page: http://kplab.tuke.sk/trac/wiki/hpa
3. HPA Specification document: http://www.kp-lab.org/intranet/design-teams/wk-shared-space-and-common-tools/awareness/participation-and-history-based-awareness/history_awareness_technical_v02.doc
4. CIS Manual for CIS services: http://trac.kp-lab.org/browser/wp6/ContentItemService/branches/M44/CIS%20Manual.doc

Annex 3. Data Export tool specifications

27490

KP-LAB

Knowledge Practices Laboratory

Integrated Project

Information Society Technologies

DII.8 M42 software specification of end-user applications – Data export tool

Due date of deliverable:

Actual submission date:

Start date of project: 1.2.2006 Duration: 60 Months

Organisation name of lead contractor for this deliverable: Metropolia

Revision [1.0]

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Contributors:	Merja Bauters	Metropolia	merjab@evtek.fi
	Christoph Richter	FH OO	Christoph.Richter@fh-hagenberg.at
	Eva Zoeserl	FH OO	Eva.Zoeserl@fh-hagenberg.at
	Anna Marina Scapolla	DIBE	scapolla@unige.it
	Angela Locoro	DIBE	angela.locoro@unige.it
	Jozef Wagner	TUK	jozef.wagner@gmail.com
	Minna Lakkala	UH	minna.lakkala@helsinki.fi
	Crina Damsa	UU	c.i.damsa@uu.nl
Editor(s):	Anna Marina Scapolla		
Partner(s):	DIBE		
Work Package:	WPII		
Nature of the deliverable:			

Version history

Version	Date	Author(s)	Description
1.0	20.11.2009	A. M. Scapolla, A.Locoro	First release, the document has been produced updating the specifications of the M36 tool
1.1	18.12.2009	Jozef Wagner	Changes related to the movement of services from the HPA into the KAS (Knowledge Analysis Services).

Table of Contents

1 Introduction

1.1 Purpose

This document contains a detailed description of improvements and new functionality of the M42 release of the Data Export tool..

1.2 Scope of the software

Data export tool allows researchers and teachers to extract summary tables of user activities from the KPE for on-line investigation, and to export them for elaborations with statistical analysis packages. It gathers data about activities taking place in the KPE and extracts data from the Awareness Repository, the users database and the Knowledge Repository.
This document specifies the improvements and the extensions that the M36 tool has had on the basis of the feedback from tests of the M36 tool and of the user requirements.

1.3 Roadmap of the tool

The implemented and planned iterations of the Data Export Tool are as follows:

- Iteration 1 (M12-M24): Feasibility study and prototyping. Release v1.0
- Iteration 2 (M25-M36): Improvements and extensions of the tool functionalities.
    - Milestones:
        - M32 Specifications, M33 Prototype v2.0
        - M33 Refined Specifications, M36 Release v2.1.
- Iteration 3 (M37-M42): Extensions of functionalities according to the users' priorities on the requirements collected by the partners and integration of graphical view for "Social Networks Analysis".
    - Milestones: M42 Release v3.0 - This release provides extended functionality, according to the users' requirements and the tests done on the M36 release, improvements of GUIs design and it will add graphical views to the "Social Networks Analysis" function.
- M48 release: this is part of the major production release of KPE. The tool will be further maintained and improved in order to fulfil the requirements of exporting additional data. It will include also the possibility of exporting data in a format compatible with Reference Model.

This document describes specifications of iteration 3.

1.4 What is new in this version?

This version of the tool presents both improvements to the M36 release and a new functionality related to the graphical views of Social Networks .

The improvements origin from the observations of the partners after testing the M36 release (see the user requirements section for references to the documents produced by partners in charge of testing and evaluation of the M36 release).
Main improvements cover:

- Overall design of GUI
- Usability of the tool
- Optimization of queries and a consequent better response time
- Naming issues
- Formatting the exported data
- Adding new types of data. .

1.5 Definitions

Analysis of items by user: this analysis results in a table reporting the SSp users (one per row) and the numbers of items created by each user. The items are counted and grouped by object type (e.g. Files, Wikis, Notes, Comments, Links, SCORMs, Tasks and so on).

Content items analysis –The process of searching all the content items of a shared space, and counting the actions they have been subjected to. This analysis results in a table where each row represents a Content Item and its properties: creator name, title, type, creation date and time, number of comment threads, of comments, of links - inbound and outbound, of tags, of responsible persons.

Tasks analysis - this analysis results in a table where each row represents a Task and its properties: Title, Creator, number of SubTasks, Subtask Title, Parent Task Title, Level of the task, Creation date and time, Start Date, End Date, number of Inbound Links, of Outbound Links, of Semantic Tags, of Comments threads, of Sub Comments, total number of Comments, of Contents, of responsible persons and of Milestone

Social network analysis – The process of searching the interrelationships that have been established among the users of a shared space in operating on other users' contents. The output of this process is a square matrix summarizing the numbers of actions carried out.

User action log analysis – The process of searching the action log and reporting::

- a table summarizing the number of actions per user and per type (create, modify, delete,...) with the possibility to filter a specific time interval.
- a list of the actions carried out in a certain time interval by one specific user or by all users.

History/Participation Awareness (**HPA)** - HPA is a repository of events representing activities or changes performed by users of various end-user tools in KP-environment. Logs of these events provide source data for analysis and identification of new knowledge that can be used for analysis and improvement of on-going activities as well as for adaptation of the graphical user interface or for personalization

Knowledge Analysis Services (KAS) - KAS provides the necessary middleware functionality for KPE end user tools which require a rich set of services for selecting and aggregating proper data from underlying log repositories, as well as for defining external events, for commenting and semantically annotating all types of events, and for patterns description and identification.

Knowledge practices environment (KPE) – KPE is the application that is used to log in to/out of the KP-Lab system, to use shared spaces with the integrated tools, and to launch other loosely integrated KP-Lab tools, such as the Activity System Design Tool. The KPE covers all the (end-user) tools under development by Working Knots 1-5 (cp. DoW3.1, p. 27).

Shared space (SSp) – SSp is a virtual area that may contain tools and objects. An SSp may have one or more members. The number of different SSps is practically unlimited.

Trialogical Learning Ontology (TLO) - The core domain ontology that provides common semantics for applications and tools and defines concepts and notions of trialogical learning.

2 User requirements

2.1 KP-Lab empirical research

The tool functionalities specified in this document will be used in the following KP-Lab empirical research cases [4]:

Case 1: Trialogical work in higher education; creation and use of knowledge objects in knowledge creation practices

Case 2: Knowledge creation practices in customer projects in higher education

Case 3: Conceptual modelling in Design Projects

Case 4: Producing document management solutions and practices for distributed work settings – Perspectives of KPE usage in two work organizations

The development of the M42 version of the tool was driven mainly by the feedback of users after testing and evaluating the M36 release and their new requests. Three documents are available in the project Plone:

Usability Report UU 03-2009 Expert Evaluation
Usability Report UH 05-2009, Diagnostic Evaluation
Usability Report FH-OOE-09 2009 Expert evaluation

2.2 Driving Objectives and High-Level Requirements

The Driving objectives (DO) relevant for the tool are the same listed in the D6.6 M33 tool specifications.

2.3 System usage scenarios

These specifications are based on the tool usage scenarios presented in ([1]):

1. US-DataExport-1-2008-11: Retrieving data of user activity in a shared space
2. US-DataExport-2-2008-11: Retrieving data of content items
3. US-DataExport-3-2008-11: Retrieving data of tasks
4. US-DataExport-4-2008-11: Social networks content item analysis and task analysis
5. US-DataExport-5-2008-11: Retrieving "User Action Log" summary for analysis
6. US-DataExport-6-2008-11: Retrieving "User Action Log" of detailed analysis of users actions

2.4 Use cases

This section presents how the M42 data export tool can be used.

2.4.1 Actors

Users of the Knowledge Practices Environment tool (i.e. teachers, assistants, course administrator) who have entered a shared space.

2.4.2 Detailed Use Case

Description:
The main steps in the usage of the tool are:

- The user enter a Shared Space and selects the “Open Data Export” from the “Tool” drop down menu of the upper menu.
- The Data Export tool home page opens in a new browser window and the user selects the type of analysis s/he wants.
- The system runs the data export application on the basis of the selection.
- The tool visualizes tables containing the results of the analysis and in case a “Social network analysis” has been chosen, a new tab is opened to visualize the graph of the network.
- The user can select all or specific columns of data for exporting them Excel format.

Pre-conditions:
The user is within the Knowledge Practices Environment and has entered a shared space.

Post-conditions:
The Data Export application opens a new browser window and the user can go on selecting the type of analysis, running it, looking at results and exporting data.

[1] “M36 System usage scenarios for the Data Export analysis tool http://www.kp-lab.org/intranet/design-teams/wk-shared-space-and-common-tools/data-export-for-analysis/System%20Usage%20ScenarioDataExport0_2.doc/view “

Screen mock-ups:
Figure 1 shows the Data Export home page where the user can start analysing SSP items. The home page presents two tabs. The Shared Space Items' Analysis Tab and the User Action Log Tab. Section 2.4.2.1 describes the Shared Space Items' Analysis functionality and the output resulting from this analysis, section 2.4.2.2 deals with the User Action Log Tab.

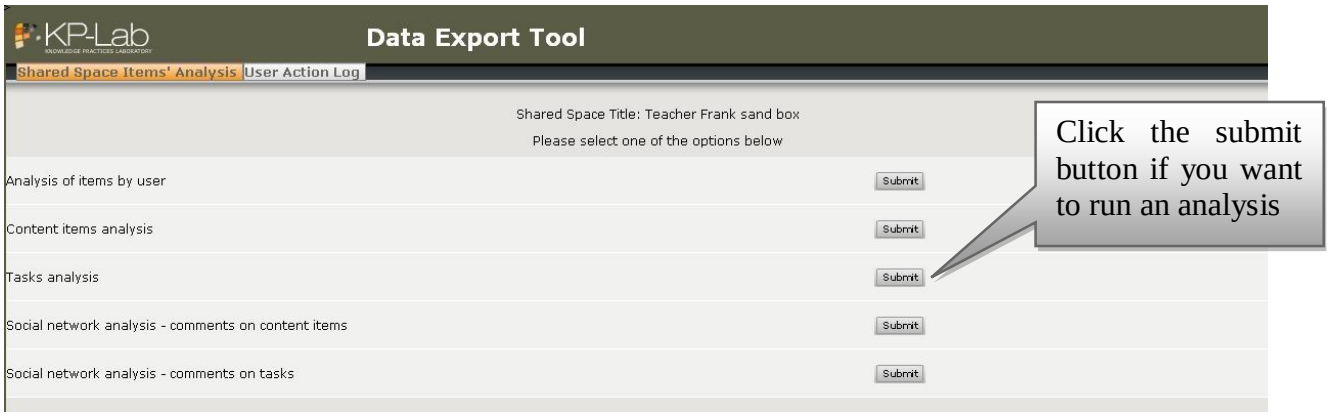


Figure 1 – The Data Export home page

2.4.2.1 Shared Space Items' Analysis Tab

Five different analyses are available. Once the user has made the choice, the request is carried out and the analysis results are presented in a new tab named "**Summary table**", This tab, when present, contains always the results of the last analysis done.

1) **Analysis of items by user**: this analysis results in a table reporting the SSp users (one per row) and the numbers of items created by each user. The items are counted and grouped by object type (e.g. Files, Wikis, Notes, Comments, Links, SCORMs, Tasks and so on).). The table is ordered by username. An example of results table is depicted in Figure 2, which contains also help for using the buttons of data selection and data export in Excel format.

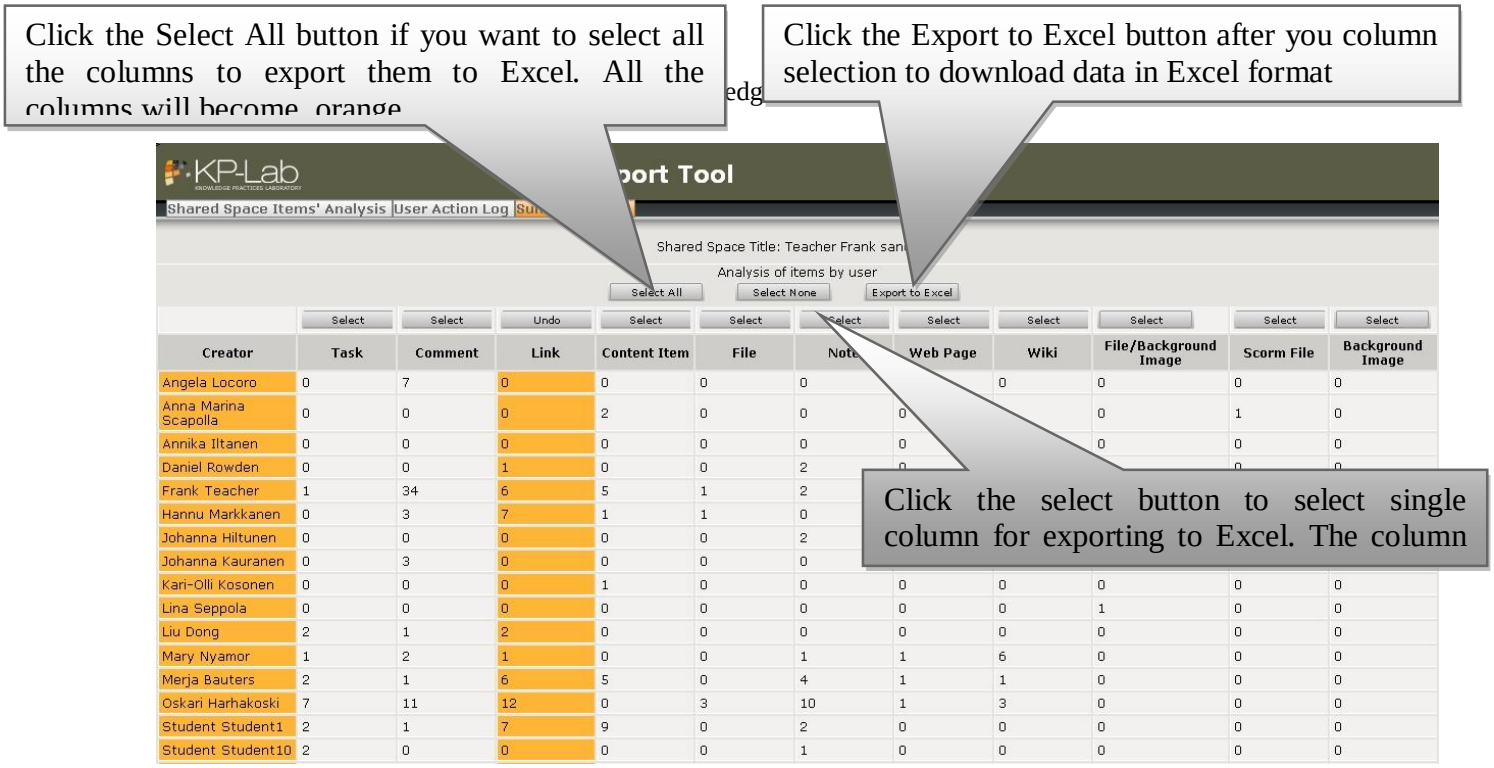


Figure 2 - Table of results for Analysis of items by user

2) **Content items analysis**: this analysis results in a table where each row represents a Content Item and its properties: creator name, title, type, creation date and time, number of comment threads, of comments, of links - inbound and outbound, of tags, of responsible persons (see Figure 3). The table is ordered by Item type and by Item title. Buttons for selecting columns and export data to Excel are the same for all the analysis.

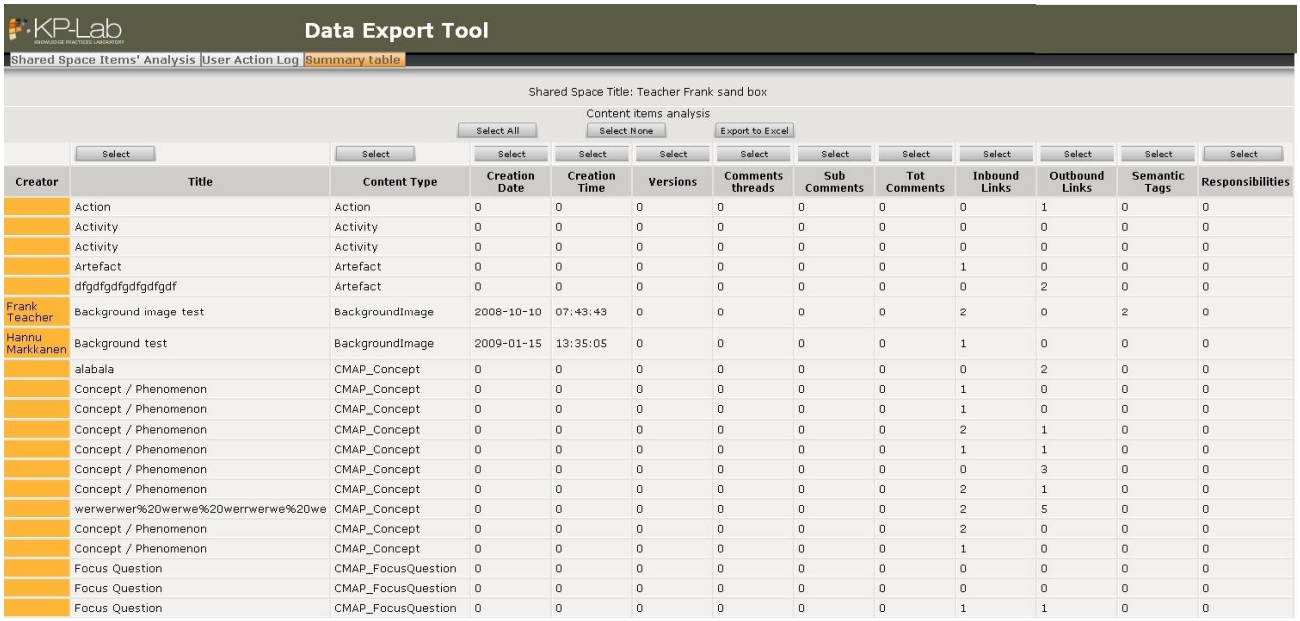
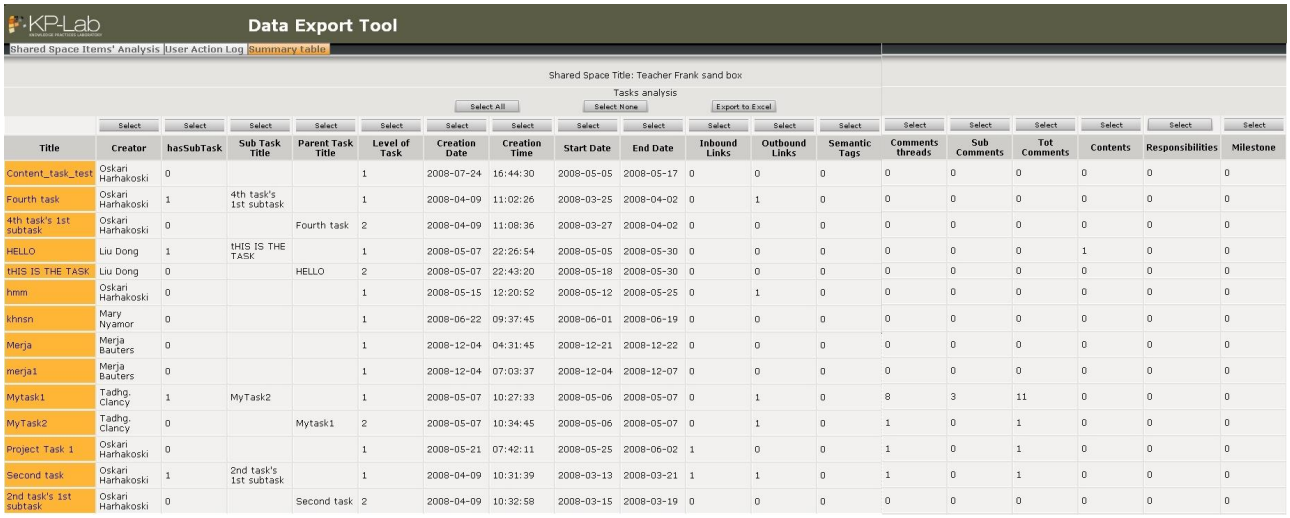


Figure 3 – Table of results for Analysis on Content Items. In the figure some Creator fields are null. This is due to inconsistencies in the Knowledge repository that should be fixed by the last software release.

3) **Task analysis**: this analysis results in a table where each row represents a Task and its properties: Title, Creator, number of SubTasks, Sub Task Title, Parent Task Title,

Level of the task, Creation date and time, Start Date, End Date, number of Inbound Links, of Outbound Links, of Semantic Tags, of Comments threads, of SubComments, total number of Comments, of Contents, of responsible persons and of Milestone (see Figure 4).). The table is ordered by Task title and grouped by Task /SubTask.


KP-Lab

Data Export Tool

Shared Space Items' Analysis | User Action Log | Summary table

Shared Space Title: Teacher Frank sand box

Tasks analysis

Select All | Select None | Export to Excel

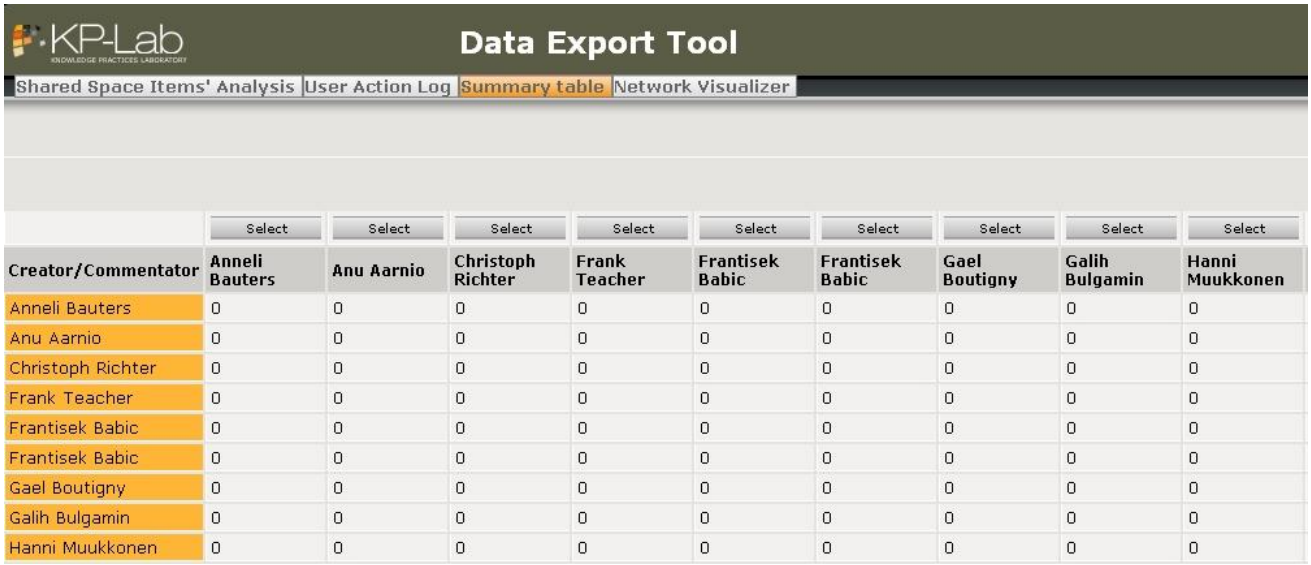
Title	Creator	hasSubTask	Sub Task Title	Parent Task Title	Level of Task	Creation Date	Creation Time	Start Date	End Date	Inbound Links	Outbound Links	Semantic Tags	Comments threads	Sub Comments	Tot Comments	Contents	Responsibilities	Milestone
Content_task_test	Oskari Harhakoski	0			1	2008-07-24	16:44:30	2008-05-05	2008-05-17	0	0	0	0	0	0	0	0	0
Fourth task	Oskari Harhakoski	1	4th task's 1st subtask		1	2008-04-09	11:02:26	2008-03-25	2008-04-02	0	1	0	0	0	0	0	0	0
4th task's 1st subtask	Oskari Harhakoski	0		Fourth task	2	2008-04-09	11:08:36	2008-03-27	2008-04-02	0	0	0	0	0	0	0	0	0
HELLO	Liu Dong	1	tHIS IS THE TASK		1	2008-05-07	22:26:54	2008-05-05	2008-05-30	0	0	0	0	0	0	1	0	0
tHIS IS THE TASK	Liu Dong	0		HELLO	2	2008-05-07	22:43:20	2008-05-18	2008-05-30	0	0	0	0	0	0	0	0	0
hmm	Oskari Harhakoski	0			1	2008-05-15	12:20:52	2008-05-12	2008-05-25	0	1	0	0	0	0	0	0	0
khnsn	Mary Nyamor	0			1	2008-06-22	09:37:45	2008-06-01	2008-06-19	0	0	0	0	0	0	0	0	0
Merja	Merja Bauters	0			1	2008-12-04	04:31:45	2008-12-21	2008-12-22	0	0	0	0	0	0	0	0	0
merja1	Merja Bauters	0			1	2008-12-04	07:03:37	2008-12-04	2008-12-07	0	0	0	0	0	0	0	0	0
Mytask1	Tadhg. Clancy	1	MyTask2		1	2008-05-07	10:27:33	2008-05-06	2008-05-07	0	1	0	8	3	11	0	0	0
MyTask2	Tadhg. Clancy	0		Mytask1	2	2008-05-07	10:34:45	2008-05-06	2008-05-07	0	1	0	1	0	1	0	0	0
Project Task 1	Oskari Harhakoski	0			1	2008-05-21	07:42:11	2008-05-25	2008-06-02	1	0	0	1	0	1	0	0	0
Second task	Oskari Harhakoski	1	2nd task's 1st subtask		1	2008-04-09	10:31:39	2008-03-13	2008-03-21	1	1	0	1	0	1	0	0	0
2nd task's 1st subtask	Oskari Harhakoski	0		Second task	2	2008-04-09	10:32:58	2008-03-15	2008-03-19	0	0	0	0	0	0	0	0	0

Figure 4 – Table of results for Tasks Analysis.

4) **Social Network analysis** – **comments on content items**: this analysis results in a square matrix where each row is relative to a creator of Content Items, and the columns report how many comments have been done by the SSP users on the content items created by the user,

5) **Social Network analysis** – **comments on tasks**: this analysis results in a square matrix where each row is relative to a creator of Tasks, and the columns report how many comments have been done by the SSP users on the task created by the user,

Social Network analysis show these tables in the "Summary table" tab, moreover a **new tab,** named **Network Visualizer**, is created. This tab contains a graphical presentation of the table. An example of a summary table resulting from Social Network Analysis is depicted in Figure 5. The Social Network Visualization is depicted in Figure 6.



	Select	Select	Select	Select	Select	Select	Select	Select	Select
Creator/Commentator	**Anneli Bauters**	**Anu Aarnio**	**Christoph Richter**	**Frank Teacher**	**Frantisek Babic**	**Frantisek Babic**	**Gael Boutigny**	**Galih Bulgamin**	**Hanni Muukkonen**
Anneli Bauters	0	0	0	0	0	0	0	0	0
Anu Aarnio	0	0	0	0	0	0	0	0	0
Christoph Richter	0	0	0	0	0	0	0	0	0
Frank Teacher	0	0	0	0	0	0	0	0	0
Frantisek Babic	0	0	0	0	0	0	0	0	0
Frantisek Babic	0	0	0	0	0	0	0	0	0
Gael Boutigny	0	0	0	0	0	0	0	0	0
Galih Bulgamin	0	0	0	0	0	0	0	0	0
Hanni Muukkonen	0	0	0	0	0	0	0	0	0

Figure 5 – Table results for Social Network. As the matrix is very large (it contains all the SSP users) a horizontal scrolling has to be done to see the analysis title and access Select All and Export to Excel buttons

The **Network Visualizer** depicts a graph where nodes are users and labelled links represent the existing connections between users and how many comments have been done. Pointing over the links a tooltip will come up with a description of who did comments to whom and how many.

With the **left Panel** you can zoom the graph, change the nodes orientation, and customize the links. Click the “Close Panel” button if you want the entire window for visualizing the graph.. At the bottom of the left Panel there is a legend of symbols: bidirectional links are black, unidirectional links are yellow, labelled links depicts the number of comments while a number between parenthesis beside the user yellow name shows self comments (e.g. comments that a user did on its Content Items) if there are such comments.

