Use Case, Goal, and Scenario Analysis of the Euronet System : Comparing Methods and Results

Thomas A. Alspaugh
University of California, Irvine
alspaugh@ics.uci.edu

Annie I. Antón
North Carolina State University
anton@csc.ncsu.edu

November 2003

ISR Technical Report # UCI-ISR-03-12

Use Case, Goal, and Scenario Analysis of the Euronet System : Comparing Methods and Results

Thomas A. Alspaugh
Institute for Software Research
University of California, Irvine
Irvine, CA 92697-3425 U.S.A.
alspaugh@ics.uci.edu

Annie I. Antón
College of Engineering
North Carolina State University
Raleigh, NC 27695-8207 U.S.A.
aianton@eos.ncsu.edu

ISR Technical Report #UCI-ISR-03-12
November 2003

Abstract: In this paper, we compare the results of three related requirements engineering efforts: an industrial requirements specification produced with a use case based process, a case study analyzing those use cases by means of goal analysis; and a second case study analyzing the original use cases with an integrated scenario analysis and management approach and software tool support. The scenario-based analysis proved more effective than either of the other two approaches. The results provide validation for both the integrated scenario analysis and the software tool.

Use Case, Goal, and Scenario Analysis of the Euronet System : Comparing Methods and Results

Thomas A. Alspaugh
Institute for Software Research
University of California, Irvine
Irvine, CA 92697-3425 U.S.A.
alspaugh@ics.uci.edu

Annie I. Antón
College of Engineering
North Carolina State University
Raleigh, NC 27695-8207 U.S.A.
aianton@eos.ncsu.edu

ISR Technical Report #UCI-ISR-03-12
November 2003

Abstract

In this paper, we compare the results of three related requirements engineering efforts: an industrial requirements specification produced with a use case based process, a case study analyzing those use cases by means of goal analysis; and a second case study analyzing the original use cases with an integrated scenario analysis and management approach and software tool support. The scenario-based analysis proved more effective than either of the other two approaches. The results provide validation for both the integrated scenario analysis and the software tool.

1. Introduction

Scenarios have increased in popularity among software engineers due,in part, to Jacobson's use case approach [14] and the more recent introduction of the Unified Modeling Language (UML) [10]. UML's availability and tool support have made scenario and use case analysis even more accessible to requirements analysts and software practitioners.

Scenarios are used to "stimulate thinking" in various disciplines. In software engineering, scenarios are used to elicit and validate requirements [8, 15]. Scenarios are also an approach for managing change and evolution during the software process. For example, evolutionary scenarios are used to envisage how a system may change due to, for instance, political or technological discontinuities [8]. In requirements engineering, scenarios are valuable for communicating with stakeholders and providing context for requirements [7, 18, 22, 20, 23]. Although valuable, scenarios and use cases can be difficult to manage, and their management is increasingly important as system complexity increases. Scenario management is receiving increased attention among researchers in the requirements community [3, 15, 23].

The state of scenario management in practice was reported by Weidenhaupt *et al.* [23], who examined the use of scenarios in 15 European projects to learn how scenarios were produced and utilized, and to identify the benefits and problems associated with scenario usage in industrial settings. Practitioners using scenarios or use cases in industrial settings face very specific challenges. Key areas needing support include the need for appropriate process guidance as well as comprehensive support for managing both scenario traceability and evolution [23].

In this paper, we discuss our experiences using our strategies for utilizing a collection of syntactic analyses of scenarios and management strategies, which we term *Integrated Scenario Analysis* (ISA) [2, 3], and a software tool *SMaRT* (Scenario Management and Requirements Tool, developed at North Carolina State University), to analyze the requirements for a business-to-business e-commerce system. We present the results of a case study on the Euronet system, providing data on the effectiveness of ISA and SMaRT. In this case study, we analyzed and managed a large collection of scenarios during requirements specification activities. Euronet is a quote management system used internally by ABB to create and manage quotes for transformers, line items that give the details of the quotes, and resulting orders from customers. Three approaches have been employed to analyze the Euronet system requirements: a use case based analysis, resulting in the original system requirements; a goal analysis [6]; and the scenario-based analyis reported here. For this particular study, we employed the ISA analysis and process guidance methods [2, 3], supplemented by scenario identification and elaboration heuristics from GBRAM (Goal-Based Requirements Analysis Method) [5] and the Inquiry Cycle [20]. The analysis was assisted in part by SMaRT, which was in a prototype stage during the case study.

In Section 2, we discuss relevant work in requirements specification. We briefly summarize our findings from the use case and goal-driven analysis in Section 3, before discussing integrated scenario analysis in Section 4. Section 4 also discusses our use of SMaRT (Scenario Management and Requirements Tool). Section 5 contrasts the three analysis methods within the context of the Euronet system and the the lessons learned are detailed in Section 6. The paper concludes with a discussion of future work in Section 7.

2. Related Work

The terms “use cases”, “scenarios”, and “goals” mean different things to different people; we thus discuss these terms in the context of other relevant work including the requirements engineering literature.

2.1. Scenarios and Use Cases

Scenarios aid analysts and stakeholders in developing an understanding of current or envisaged systems and business processes [5, 7, 8, 9, 11, 14, 18, 23]. They describe concrete system behaviors by summarizing behavior traces. Use cases, introduced by the object-oriented community [10, 14], describe groups of system interactions that external agents may have with a system. In UML, use cases are comprised of sets of actions and interactions that involve specific objects. Requirements engineering benefits from an initial emphasis on use cases, but they benefit in turn from a semantics that connects them to purposeful activities [18, 19]. A representational framework for scenarios and use cases is given by Antón and Potts [8].

Scenario analysis is a very effective and proven technique for surfacing goals during requirements engineering. The CREWS-SAVRE tool organizes scenarios hierarchically according to goals and goal obstacles; the goals serve as a grounded, shared understanding for stakeholders [16]. Goal-scenario coupling, as documented in [21, 22], provides an integrative approach to goal and scenario-oriented requirements analysis. The CREWS-L’Ecritoire approach employs bi-directional coupling to facilitate navigation between goals and their associated scenarios.

