International Journal of Computer and Communication Technology

Volume 7 | Issue 1 Article 1

January 2016

REQUIREMENT VALIDATION VS. REQUIREMENT ELICITATION

DHIRENDRA PANDEY
Department of IT, Babasaheb Bhimrao Ambedkar Universitye, Lucknow- India, prof.dhiren@gmail.com

MOHD. WARIS KHAN
Department of IT, Babasaheb Bhimrao Ambedkar Universitye, Lucknow- India, wariskhan070@gmail.com

VANDANA PANDEY
Department of Computer Science, Dr. C. V. R. University, Bilaspur- India, vandanadubey7@gmail.com

Recommended Citation
PANDEY, DHIRENDRA; KHAN, MOHD. WARIS; and PANDEY, VANDANA (2016) "REQUIREMENT VALIDATION VS. REQUIREMENT ELICITATION," *International Journal of Computer and Communication Technology*: Vol. 7 : Iss. 1 , Article 1.
DOI: 10.47893/IJCCT.2016.1324
Available at: https://www.interscience.in/ijcct/vol7/iss1/1

REQUIREMENT VALIDATION VS. REQUIREMENT ELICITATION

DHIRENDRA PANDEY[1], MOHD. WARIS KHAN[2] & VANDANA PANDEY[3]

[1&2]Department of IT, Babasaheb Bhimrao Ambedkar Universitye, Lucknow- India
[3]Department of Computer Science, Dr. C. V. R. University, Bilaspur- India
prof.dhiren@gmail.com, wariskhan070@gmail.com, vandanadubey7@gmail.com

Abstract: To support the evocation and validation of the goals achieved by the existing system and to illustrate problems of the old system, we suggest to capture current system usage using rich media (e.g., video, speech, pictures, etc.) and to interrelate those observations with the goal definitions .Therefore it becomes necessary to relate the parts of the observations which have caused the definition of a goal or against which a goal was validated with the corresponding goal in every phase of software development process. These interrelations provide the base for: 1) explaining and illustrating a goal model to, e.g., untrained stakeholders and/or new team members, and thereby improving a common understanding of the goal model; 2) detecting, analyzing, and resolving a different interpretation of the observations; 3) comparing different observations using computed goal annotations; and 4) refining or detailing a goal model during later process phase. Using the PRIME implementation framework, we have implemented the PRIME-CREWS environment, which supports the interrelation of conceptual models and captured system usage observations. We report on our experiences with PRIME-CREWS gained in a first experimental case study. The successful implementation of proposed requirement evocation and validation process can have a good impact on the production of quality software product

Keywords: Requirements management, scenario, scenario-based requirements for real world, requirements traceability, requirements evocation, requirements validation

I. INTRODUCTION

Requirements are attributes or something, which we discover before building products. It is a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents [1-(A)] A well-formed system provide all functionality and facility that satisfies customer needs. There exists a reciprocal interrelationship between human beings and machines for requirement gathering that can assist to produce quality products [2-(A)] Requirements are generally classified as functional and non functional [1-(A)]. A functional requirement is a requirement that specifies an action performed by a system without considering physical constraints. Non-functional requirement specifies system properties such as environmental and implementation constraints, performance, platform dependencies, maintainability, extensibility, reliability etc. [[1-(A)] .

In the late seventies and during the ’80s a variety of methods for defining the data (e.g., ER modelling [6]), the behaviour (e.g., State- Charts [13]), or the functions of the system (Structured Analysis [11],) had been proposed. In the late ‘80s and early ’90s, object oriented techniques (e.g., [4]) appeared which proposed an integration of the three views. Today, this stream resulted in the definition of the Unified Modelling Language [15]. As argued in many publications, it is essential to consider the history and functionality of the existing system when defining the requirements for the new system (e.g., McMenamin and Palmer [16]). There are two main reasons for this:

- The new system has to provide to a large degree the functionality and facility of the old system;
- Making the same error twice can be avoided, i.e., one can learn a lot from the success stories and pitfalls of the existing system.

