Bridging the Gap Between Past and Future in RE: A Scenario-Based Approach

Peter Haumer*, Patrick Heymans**, Matthias Jarke*, and Klaus Pohl*

(*) Lehrstuhl Informatik V, RWTH Aachen
Ahornstraße 55, 52056 Aachen, Germany
{haumer,jarke,pohl}@informatik.rwth-aachen.de

(**): Institut d'Informatique, University of Namur
Rue Grandgagnage 21, B-5000 Namur, Belgium
phe@info.fundp.ac.be

Abstract

Requirements Engineering (RE) investigates the impact of a future-oriented change vision, but the move towards this vision must consider a context heavily shaped by the past. As RE becomes a continuous process throughout the system lifecycle, it must achieve an effective combination of envisionment and traceability. In this paper, we describe a scenario-based solution to this problem which is based on an integration of five ingredients: (a) the persistent capture of context in the form of real world scenes captured in multimedia; (b) formal agent-oriented modelling with a semantics that allows distributed interactive animation; (c) message trace diagrams as a medium for exchanging animation test cases and traces; (d) a goal model to control and record the RE process; and (e) a process-integrated tool environment to ensure method-guidance and traceability with as little effort as possible. In addition to the basics of our approach, we also describe its prototypical implementation in the CREWS-EVE environment and demonstrate its usefulness with examples from a case study in the production industry.

1 Introduction

At the turn of the millennium, continuous change has become about the only constant in organisations and systems. RE should take centre stage as a facilitator but continuous facilitation is not easy to achieve. One of the biggest problems in RE is the difficulty stakeholders and requirements engineers face to mitigate their bias towards the present context. There are two aspects of this problem, one pointing towards the future, the other towards the past:

The envisionment problem: It is well-known that people have great difficulties to envision the impact of a proposed system in a future that is vastly different from the present; there are specific biases when dealing with uncertainties and risks [19] as well as an inability to play through the implications of a number of interacting features, especially where multiple agents are involved or multiple changes are happening simultaneously [20].

The traceability problem: Equally often we find the problem that organisations cannot remember how and why certain decisions were made [5], [12], [25], [30].

Taken together, both problems make coherent change management in RE close to impossible. Moreover, both problems often interact. Unfortunately, most previous work has tackled them separately rather than jointly. To address the *envisionment problem*, research and practice have experimented with tools for making concepts understandable (scenarios, animations, virtual reality) as well as with tools for making group interaction productive (brainstorming variants, making conflicts productive, etc.). Both techniques can also be combined, enabling cooperative usage of scenario techniques. Indeed, our recent survey [37] showed that practitioners levy little credence on scenario-based approaches where there is no early feedback from rapid prototyping on how realistic models and scenarios are.

One of the difficult problems in scenario management is the choice of an adequate representation. Text-based representations are very popular because they focus on use and change, and are thus often crisp, easy to maintain and manage. As they are also rough, they tend to suspend commitment and are thus quite suitable for difficult discussion with a lot of rapid change. However, text scenarios also have their limitations. First, there can be settings where pictures or sound are much more informative than textual descriptions, e.g. in many technical applications; the use of multimedia scenarios, mock-ups, and the like is well known especially in human-computer interaction. Second, the interplay of components in distributed systems is hard to capture in a small number of scenarios – and too many, or too complex scenarios destroy the advantages of short textual scenarios. Finally, any model, but also any scenario, is an abstraction of reality, which may or may not turn out to be adequate to its task. In the short term, this is not a problem and indeed one of the big advantages of scenarios. However, once the creation context of the scenario is lost, it becomes no longer criticisable; therefore, richer capture of reality scenes in multimedia, which allows scenario or model revision even after relatively long times, has become popular in documenting requirements meetings [3] and is increasingly proposed for current-state scenarios themselves [15], [22], [38]. This last point also concerns the *traceability problem*. Requirements traceability, initially mandated by procurement agencies for purposes of compliance verification, has long proven its worth as a critical tool for project documentation, change management, and as a source for organisational knowledge creation. However, with respect to our goal of continuous change management, two major problems remain. Firstly, as [12] point out, traceability does not go sufficiently backwards towards the sources of requirements, i.e. the reality from which requirements emerged and the different viewpoints on this reality. This leads to the need for documenting, in particular, the goals of stakeholders, and to link them to the RE activities. Secondly, empirical studies [12], [31] show that

traceability only works if the capture mechanisms are flexible enough to satisfy project-specific needs [7] as well as group privacy interests [11]. Yet, they must be automatic enough so that traces are produced as a side-product of normal RE work rather than creating extra effort. Summarising, our key argument is that solutions for the envisionment problem and for the traceability problem must be *combined* to achieve an effective solution to the problems of continuous requirements change. Even within a single RE process, such an integration could significantly enhance the cycle between requirements elicitation based on an as-is analysis and the definition and validation of future-system requirements models.

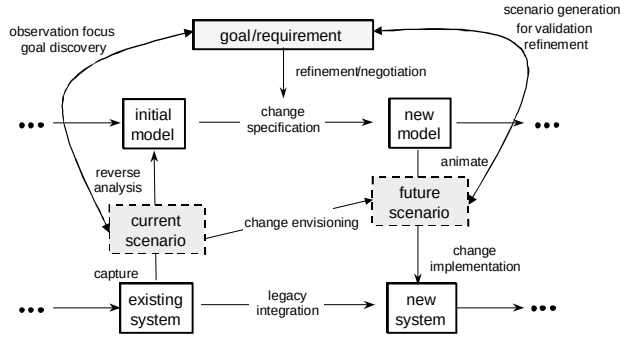

Fig. 1: Scenarios, conceptual models, and goals / requirements in change management.

