

27490

KP-LAB

Knowledge Practices Laboratory

Integrated Project

Information Society Technologies

D 2.2 Guidelines and models on implementing design principles of KP-Lab, application scenarios and best practices, v.2

Due date of deliverable: 31.7.2007

Actual submission date: 17.9.2007

Start date of project: 1.2.2006

Duration: 60 Months

Organisation name of lead contractor for this deliverable: FH OÖ F&E, INPT

Final

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):	Heidrun Allert, Hadj Batatia, Benoit Baurens, Merja Bauters, Zvi Ben Ami, Liisa Benmergui, Reuma de Groot, Liisa Ilomäki, Kari Kosonen, Minna Lakkala, Yannick Lizzy, Hannu Markkanen, Hanni Muukkonen, Christophe Piombo, Christoph Richter
Editor(s):	Christoph Richter, Hadj Batatia, Heidrun Allert
Partner(s):	UH, EVTEK, HUJI, SILOGIC, POYRY, FH OÖ F&E, INPT
Work Package:	2 – Co-Design
Nature of the deliverable:	Report

Abstract

This deliverable provides an updated and extended description of the KP-Lab co-design framework as well as a summary of main design methods used so far. The purpose of this deliverable is to explicate the underlying rationale as well as systematic structure of the co-design process. The report starts with a brief outline of the theoretical considerations underlying the design-framework as well as a comparison with current approaches in software-engineering. Against this background, the main methodological challenges are sketched. In the main part of the report the actual co-design framework, including its guiding principles, the overall process framework as well as concrete design practices are depicted. The report ends with an outlook on the next steps to be taken.

This document replaces Deliverable 2.1 “Guidelines and models on implementing design principles in KP-Lab, application scenarios and best practice, v.1” submitted at M6.

Contents

1	Introduction.....	3
1.1	Scope and Purpose.....	3
1.2	History of the Co-Design Framework and Main Changes	4
1.3	Structure of the Document	6
2	Background.....	6
2.1	Design and the Co-Evolution of Tools and Practices.....	6
2.2	Co-Design and Software-Engineering	10
3	Challenges for Co-Design	11
4	The KP-Lab Co-Design Framework	14
4.1	Guiding Principles	14
4.2	Overall Process Framework.....	16
4.2.1	Conceptual model	19
4.2.2	Process Model	23
4.3	Emerging Design Practices	34
4.3.1	Collaborative Creation of Low-Fidelity Mock-Ups	34
4.3.2	Tool Surveys.....	35
4.3.3	Multidisciplinary Design Workshops	36
4.3.4	Bricolage (Intentional Re-Purposing of Existing Tools)	39
4.3.5	Heuristic Evaluation.....	40
5	Conclusions and Outlook.....	42
	References	44

1 Introduction

A distinguishing characteristic of the KP-Lab project is its dynamic and integrative view on tools and practices. This view, which holds that tools and practices are interdependent and evolve in the course of social activity, has a direct impact not just on the software artefacts or pedagogical methods to be developed, but also on the design process itself.

While classical approaches on the development of ICT¹ make a clear distinction between an artefact's design and its usage, there is a growing body of empirical evidence that software applications and systems are often not used the way they are intended to be used or not used at all (e.g. Béguin, 2003, Rosson & Carroll, 2002, Spinuzzi, 2003). This finding, that holds both for workplace as well as educational applications, challenges not only our theoretical understanding of the design-process but also calls for design approaches able to cope with the interdependency of tools and practices. While the non-adoption of an ICT might be attributed to a failure in correct specification or implementation of the required system, it has been argued that the mere idea of separating design and usage and hence to be able to anticipate the system's usage is misleading as human activity is by no means static but evolves and unfolds against the given opportunities (e.g. Floyd, 2002).

The relationship between tools and practices has caused a lot of discussion both amongst theoreticians as well as practitioners in a variety of domains including information systems, human-computer interaction as well as software engineering. In the wake of this discussion a variety of theoretical frameworks has been proposed to conceptualize the relationship between tools and practice, such as Actor-Network Theory (e.g. Law, 1992), Adaptive Structuration Theory (e.g. DeSanctis & Poole, 1994), or Activity Theory (e.g. Leont'ev, 1978, Engeström, 1987) and diverse design approaches, which at least partly attempt to bridge the gap between design and use have merged, e.g. participatory design (Schuler and Namioka, 1993) or agile methods for software development (e.g. Highsmith, 2004). Nevertheless, there still is a lack of a comprehensive and clearly specified framework for design encompassing both tools and practices.

1.1 Scope and Purpose

Against this background the deliverable outlines the co-design framework currently under development within KP-Lab, taking into account the close interrelation of tools and practices. The co-design framework draws on state-of-the-art methods in software engineering and but also reflects the experiences made within this project. Especially, it aims to accommodate to the heterogeneity of requirements, domains and field of expertise represented in the project. The co-design framework goes beyond the majority of current approaches in that it explicitly takes into account the actual context of tool usage and feeds back emergent needs and requirements into the design process. Furthermore it takes into account the different life-cycles of tools, pedagogical designs/interventions and knowledge practices.

The purpose of this document is to explicate the KP-Lab co-design framework, including its guiding principles, an overall process framework as well as the design practices implemented so far. Besides this, the deliverable briefly discusses the foundations of the approach and its similarities and differences to other software development frameworks. Furthermore lessons learned and next steps are discussed briefly. The main focus of the co-design framework as described here is on the technical aspects of

¹ Information and Communication Technologies.

system development. This emphasis has been chosen due to the current needs of the project, especially the need for a more transparent requirements elicitation and engineering process. Processes for the development of practices are out of scope of this deliverable as these are highly dependent on the diverse priority areas established and the individual partners' expertise and background. Furthermore the tailoring of existing tools for KP-Lab is only addressed as far as it directly relates to the system development.

This document updates and extends Deliverable 2.1 “Guidelines and models on implementing design principles in KP-Lab, application scenarios and best practice, v.1” submitted at M6. The co-design framework as presented in this document is still under development in the sense that it will be continuously revised and enhanced based on internal self-evaluation as well as reflection on the emerging design practices. Therefore this document will be updated based on the experiences we make in the course of the project. Deliverable 2.3, scheduled for M30 will provide the next update.

1.2 History of the Co-Design Framework and Main Changes

The co-design framework outlined in this document provides an attempt to describe but also to systematize the design activities carried out by the multidisciplinary working knots in KP-Lab. As such the KP-Lab co-design framework is not a static document but evolves itself in the course of the project. Figure 1 depicts the interrelation of the co-design framework, the design activities and the tools and practices addressed. The co-design framework in the sense of an abstract conceptualisation thereby aims to reflect the current practices but also to provide guidance for those engaged in the design process.

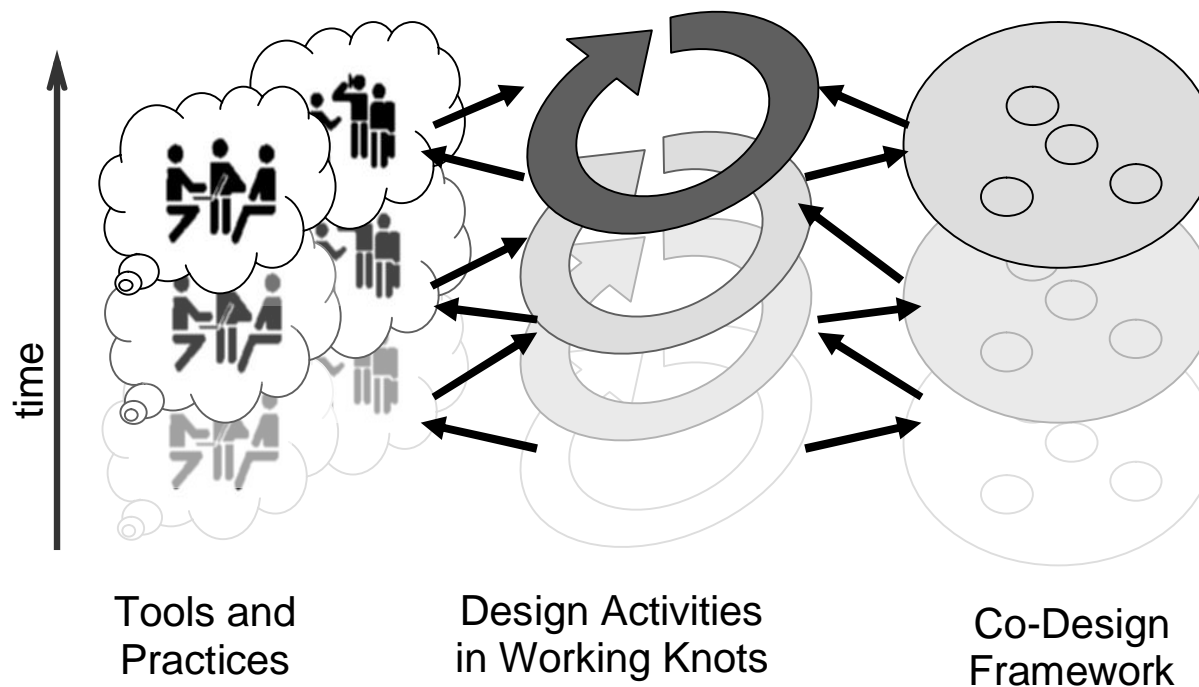


Figure 1: Interrelation between the co-design framework, design activities as well as the tools and practices addressed.

Due to its evolving nature, the co-design framework has already been cause for extensive discussions within KP-Lab project and has undergone some quite significant changes since the first version

submitted at M6. In order to better ground this deliverable the main lines of discussion as well as the most important changes are briefly reported.

Right from the beginning of the project a lively discussion on the co-design framework as well as diverse design methods to be used took place. While in the first phase of the project terminological issues as well as the quest for most suitable design artefacts (e.g. possible limitations of scenarios as design artefacts) were at the fore of the discussion, the co-ordination of the interdisciplinary design work as well as the need for a more transparent requirements engineering process became topic of discussion later on. Based on the outcomes of the first-year review the discussion on different life-cycles, alternative design models, and the impact of the so called “priority areas” on the design process were taken up again. See for example Deliverable 3.1 “Recommendations for Design Principles of Trialogical Technologies”, for a discussion on alternative design approaches and the pros and cons of scenarios as mediating artefacts. The discussions within the project have been characterized by the partners’ diverse needs, backgrounds and fields of expertise. As a consequence the co-design framework aims to provide a unifying framework for organising the design process while at the same time being open to adaptation and tailoring by the diverse stakeholder groups. In this sense the framework aims to value the diversity constitutive for the consortium.

An issue of particular relevance for the current co-design framework has been the role of theory as a driving force in the design process. While on the one hand KP-Lab Design Principles (as outlined in Del 3.1 “Recommendations for Design Principles of Trialogical Technologies”) provide guidelines for the design as well as evaluation of tools as practices, current design practices in KP-Lab also draw on the outcomes of empirical research as well as active participation of end-users. In order to accommodate to theoretical informed as well as empirical and participatory design methods, the co-design framework has been conceptualized as a practice-oriented design strategy, emphasizing practices (incl. tools) as the focal point instead of tools alone. Thereby the orientation towards practices does not preclude the reference to theoretical accounts. On the contrary most of the development work in KP-Lab is and will be theory-informed (see also chapter 3 “Challenges for Co-Design: Justification of Design Decisions”).

The changes made in the present version of the co-design framework are based both on internal discussion and reflection as well as the outcomes of the first-year review. They have already been reflected in the Description of Work 2. Due to the evolving nature of the project and the open issues described at the end of this document, further discussions will follow and additional changes to the co-design framework will be made and documented in upcoming versions of this deliverable. The following is a summary of the most significant changes made in this version of the Deliverable:

Taking into account different life-cycles more explicitly: The co-design framework has been revised in order to better accommodate the different lifecycles of tools, pedagogical designs/interventions and knowledge practices. Especially the interfaces between pedagogical designs/interventions and tool-development have been specified more explicitly, including respective design artefacts that allow to mediate the work of the pedagogical and technical partners.

Replacement of Design Teams by Working Knots: In order to support goal-oriented design work and to accommodate to the actual practices that emerged within the project, the originally envisaged Design Teams have been replaced by Working Knots. This change goes along with a stronger focus on tool development in WP2 while WP3 and the pedagogical WPs take a stronger role in the development and theoretical analysis of knowledge practices.

More transparent requirements elicitation and engineering process: The requirements elicitation and engineering process has been refined, different stages of the requirements elicitation process have been specified and terminological ambiguities have been resolved (e.g. differentiation between pedagogical/professional and usage scenario). Furthermore the process framework foresees mechanisms for the integration of high level requirements stemming from different context.

Component-oriented strategy: In order to better reflect the modular structure of the KP-Lab system, the incremental nature of the development process as well as dedicated activities on integration became a more prominent part of the overall process framework. In the same vein, the division of labour among working knots along disjunctive sets of end-user functionality (modularization), which allows for customization towards different end-user needs as well as better compensation of possible delays in tool development or requirements elicitation.

Broadened and extendable set of design artefacts: To accommodate different needs and contexts, the set of design artefacts has been broadened and is open to further extensions. Especially the role of mock-ups and prototypes in the design process has been emphasised.

1.3 Structure of the Document

The document is structured as follows: Section 2 outlines the theoretical considerations underlying the KP-Lab co-design framework and relates it to current approaches in software-engineering. Against this background resulting methodological challenges are described in section 3. Section 4 describes the general co-design framework including the guiding principles, the overall process framework as well as emerging design practices used within the framework. Finally section 5 concludes with a brief discussion of the open issues and next steps.

2 Background

This section briefly outlines the background of the KP-Lab co-design framework both from a theoretical as well as engineering point of view. While Section 2.1 introduces the concept of practice, section 2.2 positions the co-design framework in relation to current software-engineering approaches.

2.1 Design and the Co-Evolution of Tools and Practices

This section provides a tentative introduction to the concept of practice and provides reference to respective work in the fields of Human-Computer Interaction and Information Systems. The aim of this section is to ground the co-design framework theoretically and to argue for a more encompassing notion of design that explicitly takes in to account tool usage. The concept of practice as introduced here is tentative as by itself it constitutes an ongoing research topic within KP-Lab (cf. also the upcoming Del. 3.2 “A comprehensive research strategy; i.e. a research plan/program in terms of priorities, methodologies, and reporting).

At the core of KP-Lab’s design approach is the idea of co-evolution of tools, practices and agents (cp. Engeström, 1999). This perspective holds that tools, practices, and agents do not exist in isolation, but are strongly interdependent and evolve in a process of reciprocal transformation. Accordingly, the co-design approach is not focused on the development of software systems or specific artefacts per se but on providing new options for those using the tools and hence the transformation of practices. Against this background the development of a software system is not seen as an end in itself but as a means to foster novel practices which then pose new challenges and requirements for the tools used.

