MOMOCS

Model driven Modernisation of Complex Systems

DELIVERABLE # D11
XIRUP METHODOLOGY REQUIREMENTS

Dissemination Level:	Public
Work package:	1
Lead Participant:	Politecnico di Milano
Contractual Delivery Date:	January 2007
Document status:	Final
Preparation Date:	31 January, 2007
Document Version:	1.5

Revisions

Document Version	Date	Author (Partner)	Description of Changes
1.0	16/10/06	PoliMi	Initial table of contents
1.1	15/12/06	PoliMi	First set of requirements identified
1.2	17/01/07	PoliMi, TID, Softeam	New requirements added
1.3	25/01/07	PoliMi	Detailed descriptions added
1.4	30/01/07	PoliMi, Siemens, and TID	New domain-specific requirements added
1.5	31/01/07	All partners	Final version

INDEX

1 Scope

1.1 Identification

This deliverable describes the requirements for the XIRUP methodology. Together with deliverable D12, it constitutes the basis on which the XIRUP methodology will be conceived and formalised.

1.2 Project Overview

MOMOCS aims at providing a methodology, along with related tools, for the fast reengineering of complex systems. Even if a complex system comprises both hardware and software components, MOMOCS mainly deals with its software parts, but the project also aims at addressing the interdependencies among the other components and between software and hardware elements. MOMOCS studies how a complex (software) system can be modernised with the goal of keeping it aligned with the fast changes that characterize modern business processes and technical environments, and with the aims of having human beings as the centre of the modernization.

To this end, MOMOCS allows users to concentrate on what to do, that is, the actual engineering principles behind the modernization, instead of wasting time and resources on understanding how to do it (and its implications with the rest of the system). The project also aims at solving the dilemma between rigorous and bureaucratic versus agile and unstructured methodologies by proposing XIRUP, an eXtreme end-User dRiven Process methodology, along with its supporting tools.

1.3 Document Overview

This deliverable describes the requirements for the XIRUP methodology. Requirements are organized in: general requirements, to state simple and high-level directives for the whole XIRUP, methodological requirements, to cover what concerns the approaches adopted by the methodology, and technological requirements, to characterize the notations and tools supported by the methodology.

The synergies with deliverable D1.2 [6] are obvious. This document builds on the results presented in that document and tries to widen the scope of identified requirements and then tries to generalize the key features behind XIRUP. There are also clear connections with deliverable D2.1 [7], which describes what tools should be used to support XIRUP.

The document is organised as follows. Section 2 briefly explains how requirements are presented in the document. Section 3 gives a general definition of what modernization is for XIRUP, and thus for the project. Section 4, 5, and 6 describes the high-level, methodological, and technological requirements, while Section 7 presents some further requirements added by the demonstrators and not

sufficiently covered in the previous sections. Section 8 concludes the document by drawing some final comments.

2 Methodology

Requirements elicitation has been an active discipline for years: many proposals [2, 3, 5] tend to foster different rules, notations, and heuristics to improve the way requirements are specified and, in some cases, also offer a "precise" means for their analysis and validation.

In this document, we prefer to use an almost informal style to cope with our key goal of lightweight and agile approaches. It is also true that, unfortunately, none of these proposals has clearly demonstrated its merits over the others so far. In this particular case, the adoption of a precise methodology/notation would unnecessary complicate the elicitation of the key characteristics of our modernization methodology (XIRUP). The productive interaction with our demonstrators requires easily comprehensible statements and a simple domain-independent language.

These key objectives led us to adopt a simple template, based on the use of natural language, to characterize each requirement. Orthogonally, requirements are divided in classes to provide the reader with a simple taxonomy:

- **General requirements** state simple and high-level directives for the whole methodology.
- **Methodological requirements** cover what concerns the approaches adopted by the methodology, the aspects it covers, the processes it embodies, its prescriptive steps, and the foreseen degree of automation.
- **Technological requirements** characterize the tool support ascribed to the methodology, the technologies it is supposed to deal with, and the notations used in the different phases/steps of the methodology.

Each requirement is described through:

- **Unique id** makes the requirement traceable throughout the whole specification process. Given the distinction among requirements described above, the id is composed of: **G**, **M** or **T**, to state whether it is general, methodological or technological, an **R**, which means requirement, and a number, which is a dedicated counter for each category. For example, **MR3** identifies the third methodological requirement.
- **Definition** defines in natural language the requirement itself. This is a short sentence that must state a request or set a constraint on what XIRUP, along with its supporting tools, is supposed to do.
- **Explanation** explains and exemplifies the requirement to help the reader better understand what the requirement is supposed to state.
- **Demonstration** explains how we plan to demonstrate the fulfilment of such a requirement within the project.