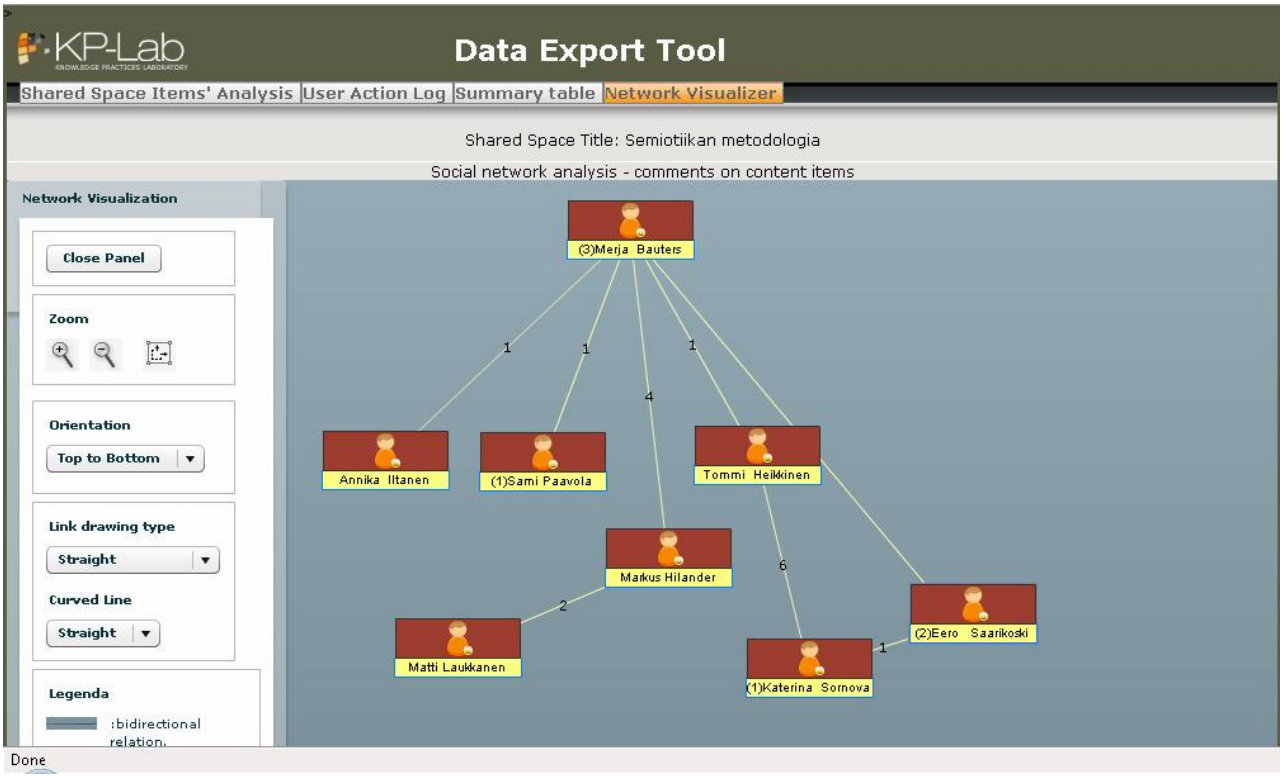


Figure 6– An example of the Network Visualizer GUI

2.4.2.2 User Action Log Tab

The **User Action Log** allows to query the SSP logs (HPA archives) and to retrieve the actions that have been done on objects belonging to the shared space. The user interface relative to this analysis is depicted in Figure 7.

There are two types of analysis:

1) **Summary**: this analysis results in a table where each row represents a SSP user and for each type of action, the sum actions logged for that user are reported (Figure 8).

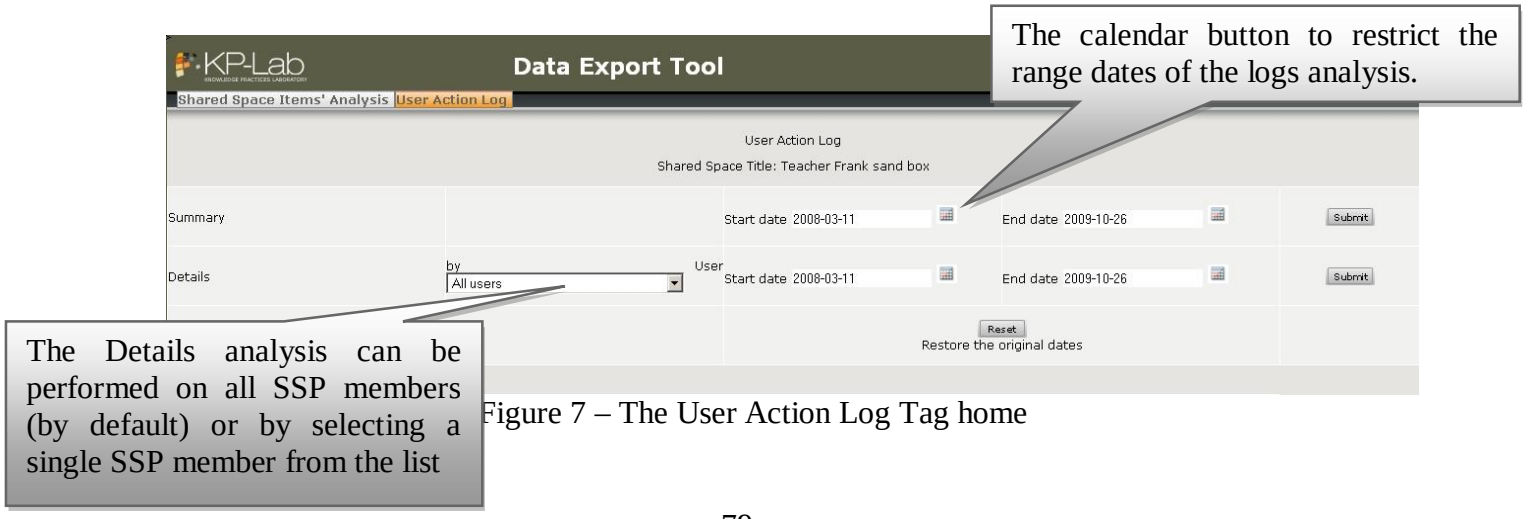


Figure 7 – The User Action Log Tag home

2) **Details:** this analysis results in a table where each row represents a single log record, an action performed by the user and logged into the awareness repository. The table is cronological ordered by date of logging. The Details analysis extracts by default all the SSP members' actions. If you want to export only the ations of a single SSP member, you can select a single member before running the query (Figure 9).

For both analysis Start and End dates are by default the SSP creation date and the today date respectively. If a user wants to change (restrict) the date range he/she should use the calendar button beside each date.

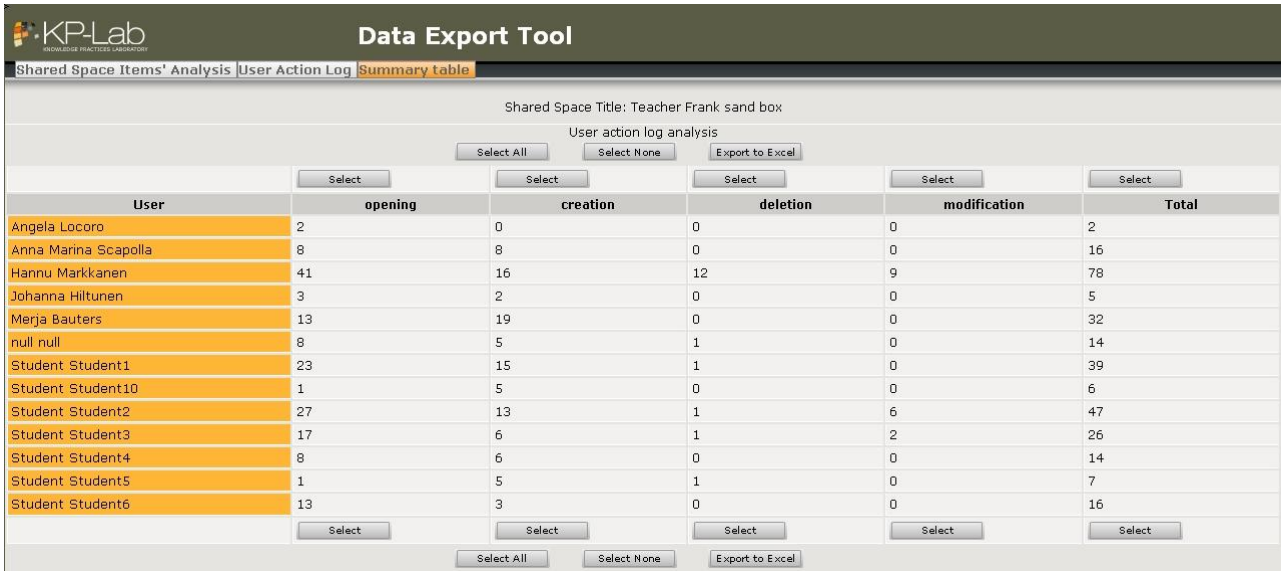
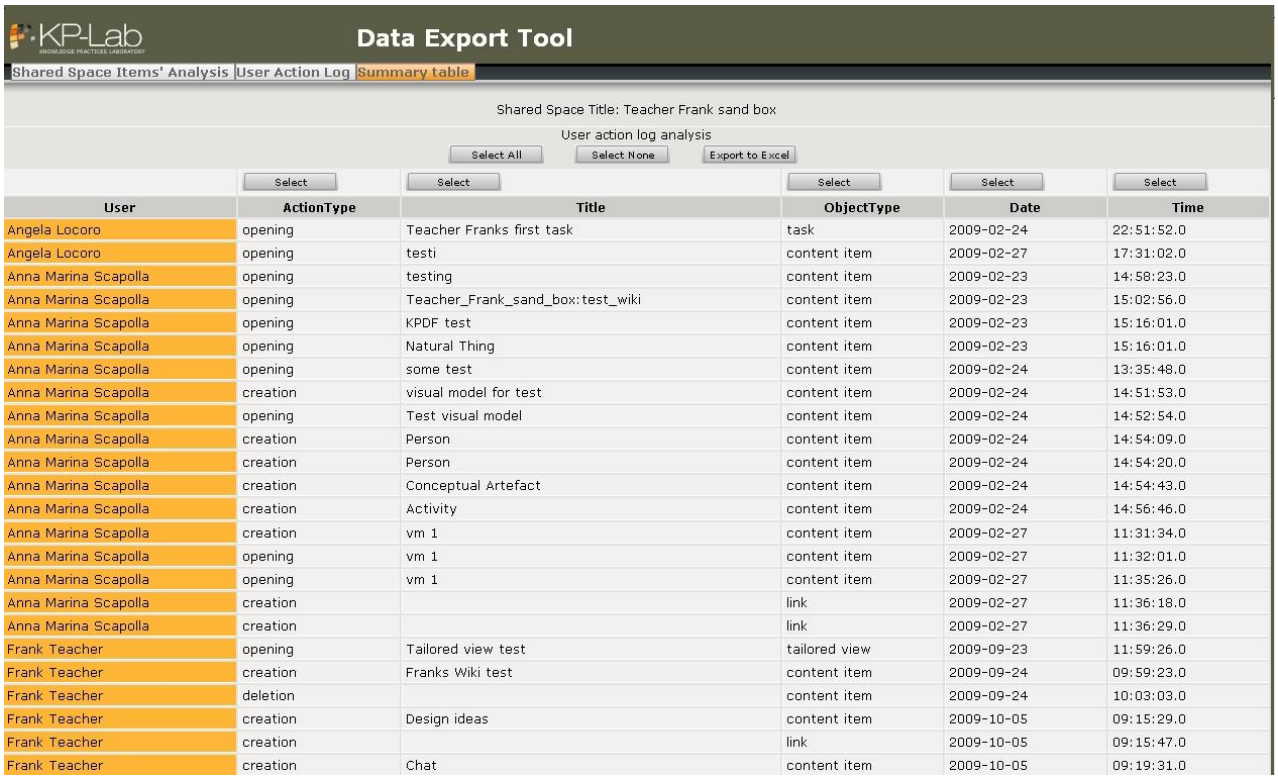


Figure 8 – The Summary analysis results of User Action Log



User	ActionType	Title	ObjectType	Date	Time
Angela Locoro	opening	Teacher Franks first task	task	2009-02-24	22:51:52.0
Angela Locoro	opening	testi	content item	2009-02-27	17:31:02.0
Anna Marina Scapolla	opening	testing	content item	2009-02-23	14:58:23.0
Anna Marina Scapolla	opening	Teacher_Frank_sand_box:test_wiki	content item	2009-02-23	15:02:56.0
Anna Marina Scapolla	opening	KPDF test	content item	2009-02-23	15:16:01.0
Anna Marina Scapolla	opening	Natural Thing	content item	2009-02-23	15:16:01.0
Anna Marina Scapolla	opening	some test	content item	2009-02-24	13:35:48.0
Anna Marina Scapolla	creation	visual model for test	content item	2009-02-24	14:51:53.0
Anna Marina Scapolla	opening	Test visual model	content item	2009-02-24	14:52:54.0
Anna Marina Scapolla	creation	Person	content item	2009-02-24	14:54:09.0
Anna Marina Scapolla	creation	Person	content item	2009-02-24	14:54:20.0
Anna Marina Scapolla	creation	Conceptual Artefact	content item	2009-02-24	14:54:43.0
Anna Marina Scapolla	creation	Activity	content item	2009-02-24	14:56:46.0
Anna Marina Scapolla	creation	vm 1	content item	2009-02-27	11:31:34.0
Anna Marina Scapolla	opening	vm 1	content item	2009-02-27	11:32:01.0
Anna Marina Scapolla	opening	vm 1	content item	2009-02-27	11:35:26.0
Anna Marina Scapolla	creation		link	2009-02-27	11:36:18.0
Anna Marina Scapolla	creation		link	2009-02-27	11:36:29.0
Frank Teacher	opening	Tailored view test	tailored view	2009-09-23	11:59:26.0
Frank Teacher	creation	Franks Wiki test	content item	2009-09-24	09:59:23.0
Frank Teacher	deletion		content item	2009-09-24	10:03:03.0
Frank Teacher	creation	Design ideas	content item	2009-10-05	09:15:29.0
Frank Teacher	creation		link	2009-10-05	09:15:47.0
Frank Teacher	creation	Chat	content item	2009-10-05	09:19:31.0

Figure 9 – The Details analysis results of the User Action Log for all SSP members.

3 Technical design

3.1 Software architecture

The overall KPE architecture documentation is available in the project Plone at the URL http://www.kp-lab.org/intranet/work-packages/wp6/result/dii-8-m46-specification-of-end-user-applications/kpe-architecture/folder_listing .

Data Export tool is designed as an independent Web Application, packaged in a Web Archive (WAR), with all its components (Apache POI - Java API To Access Microsoft Format Files to support CVS format and KME-RE-Client for communication with the KR). The creation and management of controls is delegated to proper Java classes (TagSupport classes).

The Network Visualization has been developed integrating the KapLab Data Visualization software (http://lab.kapit.fr/display/visualizer/Visualizer).

The choice of this software has been preceded by the analysis of the available not commercial tools for social networks visualization. The results of this study are available at http://www.kp-lab.org/intranet/design-teams/wk-shared-space-and-common-tools/data-export-for-analysis/SNVizualisationTools.doc/view?searchterm=social%20networks

3.1.1 Services Used

The tool queries the HPA repository through KAS Query service ([2]) and the Knowledge repository through SWKM client library.

3.1.2 Relation to Ontologies

The tool is heavily dependent on the TLO and FOAF ontologies. The TLO ontology defines the properties of the knowledge artefacts and the possible relationships among them. These artefacts and interrelationships are the objects of the data export tool's analysis processes. The FOAF ontology provides users' properties.

3.2 Collaboration with other tools

The tool queries the HPA repository through KAS Query service and the Knowledge Repository.

4 References

1 "M36 System usage scenarios for the Data Export analysis tool http://www.kp-lab.org/intranet/design-teams/wk-shared-space-and-common-tools/data-export-for-analysis/System%20Usage%20ScenarioDataExport0_2.doc/view "

1 D5.6 *(Resubmitted with respect to review 3 report)* - http://www.kp-lab.org/intranet/work-packages/wp5/result/deliverable-5.6/D5.6_update-after-3rdPrjReview_final.doc

3 Data Export M33 specifications
http://www.kp-lab.org/intranet/work-packages/wp6/result/d6-6-m33-specification-of-end-user-applications/shared-space-and-common-tools/data-export-service-gui-dibe/D6.6_Data%20export%201.0.doc/view

4. D3.2 A comprehensive research strategy (an updated and revised version for years 4 and 5). http://www.kp-lab.org/intranet/work-packages/wp1/submission-to-commission-23-9.2009/D3.2-Rev-Research-Strategy_September_submitted.doc

[2] D5.6 *(Resubmitted with respect to review 3 report)* - http://www.kp-lab.org/intranet/work-packages/wp5/result/deliverable-5.6/D5.6_update-after-3rdPrjReview_final.doc

Annex 4. Search tool specifications

27490

KP-LAB

Knowledge Practices Laboratory

Integrated Project

Information Society Technologies

DII.8 M46 specification of end-user applications – Search tool

Due date of deliverable: 30/11/2009
Actual submission date:

Start date of project: 1.2.2006 Duration: 60 Months

Organisation name of lead contractor for this deliverable: Metropolia

Revision [1.0]

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Contributors:

Markus Holi	Metropolia	markus.holi@metropolia.fi
Hannu Markkanen	Metropolia	hannu.markkanen@metropolia.fi
Merja Bauters	Metropolia	merja.bauters@metropolia.fi
Jan Paralic	TUK	jan.paralic@tuke.sk
Peter Bednar	TUK	peter.bednar@tuke.sk
František Babič	TUK	frantisek.babic@tuke.sk

Version history

Version	Date	Author(s)	Description
0.1	22.11.2009	Markus Holi	First draft of the specification

Table of Contents

1 Introduction

1.1 Purpose

This document describes the functionality of the search tool. It consists of use cases, mock-up screens, data model, and describes the required KP-LAB services.

1.2 Scope of the software

The purpose of the software is to enable advanced faceted search based on the metadata and content of items.

1.3 Roadmap of the tool

During the DoW4 one stable release will be provided as follows:

M48 stable release

- This release will add to the basic free-term and faceted semantic search features the possibility to save search results
- SWKM services used: Query, Update.
- Milestones: M46: Specification, M48: Release.

1.4 What is new in this version?

This version adds the saving of search criteria along with search results.

1.5 Definitions

Free-term search – A search that is performed by using variations and extensions of string matching to satisfy the information need specified by the user. Usually the user writes the search terms freely in a text box. This search can be directed towards free-text content, i.e. full-text of the document or metadata fields that are defined as a free text, or towards semantic content such as semantic tags.

Semantic search – A search which uses the structure of an ontology/ontologies and ontology-based metadata to fulfil the query specified by the user. This means using the relations between items, the possible class-hierarchies etc. Usually the user selects the query concepts from predefined options.

Faceted search – A search paradigm in which the searched items are indexed using multiple orthogonal dimensions or facets. The user can then search the items using these multiple facets simultaneously.

Facet – A single dimension by which the user can search the item database. An example of a facet is "Creation date" which provides search based on the creation dates of the searched items.

Search category – A single selectable value within a facet, for example "01.01.2007" as a search category within the "Creation date" –facet. This selection will retrieve items that are created on the 1st of January, 2007.

Knowledge practices environment (KPE) – KPE is the application that is used to log in to/out of the KP-Lab system, to use shared spaces with the integrated tools, and to launch other loosely integrated KP-Lab tools, such as the Activity System Design Tool. The KPE covers all the (end-user) tools under development by Working Knots 1-5 (cp. DoW3.1, p. 27).

2 User requirements

2.1 Relationship with KP-Lab research studies

The tool functionalities specified in this document will be used in the following KP-Lab empirical research cases [2]:

Case 1: Trialogical work in higher education; creation and use of knowledge objects in knowledge creation practices

Case 2: Knowledge creation practices in customer projects in higher education

Case 4: Producing document management solutions and practices for distributed work settings – Perspectives of KPE usage in two work organizations

2.2 Driving Objectives and High-Level Requirements

The Driving objectives (DO) relevant for the tool are:

DO1: Provide a collaborative environment where users can work on shared artefacts
DO 4: Users can describe the semantics of artefacts and their relations
DO 8: Enable users to plan, organize and manage tasks collaboratively

The high-level requirements (HLR) relevant for the tool are:

HLR1.4 (DO 1)	Users can search artefacts within and outside the shared environment using full text, metadata or domain ontologies.
HLR8.6 (DO 8)	Users can search the content and metadata using full text or semantic metadata search for planning and reflecting on activities.
HLR4.1 (DO 4)	Users can categorise, classify and cluster artefacts in different manners.

2.3 System usage scenarios

The tool specification is based on the following system usage scenario:
1. SUS-Search-1-2009-11: Saving search criteria and results.

2.4 Use cases

This section presents the use cases for the search tool.

2.4.1 Actors

User of the Knowledge Practices Environment

2.4.2 Use cases

1. UC-Search-01-2009-11: User saves search criteria and results.
2. UC-Search-02-2009-11: User opens the saved search object (content item organizer).

2.4.3 Detailed Use Cases

Title: User saves search criteria and results.

Unique identifier: *UC-Search-01-2009-11*

- Related usage scenarios: SUS-Search-1-2009-11: Saving search criteria and results.
- Actors: User of the Knowledge Practices Environment.
- Description: The user has entered a shared space, and selected 'Search' from the upper menu bar. The Search GUI has opened as a Flex-popup and the user has performed a search and received search results. The user decides to save the search criteria and the results to be used later. The user presses the save search button located at the top of the search result list. The search results and the criteria are saved to SWKM as Content Item Organizer objects where the search results are the content items, and the search criteria as saved as the description of the organizer. The search tool informs the user by an alert box saying "search saved successfully".
- Pre-conditions: The user has entered a shared space, and selected 'Search' from the upper menu bar. The Search GUI has opened as a Flex-popup and the user has performed a search and received search results.
- Post-conditions: The search results and the criteria are saved to SWKM as Content Item Organizer objects where the search results are the content items, and the search criteria as saved as the description of the organizer. The search tool informs the user by an alert box saying "search saved successfully". The content item organizer is displayed in the content view of that shared space.
- Nominal scenario:
    - The user presses the save search button located at the top of the search result list.
    - The search results and the criteria are saved to SWKM as Content Item Organizer objects where the search results are the content items, and the search criteria as saved as the description of the organizer. The content item organizer belongs to the shared space from where the search tool was

invoked. The content item organizer is displayed in the content view of that shared space.

- The search tool informs the user by an alert box saying “search saved successfully”.

- Non-functional requirements:
  - saving is reliable
- Screens: Figure 1 represents the saved search object (content item organizer) in content view.

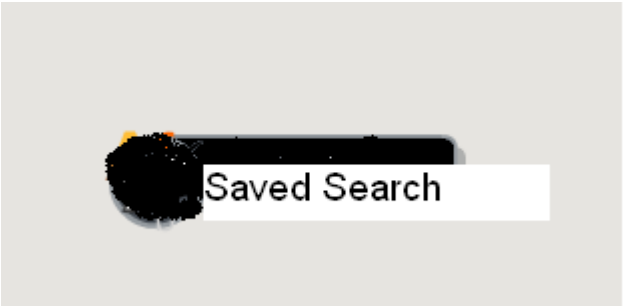


Figure 1: The saved search object as shown in content view.

Title: User opens the saved search object (content item organizer)

Unique identifier: *UC-Search-02-2009-11*

- Related usage scenarios: SUS-Search-1-2009-11: Saving search criteria and results.
- Actors: User of the Knowledge Practices Environment.
- Description: The user has entered the shared space in KPE where a saved search exists as a content item organizer. The user double clicks the content item organizer. The saved search results are displayed as a list which is similar to the list of search results in search GUI. The search criteria are displayed at the top of the list. By pressing the “Perform Search” button near the search criteria, the search GUI is opened and a search is performed using the saved criteria.
- Pre-conditions: The user has entered the shared space in KPE where a saved search exists as a content item organizer.
- Post-conditions:
  - The saved search results are displayed as a list which is similar to the list of search results in search GUI.
  - The search criteria are displayed at the top of the list.
  - By pressing the “Perform Search” button near the search criteria, the search GUI is opened and a search is performed using the saved criteria.
- Nominal scenario:

  - The user double clicks a content item organizer which represents a saved search.
  - The saved search results are displayed as a list which is similar to the list of search results in search GUI. The search criteria are displayed at the top of the list.
  - By pressing the "Perform Search" button near the search criteria, the search GUI is opened and a search is performed using the saved criteria.

- Non-functional requirements:
  - The system should be able to perform each search within sensible time limits.
- Screens: Figure 2 depicts an opened search object. The list of saved search result is displayed and the saved search results criteria on top. The user can perform a search with the saved criteria by pressing "Search".

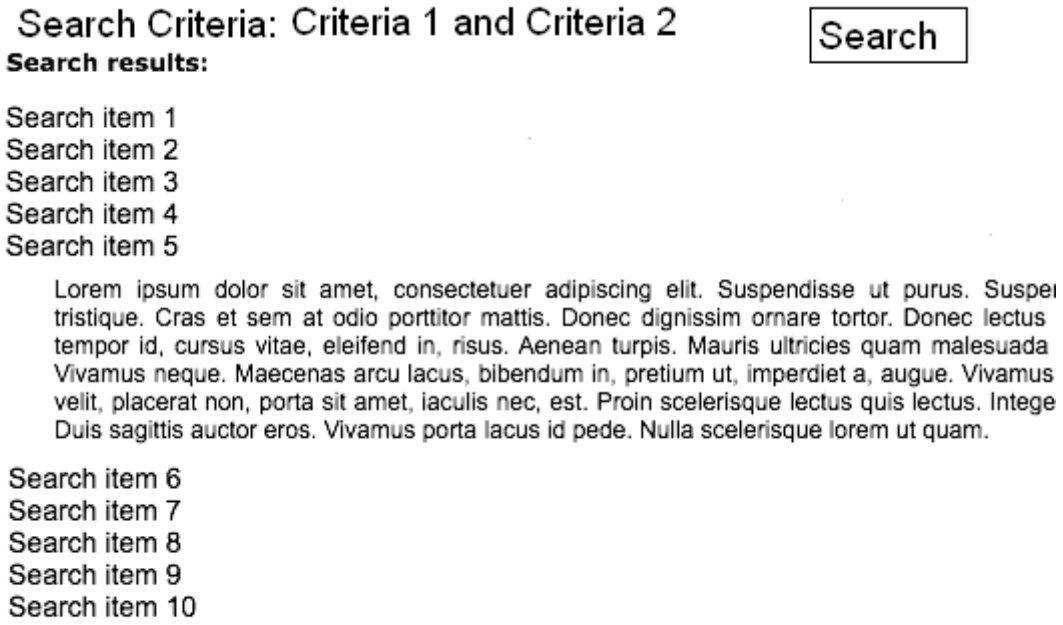


Figure 2: The opened search object.

3 Technical design

3.1 Software architecture

The overall KPE architecture documentation is available in the project Plone at the URL http://www.kp-lab.org/intranet/work-packages/wp6/result/dii-8-m46-specification-of-end-user-applications/kpe-architecture/folder_listing .

Figure 10 presents the overall architecture of the search functionality. The Search tool is one of the KPE Tools and it is invoked from the Knowledge Practices Environment GUI's upper menu bar. The Search GUI uses the Search Service for populating the facets, and retrieving the result items for searches specified by the user. The search service is based on the Solr search engine (1). The properties relevant to search are indexed by the front-end services. Indexing service is integrated with the Persistence API (P-API) and Gateways to Content Repositories (G2CR) to simplify maintenance of the index for tool developers. In the case that Persistence API is not used to manage objects of activities, it is possible to invoke Indexing service directly using the *Solr* HTTP interface

The search service is called from GUI by the Data access services. The Data access services create the HTTP-request to the Search Service based on the user selections in the Search GUI.

The Search Service contains a function saveSearch which uses P-API to create the content item organizer object to be used as the object that represents the saved search.