Broadening the scope from the design of artefacts to the deliberate creation, adoption and transformation of novel practices is backed up in the specific understanding of the concept of practice as promoted by cultural-

historical activity theory (e.g. Leont’ev, 1978, Engeström, 1999)². According to this view a practice can be defined as the way of working, grounded in tradition and shared within a collective (cp. Bødker, 1991).

In contrast to a particular activity a practice represents a recurrent pattern which can be filled out by various activities actualizing the practice, or to put it differently, a practice denotes the prototypical characteristics of a set of activities including forms of bodily and mental activity, tools and their usage, as well as certain forms of knowledge (cp. Reckwitz, 2002). In more detail practices can be characterized as follows:

- Practices are socially mediated, i.e. they are shaped by and evolve within social communities and can even become part of the communities’ identity (cp. Büscher, et al., 2001). Being bound to particular social entities also entails that a given practice broadly accepted by one community might be completely rejected by another. Furthermore, even though practices are social in nature they do not necessarily refer to collaborative activities. For example practices such as reading a book, Nordic walking or watching TV, to name a view, are socially mediated but don’t require social interaction.
- Practices entail both a momentum of stability as well as change. While practices manifest and reproduce historically developed patterns of activity they are also open for change in that the concrete activities they are based on have continuously to be adapted to new situations and changing conditions. Consequently practices are both affirmed and modified whenever a respective activity is carried out (cp. Büscher, et al., 2001).
- Practices change as a result of external and internal disturbances within and among the activities constituting the practice. While changes in practice are often attributed to external factors, such a changing resources or newly available tool, they can also be due to creative and innovative initiatives coming from within (e.g. initiatives by actors in the community), such as the introduction of a newly available tool, or the proposal for a new way of working (e.g. Béguin, 2003; Spinuzzi, 2003). In this sense design is not limited to the development of artefacts but should be seen as a way to strengthen the actor in exploring his/her potential for creating new ways of working.
- Even though practices are often characterized by the use of particular artefacts (e.g. giving a power-point presentation), practices are not determined by these artefacts in a strict sense. This difference is due to the fact that an artefact becomes a tool only when interpreted as such within in a social and historical context (cp. Floyd, 2002). Hence, while an artefact might be more or less appropriate to carry out an activity, it is the practice that defines its actual usage. In this sense activity is not determined by computer artefacts, but “it unfolds in the space of opportunities open to technology users” (Floyd, 2002, p.7).
- Furthermore practices do not exist in isolation but are part of a larger network of practices. Practices are interrelated as both individual and collective actors as well as artefacts are usually enrolled /

² It should be noted that references made to activity theory only relate to the co-design framework and do not have direct relevance for the KP-Lab research strategy or any particular priority area or case, which draw on a much broader set of theories. Activity theory has been chosen as a point of reference here, as it has been introduced as a theoretical framework for requirements analysis, system development and organisational change before (e.g. Bødker, Christiansen, & Thüning, 1995, Mwanza, 2002, Spinuzzi, 2003).

used in several practices simultaneously. Therefore changes in one practice might also trigger or inhibit changes in other practices.

Given this notion of a practice and the key assumptions underlying activity theory, the evolution of practices can be understood as a co-evolution of tools, objects, and subjects within a certain community. Neither tools, nor objects and subjects evolve independently but in a process of reciprocal transformation. Changes in practices are due to external as well as internal disturbances. While practices might change when new tools become available or circumstances and contexts shift, they can also be deliberately altered by those carrying out the activities when they invent new ways of doing things (cp. Béguin, 2003). Furthermore, even the adoption of a practice by an individual as well as the appropriation of the respective tools not simply reproduce given patterns but also entail re-interpretation and variation. As practices inevitably change when concrete activities are carried out, the evolution of practices is not a discrete act with a definite start and end but an ongoing and contingent process, which also holds for the co-evolution of tools, objects, and subjects.

This view on practices, tools, and artefacts has far reaching consequences for the design of information systems. First of all the dynamic nature of practices runs obsolete the idea to predict the future use of an artefact but promotes a view that practices change while they are carried out (cp. Floyd, 2002). Furthermore the concept of practice rules out design approaches focused on the individual user carrying out isolated activities.

In what follows some main implications for a practice-oriented design approach will be sketched, providing high-level requirements for the KP-Lab co-design framework. These requirements are provisional in the sense that they reflect the current understanding within this project. Furthermore they also only provide rough guidance for the concrete design activities as they can be implemented quite differently depending on the actual context.

Integrating design and use: While artefacts can be created independently from use, the evolution of practices cannot be separated from the execution of activities and the concrete usage of tools. Therefore a practice-oriented approach to design must not conceptualize design and development as distinct processes prior to usage, but to account for the actual appropriation of artefacts and the advancement and repurposing of artefacts during use (cp. Carroll, 2004). A practice-oriented design approach has to take into account internal and external sources of change including the opportunities and limitations provided by novel artefacts and methods as well as innovations and obstacles that arise in the execution of concrete activities. Consequently practice-oriented design must provide opportunities for the use of the artefacts produced in practice and to be receptive for changes of practice. Furthermore the design-approach must account for the integration of the discrete creation of artefacts and the continuous evolution of practices which run on different time scales. While the production of an artefact can be planned rather well, the evolution of practices is far less controllable, if at all.

Ecological perspective on usage: In contrast to design approaches which are oriented towards the creation of a distinct product, practice-oriented design has to account for the systemic and contingent nature of tools and their usage. Tools, even those labelled as “stand-alone applications”, are always only one ingredient of the toolbox available to the practitioners (e.g. Spinuzzi, 2003). Therefore the utility of a certain tool and hence its adoption depends not only on its own properties but on its fits with the overall assemblage of tools. The issue of fit goes beyond the technical interoperability or compatibility of tools as it also entails the social, cognitive and pragmatic dimension of tool usage. For example, a social-software such as instant messaging application will be of little use for the individual as long as there is not a critical mass of peers using the same technology. Similarly, the use of

educational applications might depend on the teachers' and/or students skills and attitudes towards technology as well as the availability of appropriate technical support on an institutional level. Furthermore the ecological perspective on usage also entails that the adoption and appropriation of a new tool is contingent of the current state of the activity systems involved and hence subject to historicity. Even though the degree of freedom to choose and use are certain tool, might depend on the particular context (e.g. the use of a given tool might be mandatory in an educational context and different actors might have a different say in the adoption of tools), a practice-oriented design approach has to be sensitive to the ecologies of the activity systems at stake. Given KP-Lab's interest in educational contexts it has also be taken into account that the contingency of tool usage is often restrained and adoption of tools is at least partly mediated by the teacher.

Close collaboration of practitioners and developers: As practice-oriented design is targeted at facilitating novel practices, it is necessary that practitioners (i.e. those who actually are enrolled in the respective practices) and developers closely collaborate during all stages of the design process. For example Bødker (1991) stresses that from an activity theoretical perspective “*practice should not be changed from the outside, but in a collective process by the designers and the workers themselves*” (p. 552). According to this perspective practice-oriented design has to go beyond user-centred design where the role of the user is often limited to that of an informant and test person but towards active participation giving the practitioner an active say in the design process. While the implementation of novel artefacts often requires the special expertise of technical developers these are put into use and thereby modified as tools by the practitioner.

Cyclic redefinition of design-hypotheses and requirements: Both requirements as well as the hypotheses underlying the design-decisions made, have to be redefined regularly. In contrast to design approaches focussed on the development of artefacts with predefined properties, tools and practices are inevitably open to future development. Due to the dynamic and evolutionary character of practices the design of a new artefact, e.g. a new software system, might overcome shortcomings of former artefacts but as a tool it will restructure human activities and thereby create new opportunities but also new problems and challenges (e.g. Carroll, 2000). With regard to software requirements it follows that these are not fixed but evolve themselves. Even though a certain artefact might fulfil all requirements specified, novel requirements will emerge when new practices form while others will disappear. In addition the openness of human activity also makes it impossible to predetermine users needs in advance (Bourguin et al., 2001). As a consequence requirements do not exist as such but are constructed and negotiated between practitioners and developers within the design process (e.g. Floyd, 2002, Rönkkö, 2002, Miettinen & Hasu, 2002).

Valuing design as a situated activity: Finally it can be argued that design itself is a situated and socially mediated activity that evolves in time. Like any other activity design is directed towards a certain object, mediated by specific tools, takes place in a certain community and entails rules as well as a division of labour (e.g. Kaptelinin, 2002). All these components have an impact on the way the design process is carried out. For example the division of labour between practitioners and developers defines the role of both parties and their respective tasks, while the choice of tools, such as scenarios or mock-ups, shapes possible interactions and can foster or inhibit the division of labour which might in turn have an impact on the outcome of the design process. As a consequence there is no one best way to carry out design activities but the methods and tools available have to be chosen and adapted to the circumstances at hand. In this sense each project has to develop its own design practices (cp. Floyd et

al., 1997).

2.2 Co-Design and Software-Engineering

Software engineering is a discipline concerned with the production of quality software. Its aim is to establish models, methods and tools that allow individuals or organisations create software that complies with various quality factors. Like any other engineering field, it is governed by rules and standards. Numerous of them have been created by standardization bodies (like ANSI/IEEE). These can be classified according to four categories: process, products, profession, and formalisms. Software engineering defines processes which create (non-tangible) products. Such processes are implemented by people who use methods, tools, techniques and formalisms that are accepted and shared within the profession. In this context, a process describes a series of tasks and operations that create the product. Each task uses methods, techniques and tools. Quantitative measures are used to evaluate processes, tasks and products.

Basically, the process (in software engineering) defines the work to be done, the people who carry it out, the way to conduct it, the timing of the operations, and the assessment criteria of the work. All tasks create products (i.e. deliverables) that are documented results meant for future use (by other tasks). The content and format of products are usually described by standards. Formalisms (e.g. languages) are used to present results and communicate between professionals.

Software engineering can also be described in terms of identified activities. We can distinguish three types of activities:

- product engineering: analysis, design, coding, integration, maintenance,
- quality management: testing, validation, audit, product certification, and
- technical management: process management (directing, controlling and coordinating), product management (defining, coordinating, and monitoring a product during its development), resource management (planning, estimating, assigning).

The concept of software life cycle is often used to formalize software processes. Due to the variety of contexts, domains and constraints, a number of life cycles (or life cycle models) have been proposed. Each model describes the phases (stages or steps) involved, their inputs/outputs, methods and tools, in addition to an approach to sequence the tasks during the period of creation of the software. Cascade and V are examples of life cycles. However, the concept of life cycle appears to be restrictive as it considers only the time dimension of software development. The idea of *software process* has (recently) strongly emerged to cope with the multiple dimensions of software development. Multiple attempts to create unified processes have been made.

Most of existing software processes (and life cycles) start with *Needs analysis* and end with a phase commonly called *Maintenance*. Needs analysis has been widely studied in various application domains and generated the field of requirement engineering. Maintenance is considered as the means to correct and make evolve the software in order to cope with new needs. However, little work has been done to fully integrate this task within the process.

The KP-Lab co-design process aims at wrapping a software process within a research and development context, specifically in the domain of educational technology. It has the same characteristics as a software process as it defines phases, deliverables, methods, and actors. It also attempts to reuse and redefine quantitative metrics to assess tasks, and products. However, it recommends no standards, or formalisms for the various tasks. It is a framework shared by the KP-Lab community allowing the integration of theoretical, practical (pedagogical and professional) and technical work. Its main contribution is the division of the KP-Lab process into four layers: theoretical, practical and technical,

in addition to a boundary layer that joins the practical and technical layers. Theoretical work is concerned with establishing fundamental ideas underlying the target research and development. Practical work is focussed on pedagogical design, interventions as well as the empirical investigation of particular cases and design experiments. The boundary layer translates pedagogical needs into design artefacts that are easily transformable into software design artefacts. Technical work creates software artefacts and tools used by the theoretical and practical layers. Artefacts are defined to allow communication between and within layers.

As such, the co-design process can be seen as an effort to fully integrate the maintenance and practice monitoring and analysis within the software process. This creates a situation where the same process deals with heterogeneous time scales and speeds. This comes from the fact that practices evolve through a long period of time, while software development takes place in a limited well established period.

The proposed process also allows parallel design/development of multiple practices and tools. Precisely - although having dependencies via artefacts - theoretical, practical, boundary and technical tasks work as generalized pipeline architecture. After an initialisation phase, the work on tasks becomes parallel. At the same time, for the same layer, multiple cases or tools are handled simultaneously. As an example, the technical layer is instantiated as many times as tools (or services) develop. An original structuring element, called Working Knot deals with each tool. This results in multiple Working Knots performing design and development at different stages simultaneously. Convergence of all Working Knots takes place at two different stages: the design integration and packaging and the KP-Lab system integration. In addition, the development of a single tool is incremental and iterative. Few increments (i.e. sets of functionalities) are defined and worked on in sequence. For each increment the development is iterative; the number of iterations is arbitrary and defined only by the compliance of the tool with the end user specifications. The testing tasks all feedback to the appropriate analysis, design or development phase to refine the work.

3 Challenges for Co-Design

This section provides a summary of the main challenges the co-design framework has to face. Besides a general description of the respective problem, its relevance for KP-Lab project is discussed briefly. Hereby design is seen as a collective activity often characterized by specific contradictions, ambivalences, and tensions. Even though the following challenges do not only apply to co-design framework described hereafter, they have a strong bearing on the guiding principles underlying the design approach.

Multiple languages and terminologies: Multidisciplinary collaboration, including the cooperation of users/practitioners and developers, is challenging due to the fact that each discipline has developed its own language and terminology often incompatible with each other (e.g. Floyd, 2002). For example the term ‘object’ denotes completely different and even incommensurable things when used by a software-engineer in the context of object-oriented programming or a psychologist referring to activity theory. But even within a discipline there can be remarkable variation in language and terminology rendering communication difficult and increasing the likelihood of misunderstanding.

Within KP-Lab this problem is even aggravated due to the use of a complex and sophisticated theoretical foundation of the project as well as the multiplicity of application domains addressed. While both features are constitutive for this project they put high demands on the participants. Consequently the establishment of a common terminology and the sensitivity for possible misunderstandings remains

highly important. Towards this end the “Triological Glossary” (<http://kplab.evtel.fi:8080/wiki/Wiki.jsp?page=CategoryTriologicalGlossary>) has been created in order to foster shared understanding and to establish a common terminology within the project. Similarly WP3 is aiming to create a conceptual framework for the description of knowledge practices in order to support communication among technical, pedagogical and theoretical partners.

Different perspectives and worldviews: Beyond the use of different languages and terminologies different stakeholders often also have different perspectives towards the design object as well as diverging worldviews and paradigms. Kaptelinin (2002, p. 60) notes that “*user’s and developers’ perspectives on computer technologies are fundamentally different. Users want to achieve meaningful goals, while systems developers are mostly focused on technology per se*”. In addition, with regard to the different perspectives held by computer and social scientists Floyd (2002) concludes that while the former are guided by the ideal of abstraction and objective knowledge in computer technology the latter are oriented towards an understanding of technology as part of situated and unique human activity.