SMaRT supports management and analysis of scenario-based specification [2, 3]. It provides a database for storing scenarios in a standard form, facilities for searching, comparing, and identifying duplicates and near-duplicates, capabilities for identifying, recording, and preserving dependencies between scenarios, automated analyses for process guidance, and other features. SMaRT was used in the research presented in this paper.

2.2. Goals and Scenarios

Goals are the objectives and targets of achievement for a system. Goal-driven approaches focus on why systems are constructed, expressing the rationale and justification for the proposed system. Goals are evolutionary and they provide a common language for analysts and stakeholders. Focusing on goals, instead of specific requirements, allows analysts to communicate with stakeholders using a language based on concepts with which they are both comfortable and familiar. Furthermore, since goals are typically more stable than requirements [5], they are a beneficial source for requirements derivation. Goals are operationalized and refined into requirements and point to new, previously unconsidered scenarios. Similarly, scenarios also help in the discovery of goals [7, 9, 15, 19, 22].

Goal hierarchies offer a useful way to visualize goals and their related scenarios [4, 5]. Organizing goals hierarchically provides a useful way to represent the relationships between goals and subgoals so that we can reason about those relationships [7, 12]. The Goal-Based Requirements Analysis Method (GBRAM) uses a goal topography to structure and organize such requirements information as scenarios, goal obstacles, and constraints [5, 9]. These topographies support analysts in finding and sorting goals into functional requirements while scenarios help in documenting issues, surfacing new goals and elaborating requirements.

The next section provides an overview of the ABB Euronet System as well as the use case and goal-driven analyses which provided the basis of comparison for our more recent scenario-based analysis.

3. E-Commerce & Quotation System Analysis

ABB has numerous plants throughout the Americas, Asia, and Europe that manufacture a variety of engineering products. The parent company maintains its own sales force, whose members provide quotations for product pricing and place orders for customers. Existing processes for providing quotations or ordering products were numerous and *ad hoc*, with each sales person and each plant having a different process, using various computer systems, printed catalogs or direct sales persons as plant contacts. A new, more tightly integrated system was needed to facilitate the provision of consistent lowest prices with a streamlined bidding process to aid the company’s distributed sales force. Specifically, the new system was expected to produce consistent quotations and order prices while tracking statistical information such as market trends. Another advantage identified in the development of this system was to provide better service capabilities to current and potential customers.

3.1. Euronet Use Case Analysis

A development team, comprised of software developers, IT specialists, and people with intimate knowledge about the business activities, was assembled to deliver this system. The development team began working in several parallel tracks. First, members with intimate knowledge about the business defined the business implications of uniting sales and technical support people across Europe with a single system. Second, an initial Software Requirements Specification (SRS) was created consisting primarily of a collection of use cases. This SRS serves as the basis for the goal driven analysis and ISA analysis compared in this paper. Third, based on the functionality of the prototypes developed, rather than the SRS, software developers began the Euronet system implementation while the creation of the SRS was still under way.

According to the software development team, the SRS use cases were insufficient as requirements in developing the system; they did not provide a complete and consistent specification of what the Euronet system was required to do. Nor were the prototypes a sufficient guide. Consequently, the Euronet system implementation effort ran far beyond schedule and over budget. It was because of this that ABB requested the goal-driven analysis of the SRS to analyze whether and in what ways the SRS was not sufficient.

The initial ABB SRS consists of a general overview, 52 use cases, and 26 screen designs [1]. The use cases are numbered 1 through 52 and have brief titles such as "User Log On". Each use case consists of a one- to three-sentence overview, a main scenario consisting of a list of events, a list of screens used by the use case, notes on information not otherwise given in the use case, and the author's name. Most of the use cases have informal pre- and postconditions. 25 of the use cases have one or more paragraphs describing secondary (alternative) scenarios. The use case event lists range in length from 1 to 19 events, with three events being the most common length. The use cases employ each other as events; 27 of the use cases are referred to in event lists, and individual use cases include as many as six other use cases directly, and as many as 19 indirectly. The "includes" hierarchy of the use cases is given in Figure 1. This hierarchy was determined as part of the scenario-based analysis. The use cases vary in length from one to six pages , with two pages being the most common.

The results of the initial use case based analysis and of our previous goal-driven analysis case study provide a standard against which to compare. The goal of our second Euronet case study, presented herein, was to compare the results of applying ISA and SMaRT to the results of applying

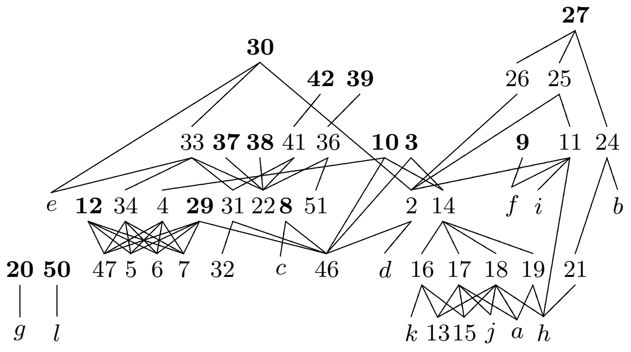


Figure 1. The "includes" hierarchy for the ABB Euronet SRS. Numbers are use case numbers; letters identify use cases referred to but not defined; boldface numbers highlight those use cases that are not included in other use cases. The 11 use cases that neither include nor are included are not shown.

goal-driven analysis, and to the results of ABB's original requirements engineering work as embodied in the specification.