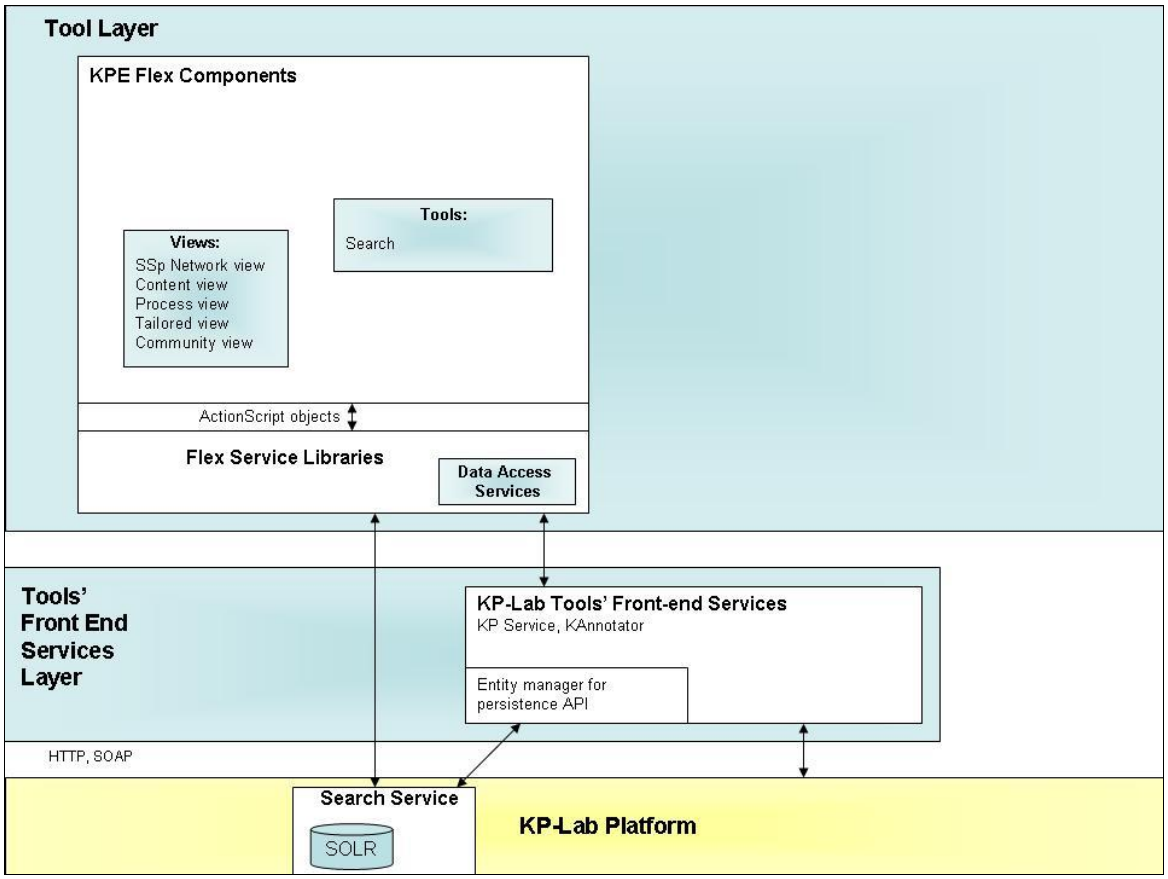


Figure 10: Overall architecture of search.

3.2 GUI

3.2.1 Screen Design

Figures 1 and 2 present the screen design of the Search GUI.

Figure 1 represents the saved search object (content item organizer) in content view.

Figure 2 depicts an opened search object. The list of saved search result is displayed and the saved search results criteria on top. The user can perform a search with the saved criteria by pressing “Search”.

3.2.2 Services Used

- Search service - The main service supporting the search tool. It contains the saveSearch function which is used to save the search. It is called with 4 parameters:
  - userURI of the savior
  - sharedSpaceURI where the search is saved
  - the search criteria as a string
  - the list of item URIs of the search objects.

  Based on these parameters the service creates a content item organizer object and saves in SWKM. The service returns “success” on success.

3.2.3 Relation to Ontologies

The saving of search criteria and results creates and instance of Content item organizer class. This class is defined in the TLO.

3.3 Front-end services

3.3.1 Front-end Service Interfaces

The Search GUI does not have its own front-end service. Instead it uses the search service which is one of the KP-Lab platform services (see 3.3.3).

3.3.2 Used Platform services

The tool uses the search service as described in section 3.2.2 of this document.

3.3.3 Search-specific Data Persistence

The search is saved in SWKM using P-API.

3.4 Collaboration with other tools

The overall KPE architecture documentation is available in the project Plone at the URL http://www.kp-lab.org/intranet/work-packages/wp6/result/dii-8-m46-specification-of-end-user-applications/kpe-architecture/folder_listing .

presents the various components and services that collaborate to provide the desired search functionalities.

The search GUI functions are dependent on KPE front-end services (KAnnotator, CIS, KBrowser, KPS) for indexing the properties of searched objects. These front-end services do not provide interface methods specific to search and they are used only indirectly. For these reason their interfaces are not defined here.

The detailed list of properties indexed by the different services can be found in (2).

4 References

1. http://wiki.apache.org/solr/

2. Search Services - Indexed Properties, http://kplab.evtek.fi:8080/wiki/Wiki.jsp?page=IndexedPropertiesForSearch

Annex 5. Time-line based analyser (TLBA) specifications

27490

KP-LAB

Knowledge Practices Laboratory

Integrated Project

Information Society Technologies

DII.8 M46 specification of end-user applications – Timeline Based Analyzer (TLBA)

Due date of deliverable: 30/11/2009
Actual submission date:

Start date of project: 1.2.2006 Duration: 60 Months

Organisation name of lead contractor for this deliverable: EVTEK

Revision [1.0]

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission	
RE	Restricted to a group specified by the consortium (including the	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Contributors:

Name	Partner organisation	Email
Michal Raček	Pöyry	michal.racek@poyry.com
Frantisek Babič	TUK	frantisek.babic@tuke.sk
Jozef Wagner	TUK	jozef.wagner@gmail.com
Jan Paralič	TUK	jan.paralic@tuke.sk
Ali Rantakari	Pöyry	ali.rantakari@poyry.com
Hannu Markkanen	Metropolia	hannu.markkanen@metropolia.fi

Version history:

Version	Date	Author(s)	Description
0.1	17.11.09	Raček	First entries to the provided template.
0.15	23.11.09	Raček	Finalising first version of TLBA Specification.
0.2	24.11.09	Raček, Babič	Incorporating Frantisek's comments. Updating technical section and SUS list, adding two UC.
0.3	27.11.09	Raček	Making corrections based on Jozef's comments.
0.4	01.12.09	Raček	Modification in roadmap based on Hannu's comments.
0.5	21.12.09	Raček	Corrections in middleware name convention.

Table of Contents

Table of Figures

1 Introduction

1.1 Purpose

This document describes M44 and M48 functional specification of the end-user Timeline Based Analyzer (TLBA) tool which is backed by Knowledge Analysis Services (KAS) a part of KP-Lab technical middleware services targeted for use in KP-Lab analytic tools.

1.2 Scope of the software

Timeline Based Analyser (TLBA) is an Adobe Flex based front-end client application that is backed by Java based service, namely KAS Query service. Communication between TLBA client application and KAS Query service is served by web services via SOAP protocol.

Timeline Based Analyser (TLBA) is a part of M44 and M48 analysis package tools releases [3]. In contrast to Visual Analyser and Data Export tool TLBA brings chronological overview of user actions into the user interface, which enables users to see and explore what kind of activities were performed on certain objects in desired space.

Main goal of TLBA is to chronologically display events that were recorded by the Knowledge Practises Environment (KPE) end-user tools, design and store possible external events which could not have been recorded by the KPE end-user tools and define 'patterns' of actions that could be searched for in historical data.

1.3 Roadmap of the tool

The planned iterations for the tool are as follows:

Iteration 1: First prototype version with basic functionalities (as launching application from KPE, displaying user actions, getting specific information about particular action.). Release 0.1
Milestones: M44 Prototype release

Iteration 2: Stable release including extensions to add external events, the possibility of commenting on user actions, defining of action 'patterns', searching the history of stored actions for defined patterns and export functionalities. Release 1.0

Milestones: M45 Technical specification, M47 Prototype, M48 Production release

Iteration 3: Final version with improvements based on the user trials. Release 1.1
Milestones: M56 Release.

1.4 What is new in this version?

TLBA is a new end-user tool and this specification serves to its first release as of which this is the first version of this application.

1.5 Definitions

History and Participation Awareness service (HPA) is a Java based KP-Lab middleware back-end service which provides its users with several awareness repository functionalities as storing an event log to the repository, retrieving event log from the repository or searching the logs on some criteria basis.

Knowledge Analysis Services (KAS) acts as an umbrella for all necessary middleware services used by (not only) end-user analytic tools, such as Visual Analyser, TLBA and Data Export. Therefore those are services build on top of the HPA and Analysis and Knowledge Mining Services (AKMS). KAS mainly contains of KAS Query services (heavily used by TLBA) and KAS Aggregation services.

Knowledge Practices Environment (KPE) a former Shared Space GUI is an internet rich application which is build on top of KP-Lab services.

2 User requirements

2.1 KP-Lab empirical research

The tool functionalities specified in this document will be used in the following KP-Lab empirical research cases [2]:

Case 1: Trialogical work in higher education; creation and use of knowledge objects in knowledge creation practices.

Case 2: Knowledge creation practices in customer projects in higher education.

Case 3: Conceptual modelling in Design Projects.

Case 4: Producing document management solutions and practices for distributed work settings – Perspectives of KPE usage in two work organizations.

2.2 Driving Objectives and High-Level Requirements

The specifications presented in this document are related to the following driving objectives (DO) and high-level requirements (HLR) [1]:

DO8	Users can plan, organize and manage tasks collaboratively
HLR8.7	Users are provided with a customized analysis of groups' working processes (e.g. identification of typical sequences of actions or interesting rules)
DO9	Users are provided with history on content development and work process advancement
HLR9.1	Users can track the evolution and changes of knowledge objects and find out their authors and contributors (sequences of performed steps in time, incl. versioning)
HLR9.2	Users are provided with customized summaries about the knowledge objects available within the shared environment (e.g. users might request an overview of the tasks completed within the last 2 weeks or the interactions of people within a shared workspace)
DO13	Users can collect data about activities and interactions systematically and over longer periods of time
HLR13.5	Users can customise the way they want to retrieve wanted data for analysis.
HLR13.6	Users are provided with summative information on performed actions (e.g. added comments, created

	tasks, modifications in metadata, background materials for decisions, etc.). (Rephrased)

2.3 System usage scenarios

The specifications in this document are based on system usage scenarios presented in [4]:

- US-TLBA-1-2009-09: Viewing all events on the timeline that have been performed on certain objects in a shared space.
- US-TLBA-2-2009-09: Viewing all events on the timeline that a user has performed on objects in a shared space.
- US-TLBA-3-2009-09: Viewing all events on the timeline that have been performed on tasks as well as on objects that are associated with the selected task.
- US-TLBA-4-2009-09: Exploring the evolution and history of a shared space on the timeline.
- US-TLBA-5-2009-09: Exploring and comparing the evolution and history of different shared spaces on the timeline (M48).
- US-TLBA-6-2009-09: Adding external events to the timeline (M48).
- US-TLBA-7-2009-11: Adding comment to the object's action (M48).
- US-TLBA-8-2009-11: Working with patterns (M48).
- US-TLBA-9-2009-11: Exporting TLBA view (M48).
- US-TLBA-10-2009-11: Storing and loading TLBA settings (M48).

2.4 Use cases

This section presents the use cases for the Timeline Based Analysis tool where actions which has been stored to the HPA awareness repository are being displayed, explored and extended.

2.4.1 Actors

- *Primary actors:* Users of Knowledge Practices Environment (KPE) along with Timeline Based Analysis (TLBA) users e.g. teachers, students and professionals.
- *Secondary actors:* KAS Query service, Knowledge Browser service (KB).

2.4.2 Use Cases

1. User launches TLBA from Shared Space context-menu
2. User launches TLBA from within the KPE tools drop down menu
3. User opens-up the TLBA from within the object's context-menu
4. User opens-up TLBA application from within the task's context-menu
5. User highlights all actions performed by one person from community view
6. User highlights object's evolution path in TLBA
7. User highlights all actions performed by one person in TLBA
8. User explores details of an action

9. User launches KPE content view
10. User reads action's comment (M48)
11. User launches TLBA to compare several Shared Spaces (M48)
12. User adds external event into the timeline (M48)
13. User comments on object action (M48)
14. User defines actions pattern (M48)
15. User loads actions pattern (M48)
16. User searches for the pattern occurrences (M48)
17. User exports current TLBA view (M48)
18. User saves current TLBA settings (M48)
19. User loads TLBA settings (M48)

2.4.3 Use Case Diagram

Following figure (Figure 1) illustrates interactions and relations between TLBA use cases (on figure use cases identifiers are simplified as follows: UC-1 represents UC-TLBA-1-2009-09, UC-2 represents UC-TLBA-2-2009-09, etc.).

Launching TLBA section shows UC-1 up to UC-5 and UC-11 where the description on how the TLBA application can be launched. Those are in direct relation to the User Action section which is divided to the two parts. First one called Passive interaction includes UC-6 till UC-10 where user can interact with TLBA only passively (no data input is required form the user) whereas second one called Active interactions, UC-12 up to UC-16, needs active user input for its functionality (providing TLBA with some sort of data). TLBA export and settings section, UC-17 up to UC-19 can be executed as a follower to the before mentioned sections.

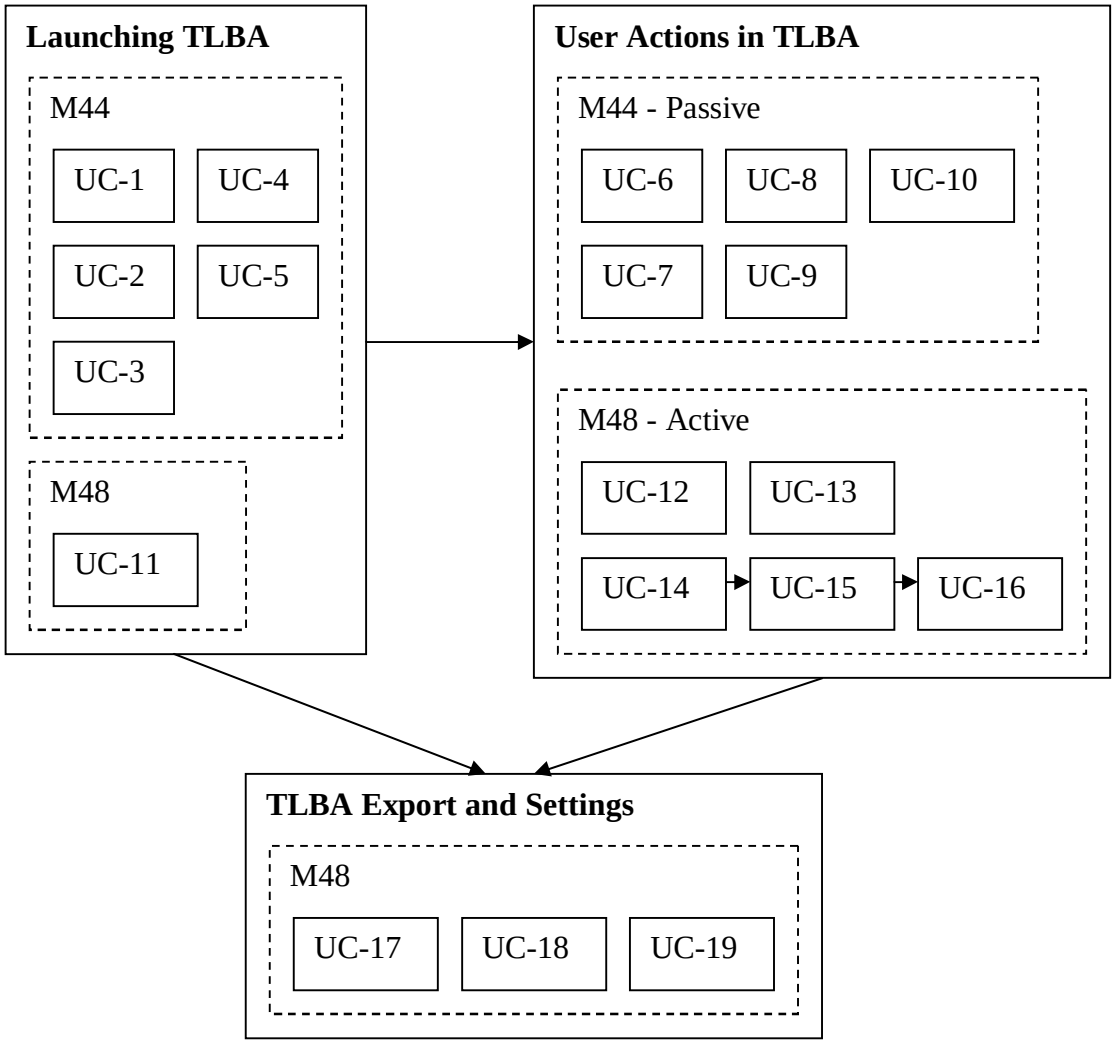


Figure 11: ***TLBA Use Case Diagram***

2.4.4 Detailed Use Cases

2.4.1.1 User launches TLBA from Shared Space context-menu

Unique identifier: ***UC-TLBA-01-2009-11***
Related System Usage Scenarios: US-TLBA-4-2009-09
Actors:

- Primary: Users of TLBA and/or KPE
- Secondary: KAS Query service, KB

Description: User has entered the KPE and brings up the network view. By selecting a Shared Space user brings up its context-menu and chooses 'open in TLBA'. TLBA is launched and all objects with their according chronological actions path are displayed.

Pre-conditions: User is logged in the KPE.
Post-conditions: View from which the TLBA was launched is un-modified. A new browser tab or view is opened with TLBA application.
Nominal scenario:

- User enters the KPE.

- User modifies its view to the network view.
- User brings up context-menu of a Shared Space and selects 'open in TLBA' option.
- TLBA is launched.

Non-Functional Requirements:

- Launching the TLBA should be available only to users that have necessary access rights to do that, i.e. read/view privileges.

2.4.1.2 User launches TLBA from within the KPE tools drop down menu

Unique identifier: ***UC-TLBA-02-2009-11***
Related System Usage Scenarios: US-TLBA-4-2009-09
Actors:

- Primary: Users of TLBA and/or KPE
- Secondary: KAS Query service, KB

Description: User has entered the KPE and navigates to the content, tailored or process view of a Shared Space. User opens up the KPE tools drop down menu and selects the 'Open TLBA' from it. TLBA is launched showing all objects and theirs chronological actions paths. No object is highlighted in TLBA.

Pre-conditions: User is logged into the KPE and has navigated itself to the content, tailored or process view.
Post-conditions: KPE view from where TLBA was called is unmodified, a new browser tab or view is opened with TLBA.
Nominal scenario:

- User enters KPE.
- User navigates itself to the content, tailored or process view of a Shared Space.
- User launches TLBA from within the KPE tools drop down menu.
- TLBA is opened in new browser tab/view.

Non-Functional Requirements:

- Launching the TLBA should be available only to users that have necessary access rights to do that, i.e. read/view privileges.

2.4.1.3 User opens-up the TLBA from within the object's context-menu

Unique identifier: ***UC-TLBA-03-2009-11***
Related System Usage Scenarios: US-TLBA-1-2009-09
Actors:

- Primary: Users of TLBA and/or KPE
- Secondary: KAS Query service, KB

Description: The user has entered the KPE environment and navigated into the Shared Space's content or tailored view. User brings up context-menu of object where the 'view object's timeline' option is chosen. TLBA application is launched and actions which belong to the desired object are highlighted in chronological order. The rest of Shared Space objects with their actions are grayed-out.

Pre-conditions: User is logged into the KPE and operates within content or tailored view.

Post-conditions: View where user operates (KPE) is unmodified. New browser tab is opened where TLBA is launched.
Nominal scenario:

- User enters KPE.
- User selects and enters Shared Space's content or tailored view.
- User selects and brings up context-menu of desired object and selects the 'View object's timeline' option.
- Application opens up new browser tab or view with TLBA application.
- Desired object's chronological actions path is highlighted.

Non-Functional Requirements:

- Viewing the TLBA should be available only to users that have necessary access rights to do that, i.e. read/view privileges.

2.4.1.4 User opens-up TLBA application from within the task's context-menu

Unique identifier: ***UC-TLBA-04-2009-11***
Related System Usage Scenarios: US-TLBA-3-2009-09
Actors:

- Primary: Users of TLBA and/or KPE
- Secondary: KAS Query service, KB

Description: User enters the KPE and navigates to the content or tailored view of a Shared Space. By right-click on a Task object, user brings up the Task's context-menu and selects the 'View task's timeline' option. TLBA is launched and all objects that belong to selected task (in KPE) with their actions are highlighted.

Pre-conditions: User has logged into the KPE and navigated to the content or tailored view of a Shared Space.
Post-conditions: View from which TLBA was launched is un-modified. A new browser tab/view is opened up with TLBA and only task's objects with their chronological actions path are highlighted.
Nominal scenario:

- User logs into KPE.
- User navigates to the content or tailored view of a Shared Space.
- User brings up the context menu of a Task object and selects the 'View task's timeline' option.
- TLBA is launched showing highlighted task's objects chronological actions paths.

Non-Functional Requirements:

- Viewing the TLBA should be available only to users that have necessary access rights to do that, i.e. read/view privileges.

2.4.1.5 User highlights all actions performed by one person from community view

Unique identifier: ***UC-TLBA-05-2009-11***
Related System Usage Scenarios: US-TLBA-2-2009-09
Actors:

- Primary: Users of TLBA and/or KPE

- Secondary: KAS Query service, KB

Description: User enters the KPE application and navigates itself to the community view. By bringing up the context menu of a user, he/she chooses the 'View user's timeline' option. TLBA is launched and all actions that were performed by one person are highlighted.

Pre-conditions: User has logged into KPE and navigated to the community view of a Shared Space.
Post-conditions: View from which TLBA was launched is un-modified. A new browser tab/view is opened up with TLBA. All actions of desired person are highlighted.
Nominal scenario:

- User logs into the KPE.
- User navigates to the community view of a Shared Space.
- User brings up the context-menu of a User object and selects 'View user's timeline' option.
- TLBA is launched highlighting all actions of desired person.

Non-Functional Requirements:

- Viewing the TLBA should be available only to users that have necessary access rights to do that, i.e. read/view privileges.

2.4.1.6 User highlights object's evolution path in TLBA

Unique identifier: ***UC-TLBA-06-2009-11***
Related System Usage Scenarios: US-TLBA-1-2009-09
Actors:

- Primary: Users of TLBA and/or KPE
- Secondary: KAS Query service

Description: The user has entered the TLBA application where Shared Space objects actions are displayed. User selects desired object by mouse's left-click on it. Selected object chronological actions path is highlighted.

Pre-conditions: TLBA application is opened up showing actions of objects in Shared Space.
Post-conditions: Highlighting of actions is done only on desired object chronological actions path.
Nominal scenario:

- User launches TLBA.
- User selects desired object by mouse's left-click.
- Object chronological action path is highlighted.

2.4.1.7 User highlights all actions performed by one person in TLBA

Unique identifier: ***UC-TLBA-07-2009-11***
Related System Usage Scenarios: US-TLBA-2-2009-09
Actors:

- Primary: Users of TLBA and/or KPE
- Secondary: KAS Query service

Description: User enters the TLBA. By left-clicking on a person's name all actions that were performed by this person are highlighted.

Pre-conditions: User has entered TLBA.
Post-conditions: TLBA display is modified only by highlighting all actions that were performed by person selected.
Nominal scenario:

- User launched TLBA.
- User left-clicks on the person name.
- TLBA highlighted all actions of desired person.

2.4.1.8 User explores details of an action

Unique identifier: ***UC-TLBA-08-2009-11***
Related System Usage Scenarios: US-TLBA-4-2009-09
Actors:

- Primary: Users of TLBA
- Secondary: N/A

Description: User has entered TLBA. By right-click on an action a context-menu appears. User selects 'Show details' option. A pop-up is brought up to the user where the detailed information about action can be found. By pressing 'Close' button detailed information pop-up is closed.

Pre-conditions: User launched the TLBA.
Post-conditions: Pop-up is closed; no modification to the TLBA view is made.
Nominal scenario:

- User launches TLBA.
- User brings up the action context-menu by right-click.
- User selects the 'Show details' option.
- Pop-up is shown to the user where detailed information about action can be found.
- By pressing 'Close' button pop-up is closed.

2.4.1.9 User launches KPE content view

Unique identifier: ***UC-TLBA-09-2009-11***
Related System Usage Scenarios: US-TLBA-4-2009-09
Actors:

- Primary: Users of TLBA
- Secondary: N/A

Description: User launched TLBA and wants to see object from timeline in a KPE content view. User navigates itself to the object action and right-clicks on it. User selects 'Open in content view' option.

Pre-conditions: TLBA is launched.

Post-conditions: New browser tab/view is opened where KPE is loaded up in content view. KPE focus is centered on object.
Nominal scenario:

- User launches TLBA.
- User navigates to the object's action and right-clicks on it.
- User selects the 'Open in content view' option from the context-menu.
- KPE content view is launched in new browser tab/view where focus is centred on object.

2.4.1.10 User reads action's comment (M48)

Unique identifier: ***UC-TLBA-10-2009-11***
Related System Usage Scenarios: US-TLBA-7-2009-09
Actors:

- Primary: Users of TLBA
- Secondary: N/A

Description: User has launched TLBA and wants to see comments posted on object's action. User navigates itself to the object's action. User invokes context-menu of action by right-clicking on it. User chooses 'Show comment' option. A pop-up is presented to the user where comment is shown.

Pre-conditions: User launched TLBA, action already has comment stored.
Post-conditions: Comment pop-up is presented to the user.
Nominal scenario:

- User launches TLBA.
- User navigates to the object's action and right-clicks on it.
- User selects the 'Show comment' option from the context-menu.
- Pop-up is presented to the user where commented text is visible.

2.4.1.11 User launches TLBA to compare several Shared Spaces (M48)

Unique identifier: ***UC-TLBA-11-2009-11***
Related System Usage Scenarios: US-TLBA-5-2009-09
Actors:

- Primary: Users of TLBA and/or KPE
- Secondary: KAS Query service, KB

Description: User has entered KPE and navigated to the network view. From KPE tools drop down menu user selects 'Open TLBA' option. A pop-up is brought up to the user where the selection of Shared Spaces that are available to user can be made. By pressing 'Open' button TLBA application is launched or by pressing 'Cancel' button pop-up is closed.

Pre-conditions: User has logged to the KPE and navigated to the network view.
Post-conditions: View from which user launched TLBA is un-modified. A new browser tab or window is opened up with TLBA.
Nominal scenario:

- User logs in to KPE.
- User navigates to the network view.
- User selects the 'Open TLBA' option from the KPE tools drop down menu.
- Pop-up is shown to the user where selection of available Shared Spaces is made.
- By pressing 'Open' button TLBA is launched showing actions in selected Shared Spaces
- By pressing 'Cancel' button pop-up is closed.

Non-Functional Requirements:

- Selection of Shared Spaces should be available only to users that have necessary access rights to do that, i.e. read/view privileges.

2.4.1.12 User adds external event into the timeline (M48)

Unique identifier: ***UC-TLBA-12-2009-11***
Related System Usage Scenarios: US-TLBA-6-2009-09
Actors:

- Primary: Users of TLBA
- Secondary: KAS Query service, HPA

Description: User had launched TLBA and wants to add 'external event' into person's timeline. By right-click on a timeline a context-menu appears and user selects the 'Add external event' option. User fills up presented form and by pressing the 'Add' button 'external event' is added into the view and HPA awareness repository. By pressing 'Cancel' button the form closes.

Pre-conditions: User has launched TLBA.
Post-conditions: A new entry is made to the HPA awareness repository, TLBA view is updated with 'external event'.
Nominal scenario:

- User launches TLBA.
- User brings up the context-menu of a timeline and selects the 'Add external event' option.
- User fills up the form with necessary information.
- By pressing the 'Add' button, external event is stored into the HPA awareness repository and TLBA view is updated.
- By pressing the 'Cancel' button, form is closed and no changes to the TLBA view or HPA awareness repository is made.

2.4.1.13 User comments on an object action (M48)

Unique identifier: ***UC-TLBA-13-2009-11***
Related System Usage Scenarios: US-TLBA-7-2009-11
Actors:

- Primary: Users of TLBA
- Secondary: KAS Query service, HPA

Description: User launched the TLBA and wants to comment (i.e. annotate, tag) the persons action. By right-click on an action the context-menu appears. User selects 'Add

comment' option. Application shows the 'Add Comment' form. User fills up the form with necessary data (a comment text) and by pressing the 'Save' button comment is stored into the awareness repository and TLBA view is updated. By pressing 'Cancel' button form is closed.

Pre-conditions: TLBA is loaded up.
Post-conditions: 'Add Comment' form closes. Comment is stored into the repository. TLBA view is updated with comment notion in the view.
Nominal scenario:

- User opens up the TLBA.
- By right-clicking on the action the context-menu appears.
- User selects the 'Add comment' option.
- 'Add Comment' form appears.
- User fills up form with necessary data.
- By pressing 'Save' button form closes. Repository is updated along with update of TLBA view.
- By pressing 'Cancel' button form closes, no updates are made.

2.4.1.14 User defines actions pattern (M48)

Unique identifier: ***UC-TLBA-14-2009-11***
Related System Usage Scenarios: US-TLBA-8-2009-11
Actors:

- Primary: Users of TLBA
- Secondary: KAS Query service

Description: User enters the TLBA and wants to define a 'pattern'. User selects the 'Extract Pattern' option from the Patterns drop down menu (user is in 'Create Pattern' mode). By left-click on actions user picks up desired actions that will form a pattern. A 'Pattern' window is presented to the user (at a time when user selects the 'Extract Pattern' option) where a pattern is represented. By pressing the 'Selected' button in the 'Pattern' window the selection of actions is complete. User broads pattern in 'Pattern' window (multiple occurrences, time frame, etc.) and finishes its definition (selecting destination and name of pattern to store) by pressing the 'Save' button. By pressing the 'Cancel' button the 'Pattern' window is closed (user leaves the 'Create Pattern' mode).