Even though both pedagogical as well as technical partners committed themselves to a multi-disciplinary project and displayed a great deal of openness towards each other, frequent virtual meetings, face-to-face workshops, as well as the periodical reflection and communication of actual design practice are essential to disclose diverging perspectives and to negotiate on common practices.

Conflicting and misleading expectations: Due to different interests and foci of those involved in the design process as well as limited insights into each others disciplines conflicting and misleading expectations can easily arise. For example Kaptelinin (2002), who discusses options to bridge the gap between the social and the computer sciences, notes that there often is an explicit or implicit but erroneous expectation from technical partners that social scientists can easily translate social science insights into practical guidelines for design. On the other hand non-technical partners often treat computer science as a rather homogeneous discipline, without being aware of different traditions, sub-domains, and methods, which might lead to the expectation that all computer scientists bring in the same expertise and skills. Incompatible expectations can also result in unclear responsibilities and therefore an unclear division of labour.

Within KP-Lab conflicting expectations regarding the design process became apparent especially with regard to the specification of requirements. It seems that technical partners, at least implicitly, assumed that pedagogical partners would be able to express end-user requirements quite easily, while simultaneously pedagogical partners were calling for more input from the technical partners regarding the technical possibilities. As a consequence the explication of mutual expectations was well as the clear definition of responsibilities constitutes an important task right from the beginning. Towards this end joint presentations and analyses of existing tools (e.g. as part of the requirements elicitation process) provide an important means to ground communication between the various partners involved and to overcome misunderstandings.

Different timescales and problems of synchronisation: Another obstacle which complicates the coordination of multidisciplinary design activities is due to the different lifecycles of tools and practices. While dependent on the product at stake, the stage of the process and the software development method chosen developers are supposed to produce new prototypes within a couple of days or weeks, the implementation of such prototypes, the evolution of new practices and associated organizational change as well as the evaluation of their application is a much more time consuming endeavour. Consequently technical developers and pedagogical researcher work on different timescales, while at the same time they are dependent on each others outcome. The evaluation of a prototype in practice presupposes its

existence while on the other hand the redesign of the prototype shall be informed by insights gained regarding its practical use.

Due to the project's commitment to a practice-oriented design approach the synchronisation of technical development and practical implementation becomes even more pressing, as it requires the analysis of tool usage over a prolonged period of time in order to allow not only for adoption but also appropriation by the end-user (cp. Carroll, 2004). Furthermore, the testing and evaluation of tools in real world contexts is often constrained by externally defined timeslots, such as the beginning or end of a term in higher education.

The situated and uncertain status of requirements: The understanding of human practices as outlined above implies that requirements are situated but also uncertain in the sense that *“the take-up, modification, and rejection of technology in a work setting, and the accommodation of work practices that will take place around a developing technology, are radically unknowable and unpredictable”* (Büscher et al., 2001, p. 5). Even though this perspective is theoretically well founded it is challenging from a practical point of view as it is at odds with most of the existing design approaches (e.g. Bødker, 1991, Büscher et al., 2001). Rönkkö (2002) with reference to McDermid (1994) states, that neither the complete specification of all requirements nor the restriction to a set of key requirements provides feasible or useful strategies. Instead he argues that it is impractical and impossible to produce complete, consistent, and implementable requirements specifications.

From an R&D project's point of view the situated and uncertain status of requirements is especially challenging as the overall duration of the project is limited while at the same time design decisions have to be made based on a necessarily tentative set of requirements. Furthermore, even though iterative and incremental design approaches allow to refine and update requirements periodically, basic design decision made before often cannot be undone or revised later on. Besides keeping in mind the tentative and hypothetical character of the requirements it seems equally important to provide heuristics and guidelines for making informed design decisions.

Context dependency and heterogeneity of requirements: In addition to the situated and uncertain character of requirements these are also highly context dependent and therefore heterogeneous across sites. What might be useful in context A can be completely unsuitable in context B. While this diversity can be problematic within a domain (e.g. higher education) it becomes even more challenging when a project such as KP-Lab is targeted at different domains (here higher education, teacher training, and professional networks). At the same time the diversity of contexts also holds the potential of cross-fertilization, boundary-crossing exchange and triangulation of results across contexts and domains and hence might lead to better understanding of the practices at stake. In order to make use of the multiplicity of context it seems especially relevant to ensure the traceability of requirements (e.g. Rönkkö, 2002) as well as to establish a conceptual framework that allows to analyse and compare practices across sites.

The situated nature of design activities: Furthermore, the situated nature of design activities is challenging by itself. Floyd et al. (1997) stress the importance of a flexible and even experimental use of methods in the design process while Bødker (1991) notes that design-teams often feel uncomfortable with the methods they are currently using and believe that other methods would suits their needs better even if they use the methods successfully. The tension between the current design practice and the (assumed) 'ideal' can of course be due to wrong choices, missing knowledge about available alternatives, as well as inadequate implementation. On the other hand it is also unlikely that there is an off the shelf method that perfectly suits the particular needs of a concrete design team. With regard to

methods of participatory design for example Törpel, Wulf, and Kahler (2002) argue that the guiding principles of this approach cannot be fulfilled in many cases and therefore the method has to be adapted to the particular situation of the project. The need for adaptation might be due to a multitude of reasons such as the lack of required competencies, restrictions and limitations regarding the availability of stakeholders or missing opportunities to meet face-to-face. As a consequence it is important that practitioners and developers are aware of design-methods available but also that they are encouraged to tailor these methods to their own needs.

Justification of design-decisions: While design decision can be based on theoretical accounts (such as the KP-Lab Design Principles) as well as empirical findings (e.g. the outcomes of a design experiment), both approaches entail particular obstacles and challenges. Due to their abstract nature theoretical accounts such as Design Principles usually do not lend themselves directly to a particular decision but have to be interpreted with regard to the particular context given. Empirical findings on the other hand often suffer the problem of missing ecological validity or their generalisation towards other contexts might be questionable. As a consequence neither theoretical accounts nor empirical findings alone are sufficient to justify design-decisions, especially when design decision are supposed to have multiple and interrelated consequences. Instead it seems necessary that those engaged in the design process make informed decisions based on both theoretical knowledge as well as empirical information available. Furthermore the pros and cons of a decision have to be weighted against the potential contexts of use.

Summarizing it has to be noted that the challenges sketched here do not exist in isolation but that they are strongly interrelated.

4 The KP-Lab Co-Design Framework

This section provides a comprehensive description of the overall framework including the guiding principles, the overall process framework as well as emerging design practices developed so far.

4.1 Guiding Principles

In general the KP-Lab Co-Design Framework is built on the following principles:

Close collaboration of practitioners and technical developers: In order to develop innovative tools that exploit technical capabilities and facilitate novel practices, it is necessary that practitioners and technical developers closely collaborate during all stages of the design process. Towards this end, so called Working Knots (WKs) provide the primary forum for envisioning, designing, implementing and evaluating the end-user tools of the KP-Lab system. Within the working knots pedagogical partners are directly involved in the requirements specification as well as the assessment and evaluation of various design artefacts (such as mock-ups) and prototypes.

Component oriented development: The co-design work within the different Working Knots is organized along the end-user applications of the KP-Lab system. According to the modular structure of the KP-Lab system, each Working Knot is centred on a particular set of end-user functionalities of the overall system. The focus on end-user functionalities allows accommodating to the situated and intertwined nature of knowledge practices by providing a set of re-configurable and tailorable tools instead of producing highly specialised but idiosyncratic solutions.

Cyclic revision of design-hypotheses and requirements: Due to the mutual interdependence of tools and practices, both requirements as well as the hypotheses underlying the design-decisions made are revised regularly. In order to accommodate the different time scales of end-user tools, their implementation in pedagogical and professional settings, and the knowledge practices itself, the design process is organised on four layers:

- Short-term development of end-user tools: Work on this layer is focussed on the development of end-user tools, such as tools for the generation, visualisation, and management of knowledge objects, the support of e-meetings or the collaborative annotation of multimedia content. The work will be organized in short iterations, in order to allow for timely feedback by pedagogical partners and allow for adaptation of evolving requirements.
- A boundary layer that translates pedagogical and professional cases into high level requirements for software tools. This work is carried out by Working Knots which group pedagogical, professional and technical experts. The aim is to move practical needs closer to technical requirements analysis. Furthermore, any single case might involve complex software functionalities that cannot be developed in the same tool. In addition to translating the case into high level requirements, we also need to package these requirements and map them to single tools for further development. This layer serves this purpose by coordinating and packaging the requirements analysis tasks.
- Middle-term implementation and exploration of field-trials and design experiments: The aim of this layer is to implement and test KP-Lab tools and methods in real world contexts, to provide feedback on the pedagogical utility and hence to refine or revise requirements. In order to gain more reliable insights in the utility of tools and methods, these have to be tested under realistic conditions also allowing the users to accommodate to the new tools and methods. Field-trials and design experiments will last from a couple of weeks up to a semester.
- Long-term investigation and analysis of evolving knowledge practices: Finally, the fourth layer is focused on the long term analysis of knowledge practices and their transformation, in order to better understand the mechanisms underlying the evolution of knowledge practices. Work on this layer includes the investigation of successive implementations of some pedagogical scenarios, longitudinal investigation of students' knowledge practices, the study of changing knowledge practices in professional networks as well as the investigation of current cutting edge knowledge practices. Insights gained on this layer will inform the co-design process via the refinement of design principles and the specification of novel pedagogical designs. As knowledge practices evolve only slowly, special attention will be paid to the analysis of cutting-edge practices in order to inform the development of new pedagogical designs or interventions.

Participatory and adaptive design methods: As both the end-user tools to be developed and the practices to be supported are heterogeneous in nature, the co-design model does not prescribe the particular design-methods or practices but provides an overall process framework. Nevertheless the end-user shall be actively involved in the design-process wherever possible.

Theory-based evaluation of end-user tools and practices: The overall design process and its outcomes will be regularly reviewed and evaluated by the scientific board to ensure its coherences with the theoretical framework. In addition, pedagogical partners will provide theoretical input to the co-design process by defining high-level and functional requirements based on the knowledge practices under investigation and providing feedback on design artefacts and prototypes.

4.2 Overall Process Framework

This section describes an integrated process model of the co-design activities within KP-Lab. It is an attempt to explicate the interrelations of the pedagogical and technical work packages with emphasize on the software development process. The document includes a description of the main phases of the development process foreseen as well as a definition of the main design artefacts used.

To be able to cope with the complexity of the project and the changing nature of the requirements, KP-Lab needs an integrated process that goes beyond classical software processes. It should comply with the following general specifications:

- it should integrate theoretical, pedagogical and technical work and handle the heterogeneous nature of the corresponding life cycles and time scales
- it should allow at the same time for sequential development as well as simultaneous work on various phases
- every phase must allow for different threads of activity to handle cases and tools
- for the same case or tool, the process should allow for dividing the work into increments
- each phase must have a mirror testing task that allows for feedback and adjustment
- the phases should share design and development artefacts
- technical development should also be multi-threaded but converges towards the integration of the KP-Lab system.

In order to accommodate to the different life-cycles of tools, pedagogical designs/interventions and knowledge practices, the process model is divided into four layers. Each layer forms an iterative loop with its lower layers. The four layers are the means used in KP-Lab to deal with the heterogeneous time scales of development of tools and practices.

The first layer is focussed on the development of end-user tools. It includes the functional specification, design and implementation, technical evaluation and integration of the KP-Lab system. The work on this layer is carried out by the Working Knots, which consist of teams of technical and pedagogical partners coordinated by a “design chair” and the technical work packages (WP4, WP5 and WP6).

The second layer, called boundary layer, is responsible for translating pedagogical scenarios and cases into high level requirements. These requirements are mapped to usage scenarios that describe the interactions of users with the system. Each of these scenarios translates into use cases that are specified and implemented in Working Knots in layer 1. This work is also carried out by Working Knots.

The third layer is focused on pedagogical designs and interventions, including the use of new end-user tools. This layer includes the realisation of design experiments and interventions, as well as the use and evaluation of end-user tools in field trials. Work on this layer is primarily carried out in the pedagogical and professional work packages.

The fourth layer is focused on the long-term analysis of knowledge practices and their transformation. This layer includes the investigation of cutting edge practices as well the intensive longitudinal investigation of knowledge practices in higher education. The respective activities are carried out in WP3 as well as the pedagogical and professional work packages.

The overall co-design process is a framework rather than a mandatory work process. It heavily relies on collaboration instead of strict documentation and planning. However, the process presents minimal organisation to document and trace activities within the project. It has been built based on existing practices. It is an attempt to formalise these practices in order to provide a base structure

for management. It is meant to be respected by all project actors, workpackages and working knots. However, methods and practices can be locally adapted.

Precisely, the proposed process has a pipeline structure (Fig.2). It consists of eight phases:

- 1- Theoretical analysis of knowledge practices
- 2- Case-based interventions and design-experiments
- 3- Boundary design
- 4- Technical development and integration
- 5- KP-Lab System integration test
- 6- Contextual usability testing in field trials
- 7- Pedagogical evaluation
- 8- Long term analysis of knowledge practices

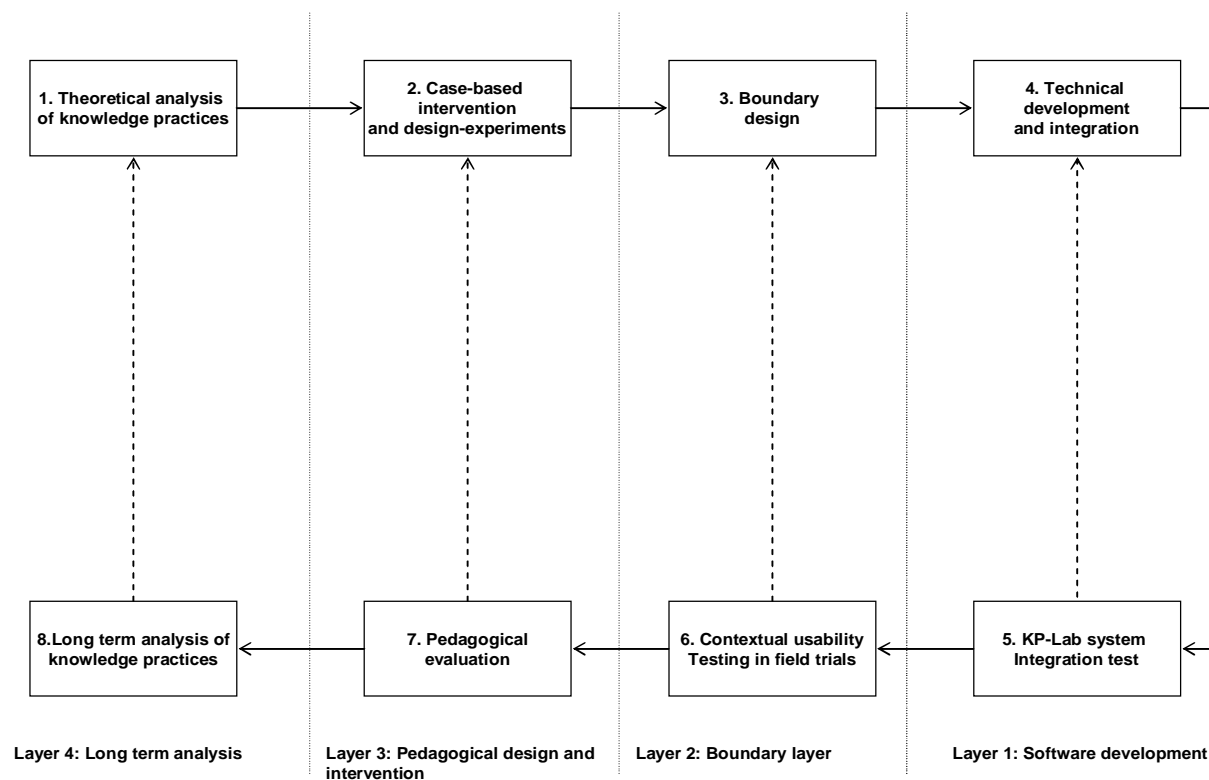


Figure 2: Overall Co-design process.