Other authors see present-state analysis as essential foundation for change integration, i.e., incrementally defining the new system based on a conceptualization of parts of the existing system, like in Lundeberg et al.’s ISAC approach [17] or in the RE frameworks proposed by Potts et al. [18] and Jarke and Pohl [19]. Another popular area where the modeling of the present-state serves as a basis for defining the required future state is business process reengineering, e.g., [20]. Thus, there exist two types of conceptual or goal descriptions in a requirements engineering process. The present state model, which (partially) defines the functionality and history of the existing system, and the required-state model, which defines the requirements for the future system. Jackson [14] calls the first type of model indicative properties whereas the second type of model is called optative properties. As depicted in Fig. 2, implementing a new system thus requires four major steps:

- Reverse analysis: Defining a present-state model by abstracting from the reality is required since, in most cases, no conceptual model of the actual system exists or the model is not up to date;
- Change definition: Integrating the change definition into the (partial) present-state model, thereby defining the model for the new system;
- Change implementation: Designing, implementing, testing, and installing the new system based on the required state model;

- Legacy integration: Considering the existing context during the change implementation to empower reuse and to avoid conflicting system implementation

The quality of the new defined (or updated) present-state and required-state models evidently depends on the knowledge elicited from the stakeholders, i.e., it heavily depends on the successful stakeholder involvement in the requirements engineering process. Scenarios are proposed as an ideal means to support the definition of the present-state model and to drive the change definition, i.e., to achieve better stakeholder involvement. In contrast to conceptual models, scenarios describe concrete examples of current and future system usage. As indicated, e.g., in a survey of industrial projects using scenarios [28], the use of scenarios, in addition to conceptual models, improves the quality of the requirements engineering process. A comprehensive comparison of existing scenario-based techniques has been developed in the ESPRIT Reactive Research Project CREWS1 [27]. The most popular form of scenarios are use cases, as proposed by Jacobson et al.'s OOSE [21] and their various extensions, e.g., by Cockburn [7]. Complementary experiences in, e.g., participatory design [3, 5], indicate that better stakeholder involvement can be achieved by using rich media (e.g., video, pictures, screen dumps, speech, etc.) for recording and discussing current system usage. Among others, the use of rich media in requirements engineering processes leads to a better understanding of the usage domain, enforces focused observation of (temporally and/or spatially) distributed aspects, avoids presumptuous abstractions, enables repeatability of results, and late reflections [5].

In this paper, we present an approach which bridges the gap between concrete examples of current system usage (scenarios) at the instance level and the conceptual present state models at the type level. We argue to capture present system usage using rich media and to inter relate those observations with the present-state model. The interrelations between the components of the present- state model and the corresponding parts of the observations provide the basis for:

- explaining and illustrating a conceptual model to, e.g., untrained stakeholders or new team members, and thereby improving a common understanding of the model;
- detecting, analyzing, and resolving different interpretation of the observations;
- comparing different observations using computed annotations based on the interrelations;
- Refining or detailing a conceptual model during later process phases.

II. THE APPROACH

We propose capturing observations of present system usage using rich media. We call a captured observation Real World Scene. The material gathered during an observation may contain information about many system usages. We therefore propose to pre structure this material into what we call a Real World Example (RWE). An RWE is a collection of material that represents one system usage. The material belonging to an RWE should be arranged in a suitable manner, e.g., if the observation was recorded using video, the video should be cut in a way that it shows the temporal sequence of a sample system usage. RWEs are used for two main purposes. On the one hand, new concepts are elicited from the RWEs. On the other hand, the current-state models can be validated against the RWEs. In both cases, we propose to inter relate the parts of the observations with the component of the conceptual present- state model which was elicited from the fragment and/or validated against the fragment (see Fig. 2). The requirements engineer has, therefore, to select the corresponding fragment of the RWE and to indicate the type of the interrelation to be created between the fragment and the component of the present-state model. An interrelated part of a real world example is called Real World Example Fragment (RWEF). The interrelation takes place at a fine-grained level since arbitrary fragments of the recorded observations (e.g., a cut-out video clip or even a part of one picture as extreme) can be linked to any element of a conceptual model—in contrast to relating the whole observation to the present state model. The types of interrelations to be created between the RWEFs and the components of the present-state models depend on the modeling primitives. Note that we do not propose a new conceptual modelling technique, but argue to embed an appropriate existing technique in our overall approach. Likewise, we neither provide any guidelines for capturing real world scenes nor for the pre structuring of real world scenes into real world examples. However, the approach described in the following can be embedded in the guidelines proposed, e.g., by McGraw and Harbinson [26] for capturing and evaluating real world observations. Obviously, our approach is not equally well applicable for any kind of project. It is well suited for projects in which the functionality and facility of the old system could be observed and in which this functionality and facility has to a large degree be provided by the new system (even if the system implementation changes significantly due to, e.g., technological progress) and/or in which observed short comings provide the basis to derive new goals for the new systems. Innovative projects where there is no precursor system will benefit less from our approach.