In the European CREWS[1] project, we have developed a solution architecture for this kind of integrated support. This architectural framework, shown in Fig. 1, places current-state and future-state scenarios as well as goal/requirements hierarchies in the context of model-based change management, following the cycle (a) reverse analysis; (b) change definition; (c) change implementation with (d) context integration. The framework is not meant to imply a specific RE process, but enables many combinations of method chunks associated with the various links in the figure.

In this paper, we present methodology, implementation, and preliminary evaluation of one particular realisation of this framework, which we call CREWS-EVE (CREWS Elicitation and Validation Environment). CREWS-EVE addresses exactly the kind of applications where text-based scenarios alone are insufficient, and therefore places great emphasis on goal-driven multimedia capture of current reality and animation of both current- and future-state models. It thus complements the text-based CREWS-SAVRE [36]. The remainder of this paper comprises two main sections. In Sect. 2, we describe in more detail how the different components in Fig. 1 are instantiated in CREWS-EVE, both from the viewpoint of the method and from the viewpoint of tool implementation. While the main benefit is expected in long-term management of continuous change, an obvious first step for evaluating CREWS-EVE concerns the support it can offer within a single change cycle. We have pursued this initial evaluation through an industrial case study with ADITEC, an engineering firm in Aachen. These experiences have led to the specification of a number of reusable "method chunks" which we describe in Sect. 3, together with example applications and benefits obtained. Finally, Sect. 4 presents our conclusions and outlines ongoing further work.

[1] CREWS stands for Cooperative Requirements Engineering With Scenarios, ESPRIT 21.903

2 CREWS-EVE Components

Fig. 2 shows how the CREWS-EVE environment instantiates the framework in Fig. 1 by specific modelling formalisms and tools. Roughly speaking, we distinguish two components which are detailed in Sect. 2.1 and 2.2 and whose technical integration is outlined in Sect. 2.3.

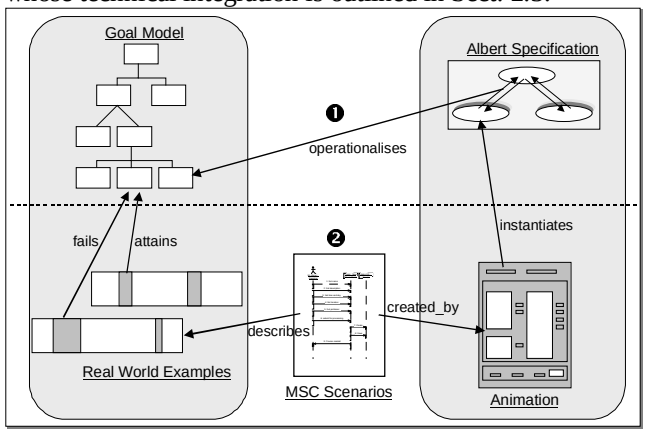

Fig. 2: Conceptual integration within CREWS-EVE

The first component (depicted on the left of Fig. 2) is an elicitation environment which supports the goal-driven analysis of real world examples captured in multimedia. It consists of a *specialised multimedia editor* and a *dependency management tool* for supporting negotiation about and traceability to real-world examples captured with multimedia (mostly for current-state scenarios) as well as a hierarchical *goals and requirements documentation tool*, as proposed in decision theory [29] and RE [39], [1], [6] and partially supported by RE tools such as Requisite Pro.

The second component (right of Fig. 2) performs formal modelling as well as scenario generation and validation within Fig. 1 CREWS-EVE offers the *agent-oriented conceptual modelling language ALBERT-II* which combines a simple way for stakeholders to describe distributed systems with a temporally oriented formal semantics that facilitates automated reasoning. Moreover, it offers the *interactive distributed animation* of current and future-state scenarios based on model interpretation.

These two components were initially developed separately and evaluated individually [15], [17]. Their concepts are integrated on the model level (marked by ❶ in Fig. 2) as well as on the instance level (marked by ❷). On the model level, we enrich the process of creating a formal Albert specification with the option to interrelate parts of the specification with goal concepts which have caused their creation or which they operationalise. On the instance level, integration is reached by expressing behaviour using the same formalism: Message Sequence Charts (MSCs) [18], [32]. This conceptual integration must be

supported by tool integration in order to keep the effort of data transfer, method guidance, and traceability low while enabling flexible learning of method chunks. This is achieved by embedding CREWS-EVE in the process-integrated modelling environment PRIME.

2.1 Elicitation and Validation of Conceptual Models Based on Real World Scenes

The quality of conceptual models, especially of current-state models, heavily depends on successful stakeholder involvement in the RE process. This can be supported by the use of rich media (e.g. video, pictures, screen dumps, speech etc.) to record and discuss current system usage, e.g. in participatory design [2], [3], [22].

Our approach [15] bridges the gap between concrete examples of current system usage at the instance level, and the conceptual current-state models at the type level. It relates the parts of the observations which have caused the definition of a concept of the current-state model or against which a concept was validated, with the corresponding concepts. Thus, it makes the abstraction process to the current-state models more transparent and traceable by

- 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, analysing 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.

