13 Model-based Requirements Engineering Framework for Systems Lifecycle Support

A. Soffer, D. Dori

Abstract: The recent migration from traditional sequential development process models to the more modern iterative and evolutionary process models has brought about an evolution in the scope of the requirements engineering process, along with new challenges of managing the requirements knowledge. In parallel, conceptual modeling throughout the development process has been receiving growing attention and wide acceptance.
Working under the premise that effective requirements knowledge management is a key factor in developing quality software that meets customer needs, the main contribution introduced in this chapter is creation and study of a new Requirements Engineering and Management (REM) framework that is tightly coupled with the evolving conceptual model of the developed system. The integration of the proposed REM process into an Object-Process Methodology (OPM)-based systems development and modeling environment is demonstrated via a case study, followed by an evaluation.
The work presented in this chapter shows that coupling the requirements knowledge management activities with the development methodology and a tool-supported modeling environment creates a comprehensive approach for the production of high quality software.

13.1 Introduction

Close examination of the typical outcome of software-intensive systems development projects introduces the challenge of narrowing the gap between the required system and the resulting implementation. In spite of efforts to improve the flow of engineering information throughout the development process, often times the implemented system does not fully match the required one, nor does it meet the customer's needs and expectations [14]. In this setting, an effective requirements engineering and management process should be used throughout the development life cycle to handle new and evolving requirements. System requirements are expected to reflect stakeholder needs by describing the system's externally-perceived functionality as well as certain properties at the desired granularity. Alt-

hough the field of requirements engineering provides means for managing this need based on establishing and maintaining traces between the requirements and their implementation, the effectiveness of these means remains a significant factor in quality and productivity of the software development process.

Requirements Engineering (RE) is the branch of software engineering that focuses on the processes of handling and management of requirements in any software development effort [34]. Regardless of the applied life-cycle development model or the software engineering methodology, RE is a critical component of any comprehensive engineering process employed for systems development or maintenance. Numerous conceivable scenarios (e.g., the typical process of change management) demonstrate that proper management of the requirements knowledge plays a pivotal role in system evolution, as it is embedded within the core of the system's development process throughout its life cycle and may take place at any point during this process.

The main contribution introduced in this chapter is creating, analyzing, and evaluating a life-cycle Requirements Engineering and Management (REM) environment that is tightly coupled with the conceptual model of the evolving system. The proposed RE process is integrated into the Object-Process Methodology (OPM) system development process. This new REM environment supports requirements documentation, tracing, testing, and management, and enables reasoning about requirements during the entire system's lifespan. Analysis and evaluation of this approach and the developed process automation show that coupling the requirements knowledge management activities with the software development methodology and a tool-supported modeling environment creates a comprehensive approach for the production of high quality software.

To present this comprehensive framework, the necessary concepts are explored in this chapter as follows: In Section 13.2 some requirements engineering challenges are presented as motivation for this work. In Section 13.3, the concept of life-cycle REM process is introduced, along with the value of integrating it with the model-driven approach to systems development. Section 13.4 includes an introduction to OPM and argues for the choice of OPCAT (Object-Process CASE Tool) as the supporting environment for REM implementation. Focusing on implementation of these ideas, Section 13.5 describes an extension to OPM's capability which enables linking textual requirements to the system's models. It then demonstrates how the requirements engineering activities are integrated into the OPM-based conceptual system model which evolves throughout the system its life cycle. An evaluation of this work is also presented in this section. Section 13.6 concludes the chapter with some observations on the utility of this mechanism.

13.2 Background: Requirements engineering challenges

Requirements should be complete, consistent, comprehensible, unambiguous, well-documented, traceable, and testable [34]. Proper communication of requirements knowledge to all the software developers in the project and across their organization helps to ensure that the requirements, as well as changes in them, are handled correctly throughout the project life cycle, and that all stakeholders maintain a shared vision [32], [6], [25]. As requirements evolve during the development or maintenance phases, it becomes necessary to modify and manage the system specification and design. Although the field of requirements engineering provides means for managing this need based on establishing and maintaining traces between the requirements and their implementation, this need remains a significant challenge in productivity of the software process.

The term "software product quality" is used as a qualifier of the extent to which the implementation satisfies stated and implied needs. The quality of a software product is directly linked to the characteristics of the process that was used to create it [32], [6]. Consequently, enhancing the development process and methodology is expected to result in higher product quality, which would manifest itself in a better match between the envisioned system, as expressed in the requirements specification, and its realization at the end of the development process.

Requirements Management (RM) is the activity concerned with the effective control of information related to system requirements, in particular the preservation of the integrity of that information with respect to changes in the system and its environment [34]. The recent trend of migration from traditional sequential software development models to evolutionary process models has brought about the evolution of the scope of requirements management. In traditional development approaches, requirements were formulated and documented (often by the acquirer without the involvement of the developer) prior to any development activities. This set of textually-expressed requirements was frozen prior to its validation for the entire duration of the system development effort. Often, the requirements list is a contractually binding document, serving as a "technical appendix" to the legal engagement between the system acquirer and provider, so deviating from it poses problems of various kinds. This may still be the case in formal, very large scale projects, where organizations such as governments are involved. In most modern systems R&D environments, however, where RE is at the heart of the development process, such requirements freezing is inconceivable. Focused on change management, the evolving requirements set serves as the glue that ties the other engineering processes together over time, ensuring that the resulting system or software-intensive product satisfies the needs and expectations of the users, the customers, and the beneficiaries.

A functional requirement is a specific need or desired behavior as seen by an external user of the system. The required capability or function must be delivered by a system through one or more of its components. Achieving high-level stake-

holders' satisfaction strongly depends on maintaining the system's life cycle conceptual integrity, i.e., faithfully reflecting the stakeholders' views and needs throughout the system life cycle. Model-driven development (MDD) [26], which has received growing attention and wide acceptance in recent years, is an adequate response to this need. High-quality requirements should give information on what a software-intensive system is and what it does, rather than how to implement the system [20]. This means that the requirements ("What") need to be linked to the implementation model ("How"), in order to facilitate the adequate connection between them.