Pre-conditions: TLBA is launched.
Post-conditions: Pattern is stored to destination specified. Pattern is available in the list of loaded patterns. User is not in 'Create Pattern' mode i.e. left-click behaves according to the UC-TLBA-4-2009-11.
Nominal scenario:

- User launches the TLBA.
- User selects the 'Extract Pattern' option from the 'Patterns' drop down menu.
- User enters the 'Create Pattern' mode and by left-click the actions are added to the pattern.
- A 'Pattern' window is created where created pattern can be observed.

- User finishes its selection by pressing 'Selected' button in the 'Pattern' window.
- User broads pattern definition by filling up data in the 'Pattern' window (multiple occurrences, time frame, destination and name of pattern to store, etc.).
- By pressing 'Save' button pattern is stored, or by pressing the 'Cancel' button user leaves the extraction of a pattern.

2.4.1.15 User loads actions pattern (M48)

Unique identifier: ***UC-TLBA-15-2009-11***
Related System Usage Scenarios: US-TLBA-8-2009-11
Actors:

- Primary: Users of TLBA
- Secondary: KAS Query service

Description: User enters the TLBA and wants to load a 'pattern'. User selects the 'Load Pattern' option from the Patterns drop down menu. A 'Load Pattern' form is presented to the user. User fills up necessary information (destination of pattern file) and by pressing the 'Load' button pattern is loaded up into the application pattern list. By pressing the 'Cancel' button the 'Load Pattern' form is closed.

Pre-conditions: TLBA is launched. Pattern is stored, accessible and un-corrupted.
Post-conditions: Pattern is loaded up to the application pattern list.
Nominal scenario:

- User launches the TLBA.
- User selects the 'Load Pattern' option from the 'Patterns' drop down menu.
- User fills up the 'Load Pattern' form with necessary data (destination of pattern file).
- By pressing 'Load' button pattern is loaded to the application pattern list, or by pressing the 'Cancel' button user leaves the form.

2.4.1.16 User searches for the pattern occurrences (M48)

Unique identifier: ***UC-TLBA-16-2009-11***
Related System Usage Scenarios: US-TLBA-8-2009-11
Actors:

- Primary: Users of TLBA
- Secondary: KAS Query service

Description: User launches TLBA and wants to search for the pattern in actions history. User selects 'Search Pattern' option from 'Patterns' drop down menu. A 'Search Pattern' form is brought up to the user. User selects pattern from a pattern list that will be searched for. By pressing 'Search' button a search will be executed and TLBA view will be updated with possible found pattern occurrences. By pressing the 'Cancel' button form is closed.

Pre-conditions: TLBA is launched. There is at least one pattern in application pattern list.
Post-conditions: Form is closed. Found patterns are highlighted.
Nominal scenario:

- User enters TBLA.

- User selects the 'Search Pattern' option from the 'Patterns' drop down menu.
- User chooses desired pattern to be searched for from the pattern list.
- By pressing the 'Search' button search is executed and form is closed. Found patterns are highlighted.
- By pressing the 'Cancel' button search is not made, form is closed.

2.4.1.17 User exports current TLBA view (M48)

Unique identifier: ***UC-TLBA-17-2009-11***
Related System Usage Scenarios: US-TLBA-9-2009-11
Actors:

- Primary: Users of TLBA
- Secondary: KAS Query service

Description: User launched the TLBA and wants to export current TLBA view as a picture. User selects the 'Export image' option from the Export drop down menu. Export Image form is presented to the user, where the format of the picture is selected along with export destination. By pressing 'Save' button picture is saved and Export Image form is closed or by pressing 'Cancel' button from is closed.

Pre-conditions: TLBA has launched.
Post-conditions: Form is closed; picture is exported to desired location in desired format.
Nominal scenario:

- User launches TLBA.
- User selects 'Export image' option from the Export drop down menu.
- Export Image form is presented to the user.
- User fills up necessary information in the form (destination, name and format of a picture to be saved).
- Picture is saved by pressing 'Save' button, by pressing 'Cancel' button the form is closed.

2.4.1.18 User saves current TLBA settings (M48)

Unique identifier: ***UC-TLBA-18-2009-11***
Related System Usage Scenarios: US-TLBA-10-2009-11
Actors:

- Primary: Users of TLBA
- Secondary: KAS Query service

Description: User opened up TLBA and made some selections in its view. By choosing 'Save settings' option from the 'Settings' drop down menu the 'Save Settings' form appears. User fills up the form. By pressing the 'Save' button the current TLBA view with its settings (selected objects, zoom scale and focus) is stored. By pressing the 'Cancel' button form is closed and settings are not stored.

Pre-conditions: TLBA is launched.
Post-conditions: 'Save Settings' form is closed. Settings are stored to desired location.
Nominal scenario:

- User opens up the TLBA.
- User chooses the 'Save settings' option from the 'Settings' drop down menu.
- User fills up the 'Save Setting' form with necessary information (destination and name).
- By pressing the 'Save' button the settings are stored and form is closed, or by pressing the 'Cancel' button the form is closed.

2.4.1.19 User loads TLBA settings (M48)

Unique identifier: ***UC-TLBA-19-2009-11***
Related System Usage Scenarios: US-TLBA-10-2009-11
Actors:

- Primary: Users of TLBA
- Secondary: KAS Query service

Description: User opened up TLBA and wants to load up TLBA stored settings. By choosing 'Load settings' option from the 'Settings' drop down menu the 'Load Settings' form appears. User fills up the form with necessary information (destination of the file). By pressing the 'Load' button the current TLBA view is updated according to stored settings (selected objects, zoom scale and focus). By pressing the 'Cancel' button form is closed and settings are not loaded.

Pre-conditions: TLBA is launched. Settings are stored, available and un-corrupted.
Post-conditions: 'Load Settings' form is closed. TLBA view is updated with desired data.
Nominal scenario:

- User opens up the TLBA.
- User chooses the 'Load settings' option from the 'Settings' drop down menu.
- User fills up the 'Load Setting' form with necessary information (destination of file).
- By pressing the 'Load' button the settings are loaded up and TLBA view is updated, or by pressing the 'Cancel' button the form is closed, and TLBA view is not updated.

3 Technical design

3.1 Software architecture

Diagram with description on the basis of the latest KPE application architecture documentation available at: http://www.kp-lab.org/intranet/work-packages/wp6/result/dii-8-m46-specification-of-end-user-applications/kpe-architecture

Timeline-based Analyser (TLBA) is an Adobe Flex based independent web application. It uses Java based KAS Query service a part of Knowledge Analysis Services (KAS), which falls into the KP-Lab Platform middleware services, as a back-end to accomplish its tasks. Communication between Adobe Flex web-based client, TLBA and Java based back-end service KAS Query service is served through the SOAP protocol over HTTP.

Tight integration with Knowledge Practices Environment (KPE) Flex Components enables users to launch TLBA directly in one of the Shared Space Views (SSp Views) through the URL invocation.

HPA awareness repository which is a collection of logs (set of recorded user's actions from KPE end-user tools), serves TLBA as a data store from where those are extracted (by the KAS Query service) and visualized. Each log represents an action/event that has happened in some KPE end-user tool and was stored by this end-user tool into the HPA awareness repository. Following figure (Figure 2) illustrates scenario, where an action is performed, stored and visualized in the TLBA.

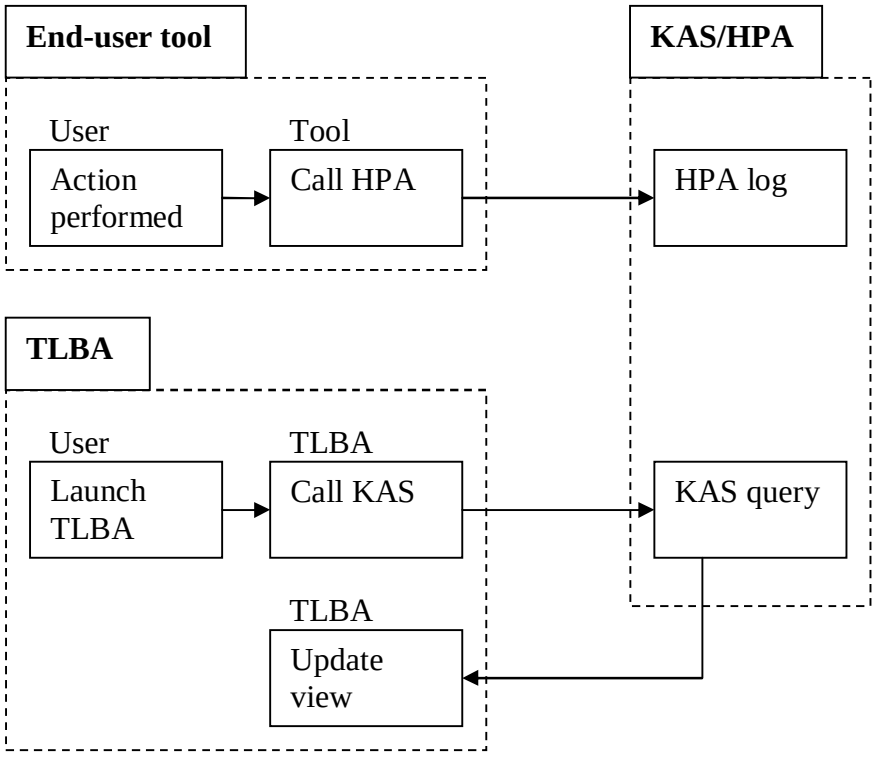


Figure 12: ***Storing and retrieving log scenario***

To store actions into the HPA awareness repository end-user tools use the HPA's Log interface and to retrieve them HPA's Query interface is used [5] or KAS Query service is used when asking for aggregated functionalities (used in TLBA) [6].

A log is represented as a single entry (row) in data-table that consists of following fields: Time stamp, Subject ID, Subject Type, Object ID, Object Type, Action Type, Custom Properties, Comment and Custom Data (for detailed information see [5]).

Event No	Subject ID	Object ID	Object Type	Action Type	Property Key	Property Value	Comment
1	http://www.kp-lab.org/system-model/TLO#Student_Mary	http://www.kp-lab.org/ontologies/CL#CLSpace_x	shared space	creation	SSPID	http://www.kp-lab.org/system-model/TLO#Root_Space	establishing a new Change Laboratory space
					title	Michal's CLSpace	
					description	CLSpace used for demonstration	

Figure 13: ***HPA log sample***

- Action types can be of following string values: creation, modification, deletion and opening.
- Object type string values determine type of object upon which an action was performed (a reference to an object can be obtained in Object ID field) and can be appended as described in [4]. Those are recommended values and serves as a hint for recognizing the object ID reference.
- Custom property field consist of a key-value string pairs which are used to establish a connection between recorded logs or serve as additional information injections. Keys that are in use are as follows: SSPID, title, description, fromObjectID, toObjectID.

3.2 GUI

Following figure (Figure 4) represents TLBA user interface layout which consist of several parts in browser's container. It represents a Shared Space named Custom Space where three persons undertaken their actions. Each person has its own timeline on which person's actions are visualized. Different shapes indicate types of actions performed e.g. creation, modification etc. Curved arrows represent chronological trajectory of actions performed on the same object. They point back in time to indicate that an object 'has been taken up' again. Labels on creation actions represent title of object that was manipulated with.

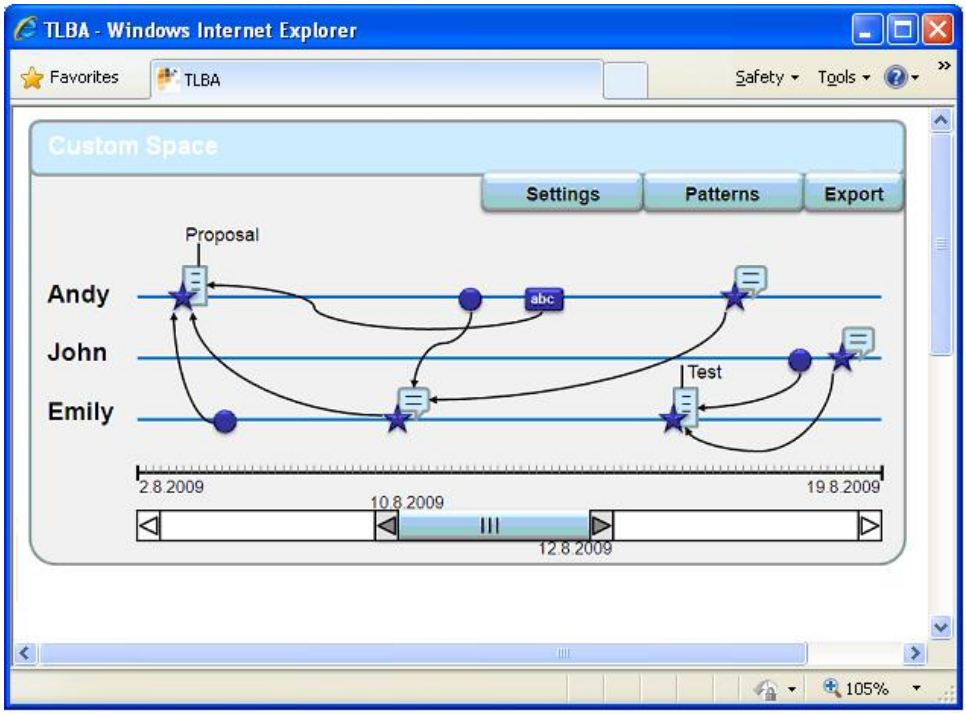


Figure 14: ***TLBA in a browser instance***

Next figure (Figure 5) describes areas of the Space layout container. As seen from previous figure, main areas are: dropdown list menus (Settings, Patterns and Export), user list, timeline data, time indicator and slider bar.

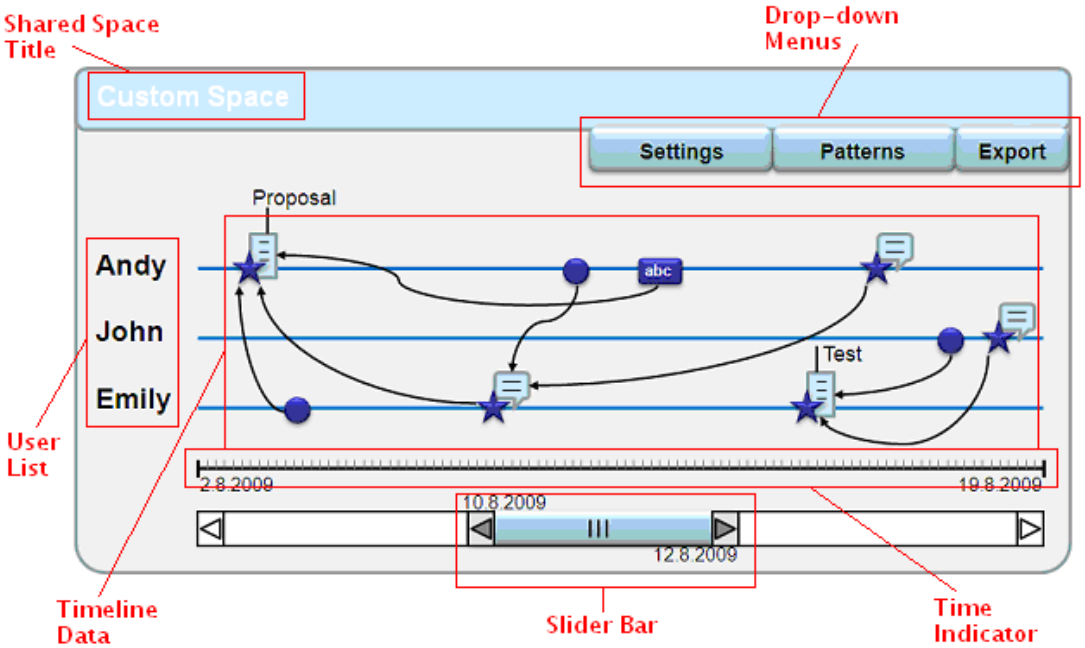


Figure 15: ***TLBA areas***

User list contains all users that performed some action upon one of objects in Shared Space which is under observation (typically those are members of that Shared Space and has read/write access rights to it). Timeline data area represents visualization of recorded logs from HPA awareness repository. Those are aligned according to the appearance of user in the user list and based on user expectations (selected objects, etc.). Time indicator represents time stamps according to which actions have been recorded. In dropdown menus functionality for storing and loading of TLBA settings, patterns or exporting TLBA vies can be found.

As described earlier, types of actions could be of four various instances: creation, opening, modification and deletion. Following table aligns action types to the visualized shapes.

Performed action	**Shape**	**Shape**
Create	Basic shape (star) including a shape that indicates the type of object (in this example: comment).	
Open	Circle	
Modify	ABC-Rectangle	
Delete	Cross	

3.3 Front end services

3.3.1 Launching TLBA

TLBA uses two front-end services for its launch (user scenarios UC-1 up to UC-5 and UC-9). Those are KAS Query service (KAS) and in addition Knowledge Browser service (KB) which is used to retrieve user details into the user list.

To display timeline data ‘getRecentEventsFiltered’ KAS Query method is called (KAS Query interface) which has following signature:

<List<Event>> getRecentEventsFiltered(<TimeRange>,<Filter>,<String>,<int>,<int>)

Parameters that are passes are as follows: timeRange, filter, includeNull, from, maxEvents. TimeRange is a object where time interval of returned logs could be specified, includeNull is a string value which determines if returned log can contain null values, from and maxEvents are integer values that which represent from which entry logs are being retrieved and maximum of returned logs. Filter parameter is a collection of properties where filter criteria are appended. It is a collection of key-value pairs. Returned object is a collection of logs that mach filter criteria.

To retrieve user details ‘getSharedSpacesMembers’ KB method is called which has following signature:

<List<Individual>> getSharedSpacesMemebers(<String>)

Parameter that is passed is a string value which represents unique shared space identifier. Returned object is a collection of objects which contain full information about shared space members.

3.3.2 M48 TLBA functionalities

M48 TLBA functionalities will use HPA interface methods for storing and retrieving necessary data to accomplish adding the external events, commenting of actions and searching for pattern occurrence in HPA awareness repository.

TLBA settings and patterns will be stored as xml files in available storage facilities of user OS. Settings and patterns are persistent in TLBA application until browser session is terminated. Settings xml file includes data about displayed shared spaces and highlighted objects. Pattern xml file includes data about inspected actions (possible multiple occurrences and their sequence), users (their number and relation), object types and time frame.

3.3.2.1 Commenting actions

For storing comments 'addComment' KAS Query service method is called, which has following signature. Comment is appended directly to the log entry in HPA awareness repository within the Custom Data field.

<int> addComment(<int>,<Individual>,<String>)

Parameters are as follows: LogID, Individual and Text. LogID is a integer that represents ID of log that is being commented, Individural represents the user that commented this log, and Text parameter is a String property which represents comment value. Returned object is an integer value which represents success or failure of operation (logID if success, -1 otherwise).

3.3.2.2 Adding external events

External events are appended to the HPA awareness repository through invocation of 'addExternalEvent' method in KAS Query interface which has following signature:

<int> addExternalEvent(<Event>)

Parameter passed represents an external event (time stamp, participants, title, description, SSPID, etc.) which has taken place outside of the KPE environment, and has to bee added to HPA awareness repository manually. Returned object is an integer value which represents ID of event log (when success log ID is returned, otherwise -1 represent failure).

3.3.2.3 Pattern search

To search for and retrieve occurrences of pattern in HPA awareness repository 'searchOccurrence' KAS Query service method is called, which has following signature:

<List<Sequence<Event>>> searchOccurrence(<Pattern>)

Parameter passed represents a pattern that needs to be searched for. This object structure is similar to the xml pattern file representation. Returned object is a list of pattern occurances – represented as a sequence of events, where the pattern is appearing.

4 References

[1] D2.4 Driving Objectives and High-level Requirements for KP-Lab Technologies (revised version).

http://www.kp-lab.org/intranet/work-packages/wp2/result/D2.4_Resubmission_290809.doc

[2] D3.2 A comprehensive research strategy (an updated and revised version for years 4 and 5).

http://www.kp-lab.org/intranet/work-packages/wp1/submission-to-commission-23-9.2009/D3.2-Rev-Research-Strategy_September_submitted.doc

[3] D4.1.1 Description of Work for Months 37-54.

http://www.kp-lab.org/intranet/work-packages/wp1/description-of-work/dow4-1-1/DoW%204.1.1.3.zip/view

[4] Timeline Based Analyzer System Usage Scenarios

http://www.kp-lab.org/intranet/design-teams/wk-analytic-tools/timeline-based-analyser/tlba-sus-uc

[5] History and Participation Awareness Specification

http://www.kp-lab.org/intranet/design-teams/wk-shared-space-and-common-tools/awareness/participation-and-history-based-awareness/history_awareness_technical_v02.doc

[6] D5.6 (Resubmitted with respect to review 3 report)

http://www.kp-lab.org/intranet/work-packages/wp5/result/deliverable-5.6/D5.6_update-after-3rdPrjReview_final.doc

Annex 6. Versioning tool specifications

27490

KP-LAB

Knowledge Practices Laboratory

Integrated Project

Information Society Technologies

D6.8 M46 specification of end-user applications – Uploadable Content Item Versioning Tool

Due date of deliverable: 30/11/2009
Actual submission date:

Start date of project: 1.2.2006 Duration: 60 Months

Organisation name of lead contractor for this deliverable: EVTEK

Revision [1.0]

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission	
RE	Restricted to a group specified by the consortium (including the	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Contributors:

Name	Partner organisation	Email
Rigolleau Benoit	AKKA	b.rigolleau@akka.eu

Version history:

Version	Date	Author(s)	Description
1	30/11/2009	B.Rigolleau	First edition of the document.

Table of Contents

1 Introduction

1.1 Purpose

This document describes the functionality of the tool for versioning the uploadable Content Items. It consists of use cases, mock-up screens, data model and describes the required KP-Lab services.

1.2 Scope of the software

Versioning allows users to create new versions and replace latest version of the uploadable Content Items, which are e.g. Scorm/IMS files, Office documents, pictures… The user is also able to see detailed information on specific versions.

M 48 Requirements:

- The user can create versions in uploadable Content Items.
- The user can replace latest version of the uploadable Content Item.
- The user can see certain information, e.g. creator, date and time, comment and file, of the versions.

1.3 Roadmap of the tool

The planned iterations for the tool are as follows:

Iteration 1: Full functionality of the tool. Release 1.0.
Milestones: M41 Requirements, M45 Specifications, M47 Prototype, M48 Production release

Iteration 2: Final version with improvements and bug fixes based on the user trials. Release 1.1.
Milestones: M56 Release.

1.4 What is new in this version?

This version is the first one.

1.5 Definitions

CIS: Contem Item Services: KP-lab platform component that exposes a service for persistency management of any Content Item. The very content is stored in various Content Repositories (mainly a JSR-170-compatible warehouse, the jackrabbit implementation is used). Semantic data is managed in the KP-Lab Knowledge Repository.

Flex: is a highly productive, free open source framework for building and maintaining expressive web applications that deploy consistently on all major browsers, desktops, and operating systems (see http://www.adobe.com/products/flex/). Used for the GUI implementation of the tool.

GUI: Graphical User Interface

Knowledge Practices Environment (KPE): KPE is the application that is used to log in to/out of the KP-Lab system, to use shared spaces with the integrated tools, and to launch other loosely integrated KP-Lab tools, such as the Activity System Design Tool. The KPE covers all the (end-user) tools under development by Working Knots 1-5 (1, p. 27).

UCI: Uploadable Content Item (SCORM/IMS files, word documents, excel documents and pictures).

UI: The user interface (also known as Human Computer Interface or Man-Machine Interface (MMI)) is the aggregate of means by which people -the users- interact with the system -a particular machine, device, computer program or other complex tool.

2 User requirements

2.1 KP-Lab empirical research

The tool functionalities specified in this document will be used in the following KP-Lab empirical research cases [2]:

Case 1: Trialogical work in higher education; creation and use of knowledge objects in knowledge creation practices

Case 2: Knowledge creation practices in customer projects in higher education

Case 3: Conceptual modelling in Design Projects

Case 4: Producing document management solutions and practices for distributed work settings – Perspectives of KPE usage in two work organizations

2.2 Driving Objectives and High-Level Requirements

The specifications presented in this document are related to the following driving objectives (DO) and high-level requirements (HLR) [1]:

DO1	Users are provided with a collaborative environment where they can work on shared artefacts
HLR1.1	Users can create structure and share various artefacts (e.g. sketches, various kinds of texts, video and audio-files, models as well as ontologies) in one place.
HLR1.2	Users are able to view the artefacts and their relations from different perspectives.
DO9	Users are provided with history on content development and work process advancement
HLR9.1	Users can track the evolution and changes of knowledge objects and find out their authors and contributors (sequences of performed steps in time, incl. versioning)
HLR9.2	Users are provided with customized summaries about the knowledge objects available within the shared environment (e.g. users might request an overview of the tasks completed within the last 2 weeks or the interactions of people within a shared workspace)

2.3 System usage scenarios

The specifications in this document are based on five different system usage scenarios presented in "*WK1: Shared Space and Common Tools: System usage scenarios for the Shared Space and Versioning*" at:
http://www.kp-lab.org/intranet/design-teams/wk-shared-space-and-common-tools/shared-space-content-view/Versioning-SUS-v0.2.2.doc

6. *US-SSp&CViV-01-2009-06*: Creating new version of the uploadable Content Item
7. *US-SSp&CViV-02-2009-06*: Replacing the latest version
8. *US-SSp&CViV-03-2009-06*: Getting information of the versions.
9. *US-SSp&CViV-04-2009-06*: Creating new version in info tab
10. *US-SSp&CViV-05-2009-06*: Replacing version in info tab

2.4 Use cases

This section presents main uses cases for Uploadable Content Item Versioning tool.

2.4.1 Actors

There is only one actor. He/She is a KP-Lab user.

2.4.2 Use Cases

This is a summary of use cases. Each use case is listed with a unique identifier and a title.

UC-VER-01-2009-10: Creating a new version of the UCI

UC-VER-02-2009-10: Replacing the last version of the UCI

UC-VER-03-2009-10: Getting information of the last UCI version

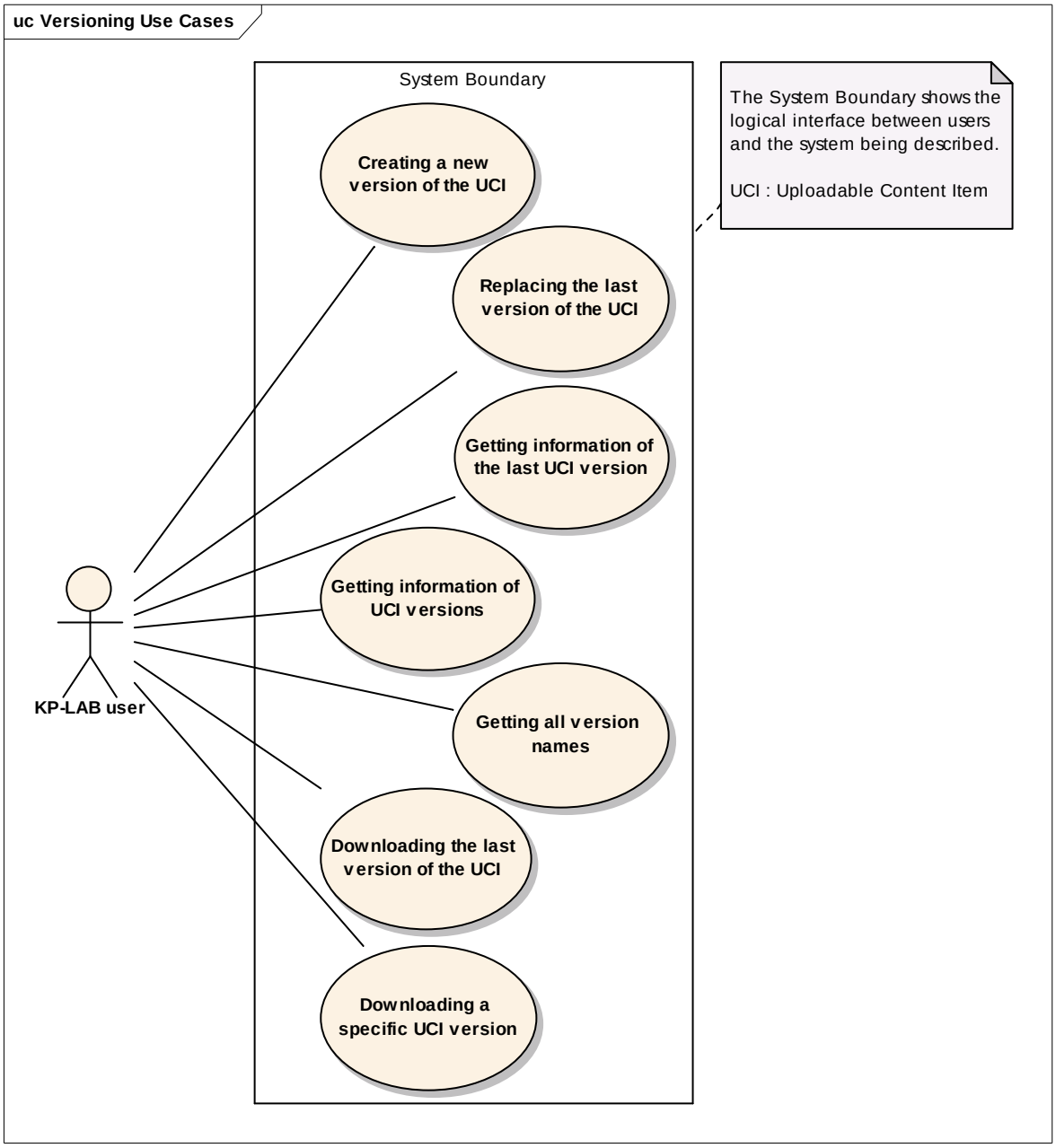
UC-VER-04-2009-10: Getting information of UCI versions

UC-VER-05-2009-10: Getting all UCI version names

UC-VER-06-2009-10: Downloading the last version of the UCI

UC-VER-07-2009-10: Downloading a specific UCI version

2.4.3 Use Case Diagram



The use case diagram showing interactions and relationships between use cases and user.