3 General Vision

This section states the vision for the whole methodology in terms of both the context in which it is supposed to operate and its interpretation of term **modernization**. Even if MOMOCS is mainly concerned with the modernization of software systems, XIRUP should be conceived with the more general, and bolder, goal of addressing the modernization of **complex systems**, no matter of the actual nature of its components.

An industry-oriented approach towards model-driven modernization of complex systems requires that some constraints be defined. A **system** is made of one or more cooperating **components**[1] (static view) that cooperate to realise one or more **processes** (dynamic view). A system becomes **complex** when we hardly understand it as a whole just by considering its components. This broad definition of (complex) system does not require that components be characterized as software elements. The distinction, between software and non-software artefacts, will become much more important when we consider the automatic transformation and deployment of selected components.

A better understanding requires that elements be grouped wisely into **sub-systems** to deliver meaningful features. Sub-systems provide a method to analyse functionalities and interfaces within a complex system in order to define exchangeable and semantic relevant components through composite structures. Particular industrial branches, along with their domain-specific best practices, provide well-known engineering patterns and standard component libraries to help identify components and sub-systems and support their interchangeability and reuse.

A complex (sub-)system is thus understandable by composing/decomposing the different elements according to the particular needs to create different **views** on the system. The ultimate goal is the creation of **models** that are enough to support, guide, and oversee the whole modernization process. A model describes an aspect of the whole system, or of one of its sub-systems (components). It might describe the architecture or the behaviour of a given (sub-)system. A model may embed a hierarchical organization and provide different abstraction layers. The ultimate goal is to provide a container for information that we want to communicate to the different roles involved in the modernization process. If roles are human, then we also need to consider the actual representation of this information to ease its understanding. Models can be inferred from existing artefacts (other models, code, designs, etc.) or be obtained (semi-)automatically by transforming existing ones.

Roughly, the model driven modernisation of a complex system means the model-assisted re-engineering of its sub-systems/components to improve available features or provide new ones. Thus, the vision behind XIRUP is *the definition of models and model-based reasoning and transformation techniques to help engineers understand existing systems and move towards modernized ones*.

[1] In this document, we use the terms component, element, and object as synonyms. We do not intend to restrict our attention to the software-related meaning associated with these terms.

3.1 Intended Modernization

Modernization is the process of understanding and evolving existing assets for the purpose of improvement, modifications, interoperability, refactoring/restructuring, reuse, integration, and migration.

MOMOCS associates the term *modernization* with the three main components of a complex (software) system, that is, data, *ware, and process. Data modernization is applied to data structures and their semantics, *ware modernization focuses on *ware reusable interfaces, componentization, and robustness of architectural design, process modernization addresses the logical flow of the complex system and aims at ameliorating the system by re-engineering its business processes and improving the overall responsiveness of the system.

Modernization activities will be based on the basic actions of Figure 1, which will be applied onto the data model, *ware models, and process models. We start by inspecting the different views/diagrams of the models to identify a subset of elements (e.g. classes, associations, dependencies, task, data structures) to modernize. This may end up adding **New functionality** or **Reengineering** existing elements. To this end, we may simply choose **Migration**, which means moving data, *ware, and process from one operating environment to another, or decide for a deeper revision of the system (**Re-architecting**), which might mean **Generalisation**, to reorganize objects or groups of object to reuse them in other models/projects, **Simplification**, to reduce the complexity of models (e.g. complex data structures, classes, interface iterations, processes/tasks), and **Maintenance**, to modify objects/group of objects by deleting (or substituting) unused/obsolete/buggy elements.