When interrelating the model components with the RWEFs, a fine-grained formal structure is gradually imposed on the initially completely unstructured observations. In this paper, we focus on four main advantages gained by the fine-grained interrelations:

1) Explanation of the present-state model: Present-state models are typically being built by one (or a group of) stakeholder(s). The abstraction of reality is typically performed in the minds of the

stakeholder, i.e., for a person not involved in the model definition, the resulting model and the abstractions made are hard to understand. The typed interrelations empower to explain the model component by retrieving the related RWEFs, i.e., the abstraction process made during the model definition is (partially) traceable. Training of people joining the project is thereby eased and a better model understanding during the whole system development lifecycle is established.

2) Comparison of different real world examples: Stakeholders may perform the same task in different ways. Similarly, the implementation of the same requirement can differ, i.e., implementing an essential feature can result in different incarnations [27]. Trying to understand the essence behind different incarnations just based on observations is sometimes very difficult. The comparison of different observations can be supported by using a more abstract conceptual description of the observations. The typed interrelations between the RWEFs and the model components support such a comparison.

3) Definition and comparison of multiple viewpoints: The definition of a present-state model based on a real world scene is always subjective, i.e., depends on the perception of the people defining the model. Since the observations are persistently recorded, each stakeholder can define his viewpoint in a conceptual model according to his perception based on the same set of observations. Viewpoint resolution can be supported based on the type of interrelations between the model components and RWEFs. Different conceptualisation of identical and/or overlapping RWEFs and similar conceptualization based on non overlapping RWEFs can be highlighted. In addition, the typed interrelations can be used to support the mediation and negotiation of the stakeholders by grounding discussions on real world observations;

4) Reviewing of conceptual models: Structured review of present-state models leads to higher model quality. The review can make use of the RWEFs to better understand the abstractions made and, thus, to better justify and comment on the model (components) under review. The typed interrelations are a prerequisite for selectively accessing the fragments for a model component.

In early phases of analyzing the existing system, it is important to understand and agree about the why behind certain properties of the system, e.g., “why does the system support this activity?”, before dealing with details about how and what, e.g., the data the system deals with, system function and/or system behavior. Consequently, more and more RE frameworks suggest the explicit definition of goal models prior to the definition of the more common conceptual data, behavior and functional models. Eric Yu has nicely summarized the various applications of goals in requirements engineering [48]. As he indicates, goals are used:

- As central means for requirements elicitation and elaboration, thereby encouraging the stakeholders to ask “why,” “how,” and “how else” questions (e.g.[2-(**B**)], [10]);
- For relating the system to organizational and business context (e.g., [29];
- To clarify requirements and stakeholder objectives without the need to go too much into detail (e.g., [7])
- To deal with conflicts (e.g., [7], [10]). Goals can be used as interconnecting mediation points supporting a focus on common objectives first, before going into the details of resolving the conflicts.

III. PRESENT STATE GOAL MODEL

In this section, we elaborate on the fundamental relationships that can exist between an RWEF and the components of a goal model, and describe a set of method fragments which establish these relationships When developing a goal model based on (a set of) RWEs, elicitation and validation can be distinguished as two conceptually different objectives. The focus of elicitation lies on building up the goal model according to the knowledge gained by analyzing the observations.3 The objective of validation is to collect evidence from the RWEs for the individual goals of an elicited goal model. In a typical analysis session, however, elicitation and validation are often heavily intertwined. Whenever the analyst encounters problems in validating a goal model against the RWEs, this leads to the elicitation of new goals which either have to be attained in addition to existing goals or may represent alternative goals for achieving a super goal. Based on the assumption that the observation material belonging to one RWE has been arranged according to some logical criteria, e.g., to the temporal sequence of events, we recommend analyzing each RWE from the beginning to the end. If the requirements engineer identifies a goal to be achieved in a part of the RWE and this goal is defined in the current goal model, he extracts this part as an RWEF and interrelates it with the goal using a typed dependency link (Section 3.1). If the goal is not yet defined in the hierarchy, he has elicited a new goal which has to be added to the goal model and linked with the RWEF (Section 3.2). To support the explanation of the goal model, we propose characterizing certain RWEFs as reference examples for attaining or failing a goal by and documenting this be defining a specific dependency link between the goal and the RWEF (Section 3.3).