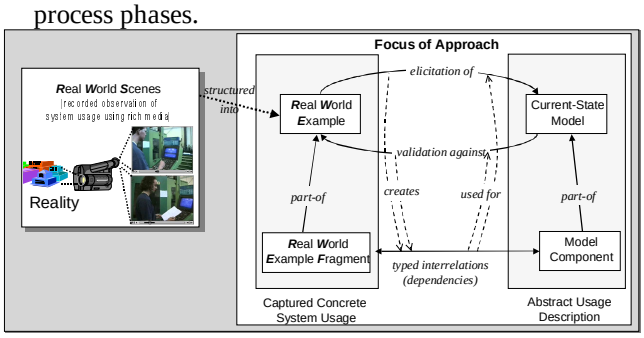

Fig. 3: Schematic overview of elicitation and validation of current-state models based on real world scenes.

Fig. 3 illustrates the approach. Current system usage is recorded on-site using rich media in what we call *Real World Scenes (RWS)*. The material gathered during an observation may contain information about many system uses. We therefore pre-structure RWS into a *Real World Example (RWE)*. An RWE is a collection of material that represents *one* coherent and complete observed system usage. The material belonging to an RWE should be arranged in a suitable manner. For example, video used to record the observation should be cut to reflect the temporal sequence of a sample system usage. RWEs have two main purposes: New model components (concepts) can be elicited from RWEs, or existing current-state models can be validated against them. In both cases, we interrelate exactly the parts of the RWE (resulting in RWEFs: Real World Example Fragments, e.g. a cut-out scene of the video-RWE) which have caused the elicitation or which have been used for validation respectively to the corresponding model component.

The interrelations result in an extended form of requirements pre-traceability [12]: (a) they trace back to concrete instance example from the real world; (b) trace on a very fine-grained manner allowing interrelations of arbitrary parts of conceptual models with arbitrary parts of real world scenes rather than just on a document level. This enables fast and selective access to the relevant parts of RWEs, documenting how model components have been elicited or validated and avoids time-consuming viewing of irrelevant information.

In the early phases of RE it is more important to understand and agree on the *why* behind certain properties of the system (i.e., "Why does the system support this activity?"), before dealing with details about *what* and *how* [6], [39], e.g., the data the system deals with, system function and/or system behaviour. In addition, examples of system usage can represent different incarnations of one task fulfilling one specific goal. This makes it hard to compare several examples at a low level of abstraction, as on the system interaction model or data model level. Concentrating on goals facilitates the detection of commonalties between different observations. If more detailed knowledge about the achievement of a goal is required (*what* and *how*), a behavioural, functional, data model can be created. The fine-grained interrelation between RWEFs and conceptual goal models is realised using specially typed dependency interrelations. The type expresses information about the analyst's interpretation of the relationship between RWEF and the goal concept. The two most important link types are Attains and Fails. Attains expresses that the analyst interprets the RWEF as an example of how the related goal is fulfilled. Fails is the direct opposite: it denotes that the RWEF shows how the goal is failed (cf. [15] for a more detailed discussion on link types).

The design of the approach enables the integration of a wide range of existing techniques for goal analysis and for capturing and pre-structuring real world scenes into real world examples such as [22].

2.2 Cooperative Animation of Formal Specifications

Albert II [9] is a formal agent-oriented language designed to express functional properties (including real-time properties) of composite systems [10]. Agents can perform *actions* and have *state components* denoting either physical or informational characteristics. An Albert II specification also includes constraints that restrict the agents' admissible behaviours. In order to provide more guidance to the analyst, these constraints are written by instantiating predefined *constraint patterns* (precondition, effect of action, action composition, invariant, visibility, etc.). Specifications are edited using the Albert II editor which provides a graphical editing environment, syntax checking facilities

and a client-side interface for invoking more elaborate reference and type checks.

An Albert II specification is a class-level description: it describes abstract properties of classes of components of a system. Because of its abstractness, it is usually difficult for stakeholders to get a detailed understanding by simply reading it (or a paraphrase), even extremely carefully. This situation is reinforced when dealing with complex/critical systems. Having a way for the stakeholders to experience behaviours defined by the specification is therefore a crucial issue in validation [13], [14], [23], [35].

The Albert II animator [16], [17] (see Fig. 4) is a distributed application that helps stakeholders construct and review instance-level behaviours step-by-step in a cooperative way. Each participant is in charge of *agent instances* that he manipulates through a client application and that interact with other agent instances. Client applications allow to perform action occurrences and observe their results in terms of state components values.

The animation server (which supports the coordinator user) conducts an animation by centralising the various requests issued by the stakeholders, and builds a new global state of the system (made of all the local instances' states). The resulting state may be inadmissible with respect to the specification. This results in a dead-end branch which cannot be further investigated. The other possible origin of a dead-end branch is when some stakeholder decides to stop building the current state transition either because he was prevented to perform some event or because some events he performed in previous stages now have unforeseen consequences. Any time an anomaly is discovered, an error message is displayed in the respective client application. In order to facilitate revisions, traceability within the animator allows such error messages to refer directly to the violated statements. It is also possible for a user to attach to any component of a state or state transition, built or being built, a comment, question or a change request. This additional information is recorded and added to the (textual) animation traces, which can then be a basis for discussion and model improvements.