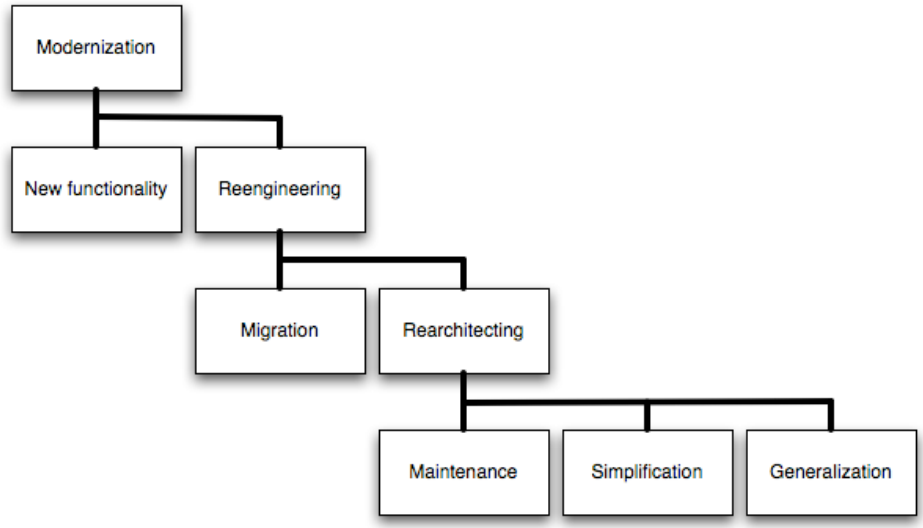

Figure 1: Intended modernization

Data, *ware, and process modernizations are iterations of the basic actions (i.e., one or more basic actions need to be combined and repeated several times in different ways). Modernization tools will support basic actions by providing automatic or semi-automatic facilities.

4 High-level Requirements

This section presents the requirements that provide a high-level characterization of XIRUP. It mainly identifies its origins and the other features that let the reader frame the approach. We start by clarifying how we plan to consider existing OMG activities, and we continue by posing the requirements on how XIRUP can be compared with well-known development processes, agile methodologies, modelling notations and software/hardware technologies. We conclude by characterizing the wide spectrum we foresee for XIRUP and by stressing the need for flexibility, which is typical of any modern development process.

GR1	*XIRUP will assist users throughout the whole development process.*
Explanation	This means that XIRUP aims at borrowing the steps of conventional development processes to identify the different phases that might be affected by modernization initiatives, or that must cope with the consequences of what done in previous (with respect to the development process) activities. For example, testing activities might be touched both per-se and because of changes in the system. It could be useful to test different aspects of the modernised artefacts using different profiles in order to have an overall impression of the modernised system and a higher covering of tested functionality. We envisage that different modernization efforts do not need to concentrate on all the phases, but they may concentrate on some particular phases, along with the consequences they produce for down-the-stream activities.
Demonstration	Our two pilot modernization efforts concentrate on quite different systems. Thus, we envisage that the fully-software scenario may want to cover and demonstrate the whole lifecycle (till the semi-automated deployment of newly produced artefacts), while the mixed scenario will mainly demonstrate how XIRUP support engineers while conceiving and analyzing changes and while testing obtained results.
GR2	*XIRUP will provide an iterative and flexible template for the modernization process.*
Explanation	Decades of software development processes have thought us that strict and prescriptive processes do not work and are not able to capture the details that characterize each development effort. To this end XIRUP aims at being as less prescriptive as possible to provide a very lightweight infrastructure to set the different modernization activities. The goal is to identify both mandatory and optional activities to provide users with a customizable infrastructure for modernization processes. Depending on the application domain, the legacy system and the kind of

	modernization, different processes can be envisaged by combining different activities/methods
Demonstration	We do not want to design fake customizations of the proposed modernization process infrastructure, but the key differences between the two pilot application scenarios will help us characterize the proposed framework to accommodate the different needs and peculiarities behind the modernization efforts, application domains, and technological settings.
GR3	*XIRUP will embody the model driven paradigm and will comply with the ADM (Architecture Driven Modernization) directives [8] provided by the OMG.*
Explanation	The most relevant example to support this requirement is the compliance of the MOMOCS Knowledge base with the OMG KDM (Knowledge Discovery Meta-model). We also plan to consider the other packages, both those that already exist (e.g., analysis and metrics) and those that will come, to assess the suitability for XIRUP and embed the main underlying concepts in our methodology.
Demonstration	We will devote particular attention to highlight the connections between XIRUP and ADM initiatives. To do this, we will stress the commonalities and the extensions that XIRUP might propose to OMG. The evaluation will be mainly based on showing the similarities among concepts. We think that any more rigorous demonstration does not apply given the high level of the requirement. Both the modernization case studies will start by populating a KDM-like knowledge base and thus conducting analyses and transformations aimed at meeting the goals set by the demonstrators, but also the compatibility, improvements, and key differences with the different ADM packages.
GR4	*XIRUP will be iterative and incremental, use-case driven (and thus based on scenarios), architecture centric, and with special attention to the risk associated with the different modernization initiatives.*
Explanation	The term modernization is so general that it cannot be addressed with single-shot approaches. The different dimensions already introduced in Section 3.1 demand for iterative and incremental approaches. The different options (e.g., different architectures, different middleware infrastructures, and different programming languages) must be evaluated incrementally by considering their effects on the to-be-modernized system [10]. Modernization use-cases and scenarios will be used to "scope" the different activities, understand their actual impact of changes and try to understand their consequences. Every modernization activity will be conceived by starting from the global (software) architecture of the system we want to modernize and by conducting special-purpose analyses on the risks associated with that activity.