3.2. Euronet Goal-driven Analysis

For the goal-driven analysis [6], our efforts entailed deriving goals from the 52 use cases in the initial ABB SRS. During this analysis, each goal was annotated with relevant auxiliary notes including agents, constraints, pre- and postconditions, scenarios, and questions as well as answers provided by various stakeholders during follow-up interviews. For traceability we tracked any changes to the goals and the associated rationale. All of the above information was documented using Microsoft Excel spreadsheets because the analysts lacked specialized tool support. Although the analysis consumed 21 analyst-hours over the course of two months, only 13 of the 52 use cases were actually analyzed. From these 13 use cases, the analysts derived 130 goals and 119 scenarios. The goals were relatively broad, but the scenarios were extremely detailed, focusing mainly on exceptions and alternatives to ensure that the goal analysis was complete and not simply reflective of normative system use.

As an ancillary part of the goal analysis, an "includes" hierarchy of use cases was created to identify those that were named but not defined in the original ABB SRS. Fifteen such use cases were located using this technique [6]. That hierarchy is no longer available, but the similar hierarchy produced during the ISA analysis appears in Figure 1.

4. Integrated Scenario Analysis (ISA)

ISA consists of a group of mutually reinforcing approaches that are amenable to automated support. The approaches include:

- structuring scenarios as event sequences plus attribute values [3];
- structuring each event as an actor-action pair;
- structuring scenario characteristics as attribute values;
- using glossaries to define attribute values and events;
- using glossaries of words and phrases that are used with system-specific meanings;
- using episodes to express scenario dependencies; and
- using syntactic similarity to measure similarity between scenarios, for searching and for uncovering duplication.

The structure ISA gives to scenarios is shown in Figure 2. Scenarios is organized into attribute-value pairs. For example, the goals of a scenario are expressed as the values of the scenario's "goal" attributes. Each event of a scenario is defined to be an actor and an action; an event is expressed as the value of an "event" attribute of the scenario, and its actor and action are expressed as the values of the event's "actor" and "action" attributes. A scenario's events are expressed in a sequence. The event sequence for a scenario may include simple events as described above and *episodes*, which are intentionally-shared subsequences of events that may appear in two or more scenarios [3]. We have extended the event sequence structure beyond our earlier work to add expressiveness and convenience, by also including iterated subsequences of events, and alternation between two or more alternative events or subsequences.

Using such a structure provides a number of benefits. Tool support for scenarios is made more straightforward, and a basis for automated scenario analyses is laid. Once scenarios are cast into this form, it is possible to begin asking whether analysis of the scenario's structure, which can be done automatically, can indicate answers that otherwise require an experienced analyst, careful consideration, and hard work. ISA is one such analysis approach, supported by SMaRT as discussed in the next section.

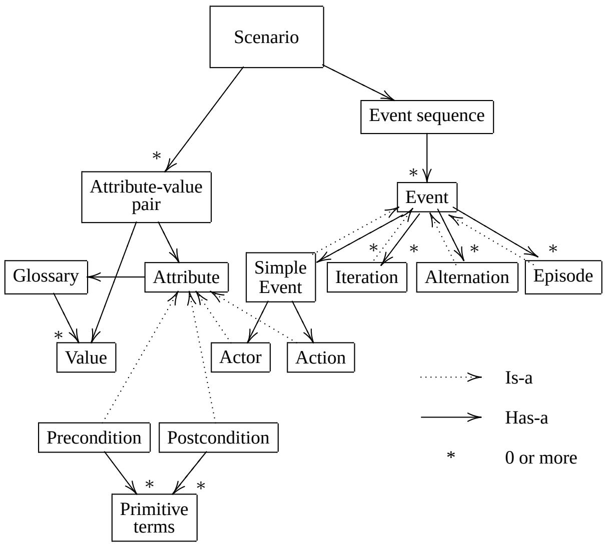


Figure 2. Metamodel for ISA

4.1. SMaRT

For this case study, we employed a software tool, SMaRT, which supports the representation of scenarios as attribute-value pairs, glossaries of attribute values, glossaries of terms, episode management, and (eventually) syntactic similarity measures [3]. SMaRT is currently implemented as a database on a server accessed over the Internet through a browser. Data presented by SMaRT is organized into projects. Each project can have one or more analysts allowed to edit or view the contents. For each project there is a set of scenarios, a set of episodes, and a glossary for each attribute (including goals) that a scenario or episode can possess. Scenarios and episodes can be created and edited using attribute values in the glossaries. These attributes include: actors, actions, events, goals, obstacles, requirements, and conditions. New values can be created for an attribute; existing values can be edited; and values that are not referenced can be deleted. A clickable list of every reference to the value can be generated automatically; clicking a reference takes the analyst to the scenario (episode, event) that uses the value. Event lists can include simple events, episodes, iteration of event sub-lists, and alternation between event sub-lists. Episode references in an event sequence can be expanded to show the episode's event sequence where it would occur, and the episode's event sequence can be edited in the context of a scenario referencing it. Clicking on episode or events takes the analyst to the episode or event editor, where all the information about that episode or event is available.

A screen shot of the Scenario Editor is shown in Figure 5. The scenario being edited is Euronet S_2 "Retrieve Existing Quote"; it exhibits a number of features of our representation, including an event list containing episodes and alternation (indicated by indentation, hierarchical event numbering, and color-coding), and attributes that include goals and pre- and post-condions. Future work for SMaRT includes implementation of similarity measures [3], expanded cross-reference capability, "includes" hierarchies for episodes, and integration of scenario network support [2].

4.2. Euronet Scenario Analysis Case Study

For the scenario-based analysis, our efforts entailed deriving scenarios from the same 52 Euronet use cases that were analyzed using the goal-driven approach. The scenarios identified from the use cases were documented in SMaRT and were analyzed as discussed below.

The analysts (the co-authors of this paper) met for marathon sessions ranging from five to eight hours in duration, daily for seven work days. During our analysis session we concentrated primarily on revising and extending the identified scenarios and discussing intricacies pertaining to alternations and iterations within certain scenarios as well as the identification and elaboration of episodes.

The scenario analysis began by identifying scenarios in the ABB SRS use cases main and secondary scenarios. Scenarios and their associated information were identified, numbered, and stored as shown in Figure 5. The information tracked (partially to ensure traceability) included the scenario number, responsible agent(s) (documented in an agent index), the use case from which each scenario was derived (documented in a included-by and includes index), pre- and postconditions, as well as any issues, rationale or questions related to the scenario.