2.4.3 Detailed Use Cases

Title: Creating a new version of the UCI
Unique identifier: *UC-VER-01-2009-10*
Related System Usage Scenarios:
Actors: any KP-Lab user
Description: the user is capable of uploading an UCI on the server. This UCI is automatically versioned on the server.

Pre-conditions: the user has a KP-lab account and he is logged.
Post-conditions: The UCI is uploaded on the server
Nominal scenario:

- User chooses a file on his/her hard disk.
- User enters information about this file.
- User validates.
- The system returns a success status and an URI to the versioned UCI.

Non-Functional Requirements:
An internet connection must be available
Screen mock-ups:

A form must be present to enter version information and to associate a file to this content item.

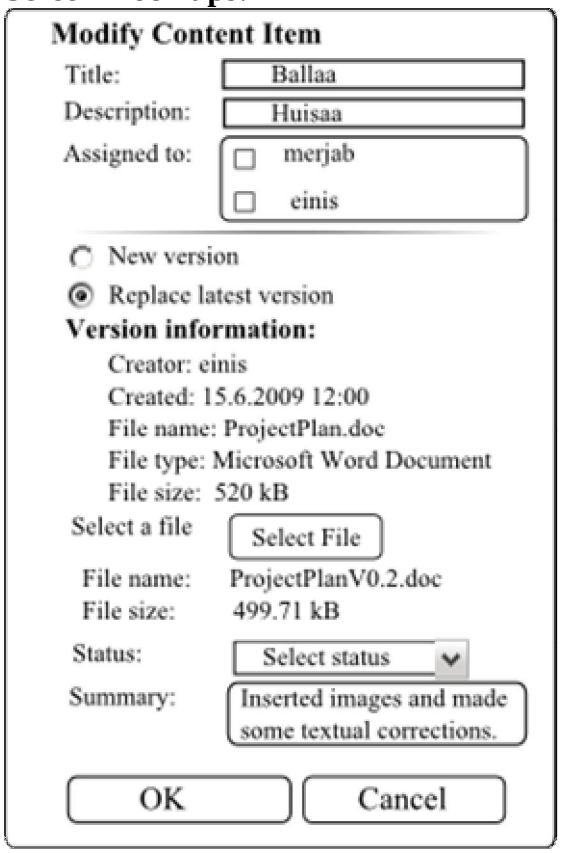
Title: Replacing the last version of the UCI
Unique identifier: *UC-VER-02-2009-10*
Related System Usage Scenarios:
Actors: any KP-Lab user
Description: the user is capable of uploading a new UCI on the server to replace the last version of this UCI.
Pre-conditions: the user has a KP-lab account and he is logged. The last version of the UCI exists.
Post-conditions: The new UCI is uploaded on the server, and it replaces the old one.
Nominal scenario:

- User chooses a file on his/her hard disk (optional).
- User enters information about this file (optional).
- User states explicitly that he/she wants to change the current version, and validates the choice.
- The system returns a success status

Non-Functional Requirements:

An internet connection must be available

Screen mock-ups:

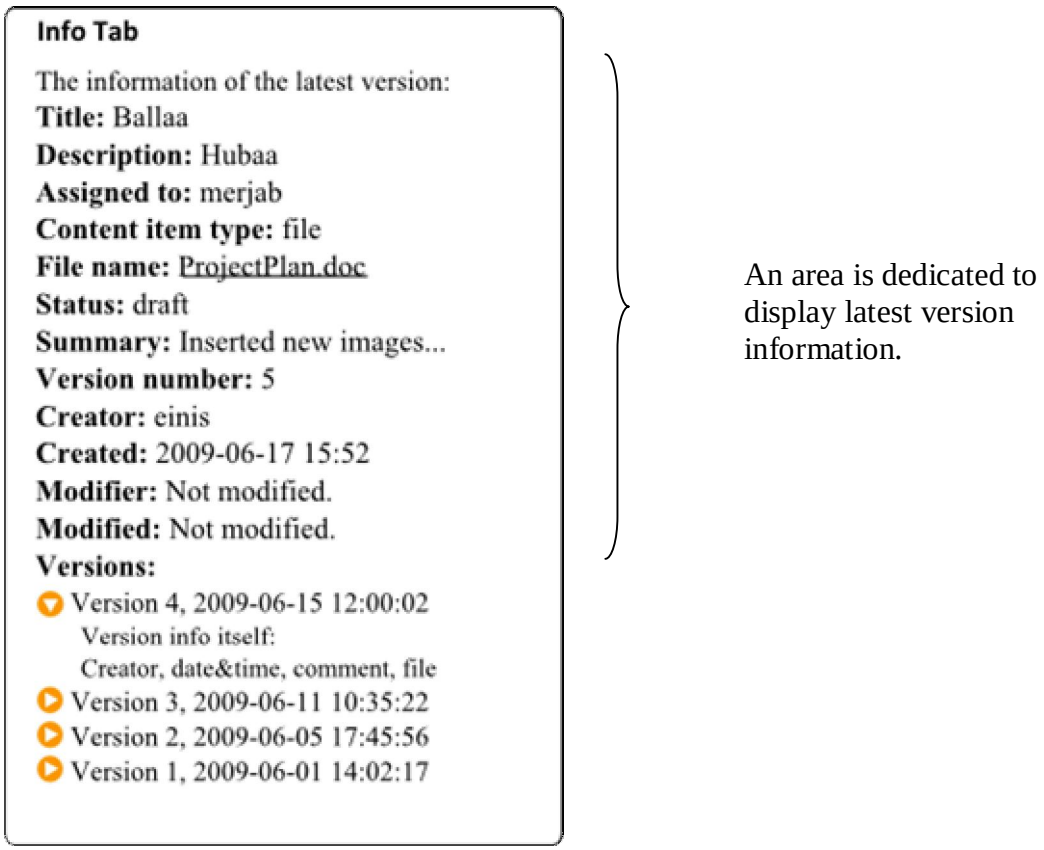


User can decide to create a new version or to replace the latest version.

Title: Getting information of the last UCI version
Unique identifier: *UC-VER-03-2009-10*
Related System Usage Scenarios:
Actors: any KP-Lab user
Description: the user is capable of getting information about the last UCI version.
Pre-conditions: the user has a KP-lab account and he is logged. A version of the UCI is exists.
Post-conditions:
Nominal scenario:

- User chooses to display information to a last UCI version.
- The system returns information

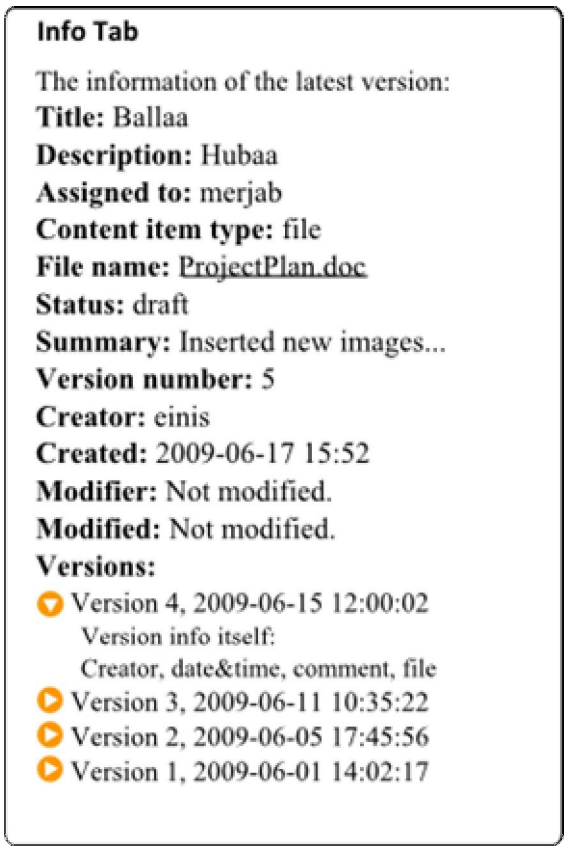
Non-Functional Requirements:
An internet connection must be available
Screen mock-ups:



Title: Getting information of UCI versions
Unique identifier: *UC-VER-04-2009-10*
Related System Usage Scenarios:
Actors: any KP-Lab user
Description: the user is capable of getting information about the specific UCI version.
Pre-conditions: the user has a KP-lab account and he is logged. A version of the UCI is exists.
Post-conditions:
Nominal scenario:

- User chooses to display information to a specific UCI version.
- The system returns information

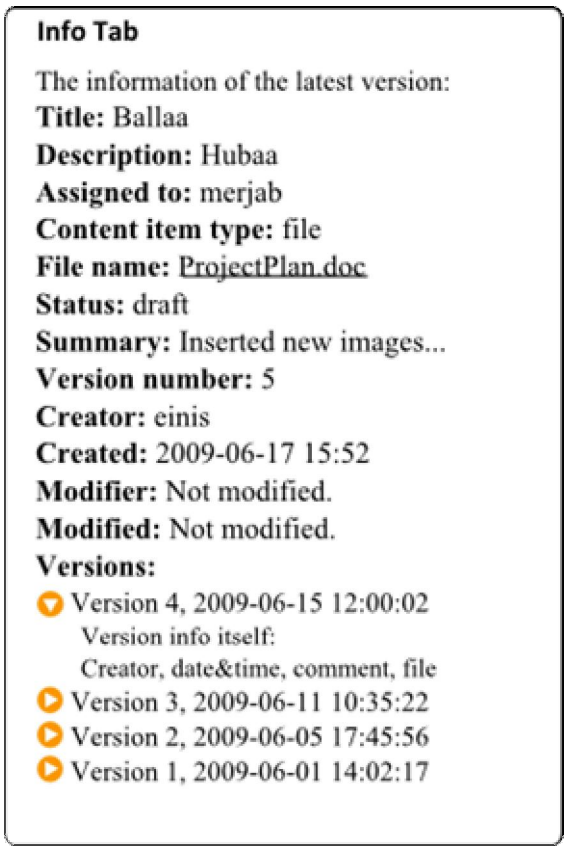
Non-Functional Requirements:
An internet connection must be available
Screen mock-ups:



Title: Getting all UCI version names
Unique identifier: *UC-VER-05-2009-10*
Related System Usage Scenarios:
Actors: any KP-Lab user
Description: the user is capable of getting all UCI version names.
Pre-conditions: the user has a KP-lab account and he is logged. A version of the UCI is exists.
Post-conditions:
Nominal scenario:

- User chooses to list all UCI versions.
- The system returns a list with all version names

Non-Functional Requirements:
An internet connection must be available
Screen mock-ups:



Title: Downloading the last version of the UCI
Unique identifier: *UC-VER-06-2009-10*
Related System Usage Scenarios:
Actors: any KP-Lab user
Description: the user is capable of downloading the last version of a UCI.
Pre-conditions: the user has a KP-lab account and he is logged. A version of the UCI is exists.
Post-conditions:
Nominal scenario:

- User chooses to download the last version of a UCI
- The system returns the corresponding file, part of the UCI.

Non-Functional Requirements:
An internet connection must be available
Screen mock-ups:

Title: Downloading a specific UCI version
Unique identifier: *UC-VER-07*
Related System Usage Scenarios:
Actors: any KP-Lab user
Description: the user is capable of downloading the specific UCI version.
Pre-conditions: the user has a KP-lab account and he is logged. A version of the UCI is exists.
Post-conditions: -

Nominal scenario:

- User chooses to download a specific UCI version
- The system returns the corresponding file

Non-Functional Requirements:

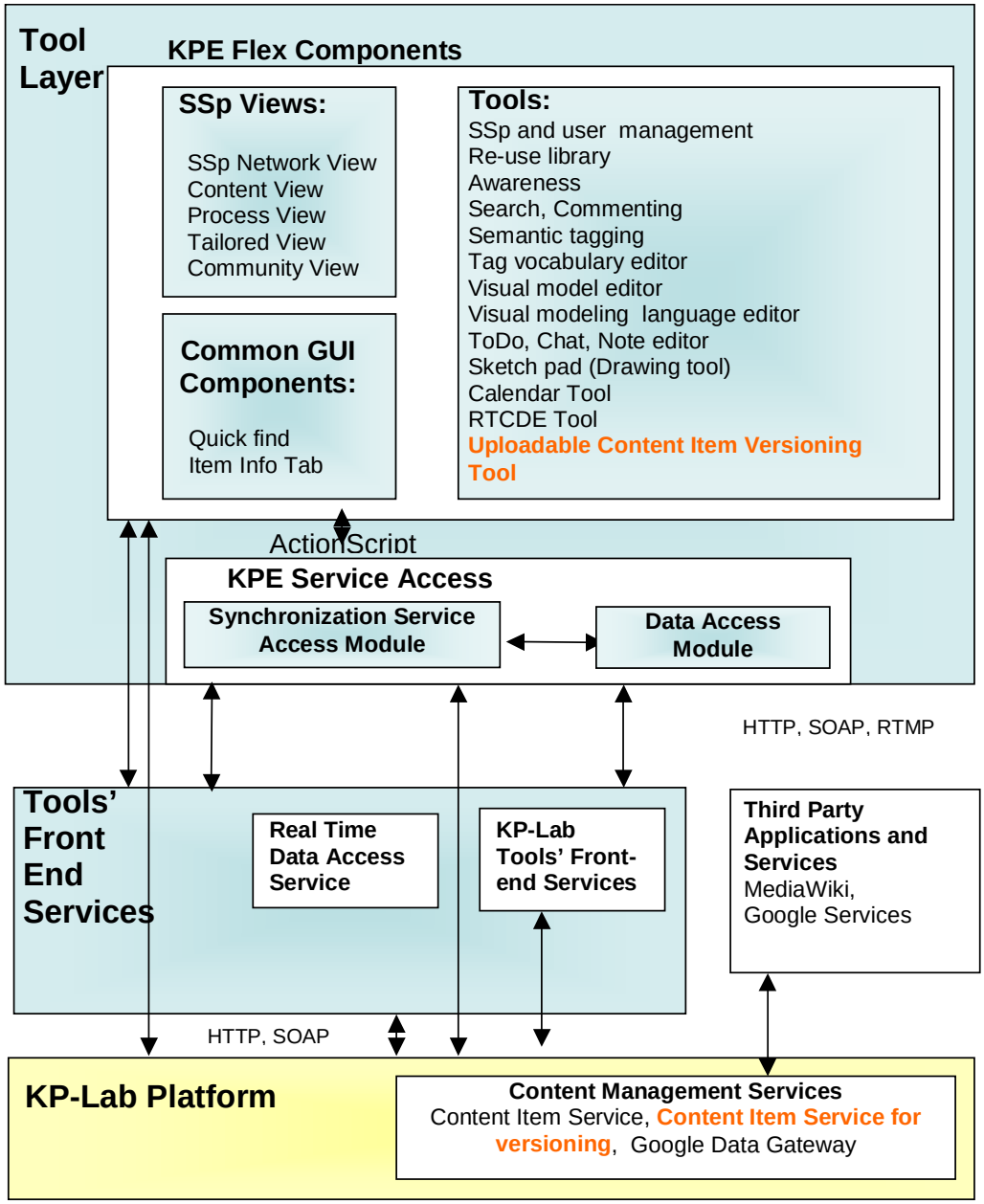
An internet connection must be available

Screen mock-ups: -

3 Technical design

3.1 Software architecture

3.1.1 The Uploadable Content Item Versioning tool in the KP-Lab System



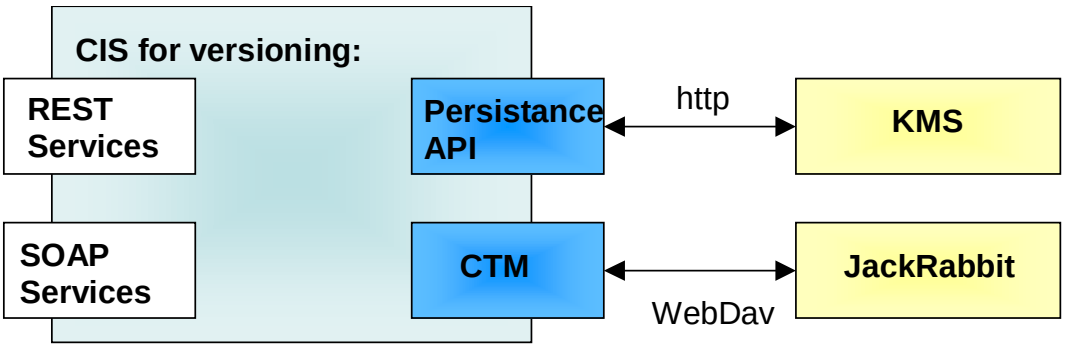
This schema represents a simplified view of the KPE environment (it does not present the loosely coupled integrated tools). The Uploadable Content Item Versioning tool (UCIV) and the services that it communicates with appear highlighted.

The tool is part of the utility tools that are tightly integrated in the KPE. Through its User interface, it provides users with a view on the different versions of Uploadable Content

Items over time. It allows to keep trace of the evolutions of UCIs and to retrieve them easily.

The tool relies on key services of the KP-Lab platform (see WPIII deliverables), namely CIS (Content Item Services), in charge of all basic treatment of Content Items with content and semantic data persistency.

3.1.2 The Uploadable Content Item Versioning Services architecture



The CIS for versioning proposes a REST and a SOAP interfaces **of exactly the same services**.

The CIS for versioning used the Persistence API to push data into KMS. In a same time it uses CTM to store and take into version Uploadable Content Item and its information. To do that, CTM communicates with a JSR170 content repository, JackRabbit. JackRabbit is in charge of the data storage and versioning. The protocol used between CTM integration (CIS here) and JackRabbit is a WebDav one.

3.1.2.1 Create service

Create is a REST web service for creating Content Items of type kp-lab/uploadable".

To create a Content Item, send a `POST` request to the following URL:

```
http://<server>:<port>//ContentItemService/Create
```

Include the relevant parameters in the body of the `POST` request.

If the request succeeds, then the response contains an xml with the URIsof the created Content Item in KMS and in CTM.

```
<xml>
            <ctmURI></ctmURI>
            <kmsURI></kmsURI>
            <message></message>
            <status></status>
</xml>
```

The created content item is instance of the following classes:

http://www.kp-lab.org/system-model/TLO# UploadableContentItem

http://www.kp-lab.org/system-model/TLO# ContentItem

Table – Upload request parameters.

Name	*Type*	*Cardinality*	*Description*
filedata	file	1	The file content
userId	text	1	The URI of the creator of this content item
sharedSpaceURI	text	1	The shared space to which this content item is created
title	text	1	the title of the content item
description	text	0..1	The description of the content item
type	text	1	The MIME type of the file
responsibleMember	text	Multiple	URIs of users who are marked as responsible for this content item
status	text	0..1	The status of the content item (for exemple draft)
__TOKEN__	Text	1	Josso token used for authentication

* Note that both imageMapURI and containerURI can not be null.

3.1.2.2 Modify service

Modify is a REST web service for modifying Content Items of type “kp-lab/uploadable“.

To modify a Content Item, send a `POST` request to the following URL:

```
http://<server>:<port>//ContentItemService/Modify
```

Include the relevant parameters in the body of the `POST` request.

If the request succeeds, then the response contains the version name, or null if the content was simply updated.

```
<xml>
            <value></value>
            <message></message>
            <status></status>
</xml>
```

Table – modify request parameters.

Name	*Type*	*Cardinality*	*Description*
filedata	file	1	The new file content
contentUri	text	1	The content URI to modify

Name	*Type*	*Cardinality*	*Description*
isVersion	bool	1	If true, a new version is created, else the actual content is updated
userId	text	1	The URI of the creator of this content item
title	text	1	the title of the content item
description	text	1	The description of the content item
comment	text	1	Comment about the new version
type	text	1	The mime type of the file
responsibleMember	text	multiple	URIs of users who are marked as responsible for this content item
status	text	0..1	The status of the content item (for exemple draft)
__TOKEN__	text	1	Josso token used for authentication

3.1.2.3 Remove service

Remove is a REST web service for removing Content Items of type ‚kp-lab/uploadable".

To remove a Content Item, send a `POST` request to the following URL:

`http://<server>:<port>//ContentItemService/Remove`

Include the relevant parameters in the body of the `POST` request.

If the request succeeds, then the response contains the default Rest response.

```
<xml>
            <message></message>
            <status></status>
</xml>
```

Table – remove request parameters.

Name	*Type*	*Cardinality*	*Description*
contentUri	text	1	The content URI to remove
__TOKEN__	text	1	Josso token used for authentication

3.1.2.4 getFile service

getFile is a REST web service for getting content item file of type ‚kp-lab/uploadable".

To get the Content Item's file , send a `POST` request to the following URL:

`http://<server>:<port>//ContentItemService/GetFile`

Include the relevant parameters in the body of the POST request.

If the request succeeds, then the response contains the file.

if the content was not found, the response has "404 Not Found" status code.

Table – getFile request parameters.

Name	*Type*	*Cardinality*	*Description*
contentUri	text	1	The file content URI
__TOKEN__	text	1	Josso token used for authentication

3.1.2.5 getFileInfo service

getFileInfo is a REST web service for getting content item file information, of type kp-lab/uploadable“.

To get the Content Item file information , send a POST request to the following URL:

`http://<server>:<port>//ContentItemService/GetInfoFile`

Include the relevant parameters in the body of the POST request.

If the request succeeds, then the response contains a xml with properties inside.

```
<xml>
            <property name="" ></property>
            <property name="" ></property>
            .
            .
            <message></message>
            <status></status>
</xml>
```

Table – getVersionedFile request parameters.

Name	*Type*	*Cardinality*	*Description*
contentUri	text	1	The file content URI
__TOKEN__	text	1	Josso token used for authentication

3.1.2.6 getVersionedFile service

getVersionedFile is a REST web service for getting versioned content item file of type ,kp-lab/uploadable“.

To get the versioned Content Item file , send a POST request to the following URL:

```
http://<server>:<port>//ContentItemService/GetVersionedFile
```

Include the relevant parameters in the body of the POST request.

If the request succeeds, then the response contains the file.

If the content was not found, the response has "404 Not Found" status code.

Table – getVersionedFile request parameters.

Name	*Type*	*Cardinality*	*Description*
contentUri	text	1	The file content URI
versionName	text	1	The name of the desired version
__TOKEN__	text	1	Josso token used for authentication

3.1.2.7 getVersionedFileInfo service

getVersionedFileInfo is a REST web service for getting versioned content item file information, of type kp-lab/uploadable“.

To get the versioned content item file information, send a POST request to the following URL:

```
http://<server>:<port>//ContentItemService/GetVersionedInfoFile
```

Include the relevant parameters in the body of the POST request.

If the request succeeds, then the response contains a xml with properties inside.

```
<xml>
            <property name="" ></property>
            <property name="" ></property>
            .
            .
            <message></message>
            <status></status>
</xml>
```

Table – getVersionedFile request parameters.

Name	*Type*	*Cardinality*	*Description*
contentUri	text	1	The file content URI
versionName	text	1	The name of the desired version
__TOKEN__	text	1	Josso token used for authentication

3.1.2.8 getVersionNames service

getversionNames is a REST web service for getting all version names for a content item.

To get the version names, send a POST request to the following URL:

`http://<server>:<port>//ContentItemService/GetVersionNames`

Include the relevant parameters in the body of the POST request.

If the request succeeds, then the response contains a xml structure with names inside.

```
<xml>
            <value></value>
            <message></message>
            <status></status>
</xml>
```

Table – getVersionedFile request parameters.

Name	*Type*	*Cardinality*	*Description*
contentUri	text	1	The file content URI
__TOKEN__	text	1	Josso token used for authentication

3.1.2.9 GetNumberOfVersionMultiple service

GetNumberOfVersionMultiple is a Rest web service for getting the exact numbers of version the files have.

To get the number of version, send a POST request to the following URL:
`http://<server>:<port>//ContentItemService/GetNumberOfVersionMultiple`

Include the relevant parameters in the body of the POST request.
If the request succeeds, then the response contains xml will match the contentURI with the number of version the file has

```
<xml>
            <value contentUri=""></value>
            <value contentUri=""></value>
            .
            .
            <message></message>
            <status></status>
</xml>
```

Table – getNumberOfVersionMultiple request parameters.

Name	*Type*	*Cardinality*	*Description*
contentUris	text	multiple	List of file content URIs
__TOKEN__	text	1	Josso token used for authentication

3.1.2.10 Services for CASS

This part is about special services for CASS. It's only to provide a compatible solution with CASS.

Upload is a REST web service for creating Content Items of type „kp-lab/uploadable".

To upload a Content Item, send a `POST` request to the following URL:

`http://<server>:<port>//ContentItemService/Upload`

Include the relevant parameters in the body of the `POST` request.

If the request succeeds, then the response contains the URIs of the Content Item, Node and the Content.

The created content item is instance of the following classes:

http://www.kp-lab.org/system-model/TLO# UploadableContentItem

http://www.kp-lab.org/system-model/TLO# ContentItem

and also

http://www.kp-lab.org/ontologies/ss# BackgroundImage

if isBgImage parameter is included in the request.

Table – Upload request parameters.

Name	***Type***	***Cardinality***	***Description***
filedata	file	1	The file content
userId	text	1	The URI of the creator of this content item
sharedSpaceURI	text	1	The shared space to which this content item is created
title	text	1	the title of the content item
xCoord	text	1	The x coordinate of the content item in the SSGUI view
yCoord	text	1	The y coordinate of the content item in the SSGUI view
description	text	1	The description of the content item
type	text	1	The mime type of the file
imageMapURI	text	optional* <=1	The image map into which this content item is created
containerURI	text	optional* <=1	The container into which tis content item is created
responsibleMember	text	multiple	Uris of users who are marked as responsible for this content item

Name	*Type*	*Cardinality*	*Description*
isBgImage		1	If this parameter is present, the resulting content item will be instance of *http://www.kp-lab.org/system-model/TLO# UploadableContentItem* *http://www.kp-lab.org/system-model/TLO# ContentItem* and *http://www.kp-lab.org/ontologies/ss# BackgroundImage* The actual value of isBgImage is of no importance – only the presence is checked.

Download is a REST service.

To use the Download service send a HTTP GET request to

```
http://<server>:<port>//ContentItemService/Download?content=contentURI
```

Table – getVersionedFile request parameters.

Name	*Type*	*Cardinality*	*Description*
contentUri	text	1	The value of the "content" parameter is an UUID of the content in the JCR. This UUID is returned as "contentURL" when the content is created.

3.2 GUI

A first prototype has already been developed based on evolutions of KP-Lab platform components CTM, CIS and CR as done in WPIII. This prototype will be integrated in M47 in KPE. It will change according to user feedbacks.

Some screenshots can be provided here. All use cases are covered, but the look&feel design is not the last one. We will apply the KPE skin, when this tool will be integrated.

3.2.1 Create a new version

The following GUI allows creating a new version of a uploadable content item.

The user can select on a file on his/her hard disk. He/she can also enter information like a title, a description. It's possible to assigned this content item to some users.

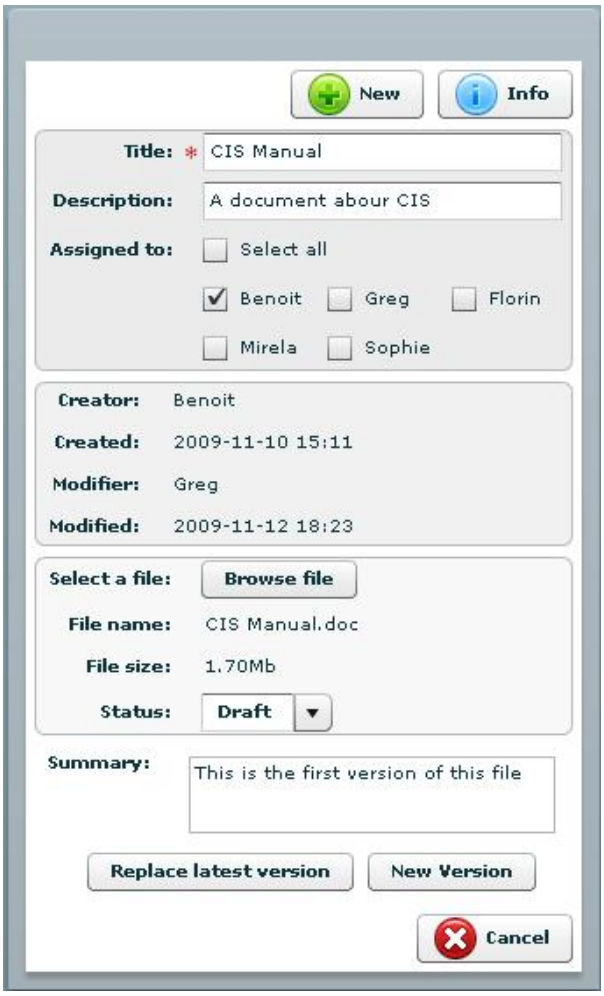
A file status and a version summary are available.