Demonstration Both the modernization efforts will be conducted as iterative and incremental processes. The impact of proposed modernization steps, along with their risk, will be properly addressed and demonstrated.

The presentation of the example applications in deliverable D1.2 [6] is already organized around scenarios to demonstrate the incrementality of proposed changes. Risk assessment and implications will be considered a key driver for the actual enactment of the proposed steps.

GR5 *XIRUP will be based on spike solutions (i.e., never add functionality before it is scheduled) and on accurate planning of the different steps. It will also focus on the problem under examination, without considering unnecessary features, and on the unit test of the different modernized elements.*

Explanation When we think of conceiving the modernization of a complex system, one of the first problems we have to face is the definition of the actual problem we want to face and its impact on the existing system [1]. Moreover, this kind of systems are usually already in use and they cannot be stopped or be stopped for long periods. This means that the modernization steps must be as scoped as possible to allow for short maintenance time and limited impact on the actual system.

The thorough test of changed elements (unit test) becomes fundamental to understand and evaluate the impact of such changes. A complete new test campaign is not feasible given the constraints listed above.

This applies especially to the need of improving automation systems, but it is also important for porting and integrating telecommunication systems onto a service-oriented platform.

Demonstration The demonstration of this requirement will be based on conceiving modernization processes that account for clear, well-defined, and focused modernization steps to smooth and mitigate the transition from the old to the new (modernized) system. Each step will be accurately planned, constrained to the minimum set of elements, and evaluated in the proper context.

GR6 *XIRUP will not embed any predefined notation or language. Different domain-specific notations can be added as plug-ins to render the different concepts.*

Explanation Since we already know that different domains are used to different notations, and also different aspects of the same system may require special-purpose notations, we think that the adoption of a single language, or a limited set of languages, would be a too heavy constraint for XIRUP. Everything must be integrated through the common knowledge base, which is a homogeneous and notation-agnostic representation of the system on which we want to work.

XIRUP must guide the user to populate such a common representation of the system, to retrieve stored information, and transform it if needed, but this must be done without forcing any concrete representation. Suitable, and special-purpose, filters must be used to interact with the knowledge base in the two directions.

Demonstration The two pilots, along with their different domains, use of tools, and notations, are already a good example of how the "same" concepts come from and must be rendered through specific notations and must be materialized in domain concepts. Most probably UML-like languages will be used, in the spirit of "not reinventing the wheel" and due to its vast diffusion among the growing community of "modernisation engineers".

The goal is to provide enough filters and adaptors to materialize XIRUP in the two domains addressed by the project, but at the same time, we also want to demonstrate the generality of proposed ideas.

GR7 *XIRUP will not be natively based on any software/hardware technology. Different technologies can be supported by means of add-hoc plug-ins.*

Explanation Similarly to UP, no software or hardware technology will be hard-coded within XIRUP. Again, the goal is provide a set of meta-concepts that can be instantiated in different contexts: purely software systems (and we already have myriads of different options), mixed systems, and maybe complete hardware ones. The more we leave the soft domain, the less XIRUP will be prescriptive in terms of the transformations needed to modernize components, but it will still be possible to maintain its underlying concepts.

Demonstration The presentation of the two modernization efforts described in deliverable D1.2 [6] should be a convincing way for demonstrating the openness and technology neutrality of XIRUP. The automation and telecommunication scenarios already contribute with different domains, different constraints, and many different technologies that seem to be enough to pave the ground to even wider possibilities.

A purely software system (the telecommunication scenario) might lead to a more automated and prescriptive instantiation of XIRUP, while the automation scenario, which is intrinsically mixed and many elements are not software components imposes a light-weight and more decentralized use of the methodology.

5 Methodological Requirements

This section presents the requirements associated with XIRUP as far as the methodology is concerned. It is true that XIRUP itself is a methodology, but for the sake of readability, we have presented its requirements around a slightly more organized taxonomy. This means that the requirements in this section concentrate on the actual support and on the different phases of the modernization process. They show why, how, when, and by whom the knowledge base should be populated, queried, and modified. They also emphasise the main development phases touched by XIRUP and its support to (automatic) transformation of system artefacts.

