A REQUIREMENTS ENGINEERING TESTBED: CONCEPT, STATUS AND FIRST RESULTS

William E. Rzepka
Rome Air Development Center
Command and Control Directorate
Software Engineering Branch
Griffiss Air Force Base, NY 13441-5700

ABSTRACT

This paper describes a Requirements Engineering Testbed currently being developed and used by Rome Air Development Center. The goals of the testbed are to support the research, development, application and evaluation of requirements engineering methods and tools. The testbed is based on a new process model for requirements engineering activities which provides a detailed description of the fundamental activities occurring during requirements engineering and their relationships to design activities. The testbed's current implementation includes a requirements analysis method and two requirements validation techniques based on rapid prototyping and simulation. Each of these tools is described from the perspective of the user and the status of the testbed is presented. Early versions of the prototyping tools have been applied in two case studies which indicate that significant productivity gains can be realized from using testbed prototyping techniques to validate user interface requirements.

Introduction

Requirements are precise statements of need intended to convey some understanding about a desired result. Regardless of whether requirements specify needs for airline reservation systems, process control software or automobiles, they describe external, user visible characteristics as opposed to internal construction. Requirements also specify the constraints placed on what is needed. Performance, reliability, safety, cost and schedule are typical constraints. In general, requirements engineering may be thought of as the activity of forming a model of what is needed and of simultaneously stating the problem to be solved.

Numerous workers in both the system and software requirements areas have indicated the importance of careful requirements analysis and validation activities. One classic study [BOEH76] indicated that savings in the order of almost two orders of magnitude could be realized if requirement errors were detected during the requirements analysis phase of system development as opposed to correcting those same errors when found in the implemented system's software.

RADC has been conducting a research and development program in requirements engineering since 1980. Its goals are to develop methodologies for problem analysis, specification and validation. To support these goals tools are being developed to insure that stating the problem, changing it and insuring its consistency are more productive. All developments are aimed at getting such techniques and tools into the hands of mission users and system acquisition engineers so that they can better understand and state their needs.

Requirements Engineering Testbed

In order to know that program efforts are proceeding toward these goals, it is necessary to try developed techniques and tools on realistic problems and use these experiences to improve the tools and our understanding of the processes involved. This constraint naturally leads to the concept of a testbed for requirements engineering.

The goals of the RADC Requirements Engineering Testbed are to carry-on research and development of requirements engineering methods and tools, to apply the R&D results to the analysis, specification and validation of requirements for large-scale command and control systems and to assess the effectiveness of the processes prescribed by the methods and the quality of the products produced by the tools.

Our approach investigates both established techniques and emerging technologies. Existing techniques which have already been installed in the testbed include stimulus-response networks and data flow analyzers. New technologies being studied and developed include very high level executable specification languages, reusable specifications, requirements process and product metrics and rapid prototyping.

RADC Requirements Panel

With the above broad goals and general approach mechanism in mind, RADC formed a panel and chartered it to develop a ten year research and development plan which would identify specific efforts to be pursued in developing the technologies needed to support the testbed.
To establish a context and direction for the R&D plan, the panel first developed a process model for requirements engineering activities, and with this understanding the panel developed a two track R&D plan. An "evolutionary track" laid out a course of action which would utilize techniques and tools currently under development by RADC, integrate them into a requirements engineering workstation environment by 1991 and then evolve this environment with additional conventional capabilities through the mid-1990s. The other "formal language track" was to develop a language which would capture the semantics of requirements in a knowledge base and exploit this information to support advanced searching, consistency preservation and deduction schemes. [KONR86a] fully documents the panel's work.

The Requirements Engineering Process Model

The panel developed a process model (see Figure 1) which goes beyond existing models in its characterization of requirements engineering. It provides a detailed description of the fundamental activities occurring during requirements engineering, but avoids prescribing specific methods for accomplishing them. It does not separate the analysis, specification and validation of requirements into discrete operations, but instead elaborates on how analysis and specification are used together from the very outset of requirements engineering, on how static analysis and validation cause the entire process to be highly iterative and on how validation is related to design activities, naturally occurring before the commitment to detailed design and implementation activities. The model recognizes three basic activities being performed: eliciting requirements from the various individual sources of a system's requirements, specifying them and insuring their consistency; insuring that the needs of all users are self-consistent and feasible; and validating that the requirements so derived are an accurate reflection of user needs.