Technically, the animator uses an interpretation algorithm. This algorithm operationalises the semantic rules which map the language's constructs to formula schemas of first-order real-time temporal logic. This way of giving a semantics to the language produces, for different instantiations of a given pattern, logical statements with similar structures. As a consequence, dealing with full real-time temporal logic could be avoided. The resulting algorithm consists checks and triggers that take place at well defined moments in the animation process. Computations are therefore very efficient (even if the language is very expressive) and traceability to specification statements is straightforward. More technical details can be found in [16], [17].

An additional advantage of animation is that, since operations are carried out by stakeholders over intangible objects (the representations of the composite system's components inside the animator tool), some restrictions about the kind and the amount of the tested operations disappear. For example, testing how operations are carried out in a manufacturing environment might be very costly in the real-world if real parts have to be produced. Another advantage of animation comes from the settings in which an animation session usually takes places (distributed but in the same room) and is that informal communication between stakeholders is possible. This is often not possible when trying out a real system where operations are performed in different places. Finally, over prototyping, animation has the advantage that it avoids to create an implementation and that it keeps therefore validation at the requirements level (since no design or implementation concerns interfere).

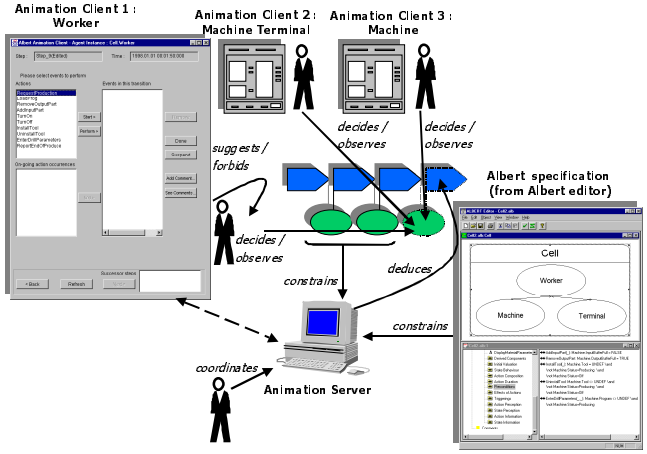

Fig. 4: Schematic overview of the animation approach.

2.3 The PRIME Environment as a Foundation for the Technical Integration

Integrating the prototypes presented in the previous subsections and supporting the definition of reusable method chunks describing their usage requires an implementation platform which allows the integration of existing tools and supports the dynamic definition and adaptability of the integrated method chunks as well as a common repository for data integration.

PRIME is an object-oriented framework for *PR*ocess *I*ntegrated *M*odelling *E*nvironments [27] which defines method and tool knowledge in explicit models. PRIME's process integration empowers requirements engineers to initiate the enactment of predefined method chunks guiding them in their activities, even across tool boundaries.

The general PRIME framework is divided into three distinguishable conceptual domains. The interactions between these domains characterise the way in which model-based method guidance is provided. Process chunks are first instantiated within the *modelling domain*. Based on the interpretation of the instantiated model, the *enactment domain* supports, controls, and monitors the activities of the *performance domain* by sending and receiving predefined message types between the domains. Thus, the performance domain has to provide 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, back-

tracks, and loops in the process model enactment. In order to be fully process-integrated, the tools of the performance domain must conform to the generic PRIME tool architecture [27]. The elicitation tools (the goal editor, the multimedia management tool, the dependency management tool) have been built this way. In contrast, the Albert II editor and the animation tool did not have the process-integration features required. We thus used techniques described in [27] and [4] to wrap the existing tools into our PRIME-based environment. We implement wrappers which empowered the animation and ALBERT tools to send and receive service calls from the enactment domain. Moreover, the wrapper provides the functionality for storing and retrieving data using the common RDBMS based repository for data integration.

3 Evaluation of the Integrated Approach

Our first evaluation effort for CREWS-EVE focused on the short-term interaction within a single project we performed at ADITEC, a manufacturing company located in Ashen, Germany. ADITEC produces machine gears for various kinds of industrial devices. The overall aim of this trial application, in which we were supported by one of the ADITEC managers (Franz) and the foreman (Fred), was to extend the integrated production management and reporting system. It consisted among other components of a central manufacturing resource planing system and a production scheduling system, which used an order dispatch system and individual machine terminals mounted at each production cell for the retrieval of production orders and report on production times.

In the following, we outline three reusable *method chunks* developed during the case study. The notion of method chunk expresses modular reusable sets of activities and emphasises that the value of an environment such as CREWS-EVE may increase over time by gradually adding method knowledge gained through experiences. We do not expect that it will be possible to predefine methodical process guidance for the whole RE process [26]. The method chunks described below focus on the added advantages gained by combining the techniques described in Sect. 2.1 and Sect. 2.2, as their individual benefits have already been discussed elsewhere.

3.1 Validation and Elicitation of Goal Abstractions and Operationalisations using Animation

Problem. Before writing an Albert specification, it is desirable to create a goal model that delimits the aspects of the system that need to be described formally, and helps elicit the information necessary to write the formal operational specification[2]. This activity is normally performed by an expert analyst who is able to write the formal specification. His source of information is at a very abstract level without the influence of users or domain experts. On the other hand, when using the elicitation technique described in Sect. 2.1 to involve stakeholders, communicate is limited to the captured scenes of the existing system.