MR1	*XIRUP will assist users while eliciting the modernization requirements (both functional and non-functional), that is, the quality dimensions that have to be improved, what has to be changed/modified/updated, and the consequences of these changes.*
Explanation	Section 3 explains how MOMOCS and XIRUP envisage the modernization of a complex modern system. Given the many modernization options and quality parameters that we should consider while starting a modernization process, the user must be thoroughly assisted during the elicitation of his/her modernization requirements. XIRUP will thus provide a dedicated methodology for the identification and prioritization of these requirements as first step towards the other phases of the proposed methodology, that is, impact and cost analyses, actual modernization (that is modification, addition, deletion, and changes to the system we want to modernize), transition (deployment) and thorough validation of obtained results. Notice that XIRUP will assist users while creating the system knowledge base of new components that are planed to be integrated to the modernized system.
Demonstration	For the sake of simplicity, the two pilot modernizations will concentrate on dedicated use cases and scenarios to elicit the main drivers for modernization. Even if other, more formal and rigorous, approaches may be used, we prefer to stick to the UML-like world given its wide diffusion and relative ease of use. Defined use cases (and scenarios), whose first seeds are already in deliverable D1.2 [6], will help the user understand what should be modernized and why, which quality dimensions have to be privileged, and also what kind of results/benefits the modernization effort is supposed to produce.
MR2	*XIRUP will assist users while creating the system knowledge base of the system they want to modernize.*

Explanation The knowledge base is the key element behind XIRUP. Besides its complete and comprehensive definition, it is mandatory that the modernization methodology assists the user while populating it. We do not only aim at creating all the different system elements, components, and artefacts, but we also need all the relationships among the entities in the knowledge base. Relationships may also come with restrictions and constraints that are mandatory for the soundness of stored data.

Its correct population must be regulated by XIRUP both in terms of who adds what and in terms of when the parts of the base should be populated and how.

Demonstration Being this requirement mandatory for the correct enactment of XIRUP as modernization methodology, both case studies will thorough explain how, when, and by whom to populate the knowledge base.

MR3 *XIRUP will support different analyses on the system knowledge base (e.g., impact analysis, cost analysis, etc.)*

Explanation Before conceiving the actual modernization steps, and thus before identifying what should be modernized and how, users need to understand what they can do and what the "costs" of these actions would be. This means that XIRUP must assist users who simply want to query the knowledge base to get enough information to plan the next steps.

Two key examples of this querying activity are the identification of the dependences among elements and the capability of predicting the consequences of possible changes. In the first case, XIRUP must advice on when and how the knowledge based should be exploited to collect all the dependences among system components, between elements and their definitions, and between definitions of similar/conflicting components. In the second case, XIRUP will be in charge of assisting the user while simulating "what-if" scenarios to discover the most convenient way for changing a given system with respect to some modernization dimensions.

Demonstration Both case studies will propose some dedicated (and domain-specific) analyses to let the user understand and plan the best modernization steps (with respect to quality dimensions used for conducting the exploratory study). We want to stress the importance of these exploratory analyses, and more in general the capability of formulating "smart queries" to the knowledge base as a way to manage the intrinsic complexity of these systems.

MR4 *XIRUP will allow users to define automated transformation of existing artefacts and elements to (semi-)automatically produce those of the modernized system.*

Explanation This requirement matches the idea of being model-based and of adopting the concepts and approaches proposed by the OMG within the model-driven architecture initiatives. Once the knowledge base is created,

	XIRUP must provide sufficient methods, and tools, to support the automatic transformation of existing artefacts into the modernized ones. XIRUP will also advice on how to consider the implications that such changes may have on the rest of the system. Besides changing/migrating some components, it must be clear also the impact, along with possible problems, that foreseen transformations have on the whole system we want to modernize.
Demonstration	This requirement will be mainly demonstrated by TID's case study. The transformation among software artefacts described in deliverable D1.2 [6] will be studied and applied according to the guidelines stated by XIRUP and they will be implemented by means of the more appropriate tools (see deliverable D2.1 [7]). In contrast, Siemens' demonstrator will mainly serve to address the problem of scoping possible transformations and thus XIRUP will be used mainly to assist the user while conceiving and evaluating possible (or already applied) improvements, but not while implementing them.
MR5	*XIRUP will assist users to define transition and deployment plans from the existing system to its modernized version(s).*
Explanation	One of the key steps of modernizing existing systems is the transition for the old to the new version of the application. The transition must be as smooth and tool-assisted as possible. We can also think of customizable deployment plans to assist users while migrating their systems, ease the transition and mitigate associated risks. If the system we want to modernize is a fully software system, we can also think of (partially) automating the deployment process, that is, the creation of the deployable artefacts and then their injection into the system.
Demonstration	The different natures of our demonstrators imply two different instantiations of these concepts. The software system allows us to conceive a fully automated process that implements the deployment of the new components, and thus oversees the whole transition process. Automation scenario imposes that XIRUP only assists and advices the experts involved in setting the new elements, but it cannot be as prescriptive as in the first case.
MR6	*XIRUP will pay specific attention to and support the regression testing of the modernized system.*
Explanation	Generally speaking validation is one of the key problems of many software systems. Testing activities are often neglected and considered scarcely important. On top of this, we can add that a modernized version of an existing system should be at least as good as its predecessor. Thus, the particular attention devoted to regression testing activities (both on the modernized components in isolation and on resulting systems) is