The requirements phase is a critical part of software development, yet difficult to enhance [28]. Brooks attests on the difficulty of establishing the requirements and their importance throughout the entire development process: "the hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work so cripples the resulting system if done wrong" [5].

Since requirements engineering in general and the traceability function in particular are quality-enabling techniques [16], improvements in RE capability are essential for developing higher quality software systems. A major expectation of RE is the quality (i.e. effectiveness, efficiency) of the requirements traceability, which is aimed at improving error detection and correction, leading to a better environment for the production of high quality software [31]. Typically, systems are tested to verify the satisfaction of all of the requirements that were deemed feasible. This way, quality is guaranteed with respect to compliance with the requirements.

In summary, the objectives of an integrated requirements engineering and management process include the continuous management of new and changed requirements in a way that ensures competitiveness in the marketplace. Successful implementation of RE depends primarily on (1) controlling requirements and changes in them, (2) managing requirement attributes for better decision making, (3) establishing and maintaining traceability throughout the lifespan of the project, and (4) ensuring the availability of flexible frameworks for the necessary communication channels among stakeholders. The integration of requirements engineering processes into the model-driven development paradigm (MDD) has motivated this work as a promising approach for supporting these needs.

13.3 Model-Based Requirements Engineering and Management

Combining Requirements Engineering processes and artifacts with model-driven development approach may offer effective solutions to some of the critical difficulties in software development. In this section we first review some basic terms relevant to this approach, and challenges they present. Then we describe the integration of RE into the system model-based development and discuss how it

improves many factors that affect the system's quality, notably change management.

13.3.1 Life cycle requirements management

The objectives of requirements management activities include collecting, documenting and organizing the requirements, linking requirements to software items, tracing requirements to all development artifacts, and tracking and communicating this information to all stakeholders. This is necessary to ensure that the basic requirements and their evolution are properly handled throughout the project life cycle. Some process models such as the SEI's CMMI [6] associate some of the requirements-related activities also to other process areas, such as product engineering, project management, and configuration management.

The requirements document is the basic source for communication among customers, end-users, system designers and implementers of the software. It also serves as a validation device for the stated requirements. In many circumstances it is used as the basis for user's manuals or other documents. Another important role that this document plays is serving as the basis for project planning and project management activities.

13.3.2 Model-based development

Model-driven development (MDD) technologies [21] have been recognized by the software engineering community as potential vehicles to alleviate the cognitive effort required for creating a software system that matches its requirements [4], [17]. In this context, a model is a domain-specific abstraction which defines in appropriate terms the structure and behavior of the modeled concepts. According to the MDD paradigm, the requirements model (specification), which is problem domain oriented, evolves into another representation of the contemplated system in a new and different context—the implementation, which belongs to the solution domain. In the solution domain, the approach to implementing the required system is described by means of system architecture, design, and code [3].

Requirements-modeling is the process of constructing abstract, formal representation of the initial textually described system requirements in a way that is amenable to unambiguous interpretation. This process ends with a requirements model, which is expected to capture as much of the relevant real world semantics as possible. Based on the need to clearly express and communicate the engineering knowledge (particularly – pertaining to requirements), the conceptual model of the system has become an essential and effective means of communication among

all stakeholders of the system's development process. Not less importantly, the conceptual model is used for the system validation and documentation needs.

13.3.3 Model-based requirements process

Creation of the requirements model is the first stage in model-driven development. The model aims to capture the real-world semantics pertaining to the system [33]. The requirements representation and documentation serve interests of the different system's stakeholders: customers, developers, and managers [24]. The requirements model focuses primarily on functional requirements, i.e., specific business needs or behaviors as seen by an external user of the system. The required capability or function must be delivered by some subset of the system, so the functional requirements set captured in this model is a basis for subsequent system development and later on possibly also for its maintenance activities.

In the subsequent phases of the development process, the requirements model is elaborated and transformed into the design model. This transition emphasizes the critical need for creating a formal, accurate, and complete requirement model from the outset, as it designed serve as the foundation of the entire software developing process.

Modeling is targeted at clear and accurate representation of the concepts that comprise the system. An important benefit of requirements modeling is that since the resulting model is available at an early stage in the system's life cycle, model analysis and simulation may be used to validate the requirements and reduce conceptual design errors. Later on, as requirements-modeling is integrated into the flow of activities in the development process; it fosters collaboration between requirements acquirers, system engineers, and developers.

A domain-model is an abstraction which defines generically the structure and behavior of the problem domain. Two types of concepts are involved in life cycle requirements modeling: (1) concepts related to customers' objectives, which represent the problem domain and are emphasized in the requirements model, and (2) engineering concepts, which are emphasized in the design model(s). A conceptual gap, which is often very wide, exists between these two model types, since one faces the acquirer with her problem domain, while the other faces the solution domain and the technological solution provider. Inevitably, the problems created due to this dissonance cast dark shadows over the remainder of the systems development process. To attenuate the adverse impact of switching from requirements to design, it is desirable to minimize the additional shake caused by the need not just to switch the attention from the problem to the solution, but to also switch between a modeling approach or diagram type used in the requirements engineering phase to the one(s) used in the design phase. In this regard, if one uses UML, for example, the problem of switching from use case diagrams, applied at the re-

quirements phase, to object-class diagrams and the rest of the UML diagram types, exacerbates the difficulties involved in this transition.