A. Validation of conceptual Goals

Refer 2.3; this phase aims at improving the quality of the requirements. This goal is achieved by performing several analyses, based on the use of formal techniques, which may help to pinpoint flaws that are not trivial to detect in an informal setting. The different analysis that are possible over the formalized requirements are: Logical Consistency to formally verify the absence of logical contradictions in the considered formalized requirement fragments (e.g. the absence of two fragments mandating mutually incompatible behaviors).Scenario compatibility to verify whether a scenario is admitted given the constraints imposed by the considered formalized requirement fragments .Property checking to verify whether an expected property is implied by the considered formalized requirement fragments. The above checks not only produce a yes/no answer, but they can also provide the domain expert with diagnostic information of different forms:

1) TRACES: When consistency and scenario checking succeeds, it is possible to produce a trace witnessing the consistency, i.e. satisfying all the constraints in the considered formalized requirement fragments. Similarly, when a property check fails the tool provides a trace witnessing the violation of the property by the formalized requirement fragments.

2) UNSATISFIABLE CORE: If the specification is inconsistent or the scenario is incompatible, no behaviour can be associated to the considered formalized requirement fragments; in these cases, the tool can also generate diagnostic information in the form of a minimal inconsistent subset. This information can be given to the domain expert, to support the identification and the fix of the flaw.

PROCESS: The validation process consists of three main steps:

M3.1: the user chooses a set of requirements to focus the validation on particular aspects of the specification.

M3.2: the user defines a set of problems, each one consisting of a set of objects and a set of scenarios and properties.

M3.3: The user checks the defined problems and analyzes the results.

The above validation steps can be iterated arbitrarily, by correcting formalized requirement fragments and/or the corresponding categorized requirement fragments if necessary, creating new scenarios, new properties, and by analyzing different aspects of the requirements specification.

TOOL SUPPORT: The definition of the set of requirements of interests (M3.1) is supported by a plug-in that takes care of checking the completeness of the set with regards to the dependencies defined in M1.2. The user may define a problem with a special class whose attributes are considered the objects of the problems, and the attached constraints are considered the scenarios and the properties of the problem. Thank to the links with the formal elements created in M2.2, the selected informal requirement fragments correspond to a set of formalized requirement fragments. With this formal model and a given problem, the tool automatically translates the problem into an equi-satisfiable problem for the model checker: the problem admits a model if and only if the model checker finds a trace. The trace is mapped back to the tool and is visualized to the user. If the problem is unsatisfiable, the user may choose to look for an unsat core. This is presented to the user as a list of formalized requirement fragments that caused the inconsistency.

B. *Elicitation of conceptual Goals*

Requirement elicitation and development phase mainly focuses on examining and gathering desired requirements and objectives for the system from different viewpoints (e.g., customer, users, constraints, system's operating environment, trade, marketing and standard etc.). Requirements elicitation phase begins with identifying stakeholders of the system and collecting raw requirements from various viewpoints. Raw requirements are requirements that have not been analysed and have not yet been written down in a well-formed requirement notation. The elicitation phase aims to collect different viewpoints such as business requirements, customer requirements, user requirements, constraints, security requirements, information requirements, standards etc. Typically, the specification of system requirements starts with observing and interviewing people [12].

C. *Development of Requirements*

Furthermore, user requirements are often misunderstood because the system analyst may misinterpret the user’s needs. In addition to requirements gathering, standards and constraints play an important role in systems development. The development of requirements may be contextual, which is shown in figure 2. It is observed that requirement engineering is a process of collecting requirements from customer and environment in a systematic manner. The system analyst collects raw requirements and then performs detailed analysis and receives feedbacks. Thereafter, these outcomes are compared with the technicality of the system and produce the good and necessary requirements for software development [3].