	twofold. It is essential to "demonstrate" the quality achieved by the modernized system, and thus orthogonally to justify the modernization effort, but it is also mandatory that testing activities be as smooth and limited as possible. To this end, we need to consider that we aim at systems that are already in operation, and thus every heavy and long validation activity might impact its actual capabilities and performance.
Demonstration	Both the modernization efforts presented in deliverable D1.2 [6] will address testing and validation. Their different natures lead to two different enactments of testing campaigns. Purely software and mixed systems require different heuristics, methodologies, and settings to carry out the validation of modernized elements and systems.
MR7	*XIRUP will support users to discover, adapt and use domain-specific and generic modernization patterns.*
Explanation	Starting from the famous book by Gamma et al. [4], and also before if we consider other design disciplines, patterns (not only design patterns) have been around for years and many domains have tried to borrow the concept and identify their incarnations. The idea of modernization pattern is in line with this trend and aims at offering user guidelines, heuristics, and tool support, to discover domain specific modernization patterns, that is, recurring solutions to tackle modernization problems, and apply them onto their systems. The hypothesis of a common knowledge base behind XIRUP leads to the idea of addressing the problem of modernization pattern at this level, and also of providing suitable means to reason on, apply, and visualize them in the different XIRUP instantiations.
Demonstration	We plan to demonstrate the approach offered by XIRUP to discover, define, and utilize modernization patterns by creating ad-hoc scenarios within the modernization exercises presented in deliverable D1.2 [6].

6 Technological Requirements

This section deals with the "technological" requirements that XIRUP is required to meet. This means that here we refine the idea of system knowledge base, how it should be used and what features it should offer to the user. Moreover, we also identify the kind of languages that will be supported by XIRUP. As already stated, XIRUP does not offer any predefined languages to render all the different elements and artefacts, but we need to be sure that a coherent and consistent instantiation of the methodology offers and supports enough notations for the different concepts that the user wants to consider while modernizing his/her systems.

TR1	*XIRUP will work on a system knowledge base that contains both the actual elements/artefacts that belong to the system we want to modernize and also their definitions.*
Explanation	Impact analysis and modernization initiatives often depend on the characteristics of existing and possibly new components. This means that the capability of relating actual elements/artefacts with their definitions is mandatory to let the user reason on the implications of some changes and also on the compatibility issues that such changes may introduce. If we render these concepts in terms of the MDA jargon, we have to say that XIRUP, and its supporting knowledge base, must deal with concepts and meta-concepts (level 0 and level 1 in the OMG hierarchy [10]) seamlessly and in a integrated way. The availability of level 1 definitions, along with constraints on their relationships with other elements, is also a way to keep the consistency of level 0 models.
Demonstration	The seamless integration of elements and their definitions is interesting for both our pilots even if they will use these features in slightly different ways. If we think of software components, the availability of sound definitions is important for both evaluating the different modernization alternatives and conceiving automatic translations. Non-software components do not allows for this last feature and thus the availability of definitions can only be used while populating the knowledge base and while performing impact and cost analyses.

TR2	*XIRUP will provide a system knowledge base that adopts the meta-modelling technologies like MOF and complies with the KDM proposed by OMG.*
Explanation	This requirement aims at stating the technological constraints behind the XIRUP knowledge base and stresses the need for compatibility with existing standards, and specifically those proposed by OMG. MOF (Meta Object Facility, [10]) is the standard notation for rendering meta-models, while KDM (Knowledge Discovery Metamodel) is the emerging standard

for eliciting the knowledge behind a (software) system.

Even if these two standards are our two main references in these days, we want to stress that XIRUP aims at being compatible with them, and not just at mimicking them. Moreover, other new standards might be considered if necessary.