Tasks one to four consist of design work; whereas tasks five to eight handle testing and evaluation at different levels. The four layers are made each of two tasks of design/development and testing/analysis. The layers lead to each other sequentially. All phases of the process are incremental. This means that the work is divided into few increments that are precisely planned for. The end of the increment P_{ij} (of phase P_i) launches the next increment (P_{i+1k}) of the following phase (P_{i+1}); and at the same time the work continues on the next increment (P_{ij+1}) of the same phase (P_i). Hence, the different phases of the co-design process do not have to be instantiated in strict linear order. While work across the different

layers is carried out in parallel it is also possible to jump forth and back to other tasks on the same layer (Fig.3).

This process is iterative at different scales. Each mirror testing task either starts a new iteration (on the same layer) to refine the design/development work, or launches the next testing phase. There are four inscribed iteration loops. The inner loop is the technical development iterative loop. It consists of developing, integrating and testing the KP-Lab system. The second loop takes care of creating high level requirements and testing the tools in field trials. The third loop iteratively designs pedagogical experiments and implements them using the developed tools. Every tool implementation leads to pedagogical evaluation that allows the adjustment of the cases and the creation of research data for long term analysis. The outer loop is the slowest. It is responsible for long term theoretical analysis of knowledge practices. Findings are used to create new cases or improve existing ones by informing new increments.

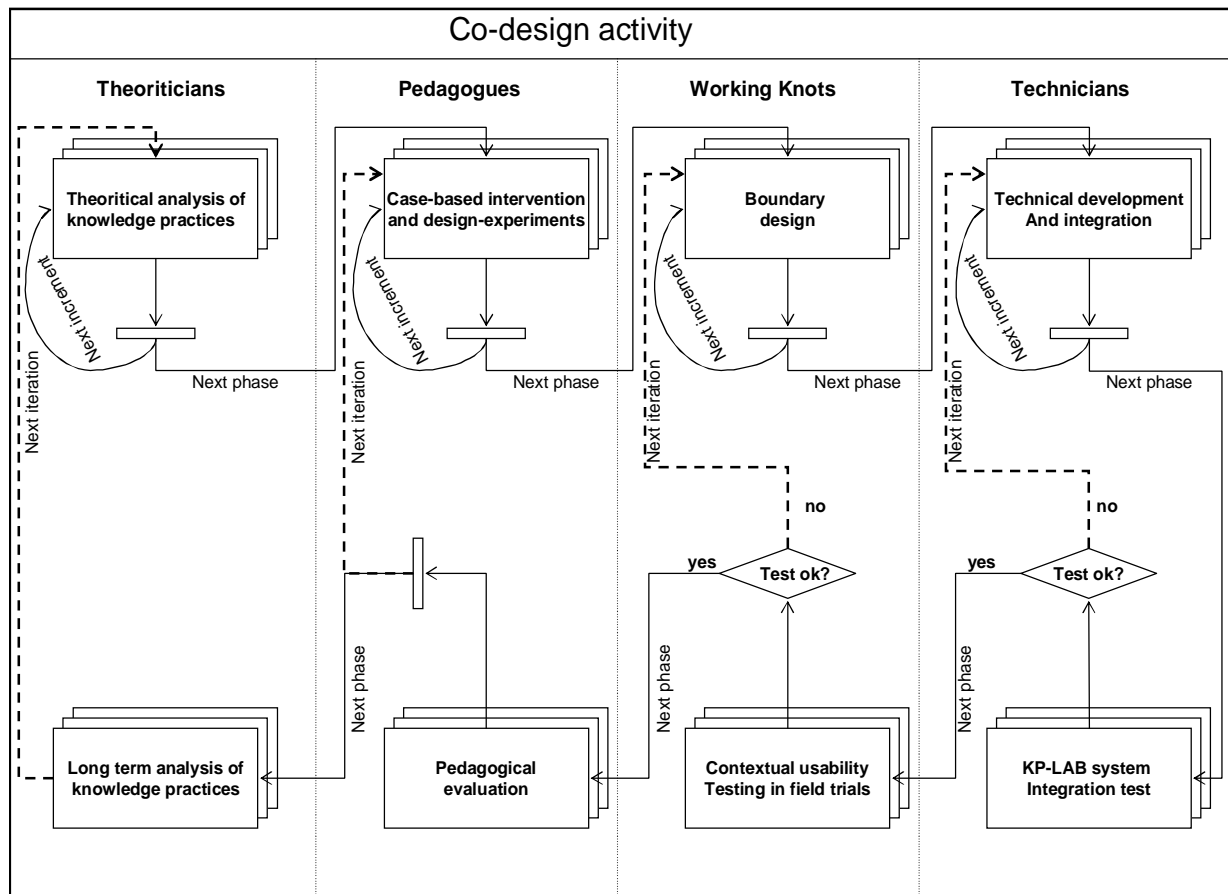


Figure 3. Activity diagram of the co-design process.

The remainder of this document explains the phases of this process in more details. Each phase is described in terms of Objective, Input, Methods and Output. The main objective of the process model is to describe the systematic nature of the co-design by showing the connections between the phases. Another aim is to create the basis for quantitative assessment of work throughout the project.

4.2.1 Conceptual model

Before describing the details of the phases, we first present the design artefacts involved in the process.

Figure 4 shows the main ones that are communicated between phases.

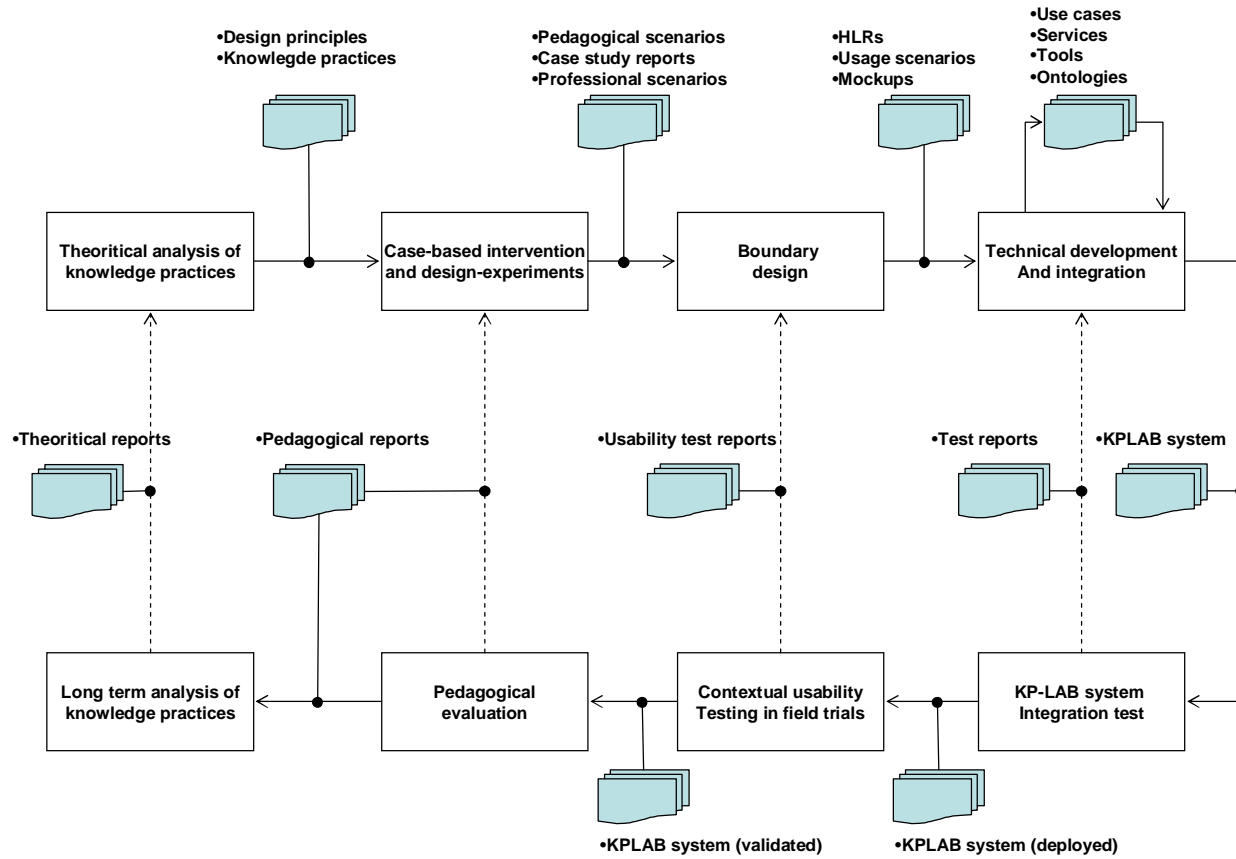


Figure 4. Communication between phases.

Figure 5 depicts an overview of their dependencies. It also depicts the relationship between artefacts and workpackages. This allows establishing responsibilities. The nature and role of each of these items are described below. Their systematic use in the process is presented in the following sections.

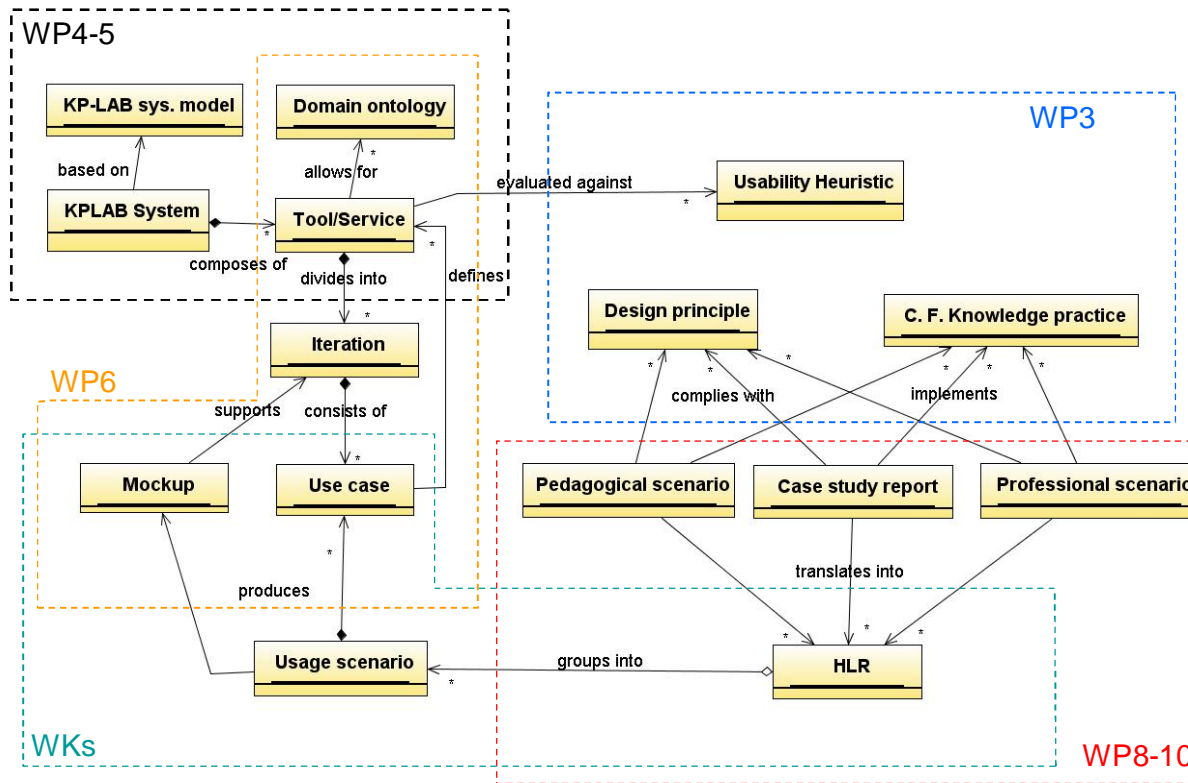


Figure 6. Main design artefacts used in the co-design process.

Design Principles

KP-Lab design principles describe abstract requirements for fostering knowledge-creation processes and are meant to guide the development of KP-Lab tools and methods. The design principles are based both on theoretical accounts as well as insight gained in case-studies and design experiments. They provide general guidelines for the design and evaluation of pedagogical and professional methods and interventions. Therefore, they will be on the background of designers throughout layer 1 and layer 2. They also influence the work of usability testing in layer 3. These are clearly documented in deliverables produced by WP3. They are accessible to all designers via written form but also communicated in workshops.

Conceptual Framework for Knowledge Practices

This conceptual framework provides a means for the description and analysis of knowledge practices in educational and professional settings in order to support the technology development. The aim of the framework is to describe knowledge practices in a systematic and comparable way and hence to foster the elicitation and assessment of high-level end-user requirements. The conceptual framework for knowledge practice developed in WP3, will be documented in R3.4 and communicated to pedagogical and technical partners via workshops, papers, and joint meetings.

Usability Heuristics

KP-Lab Usability Heuristics provide general guidelines for the design and evaluation of KP-Lab tools. The heuristics are based on the Design Principles and have been defined in a joint effort of pedagogical and technical partners. The heuristics will be updated based on experiences gained in expert evaluations as well as usability tests.