![Figure 1 - Requirements Engineering Process Model](image-url)
The result of these efforts are called "goals" and are the first documented statement of user needs. They typically are high-level in nature, specific to the relevant problem domain and stated in the user's own terminology. Goals are still relative to the user viewpoint which they represent; they may still be infeasible; and they probably conflict with the goals represented by other viewpoints. However, within each viewpoint goals are self-consistent and unambiguous as a result of several iterative analyses using static consistency checking tools.

The second part of the requirements analysis step involves integrating the goals across the various user viewpoints thereby resolving the conflicts among them. User scenarios are typically relied on to identify the activities and data which are the basis for integrating goals which involve several viewpoints. The ensuing integration analysis demands that all involved viewpoints agree on which viewpoint will accept responsibility for an activity and which will provide the data necessary to carry it out. Consistency checking is an important part of this process. Finally, the non-functional requirements, such as performance and reliability issues must also be considered and stated. The resulting requirements then, are an unambiguous, consistent, feasible subset of the goals which describe a number of possible solutions. The question now remaining is whether or not they are what the user actually intended.

The requirements engineering process model provides several options for making this determination. One is dynamic analysis of the requirements themselves. The intent is to "animate" the requirements by providing sufficient functionality to enable the requirements engineer to perform a "walkthrough" of requirements with the user. The walkthrough is an exercise conducive to exposing misunderstandings, omissions and interface problems. The dynamic analysis utilizes the activities derived during the second step of the requirements analysis and activates them with actual data which is representative of classes of system input. An activity is "animated" by utilizing algebraic and logical functions to represent the activity or by actually programming the data transformation which it represents. The walkthrough typically produces an interchange between the user and requirements engineer which results in elicitation of valuable user commentary and the validation of requirements.

Most of the validation activities prescribed by the process model do involve design of critical system components. The resulting partial system designs are then made the subject of either analytical studies or of prototyping. The analyses focus on mathematical modeling of system performance and reliability. Various models [BOEH81] may also be utilized to study the cost and schedule risks associated with a particular system development. Prototyping involves the construction of user interfaces, actual system functions and discrete event simulation models. Behaviors exhibited by the resulting user interface and functional prototypes are validated through actual usage by the user. Performance modeling results are usually analyzed by individuals skilled in this discipline.

Regardless of which path through the model is followed or if all are pursued simultaneously, the outcome is a new appreciation of user needs which invariably leads to a re-evaluation of the requirements as previously stated. This activity in the process model leads to a new set of requirements which once again is the subject of analysis and validation activities. This process is repeated until users, requirements engineers and designers are satisfied that the requirements and partial designs have reached a maturity which permits development of the target system to proceed.

Comparison with Existing Life Cycle Models

Direct comparison of the requirements engineering process model (REPM) with existing life cycle models is inappropriate because, as explained above, the REPM is not a full life cycle model. The intent of the REPM is to detail those requirement and design activities related to analysis, specification and validation. Since REPM is restricted to a portion of the life cycle it does not address management issues, such as resource management, configuration control and traceability, which span multiple development phases. Comparison then reduces to an examination of the degree to which the REPM is compatible with other life cycle models and whether the REPM could be used to complement or replace the requirements and/or design steps of these models in some way.
The waterfall model [ROYC70] does not detail all the technical activities of its requirements phase. Consequently, the REPM is compatible in the sense that if it is considered as a "black box", it too produces a specification product as an indication that the requirements phase has been completed. The waterfall model does incorporate a "build it twice" step running in parallel with requirements analysis and design, and it also prescribes feedback between successive development phases. The REPM's validation activity, which includes the prototyping of system functions, could be viewed as a build of the entire system. However, the intent is to actually construct only critical system elements whose risk level warrants the time and resources required for prototyping. Feedback between the REPM's validation and specification activities is explicit, while feedback between requirements activities is implicit in the consistency checking processes involved in transforming wish lists to goals and goals to requirements.

The "design driven" variation of the waterfall model recognizes that feedback from later phases necessitates changes in some predecessor phase, and so there must be some overlap between successive stages of the waterfall model in order to facilitate that change. The REMP has activities which are initially performed sequentially. However, there is much parallelism and overlap implied by the iterative nature of the consistency checking activities of its first two steps and similarly in its validation activity. The REMP would be a particularly useful refinement of the early phases of the design driven waterfall model, since it could elaborate on unknowns early in the development process, especially for large systems. The inability to expose risk elements in large system developments is principally why this model is usually used on small projects.