Our goal is therefore to adopt a single modeling approach, language, and methodology, which can be used across this chasm and help bridge it. Such a language must have the traits of simplicity, clarity, and dynamic aspects that use cases possess, combined with formality, the object-orientation of object-class diagrams. In addition, it should have the state transition aspects of state charts, and the workflow and time sequence of activity, collaboration, and time sequence diagrams in a single modeling framework.

The use of domain knowledge can significantly reduce the requirements engineering effort and the amount and severity of errors [33]. Using domain ontologies that include abstractions of typical structure and behavior as modeling patterns facilitates reuse and error reduction. Here too, the ability to apply the same modeling language and diagram type for domain knowledge modeling across the gap between requirements engineering and design can significantly reduce the cognitive load that impedes comprehension of the two sides of this canyon.

13.3.4 Integrating the requirements process

The requirements engineering processes constitute a fundamental life-cycle chain of engineering activities, which start early on with the establishment of positive relationships among success-critical stakeholders, and continue throughout the lifespan of the system. Modeling the requirements and integrating the requirements model into the system's life cycle enables extending the benefits of a solid REM process from the early stage to all the remaining stages of the development process, as depicted in Fig. 13.1.

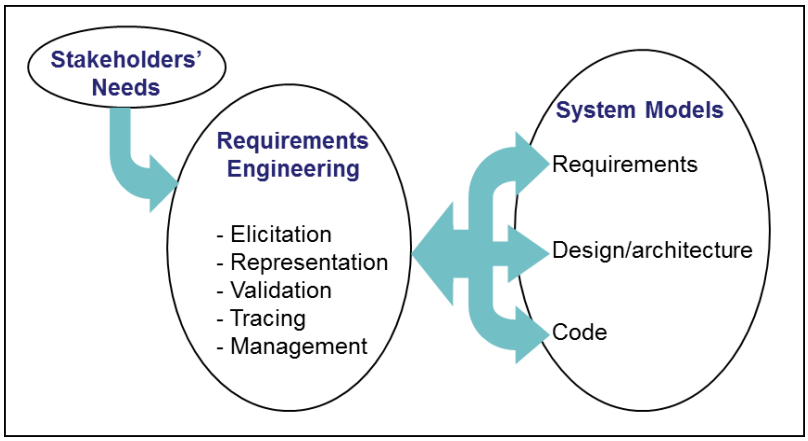

Fig. 13.1 Model-integrated requirements engineering

A natural way of connecting requirements to a model is to represent the RE information and activities according to the adopted product and process models [23]. System model integration adds value to the knowledge captured in the RE process by enabling validation and better understanding of the causal relations underlying traces.

A good requirements specification is one in which requirements are arranged hierarchically. Few high-level, broadly defined requirements are specified in increasing levels of detail, where each level contains a set of requirements that elaborate upon one or more requirements at the level above it. Current commercial tools indeed support such structure for requirements management. A hierarchical structure of requirements may also facilitate the process of their modeling. In general, high level requirements correspond to abstract processes and aggregate objects or agents (actors) and interactions between them at lower levels.

How are requirements integrated into a model? Each requirement is a specification of an individual system function. Thus, a model component designed to meet this requirement is created if it does not yet exist in the model, and it is linked to other model elements to express its relation to the rest of the model. This component should also be associated with the corresponding requirement (or requirements set) with which it is related, creating the traceability information. This traceability information is embedded into the model and is built on the fly during the modeling process, eliminating the need for an external tracing mechanism, such as a traceability matrix. This approach is becoming particularly valuable as traditional phase-oriented (waterfall-type) system development processes are being replaced by concurrent and iterative approaches, such as the spiral process and RUP [18]. In such non-linear development processes it is difficult to maintain accurate and reliable traceability information over time due to their concurrent and repetitive nature.

Model-integrated REM helps bring down the communication barriers between the various stakeholders involved, including system engineers, analysts, developers, and customers. Such barriers, misunderstandings, and potential contradictions typically exist in requirements that are based on text documents. Linking the requirements to the system model facilitates forward and backward traceability to and from the model (possibly including its test plan and model) to the requirements set and vice versa. This two-way traceability enables impact assessment of a proposed change in a requirement using the design model.

The Requirements Engineering and Management (REM) process relates to the information captured in the specification and implementation models through requirements traceability checks, validation, error-detection and analysis, and change management. Since the required capability or function must be delivered by the system to be, the information captured in the implementation model is used later for driving system development or maintenance activities.

A complete and effective model-integrated REM process includes the following steps:

1. Elicitation, acquisition, and text-based documentation of domain-dependent knowledge that pertains to the requirements from the system to be developed,
2. Modeling: formal, semantics-conveying graphic representation of the contents of the text-based requirements documentation elicited in (1), and
3. Mapping of the text-based requirements description to the conceptual requirements-level system model (the problem domain model), to the system's architectural model (the solution domain model), and to the system implementation in software and/or hardware.

13.3.5 Model-integrated requirements traceability

Requirements traceability (RT) is the ability to describe and follow information about the life of a requirement in both forward and backward directions [15]. RT is aimed at improving error detection and correction, leading to a better environment for the production of high quality software [31]. A major activity in the REM process, RT helps stakeholders understand the relationships between software artifacts created during the software development life cycle. RT is fundamental to change management, as it enables one to assess the impact of a change in some part of the system on the rest of the system by analyzing dependencies between the associated requirements and the system design and implementation.

The scope of RT is evolving as the migration from traditional sequential software development models to evolutionary development approach is taking place. Consequently, a number of issues should be considered: when should the trace be created, what is the detail level (granularity) of traces, how to relate traces to the hierarchical structure of requirements and to designed system artifacts, and how to report traces.

13.3.5.1 Requirements traceability challenges

Requirements traceability (RT) is a facilitator of system quality, as it provides means to keep track of the relationships between stakeholders' requirements and artifacts produced during the software development life cycle. Despite being introduced as a mandatory activity in development standards [6], RT is highlighted as an area in need of improvement [33], [30].