SMaRT facilitated our efforts by ensuring that all referenced terms (e.g. actors, actions, scenarios, episodes, pre- and postconditions) were defined within the project repository, thereby enforcing a minimal level of consistency. Higher-level and more sophisticated consistency checks were conducted manually by the analysts. Some of these checks are automatable and planned for future SMaRT releases. For example, we used an episode in one scenario and before subsequently reusing the episode in another scenario, we manually checked the episode to ensure it was suitable for use by both scenarios. Although such checks were manual, they were made much easier by the tool. Scenarios were named using a descriptive title that reflected what each scenario achieves, numbering them based upon the original use case number. Our initial ISA analysis of the use cases produced 28 scenarios and 34 episodes, 321 events composed of 8 actors and 232 actions, and 29 conditions used in pre- and postconditions.

A careful syntactic examination of the Euronet use cases immediately revealed a large number of problems. For example, the use cases named in the "Utilizes" section of each use case description frequently did not match the use cases that appeared in the event list. Twelve use case names were referred to in events but not defined. Ten screens named in the use cases were not defined; four of the ten were deemed equivalent to similarly-named defined screens, leaving six that were definitely undefined. Of the 26 defined screens, three were never referenced in any use case.

The Euronet "includes" hierarchy highlights the presence of use cases that are referenced but not defined. A "includes" hierarchy is a graph whose nodes are use cases (or scenarios and episodes), and whose edges show inclusion or reference of one node by another. For example, in Figure 1, there is a line from use case 30 down to use case 33 because 30 includes 33 as part of its event list. Although the analysis of "includes" hierarchies has not been a part of ISA, we found it to be extremely useful for Euronet. Our earlier goal-driven analysis included the production of "includes" trees for individual use cases, and discovered the same instances of undefined use case names, [6]. The complete hierarchy for the entire collection of use cases in the ABB SRS (Figure 1) draws attention to other potential areas of interest: the depth of the hierarchy relative to the number of use cases involved, and two patterns of strongly interconnected groups within the hierarchy.

The hierarchy is surprisingly deep compared to the number of use cases (see Figure 1). There are 64 use cases named in the Euronet specification, 52 with definitions and 12 without. Of these 64 use cases, 53 appear in the hierarchy, and the remaining 11 neither use nor are used by any other use case. The four deepest "includes" paths connect five use cases each (about 9% of the nodes in the hierarchy). Twenty-six paths connect four use cases each. Deep "includes" hierarchies indicate likely problems, especially in the absence of tool support to display the nested inclusions:

- analysts may not have examined the consequences of including deeply-nested events;
- the specification is unlikely to have been reviewed thoroughly, due to the inconvenience of manually looking up the included event sequences in context;
- spurious dependencies between scenarios (or use cases) are implied by the incompletely-considered inclusions; and
- unintended behavior is specified by the deeply-nested inclusions.

We can draw more specific conclusions from the interconnections in some areas of the hierarchy in Figure 1. The complete connections from use cases 12, 34, 4, and 29 to 47, 5, 6, and 7 indicate that the upper use cases should be similar in other ways, and raise the question of whether 47, 5, 6, and 7 should be separate use cases since they only appear together (and in the same sequence, although the hierarchy does not show this). Use cases 16, 17, 18, 19, and 21 are less completely connected to *k*, 13, 15, *j*, *g*, and *h*, but still indicate that attention should be paid to them; 16, 17, and 18 should be similar in other ways, and one should inquire whether 13, 15, *j*, and *a* should be separate use cases.

The analysis of "includes" hierarchies was so unexpectedly fruitful for Euronet that we are incorporating it into ISA and adding support for it to SMaRT. Tool support is needed because it is time-consuming and tedious to extract the “includes” relationships from the scenarios and episodes and collate them into a hierarchy; and this tedious task needs to be re-done whenever an event list is changed to include an episode or not. Consequently we did not produce “includes” hierarchies regularly during the ISA analysis. As we discuss below, had we regularly examined the “includes” hierarchy, it would have guided us to produce a different (better) collection of scenarios and episodes.

The “includes” hierarchy resulting from ISA applied to Euronet is shown in Figure 3. As in Figure 1, only the scenarios that include episodes are depicted. From this diagram we can see that ISA, without the explicit guidance of “includes” hierararchy diagrams, took care of some of the problems we noted in Figure 1 on other grounds. Fewer episodes are used, and those that are used are included in simpler ways. The parts of the hierarchy that were nested 5 deep turned out to have problems that ISA identified and that the analysts repaired guided by ISA. For example, when we examined the event sequence of S_{27} “Revise Quote” with the events that were represented by its inclusion of S_{25} “Approve Quote”, S_{26} “Assemble Quote Package”, and S_{24} “Submit Quote”, it became clear that S_{25} and S_{26} did not belong within S_{27}. Rather, they described event sequences that were associated with S_{27}, before or after the other events that S_{27} listed. For example, a desired behavior of Euronet was clearly

1. assemble a quote package (S_{26}),
2. revise it once (much of S_{27}’s own events),
3. examine and approve the quote package (S_{25}), and
4. submit the quote (S_{24}).

Other sequences described desired Euronet behavior as well, including re-assembling a quote package after substantial revisions; examining a quote package several times, revising it in between until it is finally approved completely; and revising a submitted quote package in the face of new information from the customer. Most of this behavior had no place in S_{27}, and when it was removed the long inclusion paths beginning at S_{27} vanished. Other changes that flattened and simplified the “includes” hierarchy can be seen by comparing the two diagrams.