The rapid prototyping model [SEEW82] abandons the single, monolithic system development strategy and instead prescribes multiple, incrementally delivered developments, where each delivery remedies a previous build's problems and introduces new functionality for user scrutiny. The REMP is an ideal refinement of this model because it is capable of insuring that the requirements for an incremental build are a valid reflection of user needs before the increment is designed and implemented.

The operational model [ZAVE84] uses processing abstractions to develop a problem-oriented specification of a system's requirements rather than structural abstractions to specify their decomposition. As a result its specifications contain sufficient design information to permit immediate interpretation as a prototype. Producing a system is the result of a series of implementation-oriented transformations performed on the specification. Although requirements analysis activities are not an integral part of the operational model, specification and validation are. Consequently, the REPM would complement the operational model, since it elaborates on the activities which develop the requirements specification, although it certainly stops short of being a requirements analysis method. The REPM also prescribes several approaches to validation, one of which is dynamic analysis of the requirements. The requirement necessary to do this is exactly the information contained in the operational specification. Since the REPM does not dictate particular specification or dynamic analysis mechanisms, the operational specification and its interpretation could serve as a basis for REPM specification and validation activities, respectively.

The knowledge based model [BALZ83] integrates an operational specification and transformation based software development process with expert knowledge on software engineering and the target application domain. Because the specification is operational, it can be executed as a prototype. This specification is evolved into an efficient implementation through a series of human-assisted, mechanical optimizations which successively introduce additional implementation and run-time detail. Expert knowledge is provided to the software developer by the development environment's knowledge bases and the tools which manipulate them. Like the operational model, the knowledge based model would also benefit from an explicit requirements analysis activity as provided by the REPM, and since the knowledge based model is also based on specification and validation activities, the two models are compatible with respect to these two activities.

The two-leg model of Lehman, Stenning and Turski [LEHM84] views the software process as a series of representation transformations from user needs to implemented system. Each transformation step refines the current representation closer to the implemented system. First, user needs are captured and transformed into a formal (requirements) specification through the development of an appropriate set of abstractions embodied in a domain model. Then, this representation is progressed to an implemented system through a sequence of linguistic transformations of successive representations. Each activity in this sequence takes as input a representation of the system (e.g., system requirements specification) in a "base" linguistic system, which is capable of expressing the understanding of the system at its present stage of development. This representation is transformed into a "target" linguistic system (e.g., functional specification) by means of a canonical step, that is, the same type of design activities for each transformation. The canonical step involves the refinement of the representation through the addition of new concepts resulting from the creative aspects of design, through formal verification that the target design representation is consistent with the base design representation.
and through validation that the target design satisfies the domain model of user needs.

The REPM is compatible with the two-leg model since it includes both the domain modeling activities leading to the first formal system specification and the specification and validation activities of the canonical step. However, the compatibility is only partial and must be qualified in at least three important respects. First, the two-leg model is intended to take the users' needs to formal specification and then to system implementation. The REPM deals with only the first leg of the two-leg model, through validation of the initial system specification. Second, the REPM prescribes the activities progressing from user needs to a first formal specification of system requirements as a two step process (wish list - goals and goals - requirements). This partitioning recognizes the two fundamental analysis activities which occur: defining consistent sets of user needs extracted from individual classes of users of the system; and making each class of user needs consistent with all other need classes. The two leg model does not place such restrictions on this part of its process. Third, the REPM does not require an explicit formal verification activity. In the two-leg model's canonical step this is necessary to insure that the design step just completed is a correct progression to the eventual system realization. An equivalent activity is not required by the REPM because the design is intended only to validate requirements and not necessarily to be refined into the target system. Some "designs" may have little or nothing in common with the target system, as would be the case of a discrete event simulation. The second and third steps various risk analyses are performed (e.g., survey, cost model, past project study, literature search) and prototyping may be undertaken. During the fourth step typical life cycle development activities are pursued (e.g., requirements validation, implementation, unit testing) and products are produced (e.g., requirements specification, software product design, code). The spiral model is very general and is intended to include other models as special cases. This would seem to include the REPM because during a cycle whose objective is requirements development, the spiral model would identify alternatives (goals to the REPM), perform risk analysis to determine the requirements and then validate the requirements. Prototyping could be involved during either the risk analyses or the requirements validation processes. Consequently there seems to be no significant incompatibility between these two models for the portion of the life cycle which both address.

**Testbed Implementation - Methods and Tools**

RADC has developed and enhanced three requirements engineering methods and tools which support a subset of testbed activities including requirements analysis, specification, prototyping and validation. Each tool will be briefly described below.