Management of change is an essential part in the development and maintenance of systems in general and software-intensive systems in particular. RT is fundamental to the management of change, and its scope is evolving due to the migration from traditional sequential development models to evolutionary ones. This transition raises a number of issues, including the timing of creating each trace, the granularity (detail level) of a trace, and how to relate a trace to the hierarchical

structure of the requirements on one hand and to the system conceptual model on the other hand.

Studies that examined the nature of the relations between requirements, artifacts, and people [15] propose that a more detailed semantic relation than statements such as "X contributed to Y" is needed. Research based on empirical investigations with practitioners and extensive surveys of techniques and tools for requirements engineering [15], [28] suggests that documenting and preserving information generated during early stages of a project is of crucial importance, This is so because this information is potentially relevant to later stages of the system under development, but it cannot be reconstructed at those stages.

A significant practical problem underlying RT is the difficulty to maintain the trace information up-to-date as development artifacts change over time [13]. The challenge here is to provide a low-effort method to continuously update the traceability picture. Performing the REM activities efficiently and effectively cannot be achieved without adequate tool support for automating tedious and error-prone REM tasks. The desired properties of such tools and methods of using them are discussed next.

13.3.6 Tool support for model-based requirements engineering

Years of attempts to achieve progress in RE have focused primarily on establishing procedures and developing practice-oriented methodologies [2] and standards, such as CMMI [6]. However, the high expectations for quality gains cannot be completely met by using methodologies and tools that address the needs of the requirements engineering phase alone. These specialized tools focus mainly on traceability between requirements and code, i.e., relating sections of the systems code to the relevant requirements [15], [19]. However, the amount of knowledge captured by means of traceability can be used in many more ways to improve the gamut of software engineering activities, including requirements validation, change management, complexity and cost measurement and estimation, and test management. Unfortunately, most RE tools fall short of exploiting this full potential of the knowledge represented by traceability, since their methods of operation do not reflect the complexity of the software development scenarios and process [26].

Requirements are the first artifact in the development process, while code is the last in line, the most obscure and least accessible to most stakeholders. To achieve effective traceability, we need to fill the gap between requirements and code. Relating requirements to pre-code artifacts, such as conceptual model elements, test plans, and test outcomes is far more effective than connecting requirements directly to the code.

While moving from high to lower levels of abstraction in system modeling, the complexity of the associated traceability structure grows. The challenge then is to

maintain the accuracy and effectiveness of the trace information. Model-based REM is therefore most suitable for RT, as it can exploit the potential for automation through proper tools [1]. Many automation solutions employ an array of tools to support a variety of functions, creating the challenge of consolidation and unification of these tools.

The proposed approach suggests embedding the requirements engineering and management process as an integral part of the system engineering process based on Object-Process Methodology (OPM) through the use of OPCAT—Object-Process CASE Tool [10]. Spanning across the entire system life cycle and supported by automation, REM, including RT, is thus fully integrated into the modeling and development methodology.

13.4 The Choice of OPM

This section includes an introduction to Object-Process Methodology (OPM), and analysis of principles to establish the theoretical foundations for considering OPM as a basis for implementing the required solution.

An effective requirements engineering process that yields a successful system and high-level stakeholders' satisfaction should be supported by a model that clearly and accurately expresses the system's requirements. The requirements model is a critical artifact in the development life-cycle, since it serves as the basis for the system's architecture, detailed design, testing, and change management. Following the premise that effective modeling of complex systems should combine structure and behavior representations in the same model [8], OPCAT is a comprehensive system modeling, engineering, and life-cycle support environment, based on Object-Process Methodology (OPM) [9], [27].

OPM integrates the object-oriented and process-oriented paradigms into a single frame of reference. OPM's core entities are objects, representing things that exist, and processes, which are things that transform objects. A process transforms an object by creating it, consuming it, or changing its state. Processes are connected to the involved objects through procedural links, which describe the function and behavior of a system. In addition, a set of structural relations provides for modeling the system's structure. The premise of the OPM paradigm is that objects and processes are two types of equally significant entities. Contrary to the object-oriented approach, processes in OPM are not necessarily properties of objects or operations owned by objects, but can rather stand alone. A set of interrelated Object-Process Diagrams (OPDs) constitutes the graphical OPM formalism. The same set of symbols is used in all the Object-Process Diagrams, which is the only diagram type. This way, both the static (structure) and dynamic (function, behavior) aspects that each system exhibits coexist as integrated and equivalent components in the same OPM model.

One of the most progressive ideas of OPM is the approach that promotes integration of the various system life-cycle perspectives into a unified continuous model that may reflect the complete system's life cycle. The OPM philosophy is that the same system model that started in the requirements stage as an analysis model of the requirements is continuously developed, evolved, and refined to represent the system all the way to implementation and deployment.

13.4.1 Bi-modal representation

An OPM model is jointly expressed by two semantically equivalent modalities, one graphic and the other textual. A set of inter-related Object-Process Diagrams (OPDs) constitute the graphical, visual OPM formalism. The graphical OPD set has a textual counterpart modality named Object-Process Language (OPL). OPL is a dual-purpose language, oriented towards humans as well as machines. Catering to human needs, OPL is designed as a constrained subset of English, which serves domain experts and system architects engaged in analyzing and designing a system. Every OPD construct is expressed by a semantically equivalent OPL sentence or phrase. Designed also for machine interpretation through a well-defined set of production rules, OPL has an XML-based notation that provides a solid basis for further automated processing, such as information extraction or code generation.