Pedagogical / Professional Scenario

A pedagogical and professional scenario describes an existing or foreseen pedagogical or professional intervention or change process. Their purpose is both to concretize the underlying rationale of an intervention by describing the specific methodical means (e.g. didactical strategies) and the context of the intervention or change process as well as to provide the background for the design of tools relevant to the described intervention. They provide a mediating artefact for those involved in planning, implementing and carrying out a pedagogical or professional intervention or change process and describe the activities that shall be supported by KP-Lab software. Scenarios are documented in plain English and made available to tool designers and field researchers.

Case Study Report

A case study report describes and analyses the outcomes of a case study or design experiment. Case study reports provide input for the specification of high-level end-user requirements and are the basis for meta-analyses and the theoretical-analysis of knowledge practices.

High-level end-user requirement (HLR)

Given a pedagogical or professional scenario, a High-Level end-user Requirement (HLR) is the description of a desirable behavior or property of the tool necessary or useful to implement the scenario. A HLR describes what is needed from the end-user's point of view and not how this is to be achieved. High-Level end-user requirements encompass both functional as well as non-functional requirements such as interoperability with existing systems, maintainability, safety, usability, etc.. HLRs are focussed on the actual stakeholders needs. In practice, an HLR consists of a title and a short description showing the corresponding list of activities. HLRs are the basis for software development. They are documented and traced throughout the development process.

Usage Scenario

A usage scenario illustrates how the user will interact with the system/tools in order to accomplish certain tasks. In contrast to the pedagogical and professional scenarios, the usage scenarios are focused on the interaction with the system/tools instead of accomplishing some higher-level objectives. The usage scenario shall describe 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. These are an informal way of writing use cases. No formalism, such as UML, is used. The aim of this intermediate artifact is to facilitate the collaboration between pedagogical and technical partners when analyzing requirements.

KP-Lab System Model

The KP-Lab System Model is a technical artefact aiming at providing the common “semantics” shared by KP-Lab tools. It shall guide and facilitate the implementation of their interactions, data exchanges and their integration with the KP-Lab Platform. It supports the fulfilment of interoperability requirements between the different components of a KP-Lab system. The model is developed and maintained in the frame of WP4 work.

Use Case

A use case is taken here in the strict sense of software engineering. It describes the interactions between a user and the tool. The use case model is a structured document that describes all UML use case diagrams for a given tool. It describes each use case using a standard template. Each scenario is also described using UML system sequence diagrams. Each usage scenario results in writing one or several use cases.

Mock-Up

Mock-ups draft the elements of the graphical user-interface and give an idea of the envisaged functionality of a prospected artifact. They illustrate the screens and possibilities of user interaction. Their main purpose is to test design ideas and to elicit new requirements in close cooperation of technical and pedagogical partners. Mock-up can be created both by technical as well as pedagogical partners and range from paper-mock-ups and ppt-slides to complex flash-animations.

Increment

An increment is a structuring item that allows the management of the development in terms of content (the “what” we want to achieve). For a given tool, all use cases are prioritized based on their need during the implementation of the related scenarios. Few increments are established for every tool. The first increment contains basic use cases that must be developed first. The second increment contains use cases necessary for the scenario. The subsequent increments contain complementary use cases that are last developed.

Tool

High-level requirements are analysed and grouped according to the nature of data processing they require. A tool is chosen (in the case of existing tool that are identified to potentially fulfill the desired requirements) or developed to cope with each coherent set of HLRs. It is usually made of basic components, which functions (services) are specified, developed and tested individually. They are then integrated in the tool; which is a software piece that implements a meaningful user interaction. However, KP-Lab tools are meant to be components of KP-Lab system, although they can work autonomously. Precisely, they interact with various other components and services (e.g. such user management, knowledge middleware). Integration within KP-Lab system is therefore required. It requires integration with other tools.

Technical Services Framework

The TSF is a sub-system of the KP-Lab platform. It provides shared services interfaces, available and usable for cross tools interactions in a KP-Lab System. This includes repositories of services, registration of services and services technical management, communication and messaging, security (identity/authorization), session

management, history, logging management. The TSF also proposes technical services for referencing and accessing shared knowledge and content through various resources management facilities.

KP-Lab System Architecture

The architecture defines the KP-Lab System components, the services they offer and their interconnections rules. A tool is a particular assembling, and implementation of their interactions of these components.

KP-Lab System

In addition to theoretical models, pedagogical and professional methods related to knowledge practices, KP-Lab project produces a *software system*. The KP-Lab system supports and implements KP-Lab models and methods. Although flexible and scalable, it represents a coherent entity aggregating different tools developed within the project. It also makes possible seamless use of selected third party software and services as well as various interactions (data exchanges, services invocations...) with these. It is built upon a Services Oriented Architecture that supports semantic and social web concepts and technologies.

Other Documents

Besides the design artefacts mentioned before, various types of documentations and reports are used to record and communicate specifications and report testing results. Main outcomes for every co-design activity are mentioned in the process model.

4.2.2 Process Model

This section describes each of the four layers of the co-design process.

Layer 4: Long Term Analysis

Layer four is concerned with theoretical analysis of knowledge practices. Its role is to synthesize empirical findings on knowledge practices over a prolonged period of time and across sites as well as to identify design principles. It is made of two main tasks (fig. 6) theoretical analysis of knowledge practices, and long term analysis of these practices. The second follows multiple pedagogical evaluations and feeds back insight to theoretical analysis.

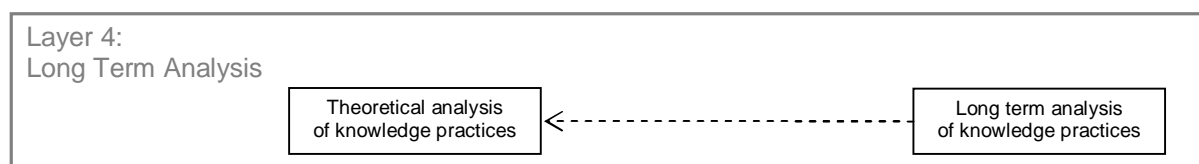


Figure 6: Layer 4.

Theoretical analysis of knowledge practices

- Objective:** The aim of this activity is to provide a thorough theoretical analysis of the knowledge practices under investigation, to develop models and frameworks for the description and analysis of knowledge practices, and to provide criteria for the evaluation of end-user tools and practices.
- Input:** Input includes both theories of human activity and learning as well as the outcomes of case-studies and design experiments.
- Output:** Three main artefacts
- Conceptual framework for knowledge practices
 - Design principles
 - Usability heuristics
- Actors:** This work is carried out within WP3 together with scientific board and pedagogical partners, and informs the evaluation of cases and end-user tools carried out in the pedagogical and professional WPs as well as in WP2.
- Methods:** Collaborative analyses of cases in higher education and workplaces at workshops, working knots, conference symposiums, and virtual meetings.

Long term analysis of knowledge practices

- Objective:** The objective of this activity is to capture and trace individual and collective transformations of knowledge practices in educational as well as organizational contexts over a prolonged period of time and across sites. The outcomes of this activity will shed light on the sustained appropriation of tools and practices along the priority areas.
- Input:** This task takes as input case study reports from pedagogical and professional cases and design experiments.
- Output:** Reports fed to the theoretical analysis in order to adjust practices and/or principles
- Actors:** Long term analyses are carried out within the pedagogical and professional work packages.
- Methods:** Comparative analysis of data gathered in case studies and design experiments. Results are analysed in various ways relatively to both knowledge practices and design principles.

Layer 3: Pedagogical Design and Intervention

The second layer is concerned with empirical research in educational and professional contexts. It consists also of two tasks (Fig. 7): Case – based interventions and design – experiments, as well as empirical investigations in the form of case studies and evaluations. Tools used in the context of these interventions and design experiments include but are not limited to those developed in KP-Lab.

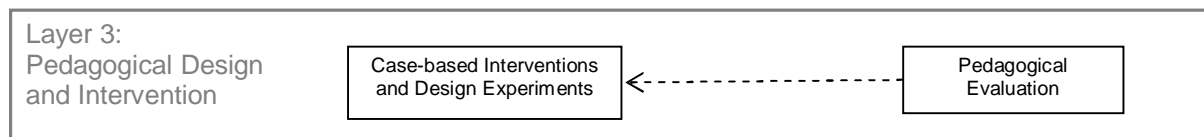


Figure 7: Layer 3.

Case-based Interventions and Design Experiments

- Objective:** The aim of this activity is to design and carry out case-based interventions and design-experiments both in educational as well as professional contexts. This activity also includes the envisioning of novel practices, methods and tools to facilitate processes of knowledge creation.
- Input:** It gets input from Layer 3 in the form of Design principles and Models of knowledge

practices as well as the outcomes of previous case studies and experiments.

Output: Main outputs, besides the interventions themselves, are pedagogical or professional scenarios.

Actors: Work is located in the pedagogical and professional work packages.

Methods: Depending on the context, methods range from instructional and pedagogical design to participatory and emancipatory approaches.

Pedagogical Evaluation

Objective: The aim of this activity is to gain a better understanding of knowledge practices in higher education and professional networks as well as to investigate the impact of interventions and design-experiments on these knowledge practices. The outcomes of this activity are meant to inform the envisioning of new end-user tools and methods to facilitate processes of knowledge creation. This activity goes beyond usability testing as it is aimed to evaluate the tools utility against the students' or professionals' knowledge-practices and hence to inform the definition/revision of high-level user-requirements. Furthermore these case studies and evaluations are also meant to investigate the utility, viability and interoperability of KP-Lab tools with pre-existing tools and information systems.

Input: Inputs to this activity come from layer 2, 3 and 4:

- Pedagogical/Professional scenarios
- KP-Lab system (validated)
- Design principles

Output: The results are evaluation reports:

- Case study report

Actors: The work on this strand is carried out within the pedagogical and professional work packages.

Methods: Conduct field trials using KP-Lab system and gather data on the pedagogical and professional impact of the practice.

Layer 2: Boundary Design

As explained above, this layer is concerned with creating mediating artefacts between the practical layer and the technical layer. Its aim is to seamlessly translate the needs of the pedagogical cases and scenarios into user requirements. At this stage, the needs are first expressed into High level requirements (HLRs). These are then analysed and grouped to form use scenarios which describe, without any formalism, the interactions between the user and the KP-Lab system (Fig. 8).

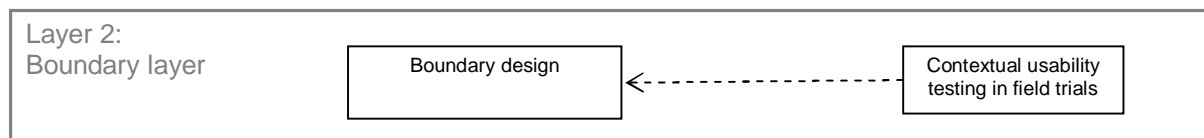


Figure 8: Layer 2.

It consists of two tasks: 'Boundary design' and 'Contextual usability testing in field trials'.

Boundary Design

This task consists itself of two sub-tasks: High Level Requirements (HLRs) specification and High level requirements (HLRs) analysis. The first translates pedagogical cases/experiments into HLRs. The second packages the HLRs into usage scenarios.

High Level Requirements specification

Objective: The basic objective of this task is to start deriving end-user requirements for the KP-Lab System. The requirements at this stage are very general. We call them *High Level Requirements*.

Input: In addition to KP-Lab theoretical and educational technology experience, this task takes input from higher tasks in layer 3 and 4.

- Design principles
- Conceptual framework for knowledge practices
- Pedagogical/professional scenarios
- Case-study reports

Output: High level requirements

Actors: This work is conducted within a multi-disciplinary team composed of chairs of Working Knots within WP2 as well as members of WP8-10.

Methods: The specification of high-level end-user requirements takes place in close collaboration between WP2 and the pedagogical WPs 8-10. Both pedagogical/professional scenarios as well as case-study reports are analysed to identify core activities conducted, actors involved, information manipulated, and problems and breakdowns encountered. Based on this analysis an initial set of end-user requirements is proposed by the chairs of the Working Knots and/or pedagogical partners. The list of end-user requirements is then discussed and revised in an iterative process, whereby technical partners inform the discussion by presenting existing technologies. After pedagogical partners decided on the priorities for the different requirements the chairs of the Working Knots group them into coherent classes which form the High Level Requirements (HLR). For example the proposed requirements “annotate image”, “annotate text”, and “annotate video” are grouped in the high-level requirement “annotating documents”. Afterwards high-level requirement are mapped to individual tools. Each HRL is mapped to one (and only one) tool and hence one WK. Multiple HLRs can be mapped to a single tool (WK). HLRs are traced throughout the development. They are stored in the process database for progress assessment and requirement testing.

HLR analysis

Objective: HLRs are further specified into more explicit documents called usage scenarios. A usage scenario is a natural language description of a set of use cases. Interaction scenarios between users and the system are described. Any artefact, such as rough drawings of screens, can also be used to illustrate the usage scenario. This step helps pedagogical partners to express in a non-formal way their needs. A single usage scenario maps to multiple HLRs.

Input: HLRs

Output: Usage scenarios

Actors: WK

Methods: Analysts, with pedagogical background, work closely with professional and pedagogical partners who defined the pedagogical scenarios of concern. Face to face and virtual collaborative work sessions are organised. A session concentrates on a single usage scenario. It attempts to group logically HLRs under the same scenario. The outcome of these sessions is natural language descriptions of user interactions. These are documented in usage scenarios. Drawings are also used to express interactions.

Contextual usability Testing in field trials

Objective: In order to fully evaluate KP-Lab tools and system, various testing stages are mandatory. As described above, “Pedagogical evaluation” conducts testing against high-level pedagogical and professional objectives. Usability testing is an intermediate stage that validates KP-Lab system and tools against usability heuristics and usage scenarios.

Input: Input comes from layers 1, and 2:

- Usability Heuristics
- Usage Scenarios
- KP-Lab system (deployed)

Output: The results are evaluation reports:

- Usability report

Actors: Work is conducted within Wks and supported by a usability panel in WP2.

Methods: Testing is done by end-users under controlled conditions (lab-tests) as well as in field trials. Besides explicit usability studies, testing questionnaires and bug reporting mechanisms are used to collect user-feedback. The latter is supervised by WP4. Reports are fed back to the working knots. This launches a new inner cycle to refine the tools.

Layer 1: Software Development

This layer takes care of creating the software tools, models, and ontologies required to implementing KP-Lab cases. The main input to this layer comes from the “Boundary Design” in the form of usage scenarios. It consists of a sub-process (Fig. 9) made of two phases. The ‘Technical development and integration’ phase is a sub-process that describes the technical work conducted in working knots and technical work packages. It is first described in general terms, before being details in the following section.

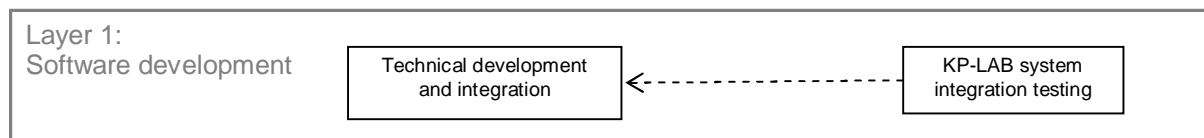


Figure 9: Layer 1.