Solution. The animator simulates usage situation of either the existing or the desired system. Interrelating goals with their operationalisations in the specification enables stakeholders to discover that occurrences of some actions provide evidence for or against the assumed attainment of a given goal by some elements of the Albert II specification. Moreover, unforeseen side-effects of the operationalisation can be discovered, that is, (positive or negative) influence on other goals. Therefore, we also interrelate action occurrences taking place during animation with goals, thus treating animation traces in the same way as RWEs in Sect. 2.1. Guidance for the different situations mentioned above is described in [15].

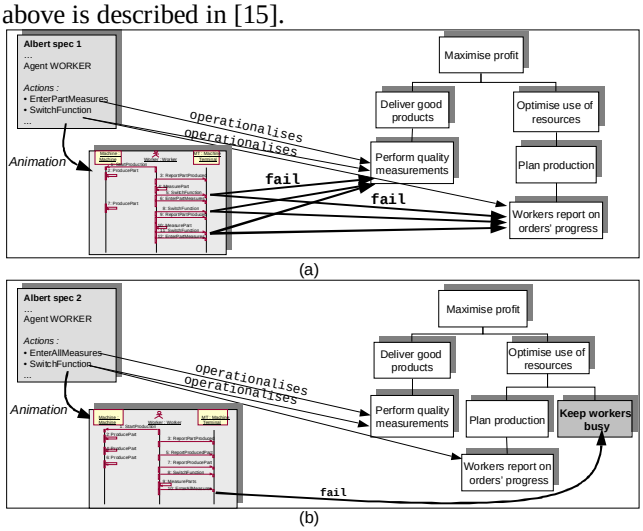

Fig. 5: Validation of goal operationalisations, an example.

Example. The ADITEC re-engineering team was asked to extend the existing production management system with the ability to handle quality measurements of the produced parts. As a consequence, the team introduced a new goal "Perform quality measurements" which was operationalised in a way that the workers have to measure gear parts as they are produced. The formal description of machine terminal functionality was modified so that it allows to enter measurement data and records them in a database. Then, an animation session took place in which Fred, the foreman, played the role of the *worker* agent instance. As the animation was progressing, Fred noticed that the actual worker might find it annoying to repeatedly switch functions on the terminal (report on the order's progression and enter the quality measurements) and would be likely to skip through these tasks. To mention this risk, the instances of action *SwitchFunction* were related to the goals "Perform quality measurements" and "Workers report on orders' progress" using `Fail` links (see Fig. 5(a)). An alternative operationalisation was then specified and animated: all the quality measurements are done after the entire order is finished. But, in this case, Fred objects that the worker remains inactive during production while he could do valuable work. To record this side-effect, he related the in-

[2] How goal operationalisation actually takes place is not a topic of the paper. For this, we rely on other research and, in particular, on [6] and [8].

stance of *EnterAllMeasures* with a `Fail` link to a new goal "Keep workers busy" he adds to the model (see Fig. 5(b)).
Benefits. With this integrated method chunk we provide a uniform treatment of current- and future-state scenarios, represented either by real world scenes and animations respectively, for validation of and interrelation with goal models. Animation can be used for the validation of changes, but also for the current-state in places where video capture is to cost and time intensive. Additional benefits of the new method chunk are that (a) it provides a more systematic way of recording animation results by directly relating the animations to the goal model; (b) the created links can be reused to support negotiation and choice between alternative operationalisations; and (c) animated behaviours can be retrieved to serve as examples for abstractly formulated goals.

3.2 Validating the Albert II Specification by Reconstructing Real World Scenarios

Problem. Animation can also be used to check if a specification of a new system adequately and correctly represents critical parts of the current application domain, e.g., with respect to scenarios elicited from real world examples. In the same way, one wants to check if retained system procedures can still be performed correctly after having specified changes on an existing system.
Solution. The integrated CREWS-EVE environment provides a direct access to MSC scenarios elicited from RWSs or additional elicitation techniques. These MSCs can be replayed, by trying to match sets of interactions on sets of Albert II actions and events. Success and failure of these matches are recorded and presented to the user. The environment supports the user in stepping through the MSC by choosing between alternatives and highlights current sets of interactions for which he will try to find a correspondence in the animation.
Example. Before extending the specification with quality measures, the ADITEC reengineering team performed an animation of the current procedures. The scenarios elicited from video taped worker interactions are replayed using our technique. After the animation, the stakeholders impressions, the created animation traces and the recorded success and failure of matches are used to discuss and decide on the accuracy of the specification to represent the current system. After an agreement has been reached on this issue and the specification modified accordingly, the specification can be modified and animated to evaluate the impact on the system.
Benefits. Being able to explicitly check if specific scenarios of the existing system are matched by the specification allows to decide if the Albert II specification appropriately represents the current system. It is also essential for checking if animated potential system evolutions still guarantee basic features.

3.3 Explanation Support for Animation Steps

Problem. While performing the animation, client users controlling their agents do not only have to understand the actual meaning of state components and executable actions, they need to get a broad idea of what will happen and why if they decide for a certain alternative which will determine the subsequent state transitions of the animation. To make a reasonable decision the user needs to be provided with more information than the animator normally offers, because, e.g., he might not have enough domain knowledge or the names in the specification might be badly chosen.

Solution. The integration provides additional information for actions and state components on different levels of abstraction, which makes them much more transparent for the user of the animation and support him in various ways in his decision making. Fig. 6 and Fig. 7 sketch this explanation support. The integration makes the full traceability support provided by PRIME available for the Albert specification and animation tools. A set of integrated method chunks semi-automatically establishes typed interrelations between objects of different types. For instance, the `operationalises` link in Fig. 6 indicates that an agent interaction has been elicited from a goal. Alternatively, properties of an agent's state component have been directly elicited from a RWE, indicates by a `based_on` interrelation, etc.