3.2.2 Modify a version

When user wants to modify an existing version, a window, likes the one for creating a version appears. Information are the same than for the creation, with some additional items. A new area is present. It is possible to get general information like creator's name, the date of last modification, etc.

The user can edit some fields, and he/she can select a new file. The "Replace latest version" button leads to the replacement of the most actual version. It corresponds actually to a standard edit. The "New version" button, gives the possibility to create a new version of the content item on front of the latest version known. After that, the "latest version" is the new one, and the former one remains accessible like on information window.

Two news buttons are also present. The first one, "new" is a link to the "create a new version" window. The other one, "info", opens the information perspective, and gives the possibility to access all information about this content item.



3.2.3 Get all information

The following GUI is the information window. It gives all information about a content item. Information about the actual version is present, and the associated file is downloadable. The bottom of this GUI provides an area dedicated for listing older versions of this content item. A list of all versions is always present, and it is possible to expand/collapse each version display/hide more information about a specific version. It is again possible to download the associated file corresponding to a previous version.

By double clicking a field, the user can switch to the edit mode. Another way to do that is to use the “Edit” button.

3.3 Collaboration with other tools

This tool is a KPE module. It is started directly within the KPE.

4 References

[1] D2.4 Driving Objectives and High-level Requirements for KP-Lab Technologies (revised version). http://www.kp-lab.org/intranet/work-packages/wp2/result/D2.4_Resubmission_290809.doc

[2] D3.2 A comprehensive research strategy (an updated and revised version for years 4 and 5). http://www.kp-lab.org/intranet/work-packages/wp1/submission-to-commission-23-9.2009/D3.2-Rev-Research-Strategy_September_submitted.doc

Annex 7. Visual analyser specifications

27490

KP-LAB

Knowledge Practices Laboratory

Integrated Project

Information Society Technologies

D6.8 M46 specification of end-user applications – Visual Analyser

Due date of deliverable: 30/11/2009
Actual submission date:

Start date of project: 1.2.2006 Duration: 60 Months

Organisation name of lead contractor for this deliverable: EVTEK

Revision [1.0]

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission	
RE	Restricted to a group specified by the consortium (including the	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Contributors:

Name	Partner organisation	Email
Ekaterina Simonenko	UPS	ekaterina.simonenko@lri.fr
Tsuyoshi Sugibuchi	UPS	buchi@lri.fr

Version history:

Version	Date	Author(s)	Description
0.1	13/11/2009	Ekaterina Simonenko	Initialization document

Table of Contents

1 Introduction

1.1 Purpose

This document contains a detailed functional specification of both, the M44 and M48 releases of the Visual Analyser tool.

1.2 Scope of the software

The Visual Analyser tool is one of the three main analytic end user tools [1]. It provides services for analyzing participation and activities within past or ongoing processes, by visually representing them based on information stored in the produced logs. More precisely, it visualizes frequencies of object-related activities in KPE and provides detailed information on the nature of the activities performed on particular knowledge objects. These visualizations stimulate teachers and students to reflect on object progression, and on their teaching and learning practices.

1.3 Roadmap of the tool

The current roadmap of the Visual Analyser tool has one finished and one planned iteration as follows:

Visual Analyser 1.0

Iteration 1:
This version includes:
• Request formulation by clicking on the attributes
• Summary visualization in the form of bar chart or line chart
• First integration into KPE (as a plug-in)
Milestones: M40: Specification, M42 Prototype, M44 Release

Visual Analyser 2.0

Iteration 2:
This version will include:
• Request formulation by drag-and-drop
• Possibility to create, load and manage predefined queries
• Improved user interface, new analysis attributes included (shared space Id, object title)
• Interactive summary visualization - requests sent "on the go", any time a query parameter is modified
• Suggestions of values when filtering events by object name, user name etc.
• More flexibility for the events filtering (connecting the filters by "OR" condition, specifying a time interval)
• Filtering of events by week, in addition to the filtering by year, month or day, filtering by defining a time interval.
• Possibility to export (save) the resulting summaries and their visualization in a file

• Final integration into KPE
Milestones: M44: Specification, M46 Prototype, M48 Release

Visual Analyser 2.1
Milestones: M56 Release
This is the final version with improvements and bug fixes of v2.0 based on the user trials.

1.4 What is new in this version?

The present deliverable describes specifications of both, M44 and M48 releases of the Visual Analyser tool. The M48 release aims at improving the M44 release on the basis of the feedback from tests and evaluation and extending the tool by adding new functionalities such as better filtering, saving the query result, etc.

1.5 Definitions

The basic units of analysis handled by the Visual Analyser are user's actions with the KPE recorded in the HPA repository.

History/Participation Awareness (**HPA)** - HPA is a repository of events representing activities or changes performed by users of various end-user tools in KP-environment. Logs of these events provide source data for analysis and identification of new knowledge that can be used for analysis and improvement of on-going activities as well as for adaptation of the graphical user interface or for personalization.

Knowledge Analysis Services (KAS) - KAS provides the necessary middleware functionality for visual analyser, which requires a rich set of services for selecting and aggregating proper data from underlying log repositories.

Knowledge practices environment (KPE) – KPE is the application that is used to log into/out of the KP-Lab system, to use shared spaces with the integrated tools, and to launch other loosely integrated KP-Lab tools, such as the Activity System Design Tool. The KPE covers all the relevant (end-user) tools.

Shared space (SSp) – SSp is a virtual area that may contain tools and objects. An SSp may have one or more members. The number of different SSps is practically unlimited.

Each of the user's action (i.e. a user induced change in the KP-Lab system) is recorded as an event. Events relevant to the visual analyser are the creation, opening, modification, commenting, tagging, linking, and deletion of objects available within a shared space (i.e. tasks, content items, vocabulary, tailored view).

The records for each event logged by the HPA are structured as follows:

- **Event ID -** identifier of the log entry
- **Time** when the event was logged
- **Type** and **ID** of the subject: user that performed this action

- **Type** and **ID** of the object: object of performed action
- **Type of the action** performed in the end-user tool
- **Comment**
- **Properties** – custom property, e.g. SSpID (Shared Space ID) and Object title
- **Custom data**

Table 1 provides a description of the attributes which can be tracked through the VA and a domain of each attribute.

AttributeDomain	
Time	"Time" can be specified as: - Year, ex: 2008 - Month, ex: 12.2008 - Week number, ex: 52.2008 - Date, ex: 28.02.2008
User Name	The names of the users working with the objects in KPE.
Object Type l	Currently Object Types are the following: l Task l Content Item l Vocabulary l Tailored View l Visual Model
Object Title	The titles of objects, specified by the users in KPE.
Action Type l	The current Action Types are the following: 1. create 2. open 3. modify 4. comment 5. tag 6. link 7. delete
Shared Space Name	The shared space names stored in the log.

2 User requirements

2.1 KP-Lab empirical research

The tool functionalities specified in this document will be used in the following KP-Lab empirical research cases [1, 2, 4]:

Case 1: Trialogical work in higher education; creation and use of knowledge objects in knowledge creation practices
Case 2: Knowledge creation practices in customer projects in higher education

Case 4: Producing document management solutions and practices for distributed work settings – Perspectives of KPE usage in two work organizations

2.2 Driving Objectives and High-Level Requirements

The specifications presented in this document are related to the following driving objectives (DO) and high-level requirements (HLR) [1]:

HLR9.2	Users are provided with customized summaries about the knowledge objects available within the shared environment (e.g. users might request an overview of the tasks completed within the last 2 weeks or the interactions of people within a shared workspace)
DO13	Users can collect data about activities and interactions systematically and over longer periods of time
HLR13.5	Users can customise the way they want to retrieve wanted data for analysis.
HLR13.6	Users are provided with summaries on performed actions (e.g. added comments, created tasks, modifications in metadata, background materials for decisions, etc.). (Rephrased)

2.3 System usage scenarios

The specifications in this document are based on three different system usage scenarios presented in document [1]:

1. US-VA-1-2009-09 : Analysing data based on a user defined query
2. US-VA-2-2009-09: Reusing queries
3. US-VA-3-2009-09: Ad hoc definition of groups

2.4 Use cases

This section presents the use cases of the Visual Analyser tool (in its final version).

2.4.1 Actors

Users of Knowledge Practices Environment (KPE) such as teachers, students, course administrators, and professionals.

2.4.2 Overview of Use Cases

The behavior of the Visual Analyser tool is defined by eight use cases as follows:

1. UC-VA-1-2009-09: Query definition and result exploration
2. UC-VA-2-2009-09: Refining the query adding or removing parameters
3. UC-VA-3-2009-09: Adding a parameter to obtain a stacked diagram.
4. UC-VA-4-2009-09: Downloading a predefined query.
5. UC-VA-5-2009-09: Saving the query as a predefined query.
6. UC-VA-6-2009-09: Managing predefined queries.
7. UC-VA-7-2009-09: Saving the result of the query.
8. UC-VA-8-2009-09: Defining groups of users for a query.

2.4.3 Detailed Use Cases

Title:
Query definition and result exploration.

Unique identifier: UC-VA-1-2009-09

Related Usage Scenarios: US-VA-1-2009-09

Actors:
Users of KPE, mainly teachers, students, course administrators, and professionals.

Description: Basic query formulation, by defining parameters and filters for the query.

Pre-conditions: User is logged into the KPE.

Post-conditions: Visual Analyser application opens in a browser window and the user can go on formulating his query.

Nominal scenario:
Goal: Formulate a query "Number of events by Year and by Object Type, concerning the user Teacher Frank", and explore the result.
Step 1: User sets the first grouping parameter by drag-and-dropping the "Year" data field from the up-left area to the Data Table area below.
Step 2. The system shows the result of the query "Number of events by Year".
Step 3. To add a second parameter - the user drag-and-drops the "Type" data field of the "Object" to the Data Table area.
Step 4. The system shows the result of the query "Number of events by Year and Object Type".
Step 5. To filter the events by user name, the user drag-and-drops the "Name" data field of the "User" down to the "Restrictions" box.
Step 6. A little box appears where user types the user name he wants, or click on the "Browse" button on the right of the little box.
Step 7. The system shows suggestions to the user name, based on the existing user names.
Step 8. A valid user name is chosen by the user.
Step 9. The system processes the query and presents the result on the screen.

Non-Functional Requirements:
The user knows how drag-and-drop method work.

Screens:

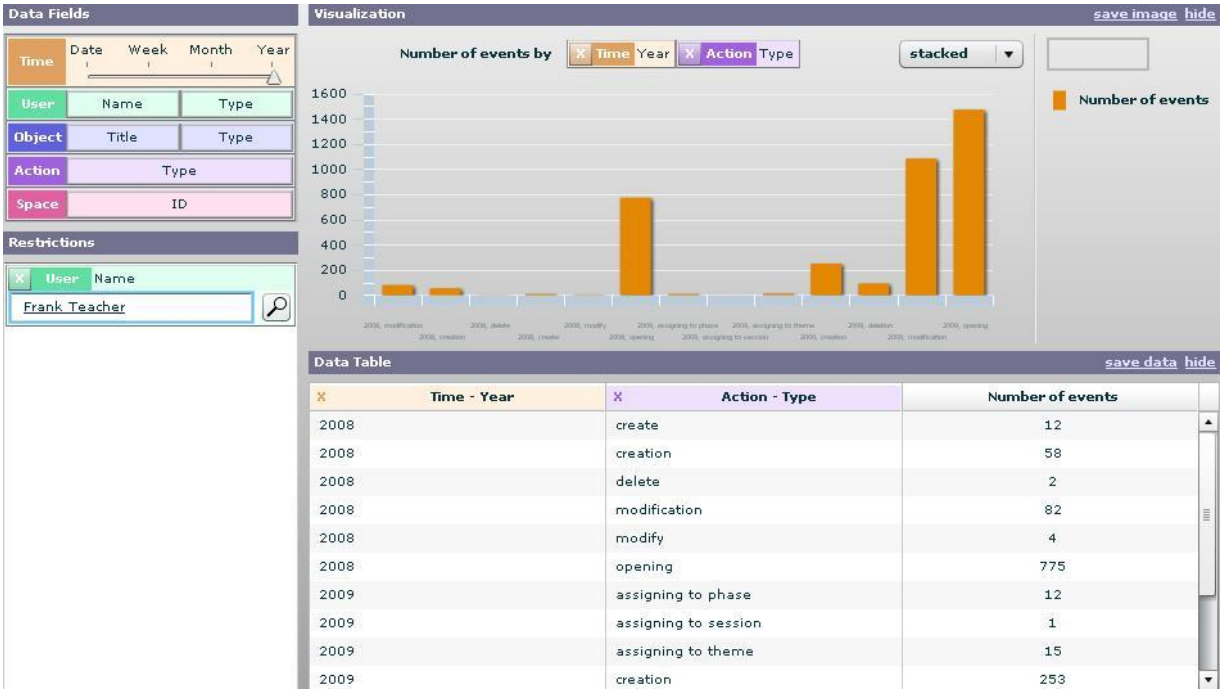


Figure 16: Basic query formulation and result.

Title: Refining the query adding or removing parameters.

Unique identifier: UC-VA-2-2009-09

Related Usage Scenarios: US-VA-1-2009-09

Actors:
Users of KPE, mainly teachers, students, course administrators, and professionals.

Description: Adding and removing parameters and filters to the query.

Pre-conditions: User is logged into the KPE.

Post-conditions: Visual Analyser application opens in a browser window and the user has formulated a modified query.

Nominal scenario:
Goal : Modify the query by refining to the Actions of type "creation" and generalizing to all the users.
Step 1: The user adds a filter to the Action Type : drag-and-drops the "Action Type" data field to the Restrictions area.
Step 2 : A field appears in the Restriction area, allowing to enter a specific user name.
Step 3: The user starts to type "creation" or uses à "Browse" button on the right to see existing action types.
Step 4: The tool suggests the action types of the events stored in the log.
Step 5: The user confirmes his choice.

Step 6: When a valid value is entered, the result is shown by the tool (see Fig. 2)..
Step 7: To remove the filter on user name, the user clicks on the "x" button on the left of the corresponding filter on the Restriction area.
Step 8: The tool removes the filter and the shows the result.

Screens:

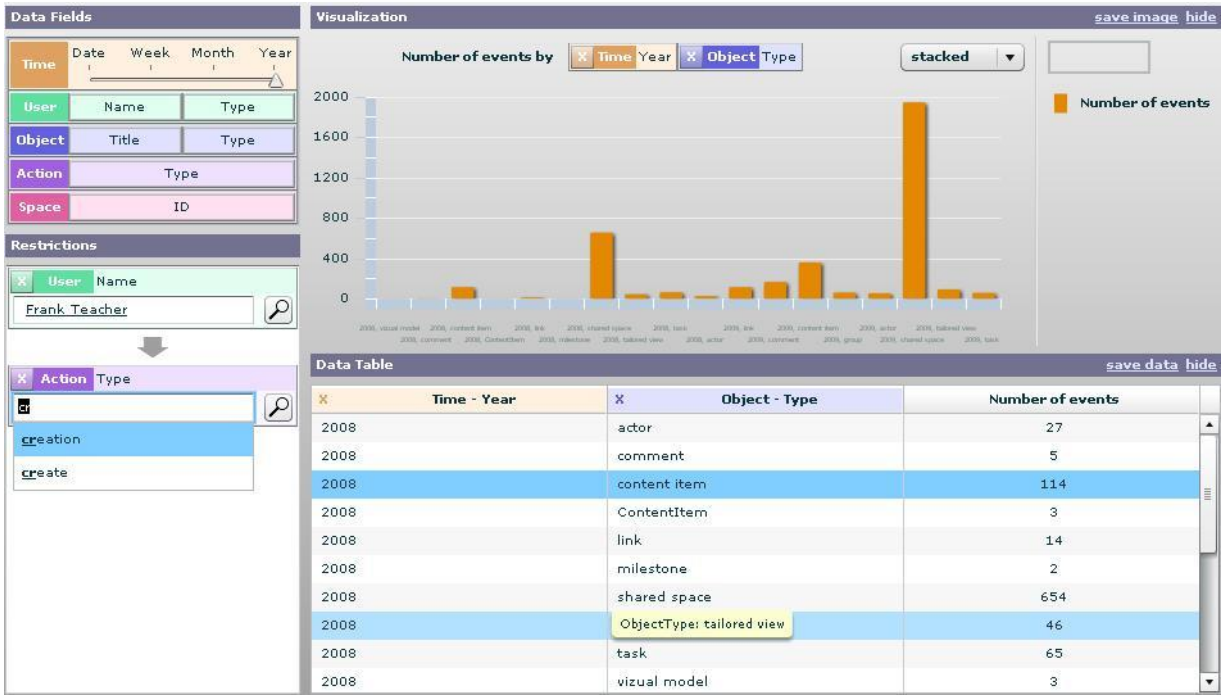


Figure 17: Adding a filter on the Action Type.

Figure 18: Filter on User Name being removed.

Title: Adding a parameter to obtain a stacked diagram.

Unique identifier: UC-VA-3-2009-09

Related Usage Scenarios: US-VA-1-2009-09

Actors: Users of KPE, mainly teachers, students, course administrators, and professionals.
Description: Modifying the query by adding a parameter to obtain a stacked diagram.

Pre-conditions: User is logged into the KPE.

Post-conditions: Visual Analyser application opens in a browser window and the user has formulated a query resulting in a stacked diagram.

Nominal scenario:
Goal : Compare object creation activities for different object types, using stacked diagramm.
Step 1: The user defines "Object Type" as a parameter for stacking, by drag-and-dropping it to the little box in the up-right corner of the Visualization area.
Step 2: The tool shows the result, by visualizing the query result as a stacked diagramm.

Screens:

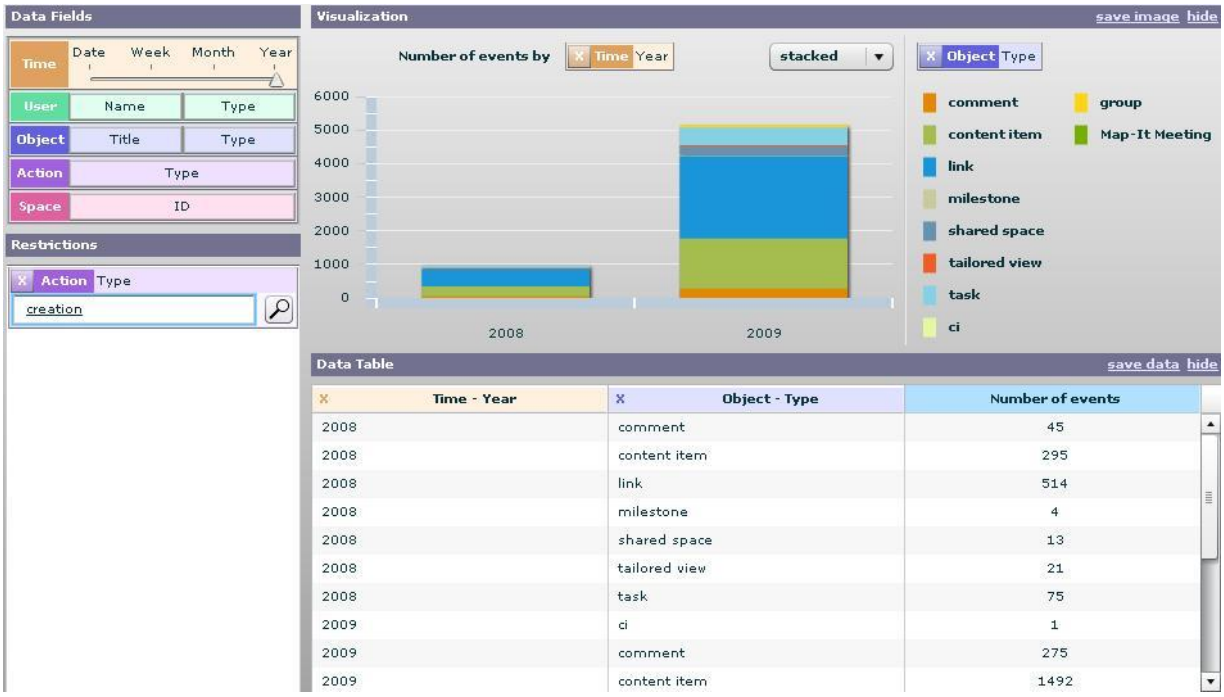


Figure 19: Stacked diagram.

Title: Downloading a predefined query.

Unique identifier: UC-VA-4-2009-09

Related Usage Scenarios: US-VA-2-2009-09

Actors: Users of KPE, mainly teachers, students, course administrators, and professionals.
Description: Downloading a query which has been saved in the system previously.

Pre-conditions: User is logged into the KPE.

Post-conditions: Visual Analyser application opens in a browser window and some of the previously stored predefined queries have been loaded.

Nominal scenario:
Step 1. The user clicks on the link "Predefined queries".
Step 2. The system opens a pop-up menu with available queries, stored in the system.
Step 3. The user chooses the query he wants to execute and confirms the choice by clicking OK.
Step 4. The system loads the query and presents the result and its visualization.

Title: Saving the query as a predefined query.

Unique identifier: UC-VA-5-2009-09

Related Usage Scenarios: US-VA-2-2009-09

Actors: Users of KPE, mainly teachers, students, course administrators, and professionals.
Description: Saving a query as one of the predefined queries for future use.

Pre-conditions: User is logged into the KPE.

Post-conditions: Visual Analyser application opens in a browser window and the user has formulated a saved a new query.

Nominal scenario:
Step 1. The user clicks on the link "save the query".
Step 2. The system adds the formulated query to the set of predefined queries.

Title: Managing predefined queries.

Unique identifier: UC-VA-6-2009-09

Related Usage Scenarios: US-VA-2-2009-09

Actors: Users of KPE, mainly teachers, students, course administrators, and professionals.
Description: Editing or deleting predefined queries.

Pre-conditions: User is logged into the KPE.

Post-conditions: Visual Analyser application opens in a browser window and some modified queries are stored in the system.

Nominal scenario:
Step 1. The user clicks on the link "Predefined queries".
Step 2. The system opens a pop-up menu with available predefined queries.
Step 3. The user browses the queries using usual keyboard keys, for each one (s)he can chose either to click on "edit" or "delete".
Step 4. If "edit" is pressed, the system loads the query and shows its result, allowing the user to modify parameters of the query.
Step 5. The user modifies parameters of the query as desired, and presses OK button of the pop-up when finished.
Step 6. The query is saved in the system.

Title: Saving the result of the query.

Unique identifier: UC-VA-7-2009-09

Related Usage Scenarios: US-VA-2-2009-09

Actors: Users of KPE, mainly teachers, students, course administrators, and professionals.
Description: Saving the result of the query and its visualization in a file.

Pre-conditions: User is logged into the KPE.

Post-conditions: Visual Analyser application opens in a browser window and a result of a query has been saved.

Nominal scenario:
Step 1. The user clicks on the link "save data" on the Data Table area if the result table needs to be saved.
Step 2. The system opens a pop-up allowing choosing the location for the file.
Step 3. The user chooses file name and a location on the disk.
Step 4. The system creates a text file on the disk containing the result of the query (a subset of the log satisfying conditions of the query).
Step 5. The user clicks on the link "save image" on the Visualization area if the visualization has to be saved.
Step 6. The system opens a pop-up allowing choosing the location for the file.
Step 3. The user chooses file name and a location on the disk.
Step 4. The system creates a PNG file on the disk containing the visualization of the result of the query.

Title: Defining groups of users for a query.

Unique identifier: UC-VA-8-2009-09

Related Usage Scenarios: US-VA-3-2009-09

Actors: User of the KPE, mainly teachers, students, course administrators, and professionals.
Description: Formulating a query comparing activities of groups of users, groups being defined in ad hoc manner.

Pre-conditions: User is logged into the KPE.

Post-conditions: Visual Analyser application opens in a browser window.

Nominal scenario:
Step 1. The user clicks on the "Define group" button below the Data Fields area.
Step 2. The system opens a pop-up asking to give a name to the group.
Step 3. The user types a name for the group.
Step 4. The system opens a pop-up listing the user names.
Step 5. The user chooses members of the group by clicking on the user name, then on "Add" button. When the group is complete, the user clicks on "Done" button.
Step 6. The system returns to the main Visual Analyser screen, the defined group added to the query parameters.

3 Technical design

3.1 Software architecture

The Visual Analyser tool is designed as an independent Web Application, packaged in a Web Archive (WAR), with all its server components, based on Apache Axis and client components, based on Adobe Flex.

3.1.1 Architecture details

Visual Analyser is designed as a client-server application, the client handling the visual interaction and the server managing query mapping and processing.

The client of Visual Analyser is designed as a web-application (integrated part of KPE), in charge of interaction with the user: presentation of the visual schema, transformation of users' actions into an analytical query and presentation of query result visually.

The server is designed to map the query, formulated by the user into the format suitable for the evaluation service (see details below), and to map the result into a format suitable for the client, for visualization. Only the client part of the tool is available to the users.

To evaluate the query, Visual Analyser consumes the services described above in section 3.3. The Figure 20 shows the schema of interaction between the user, the client (interface), the server and the KP-Lab Knowledge Analysis Services (KAS).

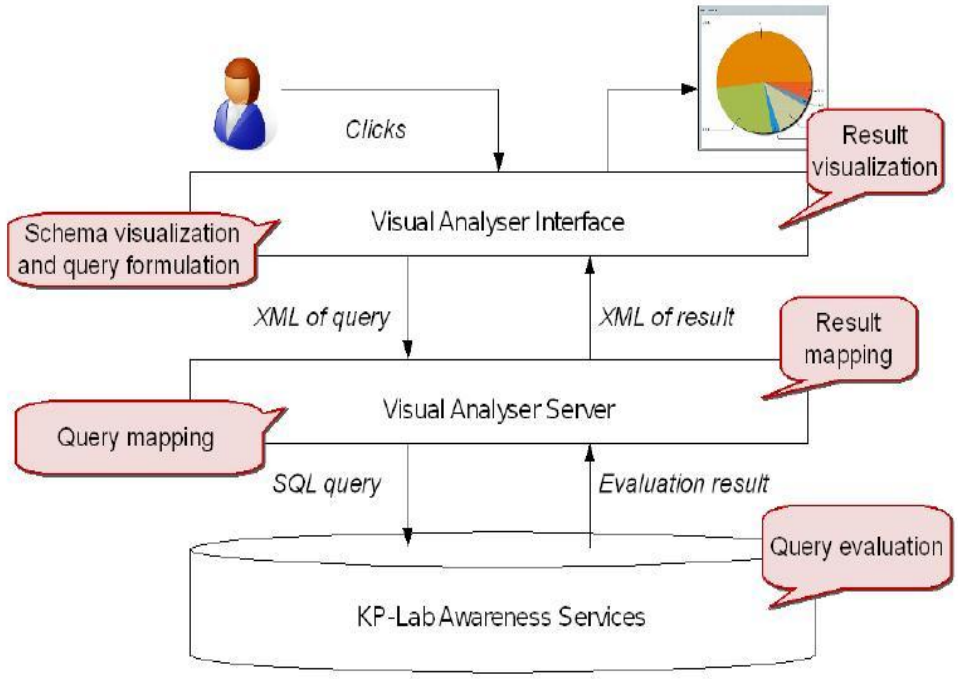


Figure 20. Visual Analyser interaction sequence

3.1.2. Implementation details and internal web services

The client of the VA tool is an Adobe Flex component, implementing VA interface, running in a browser.

The web service, provided by the client component of VA, has the following signature:

```
Object VAClient (Object query)
```

query parameter is an object, encapsulating users clicks. Users only access VA through its interface, so they do not have to handle the *query* parameter correct building, it is done by the tool.
The output is also an object, containing query result as visualized by the tool, either as a table, or a line-chart, etc.

Thus, VA client component transforms the user's clicks into a query saved in a XML string, and sends it to the VA server component. This XML string of query encapsulates the parameters of the query (classifier, measure and operation) and typically looks as follows:

```
<query>
  <classifier>
          <function name="Action"/>
  </classifier>
  <measure>
          <function name="EventId"/>
  </measure>
  <operation name="count"/>
</query>
```

VA server component is a Java program, handling query mapping into an input for the Knowledge Analysis Services (KAS) Aggregation Service, and the mapping of the result into a visualizable format (an XML string again).