13.4.2 Complexity Management

OPM mitigates the inherent complexity of model-based system development by supporting gradual refinement/abstraction of information and smooth traversal of a hierarchy of diagrams. This capability is enabled by a multi-dimensional complexity management mechanism. The three built-in refinement/abstraction mechanisms are: (1) unfolding/ folding, which is used for refining/abstracting the structural hierarchy of a thing and is applied by default to objects; (2) in-zooming/out-zooming, which exposes/hides the inner details of a thing within its frame and is applied primarily to processes; and (3) state expressing/suppressing, which exposes/hides the states of an object. Flexible combinations of these three mechanisms enable consistent system modeling at different levels of detail without losing the overall comprehension of the representation.

The unified, bimodal, and flexible representation of a system increases the user's comprehension and processing capability, which is further enhanced through automation by CASE tool support.

13.4.3 OPM Tool Support

OPM facilitates clear and concise graphical formalism along with semantically equivalent and formally structured text. Both representations are automatically kept consistent at all times by OPCAT – the OPM CASE Tool. The unified, bimodal representation of an OPM system makes use of the same methodology-supported language, which could be universal or domain-specific, throughout the system life cycle. In addition, OPM facilitates continuous life cycle modeling, thus life cycle requirements engineering can be naturally and easily achieved through OPM-based system development.

OPCAT can provide a number of functions and features to support various activities in the RE process, such as domain modeling, requirements modeling and analysis, and traceability. Based on unified modeling, combining OPCAT's modeling capabilities and the RE process needs will bring better results for system development challenges such as effective alignment of stakeholders' requirements with system evolution, and enhanced traceability and process measurements (complexity, cost, and quality).

13.4.4 A comparison to UML-based systems

The ability to trace the requirements throughout the entire life cycle is a critical part of the requirements management process [22]. Many tools enable linking requirements to a variety of life cycle artifacts, such as source code files, change requests, test case descriptions, or project tasks. However, the significant advantage of model-driven development is the ability to link requirements to the system model for various needs, such as verification and certification, impact analysis, or test case design. Tools designed for UML environments provide the capability to link requirements to use cases and to activity diagrams. However they do not support fully-integrated model traceability, since none of them have provided the ability to trace a requirement to the internals of the design model, notably to a specific class. Furthermore, since a UML representation of a particular system includes a collection of models, an attempt to achieve full requirements traceability in UML will require a very complex association mechanism in order to link the requirements to the various components included in the representation.

In contrast, the unique ability of OPCAT/REM, described below, to trace requirements to elements of the design model is highly valuable. OPM facilitates clear and concise formalism. Furthermore, OPM enables integration of the various system perspectives into a single unified model that may reflect the complete system's life cycle, based on its complexity management mechanisms that enable continuous refinement of the model [16]. Thus, a highly effective integration of

requirement engineering capabilities into a modeling environment would be with OPM.

13.5 The OPM Requirements Engineering and Management framework

Having established the value of model-integrated requirement engineering and management, as well as the main concepts of OPM, we now review and demonstrate the implementation of the proposed approach to connecting these two technologies. The *OPCAT Requirements Engineering Module* (REM) has been developed as an extension of OPCAT, the OPM-based system modeling environment. This integrated requirements engineering and management environment is tightly coupled with the model of the evolving system. It supports requirements modeling, documentation, analyzing, validating, tracing, and management during the entire system's life cycle, including its development and maintenance phases.

OPCAT/REM supports the following necessary requirements engineering activities:

1. Requirements acquisition and modeling: obtaining and maintaining a text-based requirements database.
2. Linking requirements to the system model: producing and maintaining the requirements traceability information by linking requirements and model elements, and
3. Reasoning about requirements: analyzing and validating requirements traceability and reporting deficiencies, including omissions, conflicts, and ambiguity.

Application of the OPM-based technical solution for integrated requirements management is described in detail next, using a few usage scenarios which represent common engineering tasks during the course of the development cycle. Typically, the OPM unified system model is built incrementally, including the detailed connection between the requirements database and the system specification model.

13.5.1 Requirements acquisition and modeling

The system's requirements database, which includes textual information concerning each requirements such as ID, description, level (hierarchy), originator, status, priority, owner, permissions (to modify), etc., may be created and maintained by a dedicated text-based requirements management tool. These tools (such as DOORS [12], Requisite-Pro [29], or Cradle [7]) typically support exporting the requirements data in CSV (Comma-Separated Value) format, which

OPCAT/REM is designed to accept. Alternatively, the requirements table may be created manually by the requirements engineer via a designated user interface as plain text in CSV format. At this point the requirements database is acquired and incorporated in the OPCAT modeling environment. This step sets the foundation for further processing, which includes (1) creating the traceability links by mapping requirements to model elements, and (2) analyzing and validating the integrated (text/model) requirements data.

The OPCAT/REM interface is depicted in Fig. 13.2. This example shows the textual requirements in a grid structure, which is part of OPCAT's Configurable Management View Console, which, in this case, is configured for requirements management. The representation allows flexibility in the level of detail, as each hierarchical level can be collapsed or expanded. The basic view shown in this example includes the unique identifier of the requirement (ID), its title, description, and hierarchical level. The Description field enables editing free text description of the requirement. Additional attributes may be included as needed.

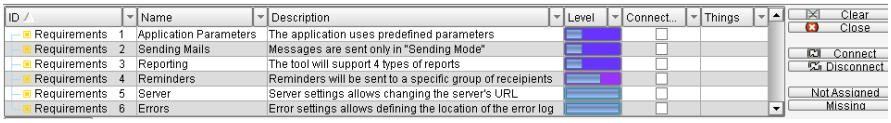

Fig. 13.2 The OPCAT/REM Requirements Database view

Once the requirements database is imported and integrated into the model, OPCAT/REM supports modifications and updates of the requirements set. In order to secure the integrity of the requirements database, which is primarily maintained through an external dedicated tool (e.g., DOORS [12]), all the changes (addition, removal, modification) made to the requirements set through the external tool are reloaded into the OPCAT environment by updating the CSV file. The update is then completed by applying additional processing, mapping, and analyzing, as described next.