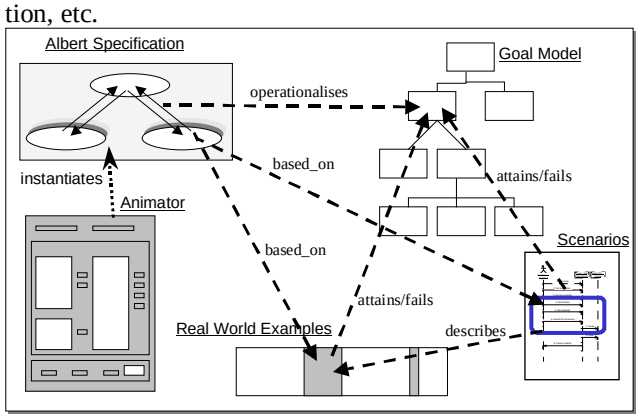

Fig. 6: Interrelating Albert II specifications with goals, scenarios and real word examples.

The interrelations are applied for information retrieval during runtime of the animation for Albert II specification objects (e.g., agent, action and state component) which have been instantiated by the animator. Before animation of a specification, the animator now evaluates for each presented object if interrelations to goals, scenarios, and RWEFs exist. When this is the case, special operations allowing to display the explanation information become available, in form of a dependency graph (see upper right windows of Fig. 7). The user can browse through this structure, expand the graph by displaying indirectly related objects, and can select goals, scenarios and/or RWEFs he wants to be presented in their native editing tool.
Benefits. The different types of information provided represent different abstraction levels, which are needed to support the understanding from the overall system goals supported by an action down to how the performance will look like concretely in a broad context. The benefits gained for each information type are as follows:

Goal supported by specification objects. The direct or indirect interrelation to abstract goal concepts provides information about the purpose of an action within the animated system part. It also facilitates the user to realise similarities and dependencies between several actions with respect to their intentions to attain certain system goals, i.e. comparing the role of different actions within the system.

Concrete scenarios representations. MSC scenarios provide an overall view on the action's temporal contexts, i.e. possible actions and interactions which might have happened before the current situation and possible ways the animation can go on by selecting one of the presented actions.

Interrelated RWEFs. Concrete video examples, especially when they originating from the user's domain, often make it easier to understand a concept than abstract names and descriptions. Interrelations to RWEFs provide a direct access to possibly several different recorded situations where the presented actions were performed and state components have been observed on the real system. The user can review these recordings in full context, e.g. he can see what happens before and after the action, which helps him to direct the subsequent animation. Further, the combination of related goals, RWEFs and the typed interrelations allows him to review attainment as well as failure examples for goals that the respective actions are tackling. This feature lets the user determine in which direction he wants to proceed within the animation (e.g. by avoiding an observed goal failure, exploring an alternative not observed, etc.).

The interrelation structure from the formal specification to RWEs can also be used to elicit missing information for specification gaps detected during animation.

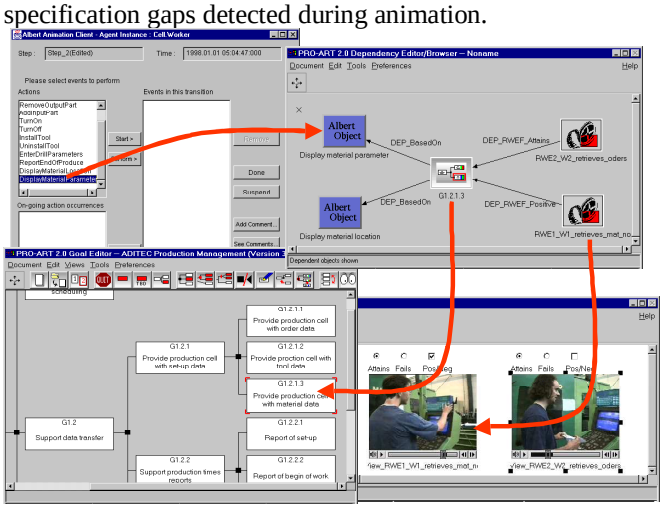

Fig. 7: Using interrelated goals and RWEFs to understand specification names.

Example. We assume that an animation has been initiated to explore with a domain expert — Fred, the foreman of ADITEC — how the Machine Terminals actually support the worker in performing a machine set-up and if this has been covered correctly by the Albert II specification. Control of the agent *worker* was given to Fred. At some stage, Fred was confronted with two alternative actions, one name he never heard before in this particular context: "Display material location" and "Display material parameter". Thus, using the explanation support from his animation window presented the graph shown in Fig. 7. The action was related to the goal "Provide material data necessary for set-up.", which Fred immediately recognised. However, guessing, but not being completely sure what was meant by "material parameter", he also decided to take a look at some video RWEFs, which he discovered after retrieving the goal's dependencies. The RWEF showed one of Fred's colleagues using the terminal's menu to retrieve the shelf number where find the production material in the storehouse. Then, he retrieved some additional numbers with a second operation, which he wrote down next to the shelf number on his notepad. Fred immediately recognised that these were parameters for the shelf's robot for collecting the material. Fred finally concluded that the specification names were imprecise because the shelving system had not been discussed with the analysis team.