Technical development and integration

Objective: This phase aims at developing individual KP-Lab tools. The tools are individually tested technically and functionally. Please notice that this phase includes system integration. This is justified by the fact that tools are integrated separately. KP-Kab has no such a task called System Integration that takes place outside the process of tool development. (Please read details of this phase in the section *Technical development: detailed description*.)

Input: Input comes from layers 1, 2, and 3:

- Usage scenarios
- High level requirements (to trace development)

Output: The results are:

- KP-Lab system domain model
- KP-Lab tools (tested individually)
- Domain ontologies
- Integrated KP-Lab system
- Technical documentation

Actors: WP4, WP5, WP6, Working Knots

Methods: The work in this phase follows a generalized 2TUP process. Each tool follows a 2TUP process. However, all tools share the technical specification branch. They also converge twice:

- First, to integrate and package the design: this leads to defining the tools and services to be developed.
- Second, to integrate the tools and services into a coherent KP-Lab system.

This 2TUP process is described below.

KP-Lab system integration testing

Objective: The integrated version of the system, in the current global cycle, is tested to ensure overall interoperability. All system-level metrics (defined within WP4) are measured. Reports, action plans, recovery actions and their follow-up are produced.

Input: Two type of artefacts:

- KP-Lab system (prototype)
- Integration testing scenarios (produced at the technical specification branch of the 2TUP).

Output: Integration report containing integration bugs, software metrics (interoperability, performance, reliability...)

Actors: WP4, WP5, WP6

Methods: An integration workshop is organised. The test of the system is completed. Integration test scenarios (produced by WP4) are individually run. Metrics and methods are used to produce data.

Tools development: detailed description

The actual development of every KP-Lab tool follows a 2TUP process (Fig.10). This sub-process is usually called Two Track Unified Process (2TUP). It is a unified process (UML-based, iterative, centred on architecture and driven by use cases). It has been chosen due to the simultaneous dual work in KP-Lab: technical and pedagogical. Fundamental technical work packages WP4 and WP5 proceed by analysing technical requirements internally in KP-Lab and technology evolution in general. They

develop and maintain the KP-Lab software architecture, supervising and monitoring a layered and service oriented design. They also specify, design and develop the various technical components (such as technical services – TSF). In addition to these, WP4 produces technical frameworks (e.g. the KP-Lab System Model) to be used by end-user tools developers. The life cycle of every tool has therefore two branches: one functional and the second technical. The functional branch specializes in specifying end-user requirements. These needs are analysed and translated into software specifications. The technical branch is shared by all tools. It concentrates on analysing technical needs, creating the KP-Lab system architecture and creating and adapting technical frameworks.

The two branches converge to the tool design phase. This is followed by tool coding and testing. The tasks of the sub-process are detailed below.

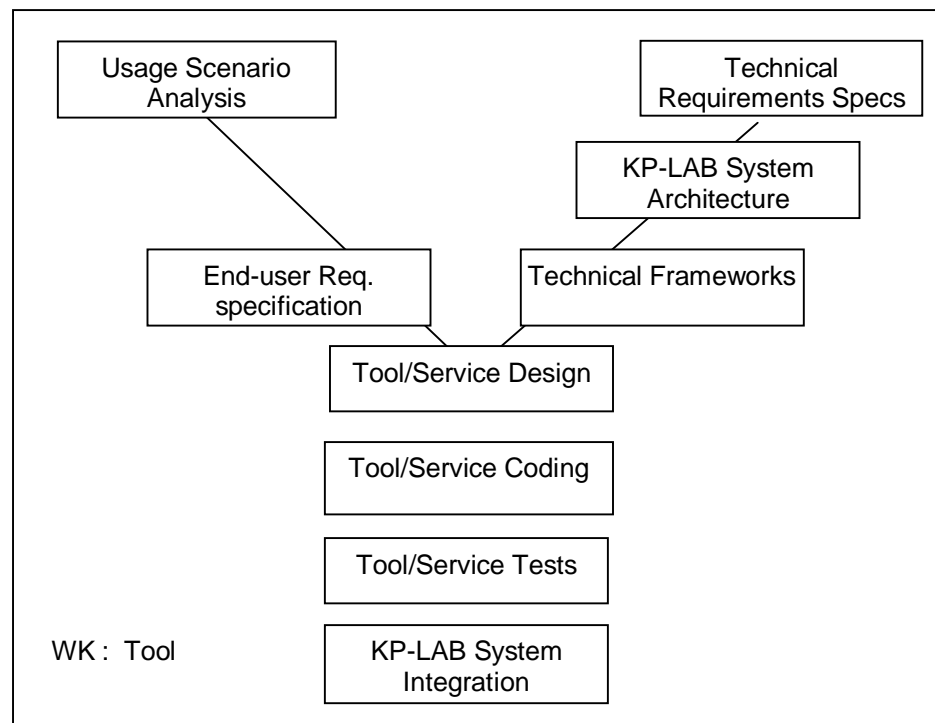


Figure 10: Tool development sub-process.

Figure 11 shows a detailed view of the ‘Technical development and integration’ activity. We notice that every working knot concentrates on one or multiple usage scenarios. They analyse them and produce software specifications (use cases, mockups, various diagrams). While the technical workpackages (WP4 and WP5) concentrate on technical analysis and specification. The results of this technical branche are combined with each WK specification to produce detailed software design. This phase leads necessarily to defining end-user tools and services that support them. In order to coordinate the design and share the coding phase, a design integration and packaging phase is established. This makes the decisions about which tools/services to develop, what architecture, what detailed design, and how to inetgrate them. The subsequent phases conduct the actual coding, testing and integration. The coding and testing of end-user tools are done within WKS and WP6. Whereas, the coding and testing of common services is carried out in WP4 and WP5. All coding tasks converge towards the KP-LAB system integration phase.

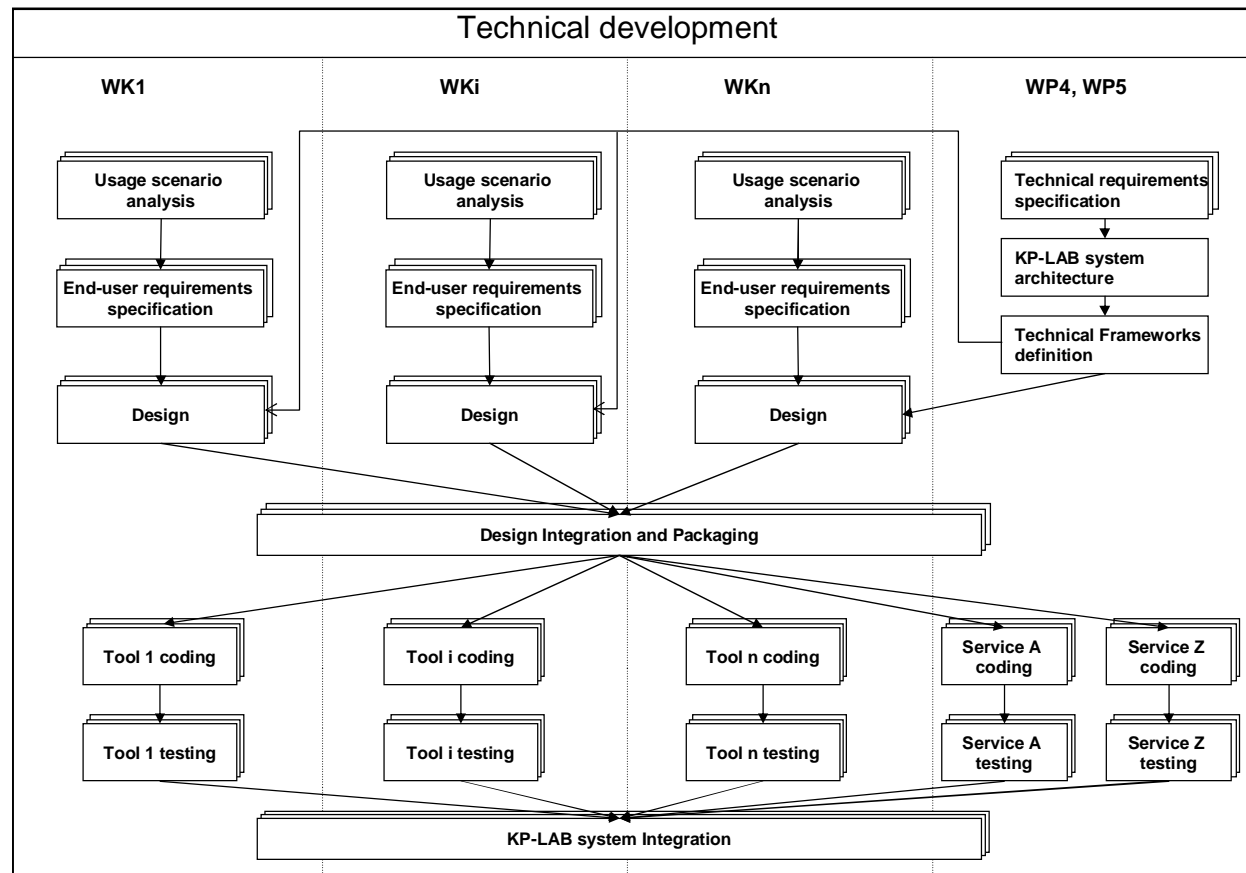


Figure 11. Technical development activity diagram.

Usage scenario analysis

Objective: The objective of this task is to further refine the requirements. UML formalisms such as use cases are used.

Input: Usage scenarios and drawings

Output: Semi-formal requirements documentation containing

- Use case model (use case diagrams, use case descriptions, sequence diagrams, domain model...)
- Mock-ups

Actors: WK

Methods: Usage scenarios are translated into semi-formal requirements. Actors use cases and relationships among them are identified. Use cases are documented according to an established template. System sequence diagrams are used to represent interaction scenarios. Mock-ups produced include low-fidelity mock-ups, based on simple drawings, as well as Software mock-ups, using prototyping languages such as Macromedia Flash.

End-User Requirements Specification

Objective: The objective is to make requirements complete, precise, measurable, verifiable, non-ambiguous and coherent.

Input: Use case model and mock-ups

Output: Revised

- Use case model
- Mock-ups

Actors: WK

Methods: Requirements are analysed to determine clarity, completeness, measurability, contradictions etc. Such problems are resolved. The artefacts and documentation is tested with scenario designers. The result is a revised use case model and mock-ups. Each of these items is uniquely identified (e.g. KPLAB UC 123) and their representation stored in the process database. Use cases are grouped into *Increments* according to prioritisation criteria, driven by the risks, in order to ease development and management. There might be one or many increments in the same tool cycle.

Technical requirements specification

Objective: This task aims at determining technical needs for individual tools and for the KP-Lab system as a whole.

Input: Design principles, knowledge practices and pedagogical scenarios.

Output: This work produces various artefacts and technical reports, including

- Technology evaluations,
- Existing tools demos and training
- Specific domains technical surveys (e.g. Educational technology standards, Web2.0 technological trends, semantic annotation tools...)
- KP-Lab system domain model

Actors: WP4, WP5, and WK

Methods: Technical partners analyse technical requirements for web based development. Technologies and trends are monitored and assessed. This produces recommendations for system and tools design. Technical requirements are also analysed for the tool under development.

KP-Lab system architecture

Objective: The objective of this task is to create, and maintain the KP-Lab system architecture.

Input: All technical documentation:

- Tools High-Level design documents
- KP-Lab System Model
- Specific Domain Model ontologies
- Technology evaluations

Output:

- KP-Lab system architecture

Actors: WP4

Methods: KP-Lab aims at creating an integrated and flexible system that implements triological learning models and methods. This system is a specific instantiation of Web2.0 conceptual framework for learning in educational and professional contexts. The system is based on two major foundations: semantic web and social software. It is service oriented and made of incremental layers of services.

Based on these principles, WP4 defines and develops a coherent architecture of the KP-Lab system. It registers and orders the different components, their assembling and and the interfaces/services they need or expose. It also defines the technical connectors (protocols, communication buses, utilities components...) that comply with the services orientation of the architecture. It also defines the deployment view of the system by addressing Logistic and Material aspects. This results in a system architecture document that will be used by all tool developers.

Technical frameworks

Objective: Depending on the technology needs and the architecture, this task specifies and/or develops technical frameworks and services to be used by tool developers.

Input:

- Technical solutions surveys
- Current KP-Lab system architecture

Output: Documentation of frameworks including:

- Developer guides
- APIs and reference implementations
- KP-Lab System Model

Actors: WP4, WP5

Methods: In order to produce a coherent and interoperable system, KP-Lab defines frameworks that should be adopted at different levels of the tool development (authentication, data access, streaming, views, session, communication...). Priority is given to open source standard APIs and frameworks. Specific frameworks and services such as semantic data storage, graphing, viewing, synchronizing... are created.

Tool/Service design

Objective: This task translates functional and technical requirements of a tool into a software solution.

Input: Functional and technical specifications

- Use case model
- Mock-ups
- KP-Lab system architecture
- Technical frameworks

Output:

- Tool detailed design documentation
- Integration test scenarios

Actors: WK

Methods: UML formalisms such as class diagrams, sequence diagrams, state-charts, and other diagrams when applicable are developed to cope with requirements.

Tool/Service coding

Objective: Coding consists of creating the actual computer programs of designed components and integrating them to create a coherent tool.

Input: All needed artefacts, especially

- Tool/service design documentation
- Technical frameworks and services
- KP-Lab system domain model

Output: The software tool:

- Packages of codes
- Technical documentation

Actors: WP6

Methods: Java is the recommended language for the core code. Other languages are possible especially for the presentation layer.

Tool/Service tests

Objective: Testing the tool individually at different levels and stages of development. The aim is to ensure correctness and validity.

Input: All pedagogical and technical requirements

Output: Testing reports for every level

Actors: WK

Methods: The code is unit tested and then tested as a whole. The prototype is also submitted for assessment and usability testing. These tests are done in an isolated environment (not integrated to KP-Lab system). Usability experts together with user testers evaluate the tool according to the usability guidelines and check lists as well as the functional specification. Results are documented in the respective functional specification and fed back to the local process for modification.

KP-Lab System integration

Objective: This task aims at integrating the developed tools/services into KP-Lab system. Please notice that this task occurs within each tool/service development. We do not wait for all tools/services to be developed in order to integrate the system. Integration is a continuous effort.