Demonstration The compliance with this requirement will be demonstrated while conceiving and implementing the tools behind XIRUP. As already said, we will pay particular attention to the compatibility with existing and emerging standards by selecting the right supporting frameworks/plug ins and also be working on their integration.

TR3 *XIRUP will allow users to exploit external libraries both to populate the knowledge base with the information about the systems they want to monitor and to create the new modernized system.*

Explanation XIRUP will support both the definition (addition to the knowledge base) of component libraries that will be used while populating the knowledge base with the data about a system we want to modernize, and also while conceiving the modernization steps. For example, the addition of a new component, whose type is known, might cause problems because of the constraints between the type of this new element and the types of the elements already in the system. The use of these definitions is mandatory to be able to reason on the consistency of stored data and enable "smart" impact analyses.

Libraries might also contain already instantiated objects that can be freely added to a given model and thus there are "cloned" to become part of the new specification.

Demonstration This feature will be demonstrated by both the pilot examples. In both cases, we will exploit it while populating the knowledge base, but we will also conceive dedicated (and domain-specific) scenarios to demonstrate its impact while evaluating possible modifications towards the design of the modernized system.

TR4 *XIRUP will support users with different views over the same data stored in the knowledge base to offer different models of the same artefacts.*

Explanation This is a key requirement for letting XIRUP be model-based. The same content in the knowledge base can be rendered in different ways according to the particular needs of the user. If we stick to UML, the same objects can be used in object/class diagrams, but they are also useful for sequence and collaboration diagrams. This capability is enabled by the common knowledge base, but XIRUP will be in charge of allowing the user to set special-purpose filters on stored data. The same information will be rendered by means of different notations (and thus different models), but also with different levels of abstraction (given the user needs and the dimensions of created models).

Demonstration The two pilot modernization scenarios will thoroughly demonstrate this option by exploiting XIRUP guidelines to create special-purpose models from stored data to properly visualize the current system, query the base and display the results, and communicate the results of automatic transformations (mainly for the TID's case study).

TR5 *XIRUP will offer means to trace the dependences among the different elements of the system we want to modernize, the versions of its elements/artefacts, taken decisions and their relationships with the evolution of the system, and the connections between elements and their definitions.*

Explanation The capability of tracing dependences among the artefacts of a modernization (development) process is extremely important to understand the consequences of changes and also to be able to conduct kind of "what-if" simulations to decide the best option.

For example, we can imagine that we want to understand all the elements that may be affected by changing (deleting) an existing element. Similarly, we might be interested in assessing the compatibility between new and existing elements by referring to their definitions.

Demonstration This requirement is intrinsic to any serious modernization effort. The capability of foreseeing what elements are touched by a change, or what components cannot/must be substituted, is mandatory. This is why we think that these features are able to autonomously demonstrate themselves.

In addition, special-purpose scenarios will be designed to stress this feature and to thoroughly support the capability of advance querying the base to assess the feasibility of a given modernization step.

TR6 *When properly instantiated, XIRUP will offer suitable notations to render modernization requirements, to visualize the models the user wants to create, along with the characteristics of the different elements, to show the dependences among the different elements of the knowledge base, to specify the automated transformations the user wants to perform on the base elements, to query the knowledge base and visualize results accordingly, to state deployment directives, and to specify the test cases used to validate the modernized system.*

Explanation This is a direct consequence of what presented so far. All the different concepts and activities must be visualized accordingly. XIRUP will not provide predefined notations, but it will be able to host many different (and domain-specific) notations to let the user render his/her concepts by means of familiar tools.

Demonstration Proposed notations will be mainly taken from the UML and UML-like set of notations. Notice that this is only a choice to ease the demonstration of component, but it is not a limitation imposed by or embedded in XIRUP.

7 Further Requirements

The additional requirements presented in this section comes (directly) from deliverable D1.2 [6] and are presented here to provide a complete description of what XIRUP will offer, to further instantiate some concepts already presented, and also to stress the key contributions of our pilot modernization exercises.

The modelling methodology must sustain the engineer in modernising a complex system.

The goal is to develop a modelling technique that supports engineers during the modernization of technological systems.

The methodology must support as much automation as feasible.

As many tasks as possible should be automated to reduce complexity and support users.

There should be a methodology guidelines document that explains how to perform modernization step by step.

To facilitate learning and application of XIRUP methodology, engineers need a clear description of the methodology. This can be specified formally, for instance by using SPEM, and informally, with guidelines, manuals, and tutorials.

The development process should allow the flexibility to iterate back to earlier steps to adapt to newly discovered problems.

This is another requirement for an agile methodology as XIRUP intends to be. The user must be able to iterate through modernization requirements and steps. The user may have one initial, and sometimes vague, idea of his/her needs, but during the modernization process can discover new needs.