However, other problems still remain, although these were not identified during the ISA analysis because we were not regularly generating and examining the “includes” hierarchy. Several episodes were only included by a single scenario or episode. While this can be reasonable if the episode represents behavior that is expected to be shared in the future, we had no reason to believe that was the case in many instances. Examples of this are Ep_{41} which is only included by S_{42}, and Ep_{32} which is only included by Ep_{31}. Another problem was the clusters of inclusions visible in the diagram. ISA eliminated the cluster of nearly identical inclu-

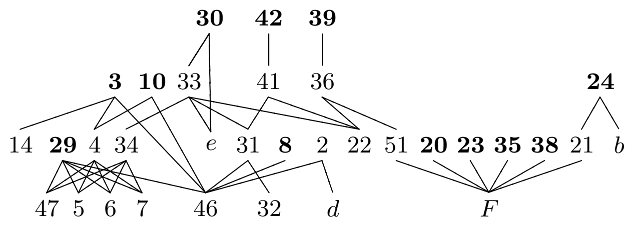


Figure 3. “Includes” hierarchy for the results of ISA applied to Euronet. Bold numbers highlight scenarios.

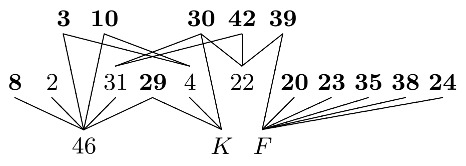


Figure 4. “Includes” hierarchy for Euronet after ISA and refinement.

sions of use cases 13, 15, *j*, and *a* on other grounds, but left the cluster including 47, 5, 6, and 7. Figure 4 shows the “includes” hierarchy after refinements guided by examination of Figure 3. 47, 5, 6, and 7 have been merged into a single episode Ep_K that is included in their place; and single inclusions such as that of Ep_b by S_{47} have been eliminated by inlining the included event sequence. The resulting hierarchy is simpler and flatter, with only two paths deeper than two, and the corresponding scenarios and episodes are easier to read, evaluate, validate, and maintain. For specifications such as Euronet’s SRS that make heavy use of “includes” between use cases or between scenarios and episodes, the techniques outlined here provide effective guidance.

5. Comparison of Analyses

This section contrasts the results of the previously mentioned requirements analysis efforts for the Euronet system. The results are summarized in Table 1.

The original use case based requirements analysis produced 52 use case definitions. References to 27 of the use cases are included as events in one or more of the 52 use cases. The use cases exhibited a number of general problems. The context of the use cases was not made clear, both the organizational context of ABB’s practices and policies and the behavioral context in the overall Euronet system function; the use cases were too focused on GUIs and system design and implementation; they were referred to with inconsistent names; and some names appeared to refer to undefined use cases [6].

In the goal-driven analysis effort the 52 original use cases were retained. Only 13 of the use cases were analyzed because this was sufficient to achieve the necessary results, which were to show whether the original SRS was complete, consistent, and a sufficient basis for software development (it was not). These use cases were annotated with substantial detail, including goals and obstacles as well as additional preconditions, postconditions, and scenarios. As previously mentioned, the analysis yielded 130 distinct goals, derived from the 52 use cases. During the goal identification and elaboration process, the analysis identified 119 new secondary scenarios. These scenarios refined the 52 use cases, representing alternative paths at a very low level rather than new high level behavior not covered by the use cases. The 119 secondary scenarios were heavily influenced by analysis of the screen sketches in the SRS. During this effort, it became evident that some use cases were referenced but not defined. The analyst team thus conducted a supplementary syntactic analysis, creating of an "includes" hierarchy to determine whether all referenced use cases had definitions; this led to the identification of 15 distinct use cases that were referenced, but not defined. Three of the 15 were deemed synonymous with other uses cases and reconciled accordingly [6].

ISA analysis was applied to all 52 of the original SRS use cases. 28 of these use cases were retained as first-class scenarios describing behavior initiated by system users. The remaining 24 use cases were characterized as episodes describing behavior that occurred only in the context of an "includes" reference by a scenario or another episode. ISA identified 12 distinct use cases that were named as included but never defined, resolving the other three found in the goal-driven analysis as *prima facie* misspellings. The analysts gave definitions to five of the twelve based upon the contexts in which they were used and their names. Three of these five were resolved as likely misspellings of other use cases. A fourth was absorbed into a new episode that generalized behavior that the initial use cases defined for some contexts but left undefined for others. The fifth was absorbed into a new scenario that defined behavior analogous to that of an existing scenario, but under different preconditions. Finally, ISA identified a new scenario based on comparison of pre- and postconditions of existing scenarios. The new scenarios and episode were of a larger scope and greater significance than the 119 new scenarios identified by the goal-driven analysis. ISA also identified these new scenarios and episode with a comparitively small investment of time. The screen sketches in the initial SRS were not considered during the ISA analysis, following the practice recommended by Antón *et al.* [6]. The ISA analysis was performed using early prototypes of SMaRT. The effects of using these prototypes was both positive and negative. The prototypes provided some automated support for the syntactic analyses that comprise ISA. The remaining syntactic analyses were done by hand. The user interface for the SMaRT prototype, particularly the early prototype, was relatively inefficient so that its use slowed the analysis during that period rather than speeding it. We can see that SMaRT, when fully functional, will be much faster to use and will automate those analyses that were done by hand in this case study, thus making ISA substantially more efficient.

It is instructive to compare the weight and scope of the new scenarios found by the goal-driven analysis to those found by ISA. An example scenario identified by ISA is S_{98} "User Log Off", defining the events needed for logging off, the preconditions for the contexts in which logging off can occur, and the postconditions defining the results of logging off. On the other hand, consider the scenarios yielded by analysis of the goal `<ENSURE adjustments for pricing for a standard product provided>` during the goal driven analysis. This goal yielded seven secondary scenarios to represent possible pricing adjustment alternatives:

- Adjustment determined by competitive level.
- Adjustment determined by item quantity.
- Adjustment determined by rush delivery.
- Adjustment determined by sales channel.
- Adjustment determined by special features.
- Adjustment determined by plant loading.
- Adjustment determined by end customer.

Clearly, the level of granularity for these scenarios is much smaller than those identified during the ISA analysis.