Input:

- KP-Lab system architecture
- Tool/service design documentation
- Technical frameworks
- Tools/services software Components

Output: Prototype of integrated KP-Lab system with documentation

Actors: WP4, WP5, WP6

Methods: Tools/Services produced every period of time (currently 12 months) are integrated into KP-Lab system. Any required connectors or complementary components to make them interoperable are created. All necessary tools and methods to evaluate software metrics are established. A test platform of the system is put in place. The outcome is the KP-Lab system that is not yet tested. The next phase will take care of technically validating the system

4.3 Emerging Design Practices

The KP-Lab co-design framework described above is adaptive in the sense that it does not prescribe any particular design method or practice but outlines the overall stages of the design process and their interrelation. Following the assumption that there is not one best design practice, this section does not present a fixed canon of design methods to be used, but gives an account of those design practices currently emerging within the project. The variation in the design practices found is at least partly due to the different scales of the tools developed. For example a tool like the Shared Space needs to be broken down to different components in order to be manageable and design activities have to be orchestrated and cross-checked periodically while rather autonomous tools such as CASS can easily be treated as a whole. Due to the heterogeneity and evolving nature of the design practices this section necessarily only provides a selective snapshot of the practices currently in place.

Due to the current state of the project the following practices are mainly focused on the early stages of the co-design process. The selection of practices presented here is based on their degree of visibility and distinctiveness within the project.

Besides the practices presented here, several trials are currently underway to explore new methods for context sensitive usability testing. As the analysis of these trials is not completed yet, these are not presented here. An overview of the ongoing trials on usability testing can be found in R2.6 “Summaries of the usability tests and Usability guidelines and recommendations for KP-Lab tools and methods.”

4.3.1 Collaborative Creation of Low-Fidelity Mock-Ups

Low-fidelity mock-ups in the form of simple drawings or power-point presentations are widespread used within KP-Lab especially in the early stages of the design process, including the design of case-based interventions and design experiments, the specification of high-level requirements as well as tool development. For example, this low-fidelity mock-ups have been used in the high-level requirements and tool development process of the KP-Lab Shared Space. While initially the creation of mock-ups has been an activity conducted by the technical partners, there is a growing amount of mock-ups created by pedagogical partners.

Objective/Purpose: Low-fidelity mock-ups provide an important means for explicating and elaborating early design ideas as well as design alternatives. Besides this, they are extremely valuable in high-level requirements elicitation, as they enable to concretize design ideas and to envision how a certain solution would look like and how it will work. Furthermore, the mock-ups often also provide hints on the author’s implicit assumptions on the usage situation. Hence, mock-ups not only allow to discuss the functionality required or the elements of the user interface for enabling the needed functionalities but also to better understand the underlying high-level requirements as well as bring forth the underlying assumptions that partners might hold.

Basic Idea: The basic idea behind the collaborative creation of low-fidelity mock-ups is to provide both developers as well as users with a mediating artifact that allows them to concretize and refine their ideas in an iterative process. Thereby it is important that the mock-up is understood as a tentative and revisable artifact rather than a fixed solution. By using low-fidelity mock-ups it becomes possible also for the end-user to directly express his or her ideas.

Main Steps and Basic Rules: Mock-up can either be build from scratch or “on top” of existing mock-ups or prototypes. The general procedure roughly is a follows:

- The author decides on the features he/she wants to describe. Thereby he/she has to figure out which aspects of the GUI or which interactions with the system he/she wants to focus on.
- The author creates one or more pictures/slides of the features chosen and describes each picture/slide briefly.
- The mock-up is discussed and evaluated both with developers and end-users. This discussion should be aimed both at understanding the underlying design idea as well as its implications from a technical as well as end-user’s point of view (see also 4.3.3 Multidisciplinary Design Workshops with Mock-Ups).
- Changes or extensions are made to the mock-up either ad hoc or after the meeting. Besides this the rationale for the changes made as well as open issues should be documented.

While the overall process is quite simple, the following rules seem to be relevant:

- The focus should be on understanding the design idea not the quality of the mock-ups appearance. Quick and dirty mock-ups might be more to the point than highly polished but distracting ones.
- Especially in the early stages of the design process, the tentative nature of the mock-ups should be highlighted.
- Discussion of mock-ups should take into account both the system’s as well as end-user’s point of view. In general it is important to have multiple stakeholders with different perspectives taking part in the mock-up review process.
- In the case of generic-purpose tools multiple contexts of use need to be considered simultaneously.

Outcomes: Besides the mock-ups itself, this practice is meant to contribute both to the elicitation and analysis of end-user requirements as well as the specification of the software artifact.

Challenges and Pitfalls: Even though in principle low-fidelity mock-ups can be created by the end-user it has to be kept in mind, that the visualization of design ideas is a non-trivial task and might require some training or exercise. The creation of mock-ups can become a very time-consuming task, especially when emphasize is put on the visual appearance of the mock-up. Finally there is the danger of getting trapped in an endless cycle of mock-up creation, discussion and update. Therefore it is important to limit the creation of mock-ups only to those features which have clearly traceable impact on the user’s interaction with the system.

For a more extensive introduction on prototyping see for example Floyd (1984), Snyder, (2003), Righetti (2006).

4.3.2 Tool Surveys

In addition to the regular state-of-the-art reviews foreseen in the Description of Work, tool surveys have also been used as a strategy to trigger the design process. For example in the early stages of the design of some note-editing functionalities recent products such as Google notebook, Livenotes and many more have been reviewed by technical as well as pedagogical partners to get a better idea of what they are looking for.

Objective/Purpose: From a design point of view the main purpose of the tool surveys is to familiarize with existing tools and hence to get a better idea of technical capabilities as well as end-user needs. By investigating and playing around with existing tools both pedagogical and technical partners are able to

rethink the high-level requirements they envision as well as to explore particular design solutions. In combination with mock-ups, tool survey may also foster the generation of new ideas in that they allow to rearrange and recombine features and functionalities treated separately before.

Basic Idea: The rationale behind this practice is twofold. While on the one hand tool surveys provide user and developers with an idea of “what works”, they also allow to better understand in which respects the product they want to develop will differ from existing tools. At the same time to mock-ups the tools surveyed provide mediating artefacts for those involved in the design process as they allow to refer to concrete examples.

Main Steps and Basic Rules: Even though tool surveys can be carried out in an ad hoc manner, the following steps are followed usually:

- Selection of criteria for the in/exclusion of tools in the survey (e.g. popularity, novelty, similarity to the product envisioned, or holding interesting features to ponder although the tool might have been meant for different use and purpose than the envisioned).
- Searching relevant tools using search engines (e.g. Google) or crawling tool repositories (e.g. SourceForge).
- Assessment of the tools. The degree of formality thereby depends on the overall purpose of the survey. Especially in the early stages of design a rather playful and explorative approach enhances creativity and open-minded approach to new ideas.
- Documentation of findings for future reference.

In case the tool survey is meant not just to assess the state-of-the-art but also to develop new design ideas on ones own, it seems especially important not just to compare the tool to some predefined criteria but play around with the tool or even test-use it for some real purpose.

Outcome: The outcome of this practice is more or less formal report, including a description of the tools surveyed as well as an account of the results of the assessment.

Challenges and Pitfalls: While this practice can bring forward new and innovative ideas it can also be limiting in the sense that it puts emphasize on what exists and not on what could be. Therefore it seems to be important to keep in mind the overall project’s objectives and also to keep a critical stance towards the tools surveyed. Besides this it is important to communicate the outcomes of the tool surveys internally as not every partner will have the time to review all tools by themselves.

4.3.3 Multidisciplinary Design Workshops

The following description is primarily based on the design workshops carried out in the development of the KP-Lab Shared Space. Face to face as well as virtual meetings took place both during the specification of high-level requirements as well as the tool development phase. In addition virtual and face to face meetings were also used as a platform to present and discuss the outcomes of various usability tests. Similar workshops have also been carried out in the context of MapIt for example. In general this type of workshop is applicable throughout the entire development process.

Objective/Purpose: The basic purpose of this type of design workshop is to elicit different types of requirements ranging from the high-level end-user needs to concrete functional requirements. These requirements are then used as a basis for the different technical specifications and tool development in general.

Basic Idea: The basic idea behind the multidisciplinary design workshops is to have series of intense and focussed meetings of developers, users, as well as pedagogical experts in order to develop a shared

understanding of the product to be developed as well as its context of use. The workshops thereby are highly interactive allowing participants to bring in and explicate their domain knowledge as well as design ideas. Furthermore the workshops are important to agree upon a common language and general practices for the team.

Main Steps and Basic Rules: Multidisciplinary design workshops are either carried out as a series of face-to-face meeting, virtual meetings or a mixture of both. In between the meetings there is usually a phase of asynchronous collaboration in which the design documents are updated and partners explore open issues in more detail before feeding it back to the group. As face-to-face meetings are somewhat different from virtual meetings these are described separately.

Face-to-Face Meetings

The face-to-face meetings are rather informal and open in nature providing a platform for brainstorming new ideas as well as the discussion of actual proposals and design artefacts, such as mock-ups.

The face-to-face meeting can either start with a brainstorming like collection of design ideas in the form of rough images or sketches or with an input-presentation followed by a phase in which participants visualize their ideas in response to the presentation. Possible input might include a pedagogical scenario, a mock-up or an existing tool.

In any case the participants' ideas are described and analysed in the plenary in order to gain a shared understanding of the underlying end-user needs as well as the proposed design ideas.

The outcomes of the meeting are documented as minutes and/or notes and used to create new visual or texts to be used as basis for the next meetings.

Even though this kind of workshops is quite informal the following rules appear to be helpful:

- Background material such as a meeting agenda should be made available to the participants before the meeting,
- The group should keep minutes of the outcomes of the meeting and make these minutes available to all relevant parties,
- Especially in brainstorming mode participants should be open to divergent ideas and try to avoid premature critique.
- In case there is an explicit chair he/she should take care of keeping the discussion on track and try to foster mutual understanding of the partners involved. Towards this end the chair should also raise clarifying questions and if needed ask for explanation of the claims made.

Virtual meetings

In contrast to face-to-face meetings virtual meetings follow a somewhat stricter procedure and also require more explicit preparation and moderation.

Background materials are delivered before the meeting and participants are asked to familiarize themselves with the contents as well as expected outcomes of the meeting.

In order to ensure a fruitful discussion all participants should read through the background material in advance. This is quite important as only selected material can be presented in virtual meetings and virtual meetings typically are much shorter than face-to-face meetings.

The virtual meeting itself is usually structured along a prepared demo of a mock-up or prototype. Thereby the presenter (preferably not the chair) breaks the presentation down into parts that are presented to group, after which all members can ask question and give comments. Just as in the face-to-face meetings the outcomes are documented as minutes and/or notes and used to create new visual or texts to be used as basis for the next meetings.

In addition to the guidelines for face-to-face meetings, the following rules seem to be especially relevant:

- Virtual meetings should be moderated by a chair which also keeps track of the time and coordinates the discussions.
- In general the length of the virtual meeting should be limited and the end of the meeting should be fixed beforehand.
- Both the presentation and comments parts should be kept to a specified time limit (e.g. 5-10 minutes each).
- If there are more than 5 participants, the comments/questions should be collected continuously, e.g. in a parallel chat. These questions are then dealt with in the question/answer period.
- If the questions/comments seem to be complex or too difficult to answer in short period of time, they should be postponed and dealt with after the meeting in meeting minutes and further (group) discussion.
- Finally care should be taken to explain all ideas fully, since this type of collaborative work is usually/always vulnerable to misinterpretations, as well as misunderstandings (concerns all materials, text, images, video, etc. Background material or materials produced/used in meetings)

Outcomes: Depending on the actual focus of a multidisciplinary design workshop the outcomes may include a description of high-level user requirements as well as concrete mock-ups or the functional requirements for the product envisioned.

Challenges and Pitfalls: Even though face-to-face meetings are easier to organize and chair a mixture of face-to-face and virtual meetings seems important, especially in a large and physically distributed consortium such as KP-Lab. Therefore virtual meetings can then extend the discussion to others who cannot participate in a face-to-face meeting.

Virtual meetings have many technical constraints such as learning how to use the specific software to share/chat/discuss the ideas efficiently. Furthermore limited bandwidth or connection problems due to local firewalls can lead to delays or prevent or even exclude a partner from participation. The problems have to be taken into account when preparing a virtual meeting.

Furthermore, changing participation in an ongoing discussion provides another obstacle. While new participants can bring in additional ideas and challenge tacit assumptions they also need to familiarise

themselves with a lot of background information including vocabulary as well as basic premises in order to "catch up" to others. In order not to re-discuss same topics over and over again it is important that new participants familiarize themselves with the existing background materials, while those already engaged in the design process should take care to record their decisions and outcomes carefully.

4.3.4 Bricolage (Intentional Re-Purposing of Existing Tools)

This practice has been used for example in internship scenario implemented by FH-OÖ. In this case the Learning Management System Moodle was repurposed to support a complex interactive scenario of remote collaboration. A more detailed description of this scenario is available in the Annex of Deliverable 8.1 'Scenarios and User Requirements for KP-Labs in Education'. The approach is focused on layer two of the co-design framework and allows to carry out design-experiments in the early phases of a project, when own prototypes are not available yet.

Objective/Purpose: The objective of this design practice is to gain a better understanding of user requirements and to anticipate needs that might emerge once an envisioned tool is in place, even in the early stage of design. The practice allows exploring new ideas in real world settings without the need of an extensive design process.

Basic Idea: The basic idea of this approach is to repurpose and tailor already existing products in order to create a working prototype which resembles important aspects of the system to be developed and to apply and test this prototype under real world conditions. The aim is not to create the "perfect" tool but to create a workable solution that can be tested under real world conditions. By drawing on existing (off-the-shelf) products it is possible to create working prototypes already in the early stages of the design process as well as to actively involve end-users in the creation and evaluation of these. The active involvement of end-users in the development of the prototype also allows them to familiarize with existing tools and hence to reflect on their needs as well as technical possibilities.

Main Steps and Basic Rules: In general the intentional repurposing of software can either be carried out in close collaboration of technical developers and end-users or by end-users on their own. The overall process roughly looks as follows.

- Specification of the intended context of use and initial requirements: Describe where, by whom and for which purpose the tool should be used. Describe the basic functionalities you envision.
- Search for and evaluation of existing products: Familiarize yourself with existing tools. Try to figure out which functionalities these tools provide and if they could be repurposed for your purposes.
- Development of a working prototype: Configure and tailor one or more tools in a way they meet the requirements best. This kind of customization might include for example the development of templates, the pre-structuring of a workplace or the sampling of a "toolbox".
- Inspection of the prototype: Evaluate the prototype against the requirements specified beforehand. Especially note those requirements the prototype does not meet. Document new requirements, that arose while creating the prototype.
- Application of the prototype: Apply the prototype in a real-world setting.
- Evaluation of the prototype: Collect information about the usability and utility of the prototype. Document new requirements and if applicable changes in practice or emerging needs.

The following rules seem to be especially relevant to this practice:

- Make sure that the prototype provides some added value to the user.
- Keep it simple! Don't make the creation of the prototype a project on its own.

- Be playful and try to make the best of the things you have at hand. Even though technical skills might be off help in some cases, often it might be sufficient to just figure out which tools already available would be most useful or if two tools just in combination could solve the problem.