**Controlled Requirements Expression (CORE)**

CORE [MULL79] is a requirements definition and analysis method whose procedures explicitly support the notion that requirements originate from several, diverse viewpoints of how the same needed capability will be used. CORE organizes these viewpoints as a functional hierarchy. CORE procedures prescribe data flow techniques to elaborate the viewpoints. Logical data relationships are described using a notation similar to that used in Jackson System Design [JACK83]. Self-consistency of each viewpoint is a goal of the CORE method and is supported by specific checking procedures. Transaction analysis is used to resolve conflicts among the various viewpoints. Constraints analysis (e.g., performance, reliability) is also an explicit step in the CORE method.

The Analyst [STEP85] is an expert system for requirements analysis and specification using the CORE method. It provides documentation and analysis tools which support and enforce the viewpoint hierarchy and data flow rules of the CORE method. Capabilities include diagram construction, management and consistency checking in support of CORE analysis activities and word processor support for textually oriented aspects, e.g., describing project objectives. Analyst provides an intelligent "help facility" [KRAM87a] which understands project progress in terms of CORE objects (e.g., diagrams) and strategically guides the user toward project completion in accordance with method rules. Analyst also implements dynamic analysis of requirements [KRAM87b] providing tools which support the requirements engineer in conducting a user "walkthrough" of typical system transactions.

**Rapid Prototyping System (RPS)**

The RPS [RZEP86] is a collection of tools which
supports the activities of building, executing and analyzing prototypes of computer based systems for the purpose of improving understanding of the requirements for those systems. A system interface provides users with menu and template access to its capabilities for prototyping critical system requirements.

A graphical modeling capability enables the development of user interface prototypes consisting of individual displays drawn with a graphical editor and linked into a "slide show" for examination and validation by the user. A dynamic graphic modeling component enables the representation of event driven, moving object displays. A performance prediction capability enables the prototype developer to construct discrete event simulation models to assess the following system performance drivers: computer hardware and software configuration; operator/analyst terminal work flows; allocation of system functions to system resources; and computer network configuration. A data modeling component enables the prototype developer to construct different logical views of a data base and to assess their potential for satisfying user query/update needs and system response requirements.

**Very High Level Prototyping Language (Proto)**

Proto [KONR87b] is a prototyping language, a set of tools and a methodology for functional prototyping, in which the logical capabilities of a proposed system are modeled and exercised, permitting validation of the requirements which pertain to system interfaces and data bases. In Proto, functional specifications of a proposed system are built and exercised. A Proto execution exhibits what data is available, what operations can be performed on that data, and what inconsistencies may exist in a specification.

The Proto style of specification is one in which a proposed system is analyzed and specified in terms of transactions, with objects representing the data elements shared between transactions. This style is partially transaction-oriented and partially object-oriented. The primary orientation of the description is a stimulus-response style, where system outputs are related to system inputs. These output-from-input relationships are shown as data flows which link activities. The interactions between data flows are shown as objects, in the object oriented programming sense, typically being shared data elements. The activities which comprise transactions are described in terms of mathematical and logical functions or they can be programmed directly.

**Testbed Status - R & D**

RADC is selectively developing the technologies needed to realize the testbed's evolutionary and formal language tracks. The first phase objective was to develop sufficient capabilities to permit analysis and validation of realistic requirements problems. As a result R&D efforts to develop, or in one case [KRAM88] enhance, the methods/tools described above were undertaken. All have been completed and were delivered by the fall of 1988.

The next phase of testbed development will be an integration of the three existing tools into a single unified methodology for requirements analysis, specification and validation supported by the three tool capabilities residing on the same hardware platform. The major effort here will be the design and implementation of a common data repository and user interface for all three tools. The common data base will store all non-local tool data, thereby allowing direct sharing of information among all tools. The user interface will provide uniform, object oriented access mechanisms to the tools. It is planned to have this capability installed in the testbed by mid-1991.

The testbed's formal language track is just getting underway. An initial effort was begun in the fall of 1987. Its objectives are to determine the suitability of existing functional programming languages, such as Prolog, for modeling the conceptual requirements of complex computer information systems and for analyzing the resulting knowledge bases to determine the impact of various real world scenarios on the system model.

**Testbed Status - Applications**

Three in-house application projects involving initial versions of the CORE/Analyst and RPS tools have been undertaken since the summer of 1986.

The first project used the Analyst tool to support an application of the CORE method to an analysis and validation of the system requirements of the RPS. The intent was to gain a better understanding of CORE, the Analyst tool and the rationale for the RPS system architecture.