The methodology should provide support for model creation and analysis.

The main artefacts of the methodology are models and transformations. Therefore, the methodology should provide support for model creation, management, and analysis.

The methodology should support the specification of existing data models, xWare infrastructures, and processes.

To be able to understand and manage existing systems, and plan for the refactoring and migration of data models, xWare infrastructures and processes, the methodology should provide a language to specify them and capture relevant information.

The methodology should support the ability to work with different viewpoints and at different levels of abstraction, in order to manage complexity.

Developers can gain understanding of the system by being able to zoom in and out on system parts and components. This requires sophisticated reverse engineering tools that can start from code to draw models, so that these models can be further abstracted into simplified views of the system. There may be several views (perspectives) for the same element (or set of elements), and with different levels of details.

The representation of models must be as simple as possible, though clearly represent the architecture and behaviour of a complex system.

A model is a document that describes the architecture and the behaviour of a technological system. Depending on the focus, a model may have several abstraction layers, until a certain depth. The goal of a model may be to provide a human understandable information container for engineers, in which case particular attention will be paid to a clear and simple representation. Sometimes the goal of a model may be to provide a formal specification in order to automate some tasks, in which case the emphasis lays on an exhaustive formal representation. These two goals are not always compatible with each other.

To enhance the abstraction layers, elements of the model must be groupable according to the functionality they reach together, enhancing several abstraction layers.

For a better understanding, one can group several elements according to the functionality reached by their cooperation. This composite structure can be called a component, and the reached functionality is the technological process associated with it. In the end, a complex system can be represented in an understandable by means of some high-level components.

Notations for models should be user friendly, and, preferably, comply with existing and well-known standards.

To be effective and useful, notations must be user friendly. When aligned with standards, this will facilitate its adoption by engineers and facilitate their support by tools.

The methodology should support the reuse of use cases and test scenarios of the existing system to be applied or adapted in the modernized system.

The modernized system should be able to pass all the test scenarios that the existing system is able to pass. Also, the knowledge on use cases for the existing system should be reused for defining better the requirements for the modernized system.

The use of UML and OCL should enhance the representation of element and dependency checks.

The idea is to translate these concepts into a UML Profile, and to benefit from standard UML techniques for the other aspects, especially behaviour modelling.

Another important aspect is that the object constraint language (OCL) gives us a powerful tool to express dependencies. The use of OCL combined with a UML-based representation of the architecture should permit an automated dependency check of our model during the re-engineering process.

8 Final Considerations

This document summarizes a conceivable amount of work aimed at defining the first set of requirements for XIRUP. The work wanted to envisage the widest context for the methodology and provide it with a general-enough frame, which goes beyond the actual needs of the two pilot applications. To this end we decided to follow a breadth-first approach. In this phase, we wanted to highlight as many meaningful requirements as possible.

While conceiving the actual modernization methodology, we will work on refining these requirements, prioritising them, eliciting their mutual dependences, and maybe also adding new ones to cover unforeseen holes.

9 References

[1] Kent Beck, "Extreme Programming Explained: embrace change", Addison-Wesley Professional, 2000.

[2] Jaelson Castro. Manuel Kolp. John Mylopoulos, "A Requirements-Driven Development Methodology". In the 13th International Conference on Advanced Information Systems Engineering CAiSE 01. 2001.

[3] Alistair Cockburn, "Writing Effective Use Cases", Addison-Wesley Longman, 2000.

[4] Erich Gamma and Richard Helm and Ralph Johnson and John Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", Addison Wesley, 1994.

[5] Michael Jackson, "Problem Frames : Analysing and structuring software development problems", ACM Press, 2001.

[6] MOMOCS consortium, deliverable 1.2, "Methodology Requirement Analysis Report: industrial solution and Telecommunications Cases", January 2006.

[7] MOMOCS consortium, deliverable 2.1, "XIRUP Supporting Tools Requirements", January 2006

[8] OMG, "Architecture-driven Modernization (ADM)", **http://adm.omg.org/**, January 2006.

[9] OMG, "Knowledge Discovery Meta-model (KDM) Specification", version 1.0, **http://www.omg.org/cgi-bin/doc?ptc/2006-06-07**, June 2006.

[10] OMG, "Meta-Object Facility (MOF) Specification", version 2.0, **http://www.omg.org/cgi-bin/doc?ptc/2006-01-01**, January 2006.

[11] Peter Schuh, "Integrating Agile Development In The Real World", Charles River Media, 2005.