One marked difference between ISA supported by SMaRT and goal analysis is the time and effort required. Our ISA analysis of all 52 use cases required 41 analyst-hours, while the goal analysis of only 13 use cases required 21 analyst-hours; extrapolated linearly to the full set of 52 use cases, the goal analyis would require approximately twice as much effort (84 analyst-hours).

We note that although the second author was involved in the goal analysis as well as the scenario analysis, nearly four years had elapsed between the two case studies, with no intervening work in that domain. The other scenario analyst was completely unfamiliar with the domain before the case study began.

Some might think that comparing the ISA tool support provided by SMaRT with the goal analysis tool support provided by Excel is not a fair comparison. We note that goal analysis requires less automated support than ISA does. The spreadsheet functioned mainly as a means of storing and organizing data for the goal analysis; further automated support was not needed. A more powerful tool would not make much difference. SMaRT also stores and organizes data, and of course for both goal and ISA analysis the data must be entered. But ISA depends on calculations and comparisons of large amounts of data, and these are time-consuming and error-prone if done manually. Automated support makes a substantial contribution here. We also note that during the ISA analysis of Euronet, SMaRT was still in a prototype stage. Some of the automatable analysis (such as the derivation and construction of Figures 1, 3, and 4) had to be done manually. Automated support for these functions would have reduced the number of analyst-hours required.

6. Lessons Learned

Our analysis yielded several lessons learned that we believe to be of interest to practitioners and researchers alike.

The return on investment is higher for ISA and SMaRT

Scenario analysis using ISA and SMaRT yielded a more complete and consistent set of scenarios than either goal analysis or the original use case analysis by itself, and did so with a comparatively small investment of time and effort. The improvements produced by ISA and SMaRT were of more significance than the smaller-scale details fixed by goal analysis. Although the collation and calculation required for ISA is time-consuming if done manually, it is well suited to automated support and the SMaRT prototype was able to assume a substantial part of the burden. A more advanced SMaRT would reduce the manual effort even further. The addition of ISA supported by SMaRT to the ABB use case analysis resulted in a markedly better specification for a small additional investment of time and effort.

Many inconsistencies should be immediately reconciled, not managed

In the inconsistency-management approach, inconsistencies are identified and then not reconciled but managed for some period of time to extract the most information from them [13, 17]. However, the great majority of the inconsistencies we identified with SMaRT and ISA were not informative but simply indicated missing information. For example, the use cases referred to a large number of statuses of various items, but these statuses were never defined and in many cases an entity's status was set but never examined, or examined but never set. We believe many of these were due to the fact that no analyst had paid attention to the inconsistent items before. SMaRT and ISA are effective in identifying and drawing attention to such inconsistencies.

Automated episode identification reveals duplication

Although SMaRT's episode identification functionality was not implemented at the time we performed this case study, the use of SMaRT helped us more readily identify duplication that we had previously overlooked. For example, we observed that UC_{42} "Change Language" partially duplicates a secondary scenario of UC_1 "Log On"; and the use case hierarchy and our episode cross-reference showed us that the four episodes Ep_{47}, Ep_5, Ep_6, and Ep_7 always appeared together in the same sequence, and never otherwise, and thus that they could be merged into a single episode. The Euronet analysts were apparently unaware of these repetitions, and SMaRT helped us identify them more easily.

We noted a number of instances of actions, events, and episodes that were identical except for one or two words. For example, we found several actions of the form "requests X" for various X's, and corresponding events of the form "U : requests X" for various users U and the same X's. A concrete example was "BA Engineer : requests Report". In a few cases this continued up to episodes; for example, the episodes Ep_{F1} "Save/Close Item," Ep_{F2} "Save/Close Order," Ep_{F3} "Save/Close Quote," and Ep_{F4} "Save/Close Quote Package." If actions, events, and episodes are parameterized, then fewer individual episodes are needed, and related ones can be standardized in a way that can be automatically supported.

Standardizing on a small number of action words makes it easier to find the right action and to express actions consistently

Although this lesson may seem jejune, the problem it addresses still occurs in industrial practice (it was prominent in the Euronet use case based analysis) and its impact is considerable. The events in the original Euronet use cases were worded in an inconsistent fashion. We found that choosing a set of standard words to employ wherever possible greatly reduced the number of separate actions, and at the same time made the events clearer and easier to understand relative to each other. For example, we standardized using the word "select" in place of "choose," "find," "click," etc. whenever it was appropriate. This lesson was analogous to the similar finding for goal wordings by Antón *et al.* [6].

Long paths in an "includes" hierarchy indicate a need for further refinements

In the "includes" hierarchy of use cases shown in Figure 1, we found two overlapping paths of 5 use cases. Several of the use cases in these two paths (UC_{25}, UC_{26}, and UC_{27}) were later found to have disagreements between their events and the events in the use cases they used, several levels down. We observed that a deep "includes" hierarchy (more than 3) indicates a proclivity for inconsistencies between the episodes being used and the context in which they are used, as well as the need to refine the set of scenarios.

Figure 3 shows the "includes" hierarchy for the scenarios and episodes resulting from the ISA analysis. The number of levels in the initial SRS "includes" hierarchy (Figure 1) suggests opportunities for refinement, as in the following

Use case based	Goal-driven	Integrated Scenario Analysis
52 use cases	13 use cases analyzed	28 scenarios retained.
27 of these included in others		24 episodes retained,
	15 undefined UCs identified,	12 undefined episodes identified;
	3 were likely misspellings	2 merged into a new episode or scenario,
	of other UCs	1 new episode identified.
	119 new scenarios identified	2 new scenarios identified.
	130 goals identified	
140 analyst-hours over 5 days	21 analyst-hours over 2 months.	41 analyst-hours over 8 days.