The RPS analysis proved extremely valuable in three respects, all related to the motivation for the application aspect of the testbed. The first was, of course, an improved understanding of the CORE method. The method worked well, but this exercise pointed out the need for additional examination of method procedures dealing with non-functional requirements, especially performance.

The second result was an increased understanding of the Analyst tool and a number of its deficiencies. These were pointed out to its developers and the problems were resolved in a later version of the tool.

The third result of this effort was a much better understanding of the RPS system requirements and architecture. The CORE analysis resulted in a data flow representation of this understanding.
and this information was used during a preliminary design review (PDR) for that work. The use of this information resulted in many changes to the contractor's system representations and several significant changes to the architecture itself.

Two other projects were undertaken to apply the user interface prototyping capabilities of the RPS. The first prototyped the displays associated with a hypothetical air defense battle management system of the future. It was intended to be the vehicle for learning the user interface prototyping tools. It resulted in 33 different, interconnected displays and took about one person-month to complete. A typical display is shown in Figure 2. The second project prototyped the displays of a system that is currently being developed as an upgrade to the NORAD Air Defense Operations Center. This project produced 16 different, interconnected displays and took 6 person-weeks to complete. This prototype was actually validated by doing a walk-through of the individual displays and the procedures for inviting them with representatives of the mission user.

Testbed Status - Evaluation

To evaluate the productivity of the testbed processes involved in the development of the two user interface prototypes using the RPS tools, simple productivity measurements based on source lines of code (SLOC) were used. The actual time spent by the requirements engineers on the design and implementation of these prototypes was carefully recorded during the course of the projects. The resulting prototypes consisted of two types of software. The first was the graphics software which generated the displays. Since this software is generated using a graphics editor and takes the form of data files which drive the same graphics primitives regardless of which project is being done, it was decided not to include this "reused" software in the productivity measurements. The second type of software is the control code which is responsible for

![Diagram](image-url)

Fig 2. User Interface Display
interconnecting the displays. This is unique to the prototype and can easily be measured. It is the basis for the productivity data which follow.

The actual productivity factors were computed by comparing the number of person-hours actually spent in developing a prototype with the person-hours computed by dividing the number of SLOCs generated by the prototyping tools by an informal standard of productivity (50 SLOCs per person-day) for development of unstructured, undocumented software. This approach provides a very conservative estimate of the productivity of this rapid prototyping process.

For the first air defense prototype it was found that use of the prototyping tools resulted in the project being completed 3.5 times faster than could be expected using conventional programming techniques. The NORAD prototype took 6 times fewer person-hours than conventional techniques. This difference is explained in terms of the learning curve which had to be overcome during the first project. Although these two points establish a positive productivity curve, until additional experiments validate them they should be interpreted only as a promising trend and not as a definitive statement on the value of this prototyping approach to requirements validation.

Conclusions

A testbed for the R&D, application and evaluation of requirements engineering methods and tools has been established at RADC. The testbed is based on a process model of requirements engineering. Research and development activities are following a 10 year plan directed toward testbed needs. Recent R&D activities have led to the development and installation into the testbed of a requirements analysis tool and two requirements validation tools based on rapid prototyping. Two of the tools have been applied in rapid prototyping. Two of the tools have been applied in requirements analysis and validation activities which have led to a better understanding of and design improvements to the tools, while validating requirements for the user interface prototypes of realistic systems. Evaluation activities have concentrated on the measurement of productivity associated with the development of user interface prototypes using a set of rapid prototyping tools. Early results indicate a positive productivity trend with as much as a factor of six reduction in development time when compared to conventional programming techniques.

Future testbed plans call for enhancements to the existing capabilities through the mid-1990s and simultaneous development of techniques based on formal representation of requirements semantics.

Acknowledgments

The author is grateful to the members of the RADC Requirements Panel for their contributions to the requirements engineering process model and the research and development plan. The panel included Bob Balzer, Mike Konrad, "Ram" Ramamoorthy, Win Royce, Steve Sherman, Leon Stucki, Terry Welch and Ray Yeh. The author also thanks the members of the System Definition Technology Group at RADC, Carla Burns, Liz Kean, Dave Marks and James Sidoran, whose efforts as first users of the Requirements Engineering Testbed produced the prototypes and productivity measurements described in this report.

REFERENCES


[KRAM87b] Kramer, J. and Ng, K.; "Animation of Requirements Specifications"; Imperial College Research Report Doc. 87/17, Imperial College of Science and Technology, University of London, London UK (October 1987).