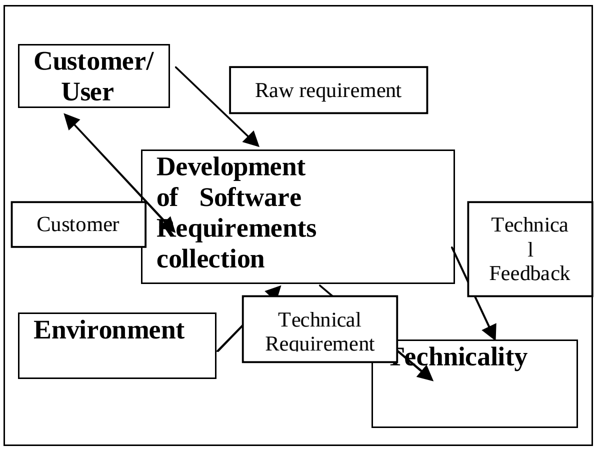

Fig 2: Development of requirements

IV. PROME CREWS MODELLING SETTING

The creation, maintenance, and use of the fine-grained interrelations between RWEFs and the components of the present state model require appropriate tool support. This has been realized in the PRIME-CREWS environment. We first sketch, in Section 4.1, PRIME (PRocess Integrated Modeling Environment), a framework for process integrated modelling environments which underlies PRIME-

CREWS. PRIME offers generic mechanisms for supporting method adaptability as a basis for incorporating method guidance for establishing and applying the interrelations. We then outline, in Section 4.2, the tools supporting the creation and visualization of the fine-grained interrelations and describe the trace repository in which the information is being recorded. In Section 4.3, we briefly describe the implementation of the PRIME-CREWS environment.

A. Method Adaptable Tools Based on the PRIME Framework

For creative modelling processes, the method guidance embodied by a tool environment cannot be fully predefined [31]. This, in particular, holds for the field of scenario-based elicitation and validation, where knowledge about suitable method guidance is just emerging. An important requirement is, therefore, that the method knowledge underlying the tool support, e.g., the sequence of steps necessary for relating an RWEF to a goal with a certain dependency link type is not hard-coded in the tools. Instead, it should be defined outside the tools for enabling an easy adaptation. Moreover, the method guidance offered by the tools clearly depends on the conceptual target models used and must, therefore, be extensible for multiple target formalisms. Process-Cantered Engineering Environments (PCEEs [12]), which define method knowledge in explicit process models, in principle enable such an adaptation. They can be divided into three conceptually distinguishable domains [23].

The modelling domain is comprised of all activities for defining and maintaining process models using a formal language with an underlying operational semantics which enables the mechanical interpretation of the models. The enactment domain encompasses what takes place in a PCEE to support (guide, enforce, control) process performance; this is essentially a mechanical interpretation of the process models by a so-called process engine. The performance domain is defined as the set of actual activities conducted by human agents and nonhuman agents (computer tools). The interactions between these domains characterize the way in which model-based method support is provided. A process model is first instantiated within the modeling domain and passed to the enactment domain, i.e., process parameters like resources and time scheduling are bound to the project-specific values. Based on the interpretation of the instantiated model, the enactment domain supports, controls, and monitors the activities of the performance domain.

The performance domain provides feedback information about the current process status to the enactment domain as a prerequisite for adjusting process model enactment to the actual process performance and enabling branches, backtracks, and loops in the process model enactment. Thus, the fundamental mechanism provided by PCEEs is suitable for process execution based on explicit method definitions. In addition to the coarse-grained project management support provided by PCEEs, fine-grained support is required to enable method-driven developer guidance and trace capture. For example, a fine-grained method fragment guides the developer in performing reviewing activities on goal concepts selected in a goal editor by retrieving related RWEFs displayed in a multimedia tool and displaying information about the annotations back in the goal editor. This would require that different tools (e.g., goal editor and multimedia editor) work systematically together according to the defined method fragment and support the developer in annotating, e.g., the right RWEFs for the reviewed goal We have extended the PCEE approach, resulting in a framework for process-integrated environments called PRIME (see [30] for a detailed description). In contrast to existing PCEEs, method fragments enacted in a PRIME based environment influence the behaviour of the tools, e.g., by restricting the selectable menu items and product parts in a tool according to the current process enactment state. In addition, the process integration provided by PRIME empowers the developer to initiate the enactment of predefined method fragments which, e.g., guide him in relating an RWEF to a corresponding goal of a goal model. Besides elementary method steps, e.g., for creating a goal, we have altogether defined about 40 method fragments for the systematic interrelation of conceptual goal models with RWEs (M3.1), for various explanation situations (Section M3.2), as well as for reviewing activities (Section M3.3) and visualization modes .These method fragments are continuously adapted and augmented as method knowledge increases.