Table 1. Comparison of the three approaches

two examples. Some episodes are included in only one scenario, such as Ep_{36} in S_{39}, and Ep_{41} in S_{42}. In most cases, such episodes should be absorbed into the scenario. Another example is that of a sequence of episodes is called in the same order by more than one scenario, such as Ep_{47}, Ep_{05}, Ep_{06}, and Ep_{07}, which are called in the same sequence by Ep_{04}, S_{29}, and Ep_{34}. Such episodes should be merged into a single episode, as shown in Figure 4. It is important not to do any absorbing or merging until the analysis is in the refinement phase, because the absorption means that the former episode no longer exists for reuse, and an episode created in anticipation of future reuse should be retained until it is clear whether that reuse will take place.

7. Discussion and Future Work

The goal-driven analysis required approximately 21 analyst-hours to analyze 13 use cases, or roughly a quarter of the initial SRS. In contrast, the ISA analysis required approximately 41 analyst-hours to analyze the entire set of 52 use cases of the initial SRS. ISA, supported by SMaRT, guided and assisted the researchers so that the analysis required approximately half as much effort as the goal driven analysis would have for the entire SRS.

The results of the goal-driven analysis are characterized as detailed and low-level. In contrast, the results of the ISA analysis are larger-scale and comparatively high-level. We believe that such high-level results are more valuable, especially in the early stages of requirements engineering work.

Traceability was both a priority and a challenge throughout this investigation. We have begun to provide support for traceability by providing appropriate tool support for scenario management in SMaRT [3].

In this paper, we demonstrate that using SMaRT provides a more consistent set of requirements, minimizing the practitioners' efforts. We have developed scenario management strategies [3] in an effort to address the challenges discussed by Weidenhaupt *et al.* [23]. Our scenario management strategies support evolution by employing shared scenario elements to identify and maintain common episodes among scenarios. Measures are used to quantify the similarity between scenarios, serving as heuristics that provide process guidance to practitioners in finding, for example, duplicate scenarios, scenarios needing further elaboration or those that may have been previously overlooked. Even though SMaRT is still under development, its automation supported this case study with great success.

8. Acknowledgements

The authors wish to thank Aldo Dagnino of ABB for his cooperation and support of this study, and to thank Paul Zaremba, William Stufflebeam, and Roksolana Antonyuk for their work on SMaRT. This research was was funded by NSF CAREER #530195.

References

[1] Software requirements specification for Euronet v.2. Asea Brown Boveri – Electric Systems Technology Institute, Mar. 1999. Confidential.

[2] T. A. Alspaugh. *Scenario networks and formalization for scenario management*. Ph.D. Thesis, North Carolina State University, Raleigh, NC, Sept. 2002.

[3] T. A. Alspaugh, A. I. Antón, T. Barnes, and B. W. Mott. An integrated scenario management strategy. In *Fourth IEEE International Symposium on Requirements Engineering (RE'99)*, pages 142–149, June 1999.

[4] A. Antón, E. Liang, and R. Rodenstein. A web-based requirements analysis tool. In *5th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 238–243, June 1996.

[5] A. I. Antón. Goal-based requirements analysis. In *Second International Conference on Requirements Engineering (ICRE'96*, pages 136–144, 1996.

[6] A. I. Antón, R. A. Carter, A. Dagnino, J. H. Dempster, and D. F. Siege. Deriving goals from a use-case based requirements specification. *Requirements Engineering Journal*, 6(1):63–73, 2001.

[7] A. I. Antón, M. McCracken, and C. Potts. Goal decomposition and scenario analysis in business process reengineering. In *Proceedings of the 6th International Conference on Advanced Information Systems Engineering (CAiSE'94)*, pages 94–104, 1994.

[8] A. I. Antón and C. Potts. A representational framework for scenarios of systems use. *Requirements Engineering Journal*, 3(3–4):219–241, Dec. 1998.

[9] A. I. Antón and C. Potts. The use of goals to surface requirements for evolving systems. In *Proceedings of the 1998 International Conference on Software Engineering (ICSE'98)*, pages 157–166, Apr. 1998.

[10] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, Reading, Massachusetts, USA, 1999.

[11] K. K. Breitman and J. C. S. do Prado Leite. A framework for scenario evolution. In *Third International Conference on Requirements Engineering (ICRE'98)*, pages 214–223, 1998.

[12] A. Dardenne, A. v. Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1–2):3–50, Apr. 1993.

[13] S. Easterbrook and B. Nuseibeh. Managing inconsistencies in an evolving specification. In *Second IEEE International Symposium on Requirements Engineering (RE'95)*, pages 48–55, Mar. 1995.

[14] I. Jacobson, M. Christerson, P. Jonsson, and G. Övergaard. *Object-Oriented Software Engineering: A Use Case Driven Approach*. ACM Press, 1992.

[15] M. Jarke, X. T. Bui, and J. M. Carroll. Scenario management: An interdisciplinary approach. *Requirements Engineering Journal*, 3(3–4):155–173, 1998.

[16] N. A. M. Maiden, S. Minocha, K. Manning, and M. Ryan. CREWS-SAVRE: Systematic scenario generation and use. In *Proceedings: 3rd International Conference on Requirements Engineering*, pages 148–155, 1998.

[17] B. Nuseibeh, J. Kramer, and A. Finkelstein. A framework for expressing the relationships between multiple views in requirements specification. *IEEE Transactions on Software Engineering*, 20(10):760–773, Oct. 1994.

[18] C. Potts. Using schematic scenarios to understand user needs. In *Proc. ACM Symposium on Designing Interactive Systems: Processes, Practices and Techniques (DIS'95)*, Aug. 1995.