4 Conclusions

In this paper we argued that the specification of a change vision for a future system is heavily influence by the context of the system's past. We identified two major problems that have to be solved in an integrated manner for bridging this gap between past and future of the system: the traceability and the envisionment problem. The CREWS Project developed a framework for this kind of integrated solution which has been instantiated by the CREWS-EVE approach presented in this paper.

The integration of CREWS-EVE's two individual components has been discussed both from a technical and a methodological viewpoint. Then we showed three typical applications of this integration in the form of reusable tool supported method chunks, which have been developed in a small case study. In the presentation of the method chunks we showed how solutions for the envisionment problem have been integrated with solutions for the traceability problem and which benefits were gained by this. Beyond these specific method chunks, the experiences show that the integration of real world multimedia scenes and animations in CREWS-EVE provides additional benefits that alleviate limitations that the two techniques have when taken separately: (i) The use of real world scenes is restricted to the re-engineering of systems where system usage is observable and recordable. This limitation can be solved by using animations of system usage in addition to direct observations. (ii) Distributed animation, even though it is an important step to improve understanding and validation of a formal specification, is still rather abstract with respect to the reality the stakeholder are used to experience. Linking animations with goals and real-world scenes reduces this gap.

Related work in this area also combines solutions for bridging the gap between future and past. For instance, Scheer et al. [34] combine virtual reality technology with Business Process Reengineering, but consider the past only by conceptual models and not real world scenes. They also have no formal semantics and no conflict modelling. Nis-

sen et al. [24] provide an approach for cooperation modelling and scenarios including conflicts, but provide only informal representation for the scenarios. Dubois and Yu [8] support formal specifications and goals of the past and the future, but do not consider scenarios. Potts and Anton [1], [28] integrate scenarios and goals of the past for managing the evolution and surfacing requirements for the future system, but only in a static manner and do not capture scenes of the past or explore animations of the future. Sutcliffe et al. [36] developed a tool supported approach considering goals and scenarios which are only in textual form. However, founding the approach on domain knowledge (e.g., hierarchies of exceptions and scenario patterns) collected in the past provides a vast potential for considering the context of the past for the future system. Loucopoulos and Lalioti [21] visualise and animate conceptual models producing scenarios but do not consider goals.

Acknowledgements. This work is founded by the European Community under ESPRIT Reactive Long Term Research 21.903 CREWS (Cooperative Requirements Engineering with Scenarios). The authors are especially grateful to L. Claes, Prof. E. Dubois, and K. Weidenhaupt for their fruitful contributions in numerous, intensive discussions.

5 References

[1] A.I. Antón and C. Potts, "The Use of Goals to Surface Requirements for Evolving Systems", *Proc. of Int'l. Conf. on Software Eng. (ICSE'98)*, Kyoto, Japan, April 1998.

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

[3] F. Brun-Cottan and P. Wall, "Using Video to Re-present the User", *Communications of the ACM*, Vol. 38, No. 5, Mai 1995, pp. 61-71.

[4] L. Claes, "Layered Components", *Proc. of Component-based Information Systems Engineering Workshop (CAISE'98)*, Pisa, June 1998.

[5] J.E. Conklin and K.C. Burges Yakemovic, "A process-oriented approach to design rationale", *Human Computer Interaction*, Vol. 6, No. 3-4, 1991, pp. 357-391.

[6] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-directed Requirements Acquisition", *Science of Computer Programming*, Vol. 20, No.1-2, Apr. 1993, pp. 3-50.

[7] R. Dömges and K. Pohl, "Adapting Trace Environments to Project Specific Needs", *Communications of the ACM*, December, 1998.

[8] E. Dubois, E. Yu, M. Petit, "From Early to Late Formal Requirements: A Process-Control Case Study", *Proc. of IWSSD9*, Isobe, Japan. April 1998.

[9] P. Du Bois, "The Albert II Reference Manual – Language Constructs and Informal Semantics", Research Report RR-97-002, Computer Science Department, University of Namur. ftp://ftp.info.fundp.ac.be/publications/RR/RR-97-002.ps.Z, Jul. 1997.

[10] M. Feather, "Language support for the specification and development of composite systems". *ACM Transactions on Programming Languages and Systems*, Vol. 9, No. 2, , Apr. 1987, pp. 198-234.

[11] L. Fuchs, "Situationsorientierte Unterstützung von Gruppenwahrnehmung in CSCW Systemen", PhD thesis, Gesamthochschule Essen, 1997 (in German).

[12] O. Gotel and A. Finkelstein, "An Analysis of the Requirements Traceability Problem", *Proc. 1st Int'l. Conf. on Requirements Eng.*, Colorado Springs, USA, April 1994, pp. 94-102.

[13] A. Grau, M. Kowsari, "A Validation System for Object-Oriented Specifications of Information Systems", In: Manthey R, Wolfengagen V. (eds). *Advances in Databases and Information Systems*. Proceedings of the First East-European Symposium on Advances in Databases and Information Systems (ADBIS'97). St. Petersburg. September 2-5, 1997. Electronic Workshops in Computing. Springer.

[14] D. Harel, H. Lachover, A. Naamad, et al., "STATEMATE: A working environment for the development of complex reactive systems", *IEEE Transactions on Software Engineering*. Vol. 16, April 1990, pp. 403-414.