B. Establishing, Managing, and Visualizing Trace Information

As described in Section 4, the key prerequisite for comparing different RWEs, explaining the current-state goal model, comparing multiple stakeholder viewpoints, and reviewing goal model components is to establish fine-grained trace relations between RWEFs and individual goals. These are some important terms which has to be discussed for information system Trace Repository for Fine-grained Product and Dependency Models,(Semi-)Automated and Adaptable Trace Capture.

C. Implementation

The PRIME-CREWS environment has been implemented with C++ using the PRIME implementation framework [30]. It runs on both Solaris and Windows 95/NT. The goal models, the multimedia artefacts, their relations, as well as any other products, are persistently stored in a repository implemented on top of a relational database. Currently, we use the Sybase 11 RDBMS and Microsoft Access, but the data storage layer is implemented to support any database providing an ODBC interface. For the user interface, we used the ILOG Views toolkit, while the multimedia facilities within the whiteboard editor have been realized using Apple’s QuickTime library. 4.1. Tool support. The definition of the set of requirements of interests (M3.1) is supported by a plug-in that takes care of checking the completeness of the set with regards to the dependencies defined in M1.2.

The user may define a problem with a special class whose attributes are considered the objects of the problems, and the attached constraints are considered the scenarios and the properties of the problem. Thank to the links with the formal elements created in M2.2, the selected informal requirement fragments correspond to a set of formalized requirement fragments. With this formal model and a given problem, the tool automatically translates the problem into an equi-satisfiable problem for the model checker: the problem admits a model if and only if the model checker finds a trace. The trace is mapped back to the tool and is visualized to the user. If the problem is unsatisfiable, the user may choose to look for an unsat core. This is presented to the user as a list of formalized requirement fragments that caused the inconsistency.

V. RELATED WORKS

Some user-centered design and ethnography approaches Provide a detailed descriptions for preparing and performing real world observations and for interpreting the observations captured on video and other recording means (e.g., SEP [25], Xerox Parc [5]). Defining such a method, however, was not the focus of our work. We concentrated our efforts on providing support for interrelating recorded observations (RWEs) and conceptual models, especially goal models, and on using these interrelations for explanation, evaluation, and comparison of RWEs and goal models. In Section 4.1, we described how RWEFs related with goals can be used to ground communication and explanation of abstract concepts on instances. Systems providing multimedia management facilities like AMORE [8], Raison d’Etre [9] and approaches supplying paper-based access to captured multimedia [5], [25] are used for explanation and communication of concepts and requirements between stakeholders with various backgrounds and less formal training. Another class of approaches relates textual scenarios and conceptual models. ScenIC from Potts et al. [[1-(B)] ,[2-(B)], and Cockburn’s extended use cases [8] are two approaches that are especially featuring goal models. Whereas, in ScenIC, goals are extended with scenarios for explanation and exploration, Cockburn extends use cases with goals for structuring them. The cited publications only support the explanation of abstract concepts with examples, but not vice versa. For instance, AMORE aggregates multimedia objects representing sources and background information to requirements. For Raison d’Etre, a keywordbased search on video transcripts is implemented for assessing media related to a special topic of interest. All these approaches do not provide explanation and information on the examples themselves. As described in Section 4.1.2, in PRIME-CREWS, the goals behind a real world example can be queried or the example can be analyzed with respect to its quality (positive or negative), problems revealed, etc. Creating conceptual descriptions of how things are performed in the real world in the form of activity lists in SEP [25], storyboards and workflow maps at Xerox Parc [5], or data and control flow diagram with AMORE makes a comparison of real world examples difficult. Stakeholders might perform the same task differently and there might be several completely different ways of reaching a goal. McGraw and Harbinson demand comparison on a higher level, but only define activity-based comparison of observations, which are evaluated using occurrence ratios for the activities. In Section 4.3.1, we presented a possible solution for this demand by comparing RWEs in respect to goals. Evaluating dependency relations between RWEs and goals enabled us to display differences and possible correspondences between two examples, as well as to determine overall relevance and success of goals in the real world. As described in Section 4.2, the abstractions performed by analyzing RWEs are always personal interpretations of the analyst. Different stakeholders may have different interpretations. Therefore, it is necessary to support several stakeholders in eliciting and validating conceptual models as well as reviewing the others interpretations against the RWE and, then, to be able to display differences between them. All approaches cited above provide only one interpretation for real world observations. They are, thus, not considering these problems.