The web-service provided by VA server side, which maps the XML string of query into an input for the KP-Lab Knowledge Analysis Services (KAS) Aggregation Service, then sends it, gets the result back and returns it as an XML string, has the following signature:

```
String VAServer (String xmlQueryString)
```

where the input parameter *xmlQueryString* is a user's query in XML form (as an example above), received from the client component of VA.
The *output string* of XML has typically this look:

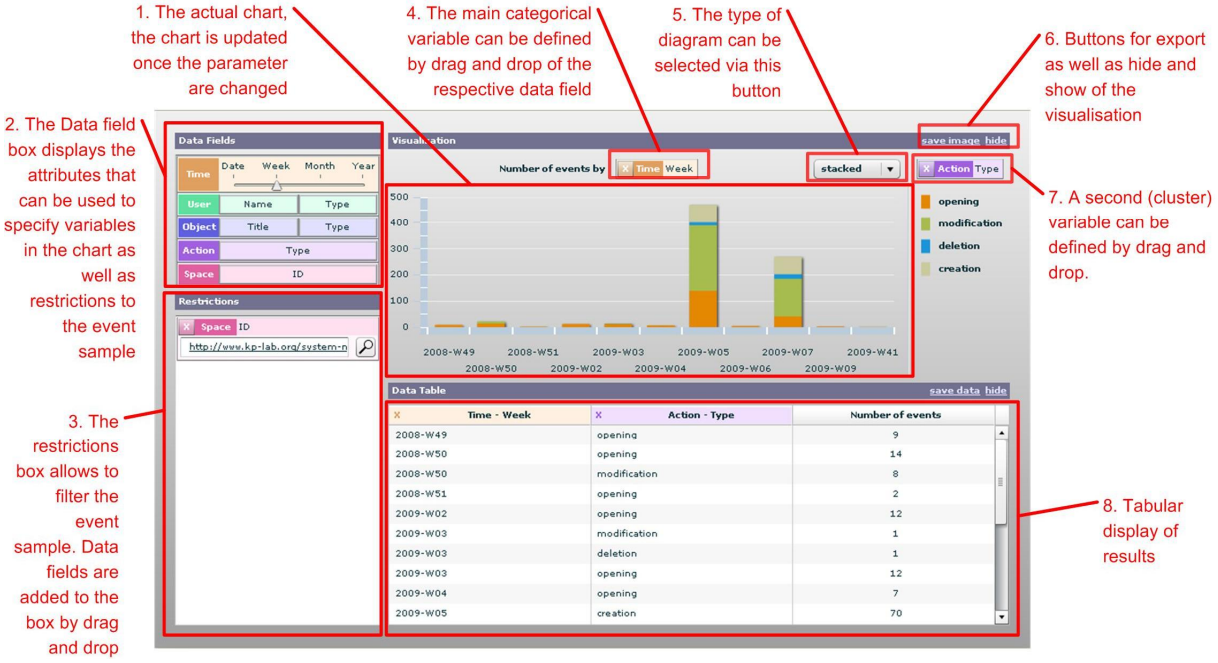
```
<Events>
          <Event>
                    <Classifier>creation</Classifier>
                    <AggResult>2503</AggResult>
          </Event>
          <Event>
                    <Classifier>deletion</Classifier>
                    <AggResult>2</AggResult>
          </Event>
          <Event>
```

```
                    <Classifier>modification</Classifier>
                    <AggResult>595</AggResult>
            </Event>
</Events>
```

This web-service is consumed by VA client component.

3.2 GUI

The VA interface includes a Data Fields area, a Visualization area, a Data Table area, and a Restrictions area. In the Data Fields area the attributes used as criteria for analysis are listed. The Visualization area is used to present the result of the query to the user. The Data Table area presents the result in a tabular form. Finally, the Restriction area allows the user to filter the result by chosen attributes values.



3.3 Collaboration with other tools

The Visual Analyser tool interoperates with the Aggregation Service from Knowledge Analysis Services (KAS) in order to evaluate users requests. The Aggregation service takes as an input a set of parameters of the query, which has been chosen and sent by the user through Visual Analyser. Then the query is evaluated over the log data, and the result is sent to the Visual Analyser in order to be visualized and presented to the user.

Here is the Aggregation Service interface used by Visual Analyser:

```
public String[][] getAggregations(String [] groupBy, Property[]
restrictions, String operation, String operationAttribute, int from,
int maxEvents);

The "groupBy" parameter can accept the following values:

   1. subjectId
   2. subjectType
   3. objectId
   4. objectType
   5. actionType
   6. time
   7. SSPID
   8. User (this is user name)
   9. Title
   10.     eventId - used to count all the events.

The used value for operation is COUNT. The values of operationAttribute
is "eventId".
```

Currently, Visual Analyser is integrated into KPE in a loose way, as a plug-in.

4 References

[1] D4.1.1 Description of Work for Months 37-54. http://www.kp-lab.org/intranet/work-packages/wp1/description-of-work/dow4-1-1/DoW%204.1.1.3.zip/view

Annex 8. Visual Model editor specifications

27490

KP-LAB

Knowledge Practices Laboratory

Integrated Project

Information Society Technologies

D6.8 M46 specification of end-user applications – Visual Model Editor

Due date of deliverable: 30/11/2009
Actual submission date:

Start date of project: 1.2.2006 Duration: 60 Months

Organisation name of lead contractor for this deliverable: EVTEK

Revision [1.0]

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission	
RE	Restricted to a group specified by the consortium (including the	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Contributors:

Name	Partner organisation	Email
Vasko Tchoumatchenko	TUS	
Tania Vasileva	TUS	
Christoph Richter	FH-OÖ	
Heidrun Allert	FH-OÖ	
Eva Zoeserl	FH-OÖ	

Version history:

Version	Date	Author(s)	Description

Table of Contents

1 Introduction

1.1 Purpose

This document describes new functionality of the Visual Model Editor (VME) tool. It consists of use cases, mock-up screens; data model and describes the required modification of the corresponding services.

1.2 Scope of the software

The Visual Model Editor (VME) is a tool for collaborative creation of visual models based on visual modelling languages.

1.3 Roadmap of the tool

The updated roadmap of the Visual Model Editor is as follows:
Iteration 1: Feasibility study, prototyping. Release v1.0
Milestones: Release M28.

Iteration 2: Basic VME functionality (creating, editing, annotating, browsing visual models). Release v2.0
Milestones: M21 Specification, M28 Prototype, M32 Stable release

Iteration 3: Advanced VME functions, integration with the Visual Modeling Languages Editor. Release 2.1
Milestones: M33 Updated software requirements specification (the present document), M35 prototype, M36 stable release.

Iteration 4: Maintenance release, which will include the new functionality described in this document. Release 2.2.
Milestones: M46 Specifications, M48 Production release.

Iteration 5: Final release with improvements and bug fixes of v2.2 based on user trials. Release 3.0.
Milestones: M56 Release.

1.4 What is new in this version?

Functionality for retrieving and analysing information on activities performed on a visual model.

The decision to integrate timeline analysis for visual models and visual modeling languages into the corresponding tools (VME and VMLE) instead of using the timeline based analyzer (TLBA) only is motivated by the need to trace the evolution of the internal structure of these types of content items in detail. While the TLBA is aimed to analyze the

evolution of a set of content items and their interrelations against the contributors involved, the timeline analysis for VMs and VMLs provides a look into the evolution of visual models and visual modeling languages as complex and semantically rich entities. As a consequence the requirements for the timeline analysis for visual models and modeling languages differ from those of the TLBA. While the TLBA emphasizes phenomena such as authorship, re-uptake of content items, referencing and annotation, the timeline analysis for visual models and modeling languages focuses on the direct comparison of structure and contents of a model or language at various points in time.

1.5 Definitions

Visual model (VM) – two-dimensional graph-structured visual representation of a certain object of interest such as flow-charts, argument-graphs, organigrams, decision trees, program logic models, use case diagrams, conceptual maps, etc.

Visual modelling language (VML) – formal specification, which defines the syntax and visual appearance of a model and the semantics of modelling elements used.

2 User requirements

2.1 KP-Lab empirical research

The tool functionalities specified in this document will be used in the following KP-Lab empirical research cases [2]:

Case 3: Conceptual modelling in Design Projects

2.2 Driving Objectives and High-Level Requirements

The specifications presented in this document are related to the following driving objectives (DO) and high-level requirements (HLR) [1]:

DO9	Users are provided with history on content development and work process advancement
HLR9.1	Users can track the evolution and changes of knowledge objects and find out their authors and contributors (sequences of performed steps in time, incl. versioning)
HLR9.2	Users are provided with customized summaries about the knowledge objects available within the shared environment (e.g. users might request an overview of the tasks completed within the last 2 weeks or the interactions of people within a shared workspace)
DO13	Users can collect data about activities and interactions systematically and over longer periods of time
HLR13.5	Users can customise the way they want to retrieve wanted data for analysis.
HLR13.6	Users are provided with summative information on performed actions (e.g. added comments, created tasks, modifications in metadata, background materials for decisions, etc.). (Rephrased)

2.3 System usage scenarios

US-VM(L)E-5-2009-09: Export of visual models

US-VM(L)E-6-2009-09: Retrieving information on activities performed on a visual model (history)

2.4 Use cases

The VME use cases, described in this document, allow users to explore information about the activities performed on a certain visual model and to trace the evolution of the model over time1.

2.4.1 Actors

VME users – both students and researchers.

2.4.2 Use Cases

UC-VME-9-2009-11 - Retrieving information on activities performed on a visual model
UC-VME-10-2009-11 - Export of visual models

2.4.3 Detailed Use Cases

2.4.3.1 UC-VME-9-2009-11 Retrieving information on activities performed on a visual model

Title: Retrieving information on activities performed on a visual model (history)

Unique identifier: UC-VME-9-2009-11

Related System Usage Scenarios: US-VM(L)E-6-2009-09: Retrieving information on activities performed on a visual model (history)

Actors: KPE user

Description:

The idea of this system usage scenario is to allow users to explore information about the activities performed on a certain visual model and to trace the evolution of the model over time.

The timeline based view on a visual model can be launched via the context menu of the visual model in content or tailored view. Once the user selects “view object’s timeline” a pop-up window appears on to of the current view.
The user can define the timeframe s/he is interested in and select single events for closer inspection. For every event and image of the visual model at that particular point as well as information about the kind of changes made are provided.

Pre-conditions: Shared space with at least one visual model.

Post-conditions: none

Nominal scenario:

3. The user right-clicks on the visual model item in the content view or tailored view and selects from the context-menu “view object’s timeline”.
4. A pop-up window, similarly to the navigator tool, appears on top of the current view.
5. The user can select a particular timeframe and scroll through the timeline. S/he can also select a particular event and an image of the visual model at that point in time is provided together with additional information about the changes made.

Non-Functional Requirements:
none

Screen mock-ups:

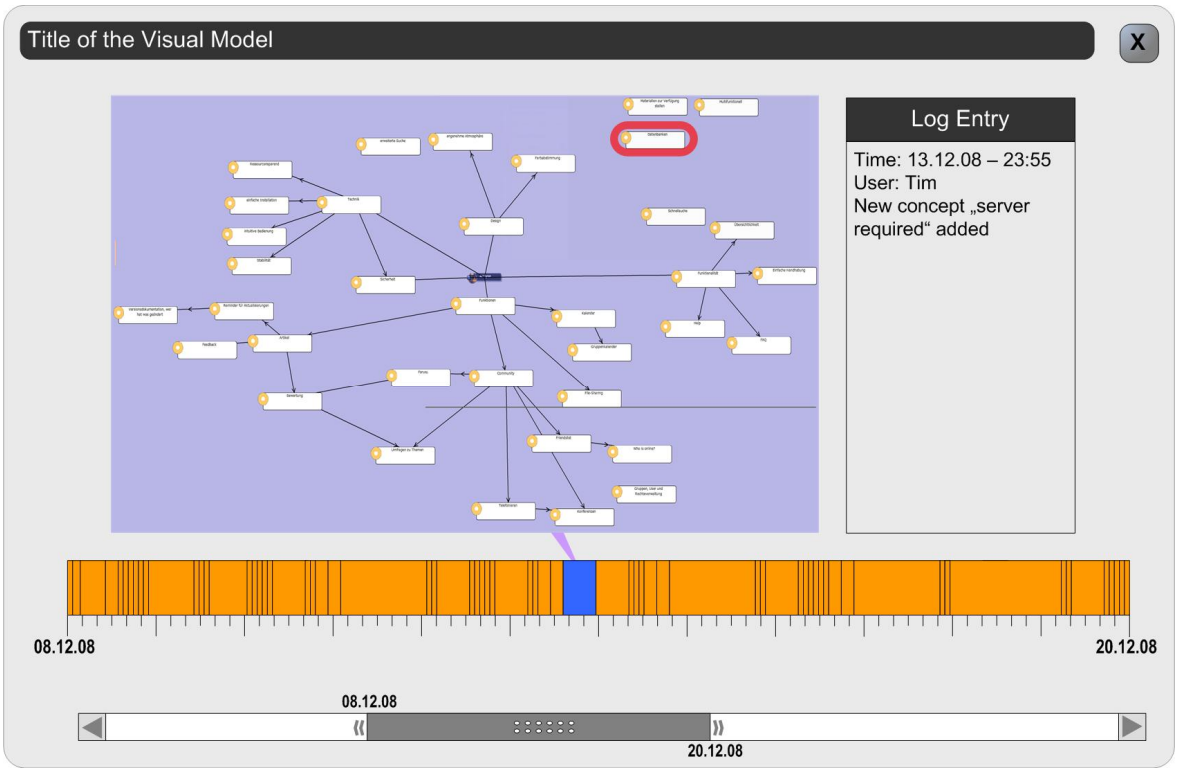


Figure 1. User interface sketch for timeline view. A slider with zooming functionality is used to specify the actual timeframe. On the corresponding timeline each event is marked as a bar, whereby the width of the bar marks the time that elapsed since the last change has been made, i.e. narrow bars indicate that changes occurred rapidly. In this case the user is provided with for only one event at a time, but s/he can “scroll” through events by ”moving” over the timeline.

2.4.3.2 UC-VME-10-2009-11 Export of visual models

Title: Export of visual models

Unique identifier: UC-VME-10-2009-11

Related System Usage Scenarios:

US-VM(L)E-5-2009-09: Export of visual models

Actors: KPE user

Description:

The user wants to export a visual model from the KPE for further analysis. The export is supposed to provide aggregated information about the structure and contents of a model.

Pre-conditions: Shared space with at least one visual model.

Post-conditions: none

Nominal scenario:

9. The user clicks on the export button next to the title of the visual model s/he wants to export
10. The system presents the exported model in textual format.
11. The user copy/pastes the text into a spreadsheet.

Non-Functional Requirements:

No synchronization is required.

Screen mock-ups:

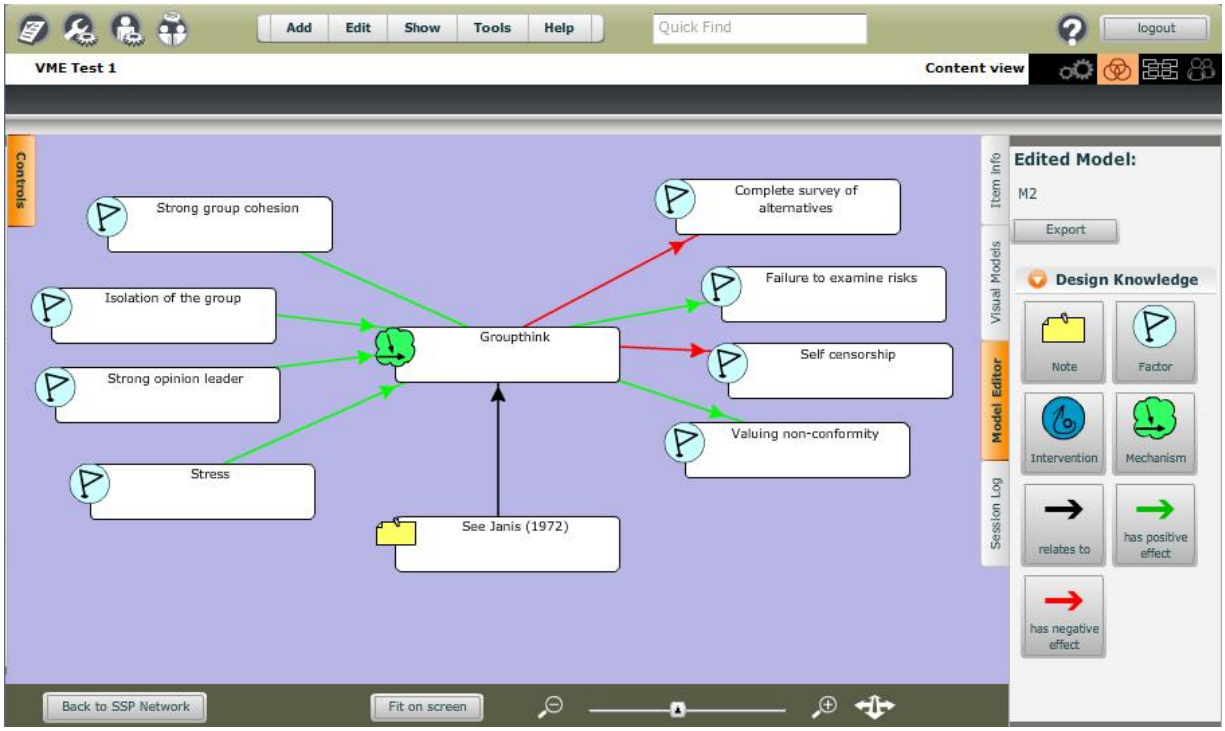


Figure 2. VME User Interface – the Export button is under the model title.

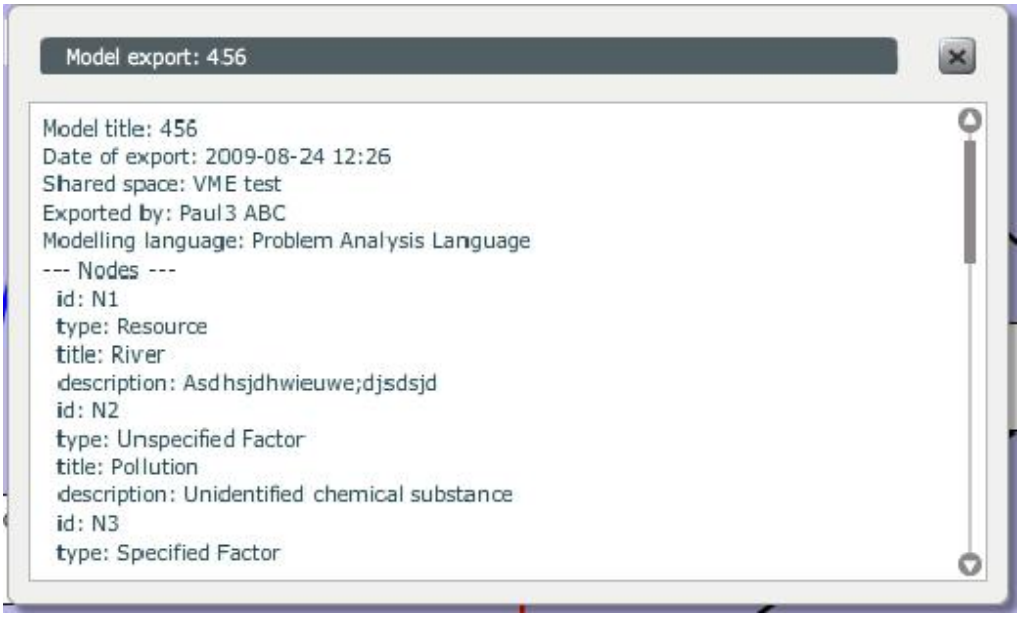


Figure 3. Model Export – Pop-up window containing textual description of the visual model.

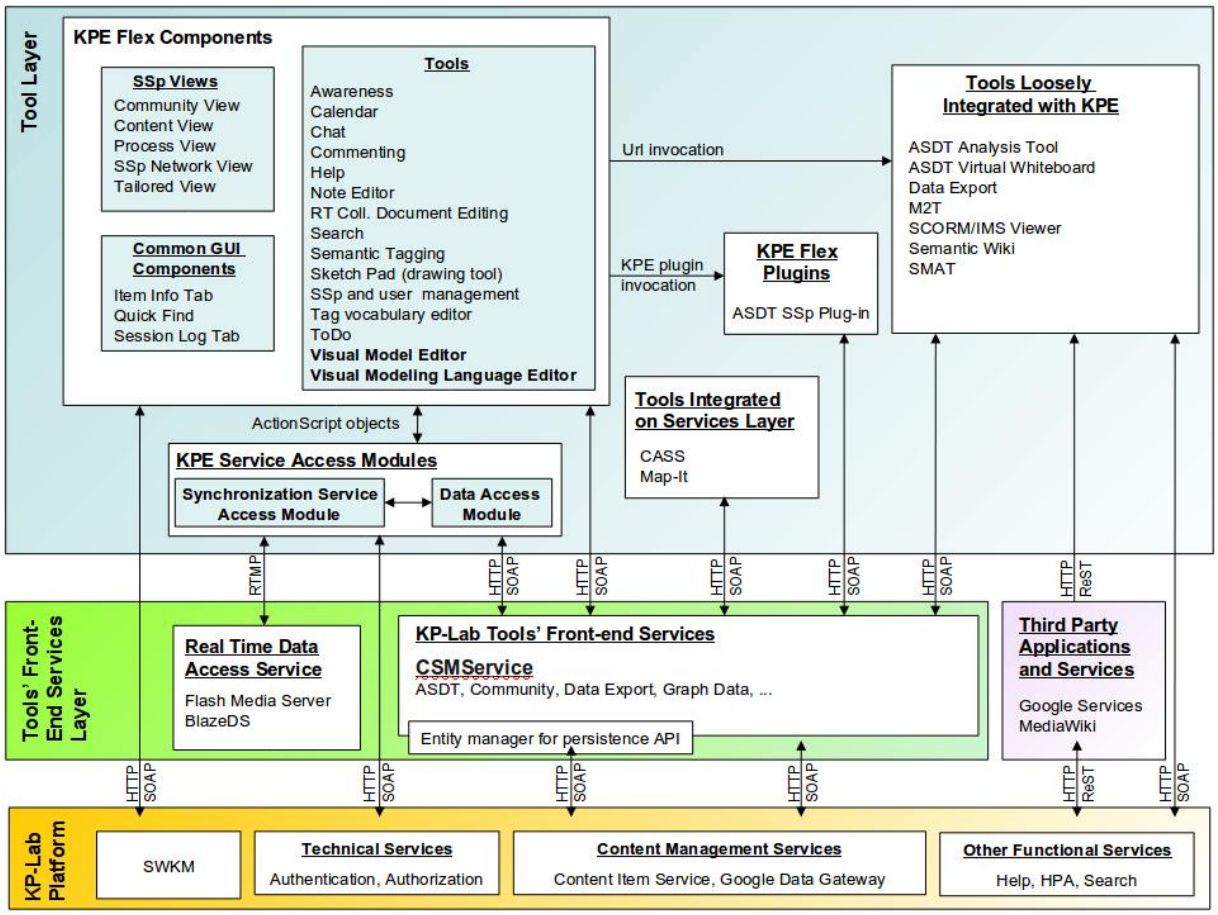
3 Technical design

3.1 Software architecture

The VME tool consists of VME Client (part of the KPE GUI) and CSM Service.

For the development of the VME analytical functionality the CSMservice API is modified to include additional information.

The VME history log records are stored in the HPA awareness repository.



3.2 GUI

- The screen design is shown on figure 2 and 3.
- Web Services used – CSMService, HPA awareness service

3.3 Front end services

The following CSMService operations are related to the VME analytical functions:

createConcept, createRelation, deleteConcept, delateRelation, modifyConcept, modifyRelation, exportModel.

For each of them the API is updated to include information, required for the recording of model history. The added information includes the full name of the user who invoked the operation, the visual model URI, the shared space URI and the coordinates of the visual model elements.

The exportModel operation is specifically developed to support the two use cases: UC-VME-9-2009-11 - Retrieving information on activities performed on a visual model (history) and

UC-VME-10-2009-11 - Export of visual models.

The request format is as follows:

```
   <soapenv:Body>
      <ser:exportModel>
         <exportModelRequest>
            <sharedSpaceUri>URI of the shared space</sharedSpaceUri>
            <sspName>Shared space name</sspName>
            <user>uer name</user>
            <visualModelUri>viusal model URI</visualModelUri>
         </exportModelRequest>
      </ser:exportModel>
   </soapenv:Body>
</soapenv:Envelope>
```

The response consists of two parts: textual description of the visual model at the moment the operation was invoked and history of all model changes.

Model Export format

Header – model title, VML used for the model, shared space name, date of export

Nodes – list of model nodes with their properties

Links – list of model links with their properties

Incidence matrix – describes the model's graph

Examle:

Model title: M2
Date of export: 2009-11-07 22:06
Shared space: VME Test 1
Exported by: Vasko Tchoumatchenko
Modelling language: Design Knowledge Mapping Language
--- Nodes ---
id: N1
type: Factor
title: Strong group cohesion
description:

id: N2
type: Factor
title: Self censorship
description:

...

--- Links ---
id: L1
type: has positive effect
title: has positive effect
description:

id: L2
type: has positive effect
title: has positive effect
description:

id: L3
type: has negative effect
title: has negative effect
description:

...

--- Incidence Matrix ---
from\to,N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,
N1,,,,,,,,L6 ,,
N2,,,,,,,,,,
N3,,,,,,,,L1 ,,
N4,,,,,,,,,,
N5,,,,,,,,L5 ,,
N6,,,,,,,,L8 ,,
N7,,,,,,,,,,
N8,,,,,,,,L7 ,,
N9,,L3 ,,L2 ,,,L9 ,,,L4 ,
N10,,,,,,,,,,

Model history format

The model history is presented as list of records. Each record corresponds to one of the following model modifications:

- create (concept or relationship)
- delete (concept or relationship)
- modify property

The record format is the following:
Event number; Date; User; Object ID ; Object Name; Object Type; Action; Details

Examples

Typical concept creation records:

1;2009-11-16 18:15:44;User Paul;N1;Factor;Factor;CREATE;
2;2009-11-16 18:15:55;User Paul;N1;Factor;Factor;DELETE;
3;2009-11-16 18:16:14;User Paul;N2;Factor;Factor;CREATE;

Property modification records:

4;2009-11-16 18:17:53;User Paul;N2;Factor: Technische Anforderungen;Factor;MODIFY;title = Factor: Technische Anforderungen
5;2009-11-16 18:17:54;User Paul;N2;Factor: Technische Anforderungen;Factor;MODIFY;description = PC, Micro Kopfhoerer, Internetverbindung

Relationships creation records:

15;2009-11-16 18:23:11;User Paul;L1;Link from Factor: Technische Anforderungen (N2) to Factor: Ort (N3);has positive effect;CREATE;
16;2009-11-16 18:23:31;User Paul;L1;Link from Factor: Technische Anforderungen (N2) to Factor: Ort (N3);has positive effect;DELETE;
17;2009-11-16 18:23:39;User Paul;L2;Link from Factor: Technische Anforderungen (N2) to Factor: Zeitersparnis (N4);has positive effect;CREATE;

3.4 Collaboration with other tools

The VME client is integrated in the KPE content view. The model languages used in VME are developed within the Visual Modeling Language Editor tool. The VME timeline view is integrated with the Timeline Based Analyser tool.

4 References

[1] D2.4 Driving Objectives and High-level Requirements for KP-Lab Technologies (revised version). http://www.kp-lab.org/intranet/work-packages/wp2/result/D2.4_Resubmission_290809.doc

[2] D3.2 A comprehensive research strategy (an updated and revised version for years 4 and 5). http://www.kp-lab.org/intranet/work-packages/wp1/submission-to-commission-23-9.2009/D3.2-Rev-Research-Strategy_September_submitted.doc

Annex 9. Visual Modelling Language Editor specifications

27490

KP-LAB

Knowledge Practices Laboratory

Integrated Project

Information Society Technologies

D6.8 M46 specification of end-user applications – Visual Modelling Languages Editor

Due date of deliverable: 30/11/2009
Actual submission date:

Start date of project: 1.2.2006 Duration: 60 Months

Organisation name of lead contractor for this deliverable: EVTEK

Revision [1.0]

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission	
RE	Restricted to a group specified by the consortium (including the	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Contributors:

Name	Partner organisation	Email
Vasko Tchoumatchenko	TUS	
Tania Vasileva	TUS	
Christoph Richter	FH-OÖ	
Heidrun Allert	FH-OÖ	
Eva Zoeserl	FH-OÖ	

Version history:

Version	Date	Author(s)	Description

Table of Contents

1 Introduction

1.1 Purpose

This document describes the functionality of the Visual Modelling Language Editor. It consists of use cases, UI screens and describes the required KP-LAB services.

1.2 Scope of the software

The Visual Modelling Language Editor (VMLE) is a tool for collaborative development of visual modelling languages to be used in conjunction with the Visual Model Editor (VME).

1.3 Roadmap of the tool

The updated roadmap of the Visual Modelling Language Editor is as follows:

Iteration 1: Design, prototyping. Release v1.0
Milestones: M33 Specification, M36 Prototype

Iteration 2: Basic VMLE functions - browsing available visual modelling languages (VMLs), creating, copying, commenting and editing VMLs. Adding, editing, commenting and deleting concepts and relations to a VML. Release v2.0
Milestones: M37 Updated software requirements, M39 prototype, M40 stable release.