[15] P. Haumer, K. Pohl, K. Weidenhaupt, "Requirements Elicitation and Validation with Real World Scenes", *IEEE Transactions on Software Engineering: Special Issue on Scenario Management*, Vol. 24, No. 12, December 1998.

[16] P. Heymans, "The Albert II Specifications Animator", CREWS Report Series 97-13, ftp://Sunsite.Informatik.RWTH-Aachen.DE/CREWS/, Aug. 1997.

[17] P. Heymans and E. Dubois, "Scenario-Based Techniques for Supporting the Elaboration and the Validation of Formal Requirements", to appear in *Requirements Engineering Journal*, 1999.

[18] ITU-T. *ITU-T Recommendation Z.120 — Message Sequence Charts (MSC)*, ITU, 1993.

[19] D. Kahneman and A. Tversky, "Subjective probability: A judgement of representativeness", *Cognitive Psychology*, Vol. 3, 1972, pp. 430-454.

[20] R. Kurki-Suonio, "Fundamentals of object-oriented specification and modeling of collective behaviors", *Object-Oriented Behavioral Specifications* (Eds. H. Kilov, W. Harvey), Kluwer Academic Publishers 1996, pp. 101-120.

[21] V. Lalioti and P. Loucopoulos, "Visualisation of Conceptual Specifications", *Information Systems*, Vol. 19, April 3, 1994, pp. 291-309.

[22] K. McGraw and K. Harbison, *User-Centered Requirements: The Scenario-Based Engineering Process.* Lawrence Erlbaum Associates, Inc., Publishers, Mahwah, New Jersey, 1997.

[23] G. O'Neill, "Automatic translation of VDM into standard ML programs". *The Computer Journal*. March 1992.

[24] H.W. Nissen, and M. Jarke, "Repository support for informal teamwork methods", In Lyytinen/Welke (eds.): *Special Issue on Meta Modeling and Method Engineering, Information Systems*, Vol. 24, No. 2, 1999.

[25] K. Pohl. "PRO-ART: Enabling Requirements Pre-Traceability", *Proc. of the 2nd Int'l. Conf. on Requirements Eng.*, Colorado-Springs, Colorado, USA, April 1996.

[26] Klaus Pohl. "Process-Centered Requirements Engineering". RSP/ Wiley and Sons, 1996.

[27] K. Pohl, , K. Weidenhaupt, R. Dömges, P. Haumer, M. Jarke, R. Klamma. "PRIME: Towards Process-Integrated Modelling Environments", to appear in ACM *Transaction on Software Engineering and Methodology*, 1999.

[28] C. Potts, K. Takahashi, and A.I. Antón. "Inquiry Based Requirements Analysis", *IEEE Software*, Vol. 11, No. 2, Apr. 1994, pp. 21-32.

[29] B. Ramesh and Sengupta, "Multimedia in a Design Discussion Support System", *Decision Support Systems*, vol. 15, no. 3, 1995.

[30] B. Ramesh, C. Stubbs, T. Powers, M. Edwards, "Requirements traceability: Theory and practice", *Annals of Software Engineering*, Vol.3, 1997, pp 397-415.

[31] B. Ramesh, "Factors Influencing Req. Traceability Practice", *Communications of the ACM*, December, 1998.

[32] Rational Software Corporation, *Unified Modeling Language*. Available on the World Wide Web: http://www.rational.com, Rational, 2800 San Tomas Expressway, Santa Clara, CA 95051-0951, USA, Jan. 1997.

[33] C. Rolland, C. Souveyet, and C.B. Achour, "Guiding Goal Modelling Using Scenarios", *IEEE Transactions on Software Engineering: Special Issue on Scenario Management*, Vol. 24, No. 12, December 1998.

[34] A.-W. Scheer, S. Leinenbach, and M. Luzius, "Nutzenpotentiale neuer Medien im Geschäftsprozeßmanagement: Virtual Reality und Multimedia zur Visualisierung und Simulation betrieblicher Abläufe", in: Karrenbauer et al. (eds.): 3. *SaarLorLux Multimedia-Kongreß 1997*, Shaker Verlag, Aachen 1997, pp. 43 - 48 (in German).

[35] J.I. Siddiqi, I.C. Morrey, C.R. Roast, M.B. Ozcan. "Towards quality requirements via animated formal specifications", *Annals of Software Engineering*, Vol. 3. 1997.

[36] A.G. Sutcliffe, N.A.M. Maiden, S. Minocha, and D. Manuel, "Supporting Scenario-Based Requirements Engineering", *IEEE Transactions on Software Engineering: Special Issue on Scenario Management*, Vol. 24, No. 12, December 1998.

[37] K. Weidenhaupt, K. Pohl, M. Jarke, and P. Haumer. "Scenario Usage in System Development: A Report on Current Practice", *IEEE Software*, Mar., 1998, pp. 34-45.

[38] D.P. Wood, M.G. Christel, and S.M. Stevens, "A Multimedia Approach to Requirements Capture And Modelling", *Proc. 1st Int'l. Conf. on Requirements Eng. (ICRE'94)*, IEEE Computer Society Press, Colorado Springs, CO, USA, Apr. 1994, pp. 53-56.

[39] E. Yu and John Mylopoulos, "Why Goal-Oriented Requirements Engineering", *4th Int'l. Workshop on Requirements Eng.: Foundation for Software Quality (RESFQ'98)*, Pisa, Italy, Jun., 1998.