VI. CONCLUSIONS

We have presented an approach for bridging the gap between concrete examples of current system usage (scenarios) and conceptual current-state models. To support the definition of a conceptual goal model of the existing system (current-state model), we have proposed capturing system usage using rich media and to interrelate those observations with the goal definitions. More precisely, we have proposed to relate the RWEFs which have caused the definition of a goal or against which a goal was validated with the corresponding goal. A fine-grained interrelation between the conceptual goal model and the recorded observations is thereby established. Thus, our approach particularly aims at making the abstraction process more transparent and traceable. We have defined basic method fragments for establishing typed traceability relations between the captured RWEs and the goal models. In addition, we have defined method fragments for using the fine-grained interrelations to visualize the evidence of the defined goals, to compare different stakeholders’ viewpoints, and to compare different RWEs. We have implemented our approach in a prototypical environment, PRIME-CREWS. PRIME-CREWS supports the creation and use of the interrelations between the goals and the RWEFs. It offers tools for multimedia management, goal modeling, and for the visualization of the various annotations computed based on the fine-grained interrelations. We have illustrated the main benefits of our approach, using examples drawn from a trial application at ADITEC. According to our experience, interrelating captured observations with conceptual goal models facilitates the definition and agreement of the main features provided by the existing system. Among others, the ADITEC trial application has shown that:

- New team members and untrained stakeholders use the fine-grained interrelation to access RWEFs as explanations for goals defined in the current-state model;
- Comparing different RWEs and comparing goal models defined by different stakeholders is facilitated by the annotated goal trees;
- Review of conceptual models is significantly improved by using the typed interrelations to understand and justify the abstraction process made during the definition of the conceptual models;
- The visualization of the review results in an annotated goal tree which improves the resolution of detected conflicts/short comings.

Our approach of interrelating RWEFs with model components can be adapted to any kind of conceptual target model, i.e., the modelling language used to define the current- state can be manifold. For example, data (e.g., entity relationship models [6] or the static structure diagrams of UML [17]), behaviour (e.g., state charts [17]), and/or functional (e.g., data flow diagrams [28] or business process models [29]) conceptual models can be used as target languages. In contrast to goal languages, such languages are mostly used during later RE phases. The adaptation of our approach to another target language, as our own experience with message trace diagrams and static structure diagrams of UML indicates, requires the definition of new interrelation types to be used to relate the RWEFs with the concepts provided by the target language chosen.

REFERENCES

[1] V. Ambriola and V. Gervasi. On the Systematic Analysis of Natural Language Requirements with CIRCE. Autom. Softw. Eng., 13(1):107–167, 2006.

[2] A.I. Antón, W.M. McCracken, and C. Potts, “Goal Decomposition and Scenario Analysis in Business Process Reengineering,” Proc. Sixth Int’l. Conf. Advanced Information Systems Eng. (CAiSE ’94), pp. 94–104,June 1994.

[3] A. Cimatti, M. Roveri, A. Susi, and S. Tonetta. From Informal Requirements to Property-Driven Formal Validation. In FMICS, volume 5596 of LNCS, pages 166–181, 2008

[4] A.I. Antón, “Goal-Based Requirements Analysis,” Proc. Int’l Conf. Requirements Eng. (ICRE '96), pp. 136–144, Colorado Springs, Colo, 1996.

[5] H. Beyer and K. Holtzblatt, Contextual Design: Defining Customer-Centered Systems. Morgan Kaufmann, 1997.

[6] G. Booch, Object-Oriented Analysis and Design with Applications. B/C Publishing Company, Inc., 1994.