Iteration 3: Advanced VMLE functions (comparing visual modelling languages, updating a visual model according to changes in the VML), Release 3.0

Iteration 4: Maintenance release, which will include optimized data model and front-end service operations in order to address functional and performance deficiencies of the previous releases. Release 3.2.
Milestones: M46 Specifications, M48 Production release.

Iteration 5: Final release with improvements and bug fixes of v3.2 based on user trials. Release 4.0.
Milestones: M56 Release.

1.4 What is new in this version?

The principal difference, compared to the previous VMLE prototypes, is a new data model, architecture and implementation of the front end service. The redesign is required because the previous prototypes were not able to achieve functional completeness and acceptable performance.

From a functional point of view a new history analysis and export capabilities will be implemented.

The decision to integrate timeline analysis for visual models and visual modeling languages into the corresponding tools (VME and VMLE) instead of using the timeline based analyzer (TLBA) only is motivated by the need to trace the evolution of the internal

structure of these types of content items in detail. While the TLBA is aimed to analyze the evolution of a set of content items and their interrelations against the contributors involved, the timeline analysis for VMs and VMLs provides a look into the evolution of visual models and visual modeling languages as complex and semantically rich entities. As a consequence the requirements for the timeline analysis for visual models and modeling languages differ from those of the TLBA. While the TLBA emphasizes phenomena such as authorship, re-uptake of content items, referencing and annotation, the timeline analysis for visual models and modeling languages focuses on the direct comparison of structure and contents of a model or language at various points in time.

1.5 Definitions

Visual model (VM) – two-dimensional graph-structured visual representation of a certain object of interest such as flow-charts, argument-graphs, organigrams, decision trees, program logic models, use case diagrams, conceptual maps, etc.

Visual modelling language (VML) – formal specification, which defines the syntax and visual appearance of a model and the semantics of modelling elements used.

2 User requirements

2.1 KP-Lab empirical research

The tool functionalities specified in this document will be used in the following KP-Lab empirical research cases [2]:

Case 3: Conceptual modelling in Design Projects

2.2 Driving Objectives and High-Level Requirements

The specifications presented in this document are related to the following driving objectives (DO) and high-level requirements (HLR) [1]:

DO2	Users are provided with a customizable tool suite for working on shared artefacts
HLR2.2	The tools are generic enough to allow users to use them in various ways of working, e.g., personalization of the tools and adaptation to different domain knowledge.
HLR2.5	Users can import and export artefacts and access data across the tools integrated or connected to the KP Environment.
DO4	Users can describe the semantics of artefacts and their relations
HLR4.3	Users can work with multiple conceptual models (vocabularies or visual modelling languages) in parallel.
HLR4.4	Users are able to save and share conceptual models (e.g. vocabularies and visual models)
DO6	Users can develop and use their own conceptual models
HLR6.1	Users can modify existing or create new visual modelling languages, ontologies and vocabularies.
HLR6.2	Users can refine, make proposals and give objections on modelling languages, ontologies and vocabularies while working on the respective knowledge representations and visual models.
HLR6.3	Users can share and integrate different visual modelling languages, ontologies and

	vocabularies.
DO9	Users are provided with history on content development and work process advancement
HLR9.1	Users can track the evolution and changes of knowledge objects and find out their authors and contributors (sequences of performed steps in time, incl. versioning)
HLR9.2	Users are provided with customized summaries about the knowledge objects available within the shared environment (e.g. users might request an overview of the tasks completed within the last 2 weeks or the interactions of people within a shared workspace)
DO13	Users can collect data about activities and interactions systematically and over longer periods of time
HLR13.5	Users can customise the way they want to retrieve wanted data for analysis.
HLR13.6	Users are provided with summative information on performed actions (e.g. added comments, created tasks, modifications in metadata, background materials for decisions, etc.). (Rephrased)

2.3 System usage scenarios

US-VM(L)E-2-2009-09 Creating and editing a visual modelling language
US-VM(L)E-3-2009-09 Migrating to a new version of a Visual Modelling Language
US-VM(L)E-5-2009-09: Export of visual models (extended to visual modelling languages)

US-VM(L)E-6-2009-09: Retrieving information on activities performed on a visual model (extended to visual modelling languages)

2.4 Use cases

This section presents the use cases for the visual model editor.

2.4.1 Actors

VMLE users – both students and researchers.

2.4.2 Use Cases

UC-VMLE-1 Browsing available VMLs
UC-VMLE-2 Creating a VML
UC-VMLE-3 Copying a VML
UC-VMLE-4 Opening VML for editing
UC-VMLE-5 Adding a concept
UC-VMLE-6 Adding an attribute
UC-VMLE-7 Adding a link
UC-VMLE-8 Deleting concepts, attributes and links
UC-VMLE-9 Updating concepts and attributes
UC-VMLE-10 Check VML consistency
UC-VMLE-11 Retrieving information on activities performed on a VML (history)
UC-VMLE-12 Export of VML

2.4.3 Detailed Use Cases

2.4.3.1 UC-VMLE-1: Browsing available VMLs

Title: Browsing the set of available VMLs

Unique identifier: UC-VMLE-1

Related Usage Scenarios: US-VM(L)E-2-2009-09 Creating and editing a visual modelling language

Actors: The users of the visual modelling language editor.

Description: The user wants to see the visual modelling languages available in a particular shared space.

Pre-conditions: The user has entered a shared space.

Post-conditions: none

Nominal scenario:

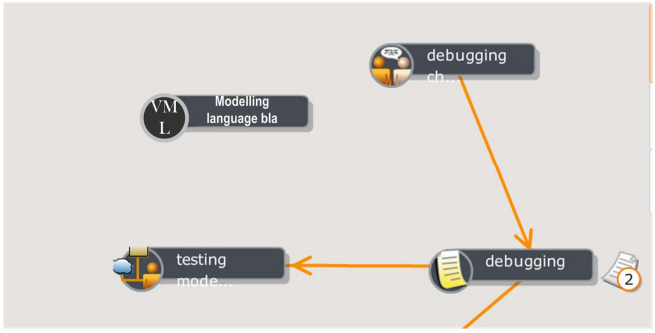
6. The user enters the content view or a tailored view.
7. The system shows the content items, available in the selected view.
8. The user identifies the visual modelling languages by their specific icon.

Extensions:

9. The user clicks the right mouse button and selects the function “add visual model” from the pop-up menu.
10. The system shows a form, which contains a list of the available visual modelling languages and a short description of each language.

Non-Functional Requirements: None

Screen mock-ups:



Content view with visual modelling language.

2.4.3.2 UC-VMLE-2 Creating a VML

Title: Creating a Visual Modelling Language

Unique identifier: UC-VMLE-2

Related System Usage Scenarios: US-VM(L)E-2-2009-09 Creating and editing a visual modelling language

Actors: VMLE users

Description: The user wants to create a new visual modelling language.

Pre-conditions: The user has entered a shared space and is in the content view or a tailored view.

Post-conditions: A new VML, together with the corresponding content item is created.

Nominal scenario:

- The user clicks the right mouse button selects the “add content item” entry and then the function “add visual modelling language” from the pop-up menu.
- The system shows a form with fields for title and description of the new visual modelling language.
- The user fills in the form for the modelling language and clicks on “ok” button.
- The system creates a new content item and adds it to the current shared space.

Non-Functional Requirements: none

Screen mock-ups: none

2.4.3.3 UC-VMLE-3 Copying a VML

Title: Copying a Visual Modelling Language

Unique identifier: UC-VMLE-3

Related System Usage Scenarios: US-VM(L)E-2-2009-09 Creating and editing a visual modelling language

Actors: VMLE users

Description: The user wants to create a copy of an existing visual modelling language.

Pre-conditions:

The user has entered a shared space and is in the content view or a tailored view.

The user has opened a VML in the VMLE.

Post-conditions:

A new VML, together with the corresponding content item is created.

Nominal scenario:

12. The user clicks on the “Save As” button in the VMLE.
13. The system shows a form with fields for title and description of the new visual modelling language.
14. The user fills in the form for the modelling language and clicks on “ok” button.
15. The system creates a new VML and adds it to the current shared space.

Non-Functional Requirements: none

Screen mock-ups:none

2.4.3.4 UC-VMLE-4 Opening VML for editing

Title: Editing Visual Modelling Languages

Unique identifier: UC-VMLE-4

Related Usage Scenarios: US-VM(L)E-2-2009-09 Creating and editing a visual modelling language

Actors: The users of the visual modelling language editor.

Description: The user wants to work on a visual modelling language.

Pre-conditions:

The user has entered a shared space and is in the content view or a tailored view.

Post-conditions:

The modified modelling language is saved.

Nominal scenario:

- The user double-clicks on a VML or selects "open" from the VML's context menu.
- The system shows the VML tree in the VMLE view (see the mock up).

Non-Functional Requirements:

None

Screen mock-ups:

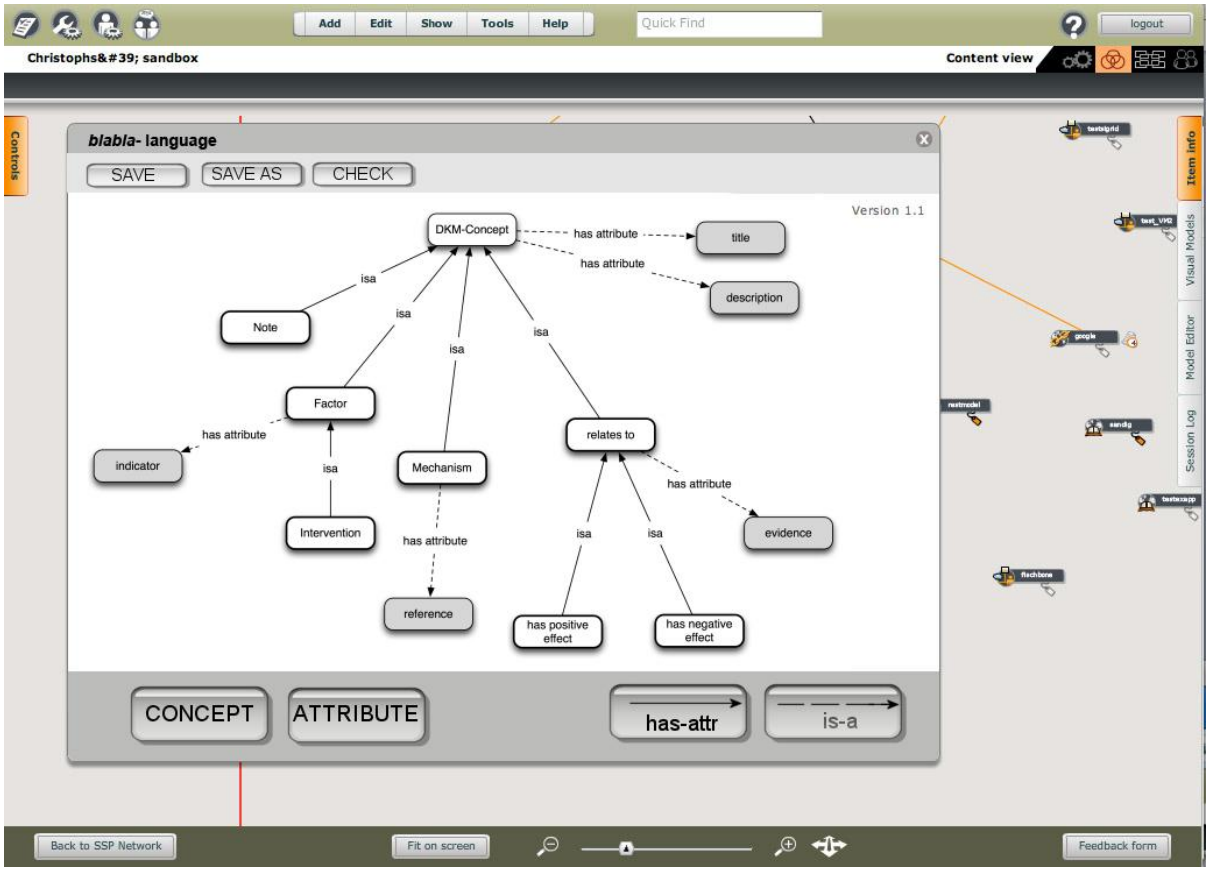


Figure 1. VMLE UI mockup

2.4.3.5 UC-VMLE-5 Adding a concept

Title: Adding a concept

Unique identifier: UC-VMLE-5

Related Usage Scenarios: US-VM(L)E-2-2009-09 Creating and editing a visual modelling language

Actors: The users of the visual modelling language editor.

Description: The user wants to add a concept to the VML

Pre-conditions:

The user has opened a VML in the VMLE.

Post-conditions:

The modified modelling language is saved.

Nominal scenario:

[2] User clicks on the "concept" button (see Figure 1).

[3] VMLE shows a form:

[4] "name" (required) - TextInput

[5] "scope note" (optional) - TextArea

[6] "vertex" - RadioButton

[7] "edge" - RadioButton

[8] "colour" - ColorPicker

[9] User fills the form and clicks OK.

[10] VMLE creates concept graph node and places it on the screen.

Non-Functional Requirements:

None

Screen mock-ups:

2.4.3.6 UC-VMLE-6 Adding an attribute

Title: Adding an attribute

Unique identifier: UC-VMLE-6

Related Usage Scenarios: US-VM(L)E-2-2009-09 Creating and editing a visual modelling language

Actors: The users of the visual modelling language editor.

Description: The user wants to add attribute to the VML.

Pre-conditions:

The user has opened a VML in the VMLE.

Post-conditions:

The modified modelling language is saved.

Nominal scenario:

11. User clicks on the "attribute" button.
12. VMLE shows a form with two fields:
    - "name" (required) - TextInput
    - "scope note" (optional) - TextArea
13. User fills the form and clicks OK

14. VMLE creates attribute graph node and places it on the screen.

Non-Functional Requirements:

None

Screen mock-ups:

2.4.3.7 UC-VMLE-7 Adding a link

Title: Adding a link

Unique identifier: UC-VMLE-7

Related Usage Scenarios: US-VM(L)E-2-2009-09 Creating and editing a visual modelling language

Actors: The users of the visual modelling language editor.

Description: The user wants to add “is-a” or “has-attribute” link.

Pre-conditions:

The user has opened a VML in the VMLE.

Post-conditions:

The modified modelling language is saved.

Nominal scenario:

Creation of “is-a” link

- User clicks on the "is-a" button.
- VMLE enters "linking" mode.
- User clicks on node of type "concept" and drags the mouse to a second node of type "concept". Nodes on type "attribute" can not be selected.
- VMLE creates "is-a" link between the selected concepts.

Creation of “has-attribute” link

- User clicks on the "has-attribute" button.
- VMLE enters "linking" mode.
- User clicks on node of type "concept" and drags the mouse to a second node of type "attribute".
  or
  User clicks on node of type "attribute" and drags the mouse to a second node of type "concept".
- VMLE creates "has-attribute" link between the selected concept and attribute.

Non-Functional Requirements:

None

Screen mock-ups:

2.4.3.8 UC-VMLE-8 Deleting concepts, attributes and links

Title: Deleting concepts, attributes and links

Unique identifier: UC-VMLE-8

Related Usage Scenarios: US-VM(L)E-2-2009-09 Creating and editing a visual modelling language

Actors: The users of the visual modelling language editor.

Description: The user wants to delete VML element.

Pre-conditions:

The user has opened a VML in the VMLE.

Post-conditions:

The modified modelling language is saved.

Nominal scenario:

- user selects Concept, Attribute or Link
- user selects Delete from the context menu or presses the Delete button from the keyboard.
- The system asks for confirmation
- If the user confirms the deletion, the system removes the selected items from the model and refreshes the screen.

Non-Functional Requirements:

None

Screen mock-ups:

2.4.3.9 UC-VMLE-9 Updating concepts and attributes

Title: Updating concepts and attributes

Unique identifier: UC-VMLE-9

Related Usage Scenarios: US-VM(L)E-2-2009-09 Creating and editing a visual modelling language

Actors: The users of the visual modelling language editor.

Description: The user wants to update VML concept or attribute.

Pre-conditions:

The user has opened a VML in the VMLE.

Post-conditions:

The modified modelling language is saved.

Nominal scenario:

- User double clicks on Concept or Attribute (or selects the graph element and chooses Modify from the context menu)
- The system locks the graph element.
- The system shows the same form which was used when creating the Concept or Attribute.
- The user modifies the form's content and clicks OK.
- The system un-locks the graph element.

Non-Functional Requirements:

None

Screen mock-ups:

2.4.3.10 UC-VMLE-10 Check VML consistency

Title: Check VML consistency

Unique identifier: UC-VMLE-

Related Usage Scenarios: US-VM(L)E-2-2009-09 Creating and editing a visual modelling language

Actors: The users of the visual modelling language editor.

Description: The user wants to validate the consistency of the created VML.

Pre-conditions:

The user has opened a VML in the VMLE.

Post-conditions:

The modified modelling language is saved.

Nominal scenario:

1. User clicks on the "Check" button.
2. VMLE shows the errors on the VML graph:
   - The problematic VML elements are highlighted (in red or similar distinctive colour).
   - The tool-tips for these VML elements are replaces with the corresponding error messages.

Non-Functional Requirements:

None

Screen mock-ups:

2.4.3.11 UC-VMLE-11 Retrieving information on activities performed on a VML

Title: Retrieving information on activities performed on a VML

Unique identifier: UC-VMLE-11

Related Usage Scenarios: US-VM(L)E-6-2009-09: Retrieving information on activities performed on a visual model (or language)

Actors: The users of the visual modelling language editor.

Description:

The idea of this system usage scenario is to allow users to explore information about the activities performed on a certain visual modelling language and to trace the evolution of the model over time. The history log is structured as table and is available to the user as part of the VMLE model export output.

Pre-conditions:

none

Post-conditions:

none

Nominal scenario:

- The user clicks on the export button
- The system presents the exported model in textual format.
- The user copy/paste the text into a spreadsheet.

Non-Functional Requirements:

None

Screen mock-ups:

none

2.4.3.12 UC-VMLE-13 Export of VML

Title:

Unique identifier: UC-VMLE-

Related Usage Scenarios: US-VM(L)E-5-2009-09: Export of visual models (or languages)

Actors: The users of the visual modelling language editor.

Description: The user wants to export a visual model from the KPE for further analysis. The export is supposed to provide aggregated information about the structure and contents of a model.

Pre-conditions:

The user has opened a VML in the VMLE.

Post-conditions:

none

Nominal scenario:

- The user clicks on the export button
- The system presents the exported model in textual format.
- The user copy/paste the text into a spreadsheet.

Non-Functional Requirements:

none

Screen mock-ups:

none

3 Technical design

3.1 Software architecture

The VMLE tool consists of VMLE Client (part of the KPE GUI) and CSM Service.

The VMLE specific log history is stored in HPA awareness service.

The VMLs are stored in a graph database.

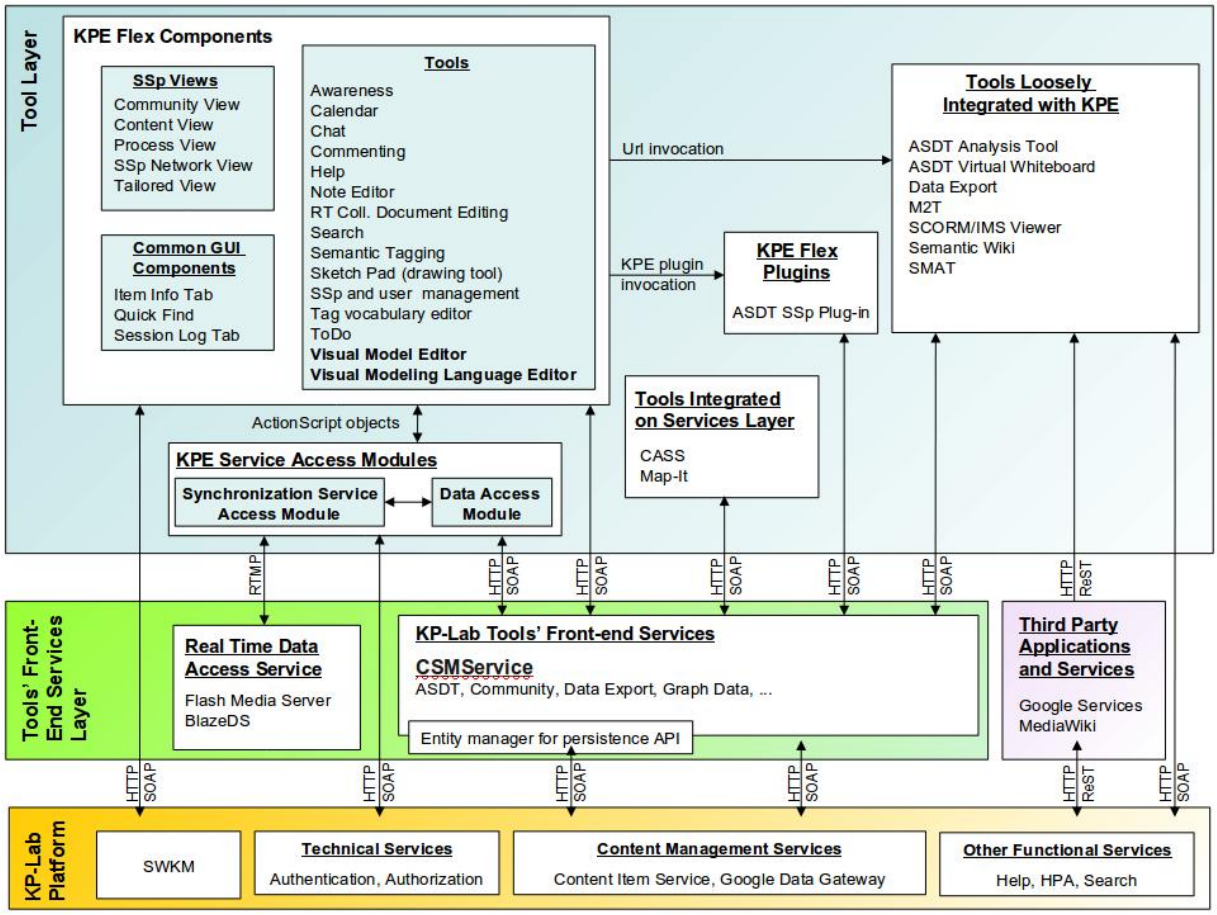


Figure 2. VMLE in the KPE Architecture

3.2 GUI

- The preliminary screen design is shown on figure 1.
- Web Services used – CSMService, HPA awareness service

3.3 Front end services

VMLE relies on the following front-end services:

- Real Time Data Access Service – for synchronization and locking
- CSMService – VML storage and retrieval, log history storage and retrieval, VML consistency checking
- ContentItem Service – for creation of VML content items.

The VMLE specific CSMService operations are described in the following technical use-cases.

3.3.1.1 Technical VMLE use cases

Open VML for editing

- In the Content View, the user double-clicks on a VML or selects "open" VML from the context menu.
- The system starts the VMLE and initializes it with the following info:
  - Shared Space URI

- VML URI (this is the URI of the VML content item)
- full name of the current user

- VMLE calls CSMService:

VmlDescription vml = getVml(String vmlUri)

The VmlDescription consists of list of nodes and links, together with properties (name, scope note, startNode, endNode, etc.) and coordinates. The detailed format is not yet specified.

- VMLE draws the VML tree on the screen.

Add "concept"

- User clicks on the "concept" button.
- VMLE shows a form:
  - "name" (required) - TextInput
  - "scope note" (optional) - TextArea
  - "vertex" - RadioButton
  - "edge" - RadioButton
  - "colour" - ColorPicker
- User fills the form and clicks OK.
- VMLE calls CSMService:

String uri = addVmlConcept(String sharedSpaceUri, String vmlUri, String userFullName , String name, String scopeNote, String colour, boolean isVertex, int xCoord, int yCoord)

where "isVertex" is true if the "vertex" RadioButton is selected.

- VMLE creates concept node display object with "uri", "name" and "scope note" properties and places it on the screen.

Add "attribute"

- User clicks on the "attribute" button.
- VMLE shows a form with two fields:
  - "name" (required) - TextInput
  - "scope note" (optional) - TextArea
- User fills the form and clicks OK.
- VMLE calls CSMService:

String uri = addVmlAttribute(String sharedSpaceUri, String vmlUri, String userFullName , String name, String scopeNote, int xCoord, int yCoord);

- VMLE creates attribute node display object with "uri", "name" and "scope note" properties and places it on the screen.

Add "is-a" link

- User clicks on the "is-a" button.
- VMLE enters "linking" mode.

- User clicks on node of type "concept" (let's call it startNode) and drags the mouse to a second node of type "concept" (endNode). Nodes on type "attribute" can not be selected.
- VMLE calls CSMService:

String uri = addVmlLink(String sharedSpaceUri, String vmlUri, String userFullName, String startNodeUri, String endNodeUri, boolean true);

- VMLE creates "is-a" link display object with "uri" property.

Add "has-attribute" link

- User clicks on the "has-attribute" button.
- VMLE enters "linking" mode.
- User clicks on node of type "concept" (conceptNode) and drags the mouse to a second node of type "attrbute" (attrbuteNode).
- VMLE calls CSMService:

String uri = addVmlLink(String sharedSpaceUri, String vmlUri, String userFullName, String startNodeUri, String endNodeUri, boolean false);

- VMLE creates "has-attribute" link display object with "uri" property.

3.3.2 Save

- User clicks on the "Save" button.
- VMLE calls CSMService:

String version = saveVml(String sharedSpaceUri, String vmlUri, String userFullName, Array nodes);

"nodes" is an Array of objects - one entry for each node from the VML tree. Each entry has three fields:
- String uri - URI of the node
- int xCoord - x coordinate of the node
- int yCoord - y coordinate of the node

- VMLE shown the returned version number somewhere on the screen.

3.3.3 Save As

- User clicks on the "Save As" button.
- The system calls the ContentItem Service createContentItem method. The contentItemType should be http://www.kp-lab.org/system-model/TLO#VisualModellingLanguage. The URI of the newly create ContentItem is stored as newVmlUri.
- VMLE calls CSMService:

saveAsVml(String sharedSpaceUri, String vmlUri, String userFullName, String newVmlUri, Array nodes);

Note: see the "Save" operation for the description of the "node" array.

- The system updates the Content View to show the new VML content item.

3.3.4 Check

- User clicks on the "Check" button.
- VMLE calls CSMService:

Array listOfErrors = checkVml(String sharedSpaceUri, String vmlUri, String userFullName);

The result contains a (possible empty) list of errors. The list entries have the following fields:

  - uri – URI of an object from the current VML;
  - message – A text, explaining the error.
- VMLE shows the errors on the VML tree:
  - First, any highlighting is removed from all VML elements;
  - The VML element, whose URIs are in the listOfErrors are highlighted (in red or similar distinctive colour).
  - The tool-tips for these VML elements are replaces with the corresponding error messages.

3.3.5 Delete

- User selects Concept, Link or Attribute.
- User selects Delete from the context menu
- The system asks for confirmation
- If the user confirms the deletion, VMLE call CSMService:

  delete(String uri, String userFullName)

- VMLE refreshes the screen

3.3.6 Modify

- User double clicks on Concept or Attribute (or selects the graph element and coosees Modify (or Open?) from the context menu)
- The system locks the graph element.
- The system shows the same form which was used when creating the Concept or Attribute.
- The user modifies the form's content and clicks OK.
- VMLE calls CSMService

updateVmlConcept(String sharedSpaceUri, String vmlUri, String userFullName , String name, String scopeNote, String colour, boolean isVertex, int xCoord, int yCoord)
or
updateVmlAttribute(String sharedSpaceUri, String vmlUri, String userFullName , String name, String scopeNote, int xCoord, int yCoord);

- The system un-locks the graph element.

3.3.6.1 VML Export format

The VML is exported as a text which describe the VML graph. It consists of:

- Header – VML title, shared space name, date of export
- Concepts – list of VML concepts
- Attributes – list of VML attributes
- Incidence matrix – describes the VML graph connectivity

3.3.6.2 VML history format

The VML history is presented as list of records. Each record corresponds to one of the following modifications:

- create concept, attribute or link
- delete concept, attribute or link
- update concept or attribute
- save – creates new VML version
- save-as – creates VML copy

The record format is the following:
Event number; Date; User; Object ID ; Object Name; Object Type; Action; Details

3.4 Collaboration with other tools

The VME client is integrated in the KPE content view. The model languages used in VME are developed within the Visual Modelling Language Editor tool.

4 References

[1] D2.4 Driving Objectives and High-level Requirements for KP-Lab Technologies (revised version). http://www.kp-lab.org/intranet/work-packages/wp2/result/D2.4_Resubmission_290809.doc

[2] D3.2 A comprehensive research strategy (an updated and revised version for years 4 and 5). http://www.kp-lab.org/intranet/work-packages/wp1/submission-to-commission-23-9.2009/D3.2-Rev-Research-Strategy_September_submitted.doc