Outcomes: The outcomes of this practice comprise a working prototype as well as a report on high-level requirements.

Challenges and Pitfalls: The practice might seduce the user or designer to make the creation of the prototype a project on its own. Secondly, even though this practice tries to build on off-the-shelf products the repurposing of existing tools might require some technical expertise or can be quite time consuming. Due to this reasons care should be taken that there is a reasonable chance to create a working prototype that provides at least some added value to the user. Furthermore the practice might not be applicable in case of genuinely new functionalities. Nevertheless even in such a case it might be worthwhile to think for example about a working paper-based prototype.

For a more extensive discussion of the concept of bricolage see for example Louridas (1999).

4.3.5 Heuristic Evaluation

Heuristic Evaluations have been applied especially in the phase of tool development, when user-tests have not yet been feasible. For example, KP-Lab heuristics were used by experts in the evaluation of the Shared Space in different phases (such as mock-up phase, and ready beta version). The KP-Lab heuristics were also used for categorising results from the field trial usability tests.

Objective/Purpose: The general goal of heuristic evaluation is to identify the usability problems in the design so that they can be addressed in the course of the iterative design process. Heuristic evaluation involves experts examining the interface and judging its compliance with recognised usability principles. The focus of heuristic evaluation is on those problems that are rather independent of the actual context of use such as issues of consistency, error prevention and handling or the visibility of system status.

Basic Idea: In general heuristic evaluation provides an analytic evaluation strategy, aimed to identify usability problems by careful inspection of the product. Thereby the product is assessed against a set of predefined heuristics. (A detailed description of the heuristics used in KP-Lab can be found in R2.5.2). In contrast to user tests and field trials heuristic evaluations do not require the existence of a working prototype but can also be applied to mock-ups. Furthermore it is less time consuming than empirical testes. Therefore heuristic evaluation provides additional value especially in the early phases of design when other methods are not applicable or when timely information on usability problems is required.

Main Step and Basic Rules: In order to reduce bias and increase the reliability of the findings, heuristic evaluations should always be carried out by more than one expert. Thereby the experts inspect the interface on their own first and discuss their finding only later on with the other experts. The individual evaluation can either be carried out by the expert on its own or together with a scribe who records the findings and interpretations made. A session for an individual expert lasts for about one or two hours. Longer evaluation sessions might be necessary for larger or very complicated interfaces with a substantial number of dialogue elements. In this case it is advised to split the evaluation into several smaller sessions, each concentrating on a part of the interface. The entire heuristic evaluation is structured as follows:

Phase 1: Individual inspection of the product

- During the evaluation session, the expert goes through the interface several times and inspects the various dialogue elements and compares them with a list of recognized usability principles (categories)
 - The first pass is intended to get a feel for the flow of the interaction and the general scope of the system.
 - The second pass allows the expert to focus on specific interface elements while knowing how they fit into the larger whole.
- The results of the evaluation are recorded by the expert or if available the scribe. If possible problems should also be documented with screenshots for easier reference in subsequent discussions with other experts and stakeholders.

Phase 2: Synthesis of findings and prioritization

- After all evaluations have been completed the experts discuss their findings and write an overall report including the problems, justification and description of the problem referring to the KP-Lab heuristic category as well as suggestions for improvement.
- Prioritizing of problems is done in collaboration with the stakeholders and developers taking into account the end-user needs, time constraints and other resources.

In addition the following rules should be obeyed:

- If the system is domain-dependent and the experts do not have sufficient domain knowledge, it will be necessary to assist the experts to enable them to use the interface. Otherwise, if the system is intended as a walk-up-and-use interface for the general population or if the experts are domain experts, no special assistance might be required.
- In order to enhance the comprehensibility and traceability of findings, the problems identified should be described exactly and justified with reference to the heuristics. Furthermore the expert should briefly explain the problem and possible implications.
- It is advised to be as specific as possible and list each usability problem separately.
- When discussing about the problems with developers and stakeholder it is advisable to state also the positive aspects of the design

Outcomes: The outcome of the heuristic evaluations are descriptions of the usability problems, preliminary suggestion of improvements referring to the appropriate heuristic category (principle) and preliminary listing of the severity and priority of improvements.

Challenges and Pitfalls: Due to the rather abstract nature of the heuristics, these have to be interpreted by the experts in order to be applicable to a given product. As a consequence findings are dependent on the experts experience and knowledge and disagreement might arise on findings and especially about the severity of the problems found. Therefore it is essential that the experts discuss and agree on what they have found.

Besides this heuristic evaluation may lack systematic analysis, which often leads to the fact that results found are not justified or explained in a way comprehensible to an outside stakeholder. Without a description and explanation of the problems found, the report might be of little value for the developer or other stakeholders.

Furthermore the assessment of a product's usability should not be restricted to heuristic evaluation only, as this method is not capable to track down context or task specific problems and hence might lead to a neglect of end-users' needs.

For a general introduction to heuristic evaluation the reader is referred to Nielsen (1994). More in-depth discussions on the strength and weaknesses of heuristic evaluation can be found for example in Coyle et al. (2006), Nørgaard and Hornbæk (2006) and Holzinger (2005).

Alternative uses for heuristics: Beside its use in heuristic evaluation KP-Lab heuristics can also be used as follows:

Firstly, KP-Lab heuristics can be used as checklist to remind about the main principles regarding usability. For example, there might be a need to check that all the error messages and/or on-line helps are consistent and include the appropriate user language and principles of error message descriptions and/or on-line help guidance descriptions.

Secondly, KP-Lab heuristics can be used to categorise usability findings from any kind of test as a background material to enable easier discussion of the problems found and how to prioritise the to-be-implemented improvements, new features etc, with the stakeholders, and developers.

5 Conclusions and Outlook

This final section provides a brief discussion of open issues we are facing and the next steps to be taken. Even though the co-design framework outlined in this document provides great advancement compared to the first version of this deliverable we still face a couple of open issues not yet resolved. To solve these problems will be a core task for the upcoming year.

Management of the Co-Design Process: The main objective of this deliverable has been to describe the rationale and systematic nature of the development process. While we outlined the main phases of the development process and their interrelations, the concrete practices as actualized in the Working Knots have to be evaluated regularly against the co-design framework and adapted in order to reduce eventual flaws or weaknesses. According to the rationale of the co-design framework, as outlined in this document, the process model should be understood as an overall guidance rather than a strict process regulation. In this sense the framework provides an overview of the entire process and might help partners to better understand the different perspectives and needs. Negotiation and reflection of working practices are clearly not rendered unnecessary by this framework. Therefore an important task for the upcoming year will be to foster the day-to-day management of the design process and to collect and disseminate viable design practices.

Documentation and Analysis of Emerging Design Practices: Besides the co-ordination of the design-process special attention will be paid to a systematic documentation and analysis of the

emerging design practices. The outcomes of this activity are meant to trigger internal exchange of best practices but also provide input to the refinement and update of the co-design framework itself.

Refinement of Layer 3 and Layer 4: Due to the actual needs of the project the main focus in this document has been on the technical aspects of system-development. Nevertheless a more detailed description of the main activities on layer 3 ‘pedagogical design and intervention’ as well as layer 4 ‘long term analysis of knowledge practice’ might be a reasonable extension in the future.

Transparency and traceability of the design process: Due to the size of the consortium, the multiplicity of tools envisioned and the heterogeneity of requirements the transparency and traceability of the design process is of great importance both for pedagogical and technical partners. In order to provide partners with a better overview of the current state of design activities we plan to develop and implement lightweight tools for the management of requirements as well as the creation and handling of roadmaps (DoW2, T2.1.4).

Theory based evaluation of KP-Lab tools: In order to ensure the quality of the tools under development as well as to make an informed decision on how to continue the further development a formative evaluation is foreseen. The aim of the evaluation will be to assess the tools from a theoretical perspective and to provide guidance for the Working Knots in charge of the tools. Towards this end a three step procedure is foreseen, including a self-assessment carried out by the Working Knots, an evaluation by members of the scientific board as well as final decision making by the Coordination Committee.

References

- Béguin, P. (2003). Design as a mutual learning process between users and designers. *Interacting with Computers*, 15, 709-730.
- Bødker, S. (1991). Activity theory as a challenge to systems design. In: H.E. Nissen, H. Klein, R. Hirschheim (eds.). *Information Systems Research: Contemporary Approaches and Emergent Traditions* (pp. 551–564). Amsterdam: Elsevier.
- Bødker, C., Christiansen, E., Thüning, M. (1995). A conceptual toolbox for designing CSCW applications. *Proceedings of COOP '95, International Workshop on the Design of Cooperative Systems*, January 1995, Juan-les-Pins, 266-284.
- Bourguin G., Derycke A., Tarby J.C. (2001). Beyond the Interface: Co-evolution inside Interactive Systems - A proposal founded on Activity Theory. *Proceedings of IHM-HCI 2001 conference*, 10-14 September 2001, Lille, France, 297-310.
- Büscher, M., Gill, S., Mogensen, P., Shapiro, D. (2001). Landscapes of practice: Bricolage as a method for situated design. *Computer Supported Cooperative Work*, 10(1), 1-28.
- Carroll, J.M. (2000). *Making use: Scenario-based design of human-computer interaction*. Cambridge: MIT press.
- Carroll, J. (2004). Completing Design in Use: Closing the Appropriation Cycle. *Proceedings of the 12th European Conference on Information Systems (ECIS 2004)*, Turku, Finland, CD-ROM, 11 pages.
- Coyle C. L., Iden R., Kotval X. P., Santos P. A., Vaughn H. (2006). Heuristic Evaluations at Bell Labs: Analyses of Evaluator Overlap and Group Session, *ACM, CHI 2006*, Experience Report April 22-27.
- De Sanctis, G., Poole, M.S. (1994). Capturing the complexity in advanced technology use: adaptive structuration theory. *Organization Science*, 5(2), 121-147.
- Engeström, Y. (1999). Activity theory and individual and social transformation. In: Engeström, Y., Miettinen, R., Punamäki R.-L. (eds.). *Perspectives on activity theory* (pp. 19-38). Cambridge: Cambridge University Press.
- Floyd, C. (2002). Developing and embedding autooperational form. In: Y. Dittrich, C. Floyd, R. Klischewski (eds.). *Social thinking – software practice* (pp. 5-28). Cambridge, MA: MIT press.
- Floyd, C., Krabbel, A., Ratuski, R., Wetzell, I. (1997). Zur Evolution der evolutionären Systementwicklung: Erfahrungen aus einem Krankenhausprojekt. *Informatik Spektrum*, 20(1) 13-20.
- Floyd, C. (1984). A Systematic Look at Prototyping. In Budde et al. (Hrsg.): *Approaches to Prototyping*. Berlin, etc.: Springer. 1-19.
- Highsmith, J. (2004). *Agile Project Management*. Boston: Addison Wesley.
- Holzinger A. (2005). Usability Engineering Methods For Software developers. *Communications of the ACM* 48(1): 71-74.
- Kaptelinin, V. (2002). Making use of social thinking: The challenge of bridging activity systems. In: Y. Dittrich, C. Floyd, R. Klischewski (eds.). *Social thinking – software practice* (pp. 45-68). Cambridge: MIT press.
- KP-Lab (2006a). *Guidelines and models on implementing design principles of KP-Lab, application scenarios and best practices, v. 1*. Deliverable 1 of WP 2, on co-evolutionary design.
- KP-Lab (2006b). *Scenarios and User Requirements for KP-Labs in Education*, Deliverable 1 of WP8, on knowledge practices in education.
- KP-Lab (2006c). *Recommendations for Design Principles of Trialogical Technologies*, Deliverable 1 of WP3, Theoretical foundations of knowledge–practices laboratory.
- KP-Lab (2007a). *Report of the Task Force on Usability activities*. Result 2.5.2 of WP2, on co-evolutionary design.
- KP-Lab (2007b) *Summaries of the usability tests and Usability guidelines and recommendations for KP-Lab tools and methods*. Result 2.6 of WP2, on co-design.
- Law, J. (1992). Notes on the Theory of the Actor-Network: Ordering, Strategy and Heterogeneity. *Systems Practice* 5, 379-393.

- Leont'ev, A.N. (1978). *Activity, consciousness, and personality*. Englewood Cliffs: Prentice Hall.
- Louridas, L. (1999). Design as Bricolage: Anthropology Meets Design Thinking. *Design Studies*, 20(6), 517-535.
- McDermid, J.A. (1994). Requirement analysis: Orthodoxy, fundamentalism and heresy. In: M. Jirotko, J. Goguen (eds.). *Requirement engineering: Social and technical issues* (pp. 17-40). London: Academic Press.
- Miettinen, R., Hasu, M. (2002). Articulating user needs in collaborative design: Towards an activity-theoretical approach. *Computer Supported Cooperative Work*, 11(1-2), 129-151.
- Mwanza, D. (2002). *Towards an Activity-Oriented Design Method for HCI research and practice*. PhD thesis, Knowledge Media Institute, Open University UK.
- Nielsen J. (1994). Heuristic evaluation. In: J. Nielsen, R.L. Mack (eds.). *Usability Inspection Methods*. New York: John Wiley & Sons.
- Nørgaard M., Hornbæk K. (2006). What Do Usability Evaluators Do in Practice? An Explorative Study of Think-Aloud Testing. *ACM, DIS 2006*, June 26–28, University Park, Pennsylvania, USA.
- Reckwitz, A. (2002). Toward a Theory of Social Practices – A development in culturalist theorizing. *European Journal of Social Theory*, 5(2), 245-265.
- Righetti X. (2006). *Study of Prototyping Tools for User Interface Design*. Information system Interfaces, Université de Genève. Unpublished. [Retrieved 16.07.2007 from: <http://thefabric.com/articles/RapidUIProto.pdf>.]
- Rönkkö, K. (2002). “Yes-what does it mean?” Understanding distributed requirements handling. In: Y. Dittrich, C. Floyd, R. Klischewski (eds.). *Social thinking – software practice* (pp. 223-242). Cambridge: MIT press.
- Rosson, M.B., Carroll, J.M. (2002). *Usability engineering: scenario-based development of human-computer interaction*. San Francisco: Morgan Kaufmann.
- Schuler, D. and Namioka, A. (1993). *Participatory Design: Principles and Practices*. Hillsdale: Lawrence Erlbaum Associates.
- Snyder, C. (2003). *Paper prototyping*. San Francisco: Morgan Kaufmann Publishers.
- Spinuzzi, C. (2003). *Tracing genres through organizations – a sociocultural approach to information design*. Cambridge: MIT press.
- Törpel, B., Wolf, V., Kahler, H. (2002). Participatory organizational and technological innovation in fragmented work environments. In: Y. Dittrich, C. Floyd, R. Klischewski (eds.). *Social thinking – software practice* (pp. 331-356). Cambridge: MIT press.