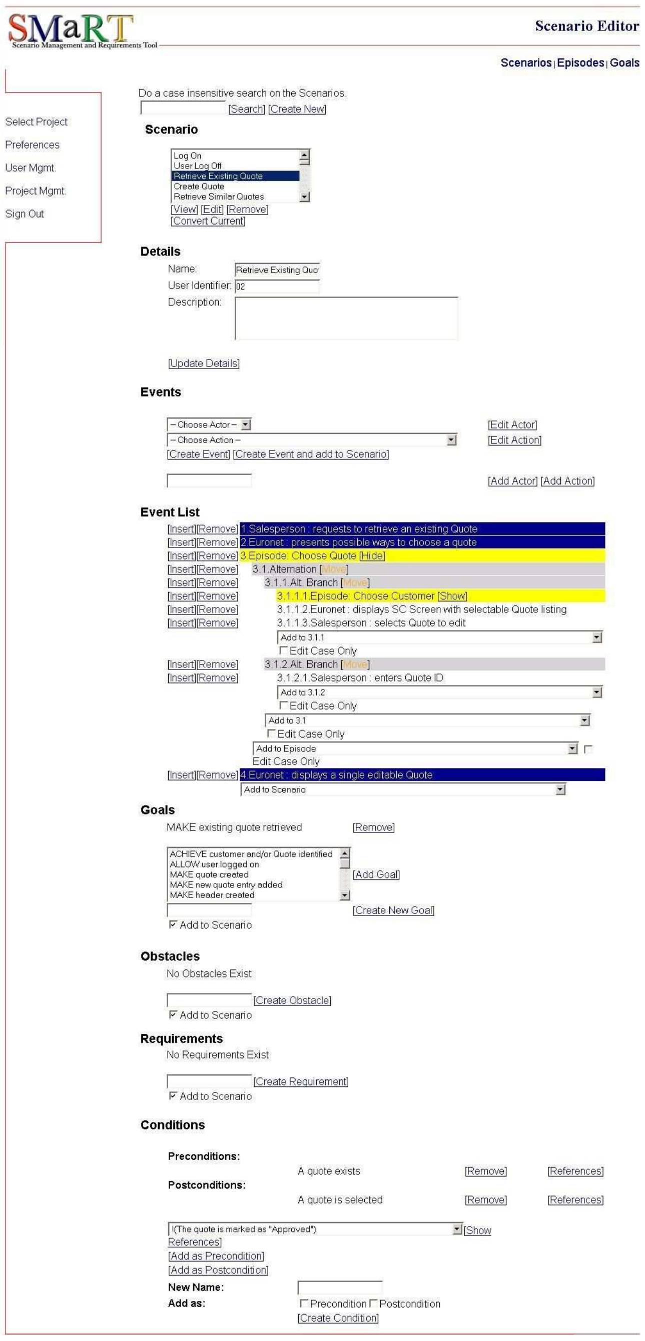
[19] C. Potts. ScenIC: A strategy for inquiry-driven requirements determination. In *Fourth IEEE International Symposium on Requirements Engineering (RE'99)*, pages 58–65, June 1999.

[20] C. Potts, K. Takahashi, and A. I. Antón. Inquiry–based requirements analysis. *IEEE Software*, 11(2):21–32, Mar. 1994.

[21] C. Rolland, K. Grosz, and R. Kla. Experience with goal-scenario coupling in requirements engineering. In *Fourth IEEE International Symposium on Requirements Engineering (RE'99)*, pages 74–83, June 1999.

[22] C. Rolland, C. Souveyet, and C. Ben Achour. Guiding goal modeling using scenarios. *IEEE Transactions on Software Engineering*, 24(12):1055–1071, Dec. 1998.

[23] K. Weidenhaupt, K. Pohl, M. Jarke, and P. Haumer. Scenarios in system development: Current practice. *IEEE Software*, 15(2):34–45, Mar./Apr. 1998.


SMaRT
Scenario Management and Requirements Tool

Scenario Editor

Scenarios | Episodes | Goals

Select Project
Preferences
User Mgmt.
Project Mgmt.
Sign Out

Do a case insensitive search on the Scenarios.
[Search] [Create New]

Scenario

Log On
User Log Off
Retrieve Existing Quote
Create Quote
Retrieve Similar Quotes

[View] [Edit] [Remove]
[Convert Current]

Details

Name: Retrieve Existing Quo
User Identifier: 02
Description:

[Update Details]

Events

-- Choose Actor -- [Edit Actor]
-- Choose Action -- [Edit Action]
[Create Event] [Create Event and add to Scenario]

[Add Actor] [Add Action]

Event List

[Insert][Remove] 1.Salesperson : requests to retrieve an existing Quote
[Insert][Remove] 2.Euronet : presents possible ways to choose a quote
[Insert][Remove] 3.Episode: Choose Quote [Hide]
[Insert][Remove] 3.1.Alternation [Move]
[Insert][Remove] 3.1.1.Alt. Branch [Move]
[Insert][Remove] 3.1.1.1.Episode: Choose Customer [Show]
[Insert][Remove] 3.1.1.2.Euronet : displays SC Screen with selectable Quote listing
[Insert][Remove] 3.1.1.3.Salesperson : selects Quote to edit
Add to 3.1.1
☐ Edit Case Only
[Insert][Remove] 3.1.2.Alt. Branch [Move]
[Insert][Remove] 3.1.2.1.Salesperson : enters Quote ID
Add to 3.1.2
☐ Edit Case Only
Add to 3.1
☐ Edit Case Only
Add to Episode ☐
Edit Case Only
[Insert][Remove] 4.Euronet : displays a single editable Quote
Add to Scenario

Goals

MAKE existing quote retrieved [Remove]

ACHIEVE customer and/or Quote identified
ALLOW user logged on
MAKE quote created
MAKE new quote entry added
MAKE header created
[Add Goal]
[Create New Goal]
☑ Add to Scenario

Obstacles

No Obstacles Exist
[Create Obstacle]
☑ Add to Scenario

Requirements

No Requirements Exist
[Create Requirement]
☑ Add to Scenario

Conditions

Preconditions:			
	A quote exists	[Remove]	[References]
Postconditions:			
	A quote is selected	[Remove]	[References]

!(The quote is marked as "Approved") [Show References]
[Add as Precondition]
[Add as Postcondition]
New Name:
Add as: ☐ Precondition ☐ Postcondition
[Create Condition]

Figure 5. Screen shot of the SMaRT scenario editor screen