13.5.2 Linking requirements to the system model

The novelty of this integrated RE tool is its ability to bind OPM model elements (mainly objects and processes, collectively called "things") from any of the system life cycle phases (e.g. conceptual modeling, requirements specification, architecture, and design) to the related system requirements. The requirements linking process is described below using Fig. 13.3.

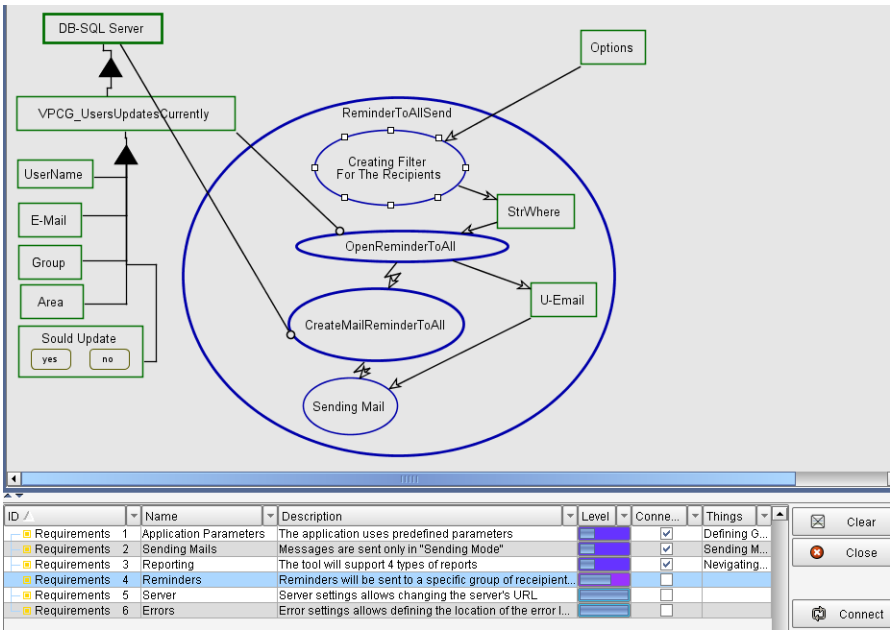

Fig. 13.3 The requirements linking process

Linking a requirement to a corresponding model element is done by first selecting the desired requirement from the available requirements database (e.g., Requirement 4 – highlighted), then the target model element is selected (e.g., the "Creating Filter for the Recipients" process). The link is made by clicking the "connect" button, and indicated by setting a check mark in the "connected?" field (Fig. 13.3 depicts the OPCAT screen just before making the “connection”). Once a requirement has been linked to one or more model elements, the requirement links become an integral part of the OPM model engine. The requirement links are used by the tracing and validation mechanisms described next.

13.5.3 Reasoning about requirements

Tracing, analyzing, and reasoning are done using the analysis and reporting capability of the tool. Through this capability OPCAT/REM facilitates efficient and accurate access to the relevant model sections based on the requirement of interest, thus providing the ability to trace requirements directly to the system model. A number of viewing, analysis, and validation functions are available, as follows:

13.5.3.1 Thing - Requirement matching

Indicating which requirements are linked to a certain Thing ‘T’ (e.g., “Sending Mails”, as depicted in Fig. 13.3). This information can be obtained by using the thing's name “Sending Mails” (='T') as a reduction filter on the things list in the requirements database view (shown in Fig. 13.2). The result will be a new list that includes just the requirements that are associated with ‘T’ (in the same format as depicted in Fig. 13.3).

13.5.3.1 Requirement - Thing matching

Indicating which model elements (‘Things’) are linked to a certain requirement ‘R’. The "Things" field in the ‘R’ row in the requirements table (shown in Fig. 13.3) includes a list of the things associated with ‘R’ (the Things list is horizontal, partly visible in Fig. 13.3 due to space constraints). Double clicking on the ‘R’ row will bring up a list of all the appearances of these things in the model, including the Object-Process Diagram (OPD) in which each thing appears, as shown in Fig. 13.4. Furthermore, double clicking on the OPD name will bring up that diagram with the relevant thing highlighted in red. For example, the requirement-model association view is depicted in Fig. 13.4. In this example, there are 4 occurrences of the 'Creating Filter for the Recipients' process which are associated with Requirement 4. The desired row is then selected from the list. For example, the first row may be selected, and as a result of double-clicking on the diagram title ("SD2.4 – Summary Report") in that row, the SD2.4 diagram is displayed with the "Creating Filter for the Recipients" process highlighted in red.

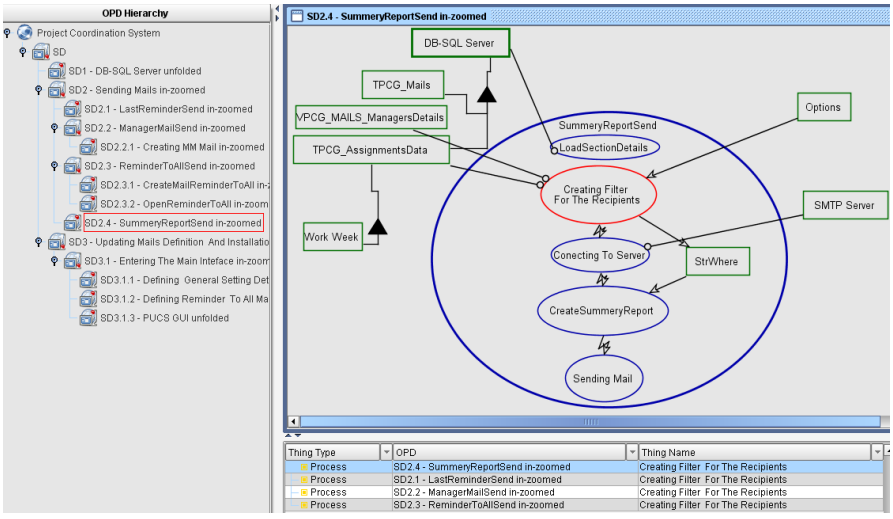

Fig. 13.4 The requirement-model association view

The integrated REM module also enables checking for gaps and inconsistencies in the designed system. For example:

- What requirements in the database are not linked to any element of the model? This information can be obtained by using the "not connected" flag (i.e., the “connected?” check mark is not sown) to apply a reduction filter on the "connected?" list.
- Which elements of the model are not linked to any requirement? This information is obtained by clicking the "Not Assigned" button.
- Where are the trace inconsistencies (broken links)? Trace inconsistencies might appear as a result of a scenario in which a requirement R is linked to a thing T such that if the requirement R is deleted (or even modified), when the requirements database is reloaded, the link to T gets broken. This situation may be detected by clicking the "Missing" button, in response to which a list of all broken links will be displayed.

13.5.4 Evaluation

Evaluation and validation of the results of this work were done based on a modeling case study and an experienced users’ survey.

The modeling case study was based on a commercial Home Management System. The purpose of this exercise was to gain experience with the proposed framework and methodology on a larger scale and with a real-world system, and to evaluate its usefulness in terms of the quality of the resulting model and the OPM-based model-integrated requirements engineering process. Analysis of these factors enables drawing some conclusions on the effectiveness of the OPM-based REM environment, as follows:

The process of creating, binding, and reasoning about requirements is convenient and user friendly. It is easy to navigate through the diagram hierarchy and quickly get to where is needed. The concurrent display of the diagram tree structure, the details of a selected component, and the requirements link table in one view was instrumental in seeing the big picture, while working on the details of one particular part of the system. The automation provided by the OPCAT tool (e.g., finding the connected things) is very helpful in increasing the productivity of the development process.

In order to further evaluate and validate the impact of this development with experienced practitioners, we conducted a survey among several software system engineers who have years of experience developing large-scale systems of different types using various modeling techniques. The objective was to evaluate three main aspects of the underlying problem:

1. Corroborating the significance of the requirements-implementation traceability problem and its impact on the quality of resulting systems.
2. Assessing the effectiveness of the OPM-based requirements engineering and management process as it is integrated in the development life cycle.
3. Evaluating the viability and usefulness of the OPM-based implementation and its potential to be applied in large-scale system development tasks.

Overall, the results of the survey provided useful insights that corroborated most of the assumptions set out to examine:

1. The participants generally agreed with our analysis and conclusion, that the requirements-implementation information gap is a significant problem that impacts the quality of the developed system.
2. The participants were comfortable with the assertion that integrated model-based requirements engineering process helps to obtain better, less disruptive information flow. However, they were slightly less certain that achieving model integration will contribute to the overall quality of the software product.
3. The participants found the system stronger in some aspects compared to others. In particular, the criteria that were ranked high were clarity, level of detail, completeness, and effectiveness. The participants were less convinced about the required learning curve, the ability of the system to scale, and the support for developers' collaboration while using this system.

13.6 Summary and Conclusions

The primary objective of the work presented here was to develop an engineering solution that can reduce the undesired mismatch that often exists between the required system and its implementation. The fundamental approach to achieving this objective is based on the premise that coupling core software engineering concepts with innovative model-driven development methodology enables creation of the desired solution.

While traditional knowledge management techniques focus on creation, organization and management of knowledge based on the properties and attributes of the data, the approach presented here is primarily focused on the utility of the requirements knowledge. The system development processed is enhanced by integration of two engineering process: requirements engineering and model-based development.

After establishing the value of model-integrated requirement engineering and management, and presenting the main concepts of Object-Process Methodology (OPM), the chapter includes a description of the implementation of the proposed approach to connecting these two technologies. OPCAT - the OPM-supporting system modeling tool with its integrated Requirements Engineering capabilities,

supports an requirements engineering process that spans across the entire software lifecycle, including design, maintenance, and evolution. System engineers who use OPCAT can quickly and accurately model and specify requirements, architecture, and any structural and behavioral aspects of the system. Furthermore, they can promptly and correctly establish and obtain the traceability between requirements and their implementation, thereby contributing to improving the system quality. These conclusions were also corroborated by the user survey presented in sub-section 13.5.4.

In summary, following this analysis and findings, this chapter exhibits the following contributions of this work:

- Establishing that coupling the requirements engineering process with a conceptual modeling environment and development methodology facilitates production of high quality software. Using the model-integrated requirements trace information, error detection and analysis, as well as change management can be done more efficiently and correctly, thus extending the value and impact of effective management of requirements knowledge to the entire system life cycle.
- Development of the OPCAT Requirements Engineering Module (REM) as an extension of OPCAT, the OPM-based system modeling environment. This integrated Requirements Engineering and Management environment is tightly coupled with the model of the evolving system, and enables creating, linking, and reasoning about requirements throughout the system's life cycle, including its initiation, development, and maintenance phases.
- Analyzing and evaluating the role of automation and CASE tool support, including knowledge management and reasoning capability, aimed at improving the effectiveness of the requirements engineering process across the software development and maintenance cycle. As has been shown by the example system and by the user survey, the result can indeed solve the system development problem stated above.

While the tip of the iceberg has been touched here with some basic analytical functions that can be performed on the requirements-integrated model, the opportunity for additional analyses is vast. Specifically, an interesting future research direction is to extend this basic capability to enable the system analyst to answer questions such as: to what extent does the requirements model faithfully represent the operational concepts of the system? Does the design fulfill the requirements?