[7] F. Brun-Cottan and P. Wall, “Using Video to Re-present the User,” Comm. ACM, vol. 38, no. 5, pp. 61–71, May 1995.

[8] P.P.S. Chen, “The Entity-Relationship Approach: Towards a Unified View of Data,” ACM Trans. Database Systems, vol. 1, no. 1, pp. 9–36, Mar. 1976.

[9] A. Cockburn, “Structuring Use Cases with Goals,” J. Object-Oriented Programming, vol. 10, no. 7, pp. 35–40, Nov. 1997.

[10] J.M. Carroll et al., “Raison d’Etre: Capturing Design History and Rationale in Multimedia Narratives,” Proc. ACM CHI ’94 Conf., pp. 192–197, Boston, Apr. 1996.

[11] M.G. Christel, D.P. Wood, and S.P. Stevens, “AMORE: The Advanced Multimedia Organizer for Requirements Elicitation,”

[12] A. Dardenne, A. van Lamsweerde, and S. Fickas, “Goal-Directed Requirements Acquisition,” Science of Computer Programming, vol. 20, nos. 1-2, pp. 3–50, Apr. 1993.

[13] T. DeMarco, Structured Analysis and System Specification. New York: Yourdon Press, 1978.

[14] A. Finkelstein et al., “Inconsistency Handling in Multi-Perspective Specifications,” IEEE Trans. Software Eng., vol. 20, no. 8, pp. 569– 578, Aug. 1994

[15] D. Harel, “STATECHARTS: A Visual Formalism for Complex Systems,” Science of Computer Programming, vol. 8, pp. 231–274, 1987.

[16] M. Jackson, Software Requirements & Specifications—A Lexicon of Practice, Principles and Prejudices. Addison-Wesley, 1995.

[17] Rational Software Corporation, Unified Modeling Language. Available on the World Wide Web: http://www.rational.com, Jan.

[18] S.M. McMenamin and J.F. Palmer, Essential System Analysis. Prentice Hall, 1984

[19] M. Lundeberg, G. Goldkuhl, and A. Nilsson, “A Systematic Approach to Information Systems Development (I+II),” Information Systems, vol. 4, nos. 2-3, pp. 1–12, 93–118, 1979.

[20] C. Potts, “Determining Requirements for Evolving Systems,” Tutorial Notes and Slides, CAiSE ’97, Barcelona, Spain, June 1997.

[21] M. Jarke and K. Pohl, “Establishing Visions in Context: Towards a Model of Requirements Processes,” Proc. 14th Int’l. Conf. Information Systems, pp. 23–34, Orlando, Fla., Dec. 1993

[22] A.W. Scheer, Architektur Integrierter Infomationssysteme [Architecture of Integrated Information Systems]. Springer, 1990 (in German).

[23] I. Jacobson et al., Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, 1992.

[24] M. Jarke and K. Pohl, “Requirements Engineering in 2001: (Virtually) Managing a Changing Reality,” Software Eng. J., Nov. 1994. Eng. (RE ’95), pp. 194–203, York, England, 1995.

[25] K. Pohl, Process Centered Requirements Engineering. England: J. Wiley & Sons Ltd., 1996.

[26] K. McGraw and K. Harbison, User-Centered Requirements: The Scenario-Based Engineering Process. Mahwah, N.J.: Lawrence ErlbaumAssoc., 1997.

[27] S.M. McMenamin and J.F. Palmer, Essential System Analysis. Prentice Hall, 1984.

[28] C. Rolland et al., “A Proposal for a Scenario Classification

[29] Framework,” Requirements Eng. J., vol. 3, no. 1, pp. 23–47, 1998.

[30] K. Weidenhaupt, K. Pohl, M. Jarke, and P. Haumer, “ScenarioUsage in System Development: A Report on Current Practice,” IEEE Software, vol. 15, no. 2, pp. 34–45, Mar. 1998.

[31] [28]_ E. Yu, “Modeling Strategic Relationships for Process Reengineering,” PhD thesis, Technical Report on Research in Data and Knowledge Eng., DKBS-TR-94-6, Dept. of Computer Science, Univ. of Toronto, 1994.

[32] [29]_ K. Pohl, Process Centered Requirements Engineering. England:

[33] J. Wiley & Sons Ltd., 1996.

◈◈◈