More future work may relate to automation of RE activities which are supported by OPCAT. One possible enhancement is automatically generating an OPM requirements model from text-based requirements specification documentation, a capability that was proven feasible [11]. Other possible enhancements are in the area of requirements trace management: addressing the need to encapsulate the engineering rationale for the link between the requirement and its design within the traceability structure. The challenge is to maintain the effectiveness of the trace information in many development-cycle situations.

References

[1] M. Badar, and D. Zowghi. "Developing a Requirement Toolset: Lessons Learned", Preceedings of the Australian Software Engineering Conference (ASWEC'04), Melbourne, Australia, April, 2004

[2] B. Boehm, and A. Egyed. "Optimizing Software Product Integrity through Life-Cycle Process Integration", *Journal for Computer Standards and Interfaces*, 21, 1999, pp. 63-75

[3] B. Boehm, and D. Port, "Escaping the Software Tar Pit: Model Clashes and How to Avoid Them", Software Engineering Notes, Association for Computing Machinery, pp. 36-48, January 1999

[4] F. Brooks, "No Silver Bullet – Essence and Accident in Software Engineering", *IEEE Computer*, 20 (4), April 1987, pp. 10-19

[5] F. Brooks, *The Mythical Man-Month*, Addison-Wesley, 1975-1995

[6] Capability Maturity Model Integration-SE/SW, Ver. 1.2, Technical Report CMU/SEI-2006-TR-008, 2006

[7] Cradle: requirements management tools. http://www.threesl.com/pages/index.php

[8] D. Dori, "Object-Process Analysis: Maintaining the Balance between Structure and Behavior". Journal of Logic and Computation, 5, 2, 1995, pp. 227-249

[9] D. Dori, *Object-Process Methodology - A Holistic Systems Paradigm*, Springer Verlag, 2002

[10] D. Dori, I. Reinhartz-Berger, and A. Sturm, "OPCAT – A Bimodal Case Tool for Object-Process Based System Development", Proceedings IEEE/ACM 5th International Conference on Enterprise Information Systems (ICEIS 2003), Angers, France, pp. 286-291

[11] D. Dori, N. Korda, A. Soffer, and S. Cohen, "SMART: System Model Acquistion From Requirements Text", *Proceedings: International Conference on Business Process Management*, Potsdam, Germany, June 2004

[12] DOORS - IBM Rational. http://www-01.ibm.com/software/awdtools/doors/

[13] A. Egyed, and P. Grunbacher, "Automating Requirements Traceability: Beyond the Record & Replay Paradigm", The 17th International Conference on Automated Software Engineering. 2002. Edinburgh: IEEE CS.

[14] W. Gibbs "Software's Chronic Crisis", Scientific American, Sep. 1994, P.86

[15] O. Gotel, and A. Finkelstein, "An Analysis of The Requirements Traceability Problem", Proceedings of the First International Conference on Requirements Engineering, Colorado Springs, Colorado, USA April 1994, pp. 94 - 101

[16] O. Gotel, and A. Finkelstein, "Modeling the Contribution Structure Underlying Requirements", Proceedings of the First International Workshop on Requirements Engineering: Foundation of Software Quality (REFSQ '94), Utrecht, The Netherlands, June 1994. 21

[17] D. Harel, "Biting the Silver Bullet: Towards a brighter Future for System Development", *IEEE Computer*, 25 (1), Jan. 1992, p8-20

[18] *IBM, the Rational Unified Process*, http://www-01.ibm.com/software/awdtools/rup/

[19] The International Council on Systems Engineering (INCOSE), "Tools Survey: Requirements Management Tools", http://www.incose.org/tools/tooltax.html, April 1999

[20] M. A. Jackson, "The Role of Architecture in Requirements Engineering", Proceedings, 1st IEEE International conference on Requirements Engineering, Colorado Springs, April 1994, p. 241

[21] A. Kleppe, J. Warmer, and W. Bast. *MDA Explained, the Model Driven Architecture: Practice and Promise*. Addison-Wesley, 2003

[22] G. Kotonya, and I. Sommervile, *Requirements Engineering: Processes and Techniques*, John Wiley & Sons, 1998

[23] L. Lavazza, and G. Valetto, "Enhancing Requirements and Change Management through Process Modeling and Measurement", *ICRE2000 Fourth IEEE International Conference On Requirements Engineering*, June 19-23, 2000, Schaumburg, Illinois

[24] B. Nuseibeh, and S. Easterbrook, “Requirements Engineering: A Roadmap”, Proceedings, Conference on The Future of Software Engineering (ICSE), pages 35-46, Limerick, Ireland, June, 2000
[25] F. Marschall, and M. Schoenmakers, "Towards Model-Based Requirements Engineering for Web-Enabled B2B Applications", 10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'03), 2003, p. 312
[26] OMG Model Driven Architecture, http://www.omg.org/mda/
[27] *OPM – The Official Web Site*, http://www.objectprocess.org
[28] B. Ramesh. "Factors Influencing Requirements Traceability Practice", Communications of the ACM. Vol. 41, No. 2, December 1998
[29] IBM Software – Rational RequisitePro http://www-01.ibm.com/software/awdtools/reqpro/
[30] B. Ramesh, and M. Jarke, “Toward reference models for requirements traceability”, IEEE Transactions on Software Engineering, 27(1), January 2001
[31] B. Ramesh, C. Stubbs, T. Powers, and M. Edwards: "Requirements Traceability: Theory and Practice", annals of software engineering, No. 3 (1997), pp. 397-415
[32] S. Robertson, and J. Robertson. *Mastering the Requirements Process.* Addison-Wesley, UK, 1999
[33] C. Rolland, and N. Prakash. "From Conceptual Modeling to Requirements Engineering". Annals of Software Engineering, 10 (2000), pp. 151-176
[34] I. Sommerville, and P. Sawyer, *Requirements Engineering: A good practice guide*, John Wiley & Sons, 1999