A REQUIREMENTS ENGINEERING ENVIRONMENT FOR ANALYZING & VALIDATING SOFTWARE AND SYSTEM REQUIREMENTS

Carla Burns
Rome Laboratory
Griffiss AFB, Rome NY 13341-5700
bumsc@rl.af.mil

Abstract
Software and system prototyping are approaches which may be used early on in the lifecycle to identify deficiencies and errors in requirements. Since 1985, Rome Laboratory has been developing tools in support of software and system prototyping. The tools provide support for prototyping user interfaces, functional algorithms and software/hardware mappings for parallel and distributed architectures. The prototyping tools have recently been integrated into a Requirements Engineering Environment (REE) which houses tools for requirements elicitation, analysis, specification and validation. The REE utilizes a common object-oriented database to store tool data and share information between tools. This paper emphasizes the capabilities of the REE prototyping tools along with their corresponding methodologies. Areas where data sharing occurs between the REE tools is also discussed.

1.0 Introduction
Software and system prototyping are approaches which may be used early on in the lifecycle to identify errors in requirements. Software and system prototyping can refer to the quick development of executable software which supports analyzing and validating critical attributes of system components such as the user interface, complex algorithms or performance. A prototype does not represent the complete system, but allows these vital aspects of the system to be analyzed.

Rome Laboratory (RL) has developed tools which support software and system prototyping. Emphasis was being placed on developing the tools such that they may be used by non-programmers. It is crucial that this goal was met since requirements analysts are not necessarily programmers. Ideally such individuals possess knowledge of various requirements methods as well as the application being prototyped. The prototypes generated by the RL-developed tools are termed “throwaway prototypes” since they do not evolve into the final product.

However, throwaway prototypes are useful for analyzing and validating requirements since they can serve as mechanisms for interacting with end users. Once a prototype is complete the end users of the system can be brought in and asked for an assessment. The prototype and system/software requirements can be modified to reflect the end user's comments. This process of modifying the prototype and obtaining feedback is an iterative process which allows requirements to be validated in the early phases of the lifecycle. Consequently, such an approach to development should not only reduce the number of requirements errors but also result in more satisfied end users.

2.0 The Requirements Engineering Environment (REE)

Rome Laboratory has recently integrated its prototyping tools into a Requirements Engineering Environment (REE). The REE utilizes an object-oriented database to store and share tool information. There is also a common user interface which provides uniform, object-oriented access to all tools. The REE is currently comprised of three tools: The Rapid Interface Prototyping System (RIP), Proto and The Analyst.

The REE architecture tightly integrates the RIP and Proto tools on a Sun/Unix platform. The Macintosh based Analyst tool is loosely coupled with REE via a file transfer facility. A conceptual view of the REE is portrayed in Figure 1. Each of the three REE tools is discussed below.

2.1 Rapid Interface Prototyping System (RIP)

The Rapid Interface Prototyping System (RIP) [1] contains a collection of tools to prototype user interfaces. A user interface prototype can model a system's screen contents and layout as well as execute its associated functions for the purpose of validating its user interface requirements. One of the strengths of RIP is that its users need not be computer scientists or programmers. RIP utilizes an Apple Macintosh "MacDraw-like" graphics editor to draw the objects required for the prototype and menus to specify the activities which will occur during the prototype execution, such as removing and displaying objects. There is also a world database facility which allows the requirements analyst to graphically extract
2.1.1 Designing a RIP User Interface Prototype

Prior to implementing a user interface prototype, the requirements for the user interface are obtained. Such baseline requirements may be in pictorial and/or textual format. A scenario which represents the functionality of the baseline requirements depicted through a sequence of events is created. The scenario should reflect how the end users would utilize the system since the scenario drives the execution of the user interface prototype and is the mechanism through which the requirements analyst communicates with the end users. Based on the user interface requirements and the scenario defined, all of the graphic objects required for the prototype are then identified along with how the end users will interact with the system. The process of identifying how the end users will interact with the system determines what object on the display the end user selects to cause a scenario event to occur.

2.1.2 Creating Static Displays

Once all of the design activities for the user interface prototype are complete the requirements analyst can implement the user interface prototype using RIP. Some systems require a capability to view a geographic location of the world. RIP provides an interface to the Defense Mapping Agency's DMA's World Database I. This interface allows the requirements analyst to graphically extract a geographical area out of the database and incorporate it into a map which can be used in the prototype. Using menus, the requirements analyst specifies the desired map's projection, granularity and its associated features which may include coastlines, international boundaries and rivers.

RIP is then used to create the numerous graphical objects required for the prototype. The graphical objects represent the real-world objects as they would be seen on the display. RIP has three categories of graphical objects: graphics, widgets and bitmap images. Graphic objects include maps, rectangles, circles and text. RIP provides an Apple Macintosh "MacDraw-like" facility to graphically create such objects. RIP widgets are predefined objects which provide a subset of OpenLook controls [2] and are typically used to trigger off scenario events when selected by the end users. Examples of RIP widgets include buttons, lists and choice boxes. RIP also provides an interface to a bitmap editor. Consequently, bitmap images can be created and directly imported into RIP.

During the process of creating the graphical objects required for the prototype, the requirements analyst partitions the objects into various overlays. An overlay is a transparent layer that contains one or many RIP graphical objects. Overlays provide a means of logically separating graphical objects for manipulation during the prototype execution. For instance, an interface requiring various categories of objects (e.g. airplanes, bases, sensors) might have each category in a different overlay so that the category can be displayed and undisplayed separately from the other categories. Any number of overlays can be displayed at a given time.

Figure 2 provides an example of a real-world Command, Control, Communications and Intelligence (C3I) display which was generated using RIP [3]. This C3I display is used to monitor the North American region of the world. The RIP tool provides the facilities to create and manage all of the graphical objects required for sophisticated displays.
2.1.3 Defining the Functionality of a RIP Prototype

Next, the functionality which is to occur during the scenario or prototype execution is defined. The types of activities which can occur include removing objects, displaying objects, ringing alarms, flashing objects and prompting the user for input. Any Unix command or program may also be executed. To define the functionality for the prototype, the requirements analyst graphically identifies the target object which is to execute a scenario event when selected by the end user. Any RIP graphical object (graphics, widgets and images) can serve as a target object. Menus are then used to specify the various scenario events which will occur when the target object is selected during prototype execution. More than one activity can be associated with a target object. Typically to transition from one state of the scenario to the next involves removing many objects and placing additional objects on the display.

For example, one of the requirements for the user interface depicted in Figure 2 is that the operator have the ability to zoom in on any region of North America. The operator zooms in on the Alaskan region of the world by selecting the "AK" textual object under the Air Defense Summary title denoted in Figure 2. A screen representing the Alaskan region is then displayed as illustrated in Figure 3. Notice that many objects have been removed and additional ones added to make the transition.

2.1.4 Creating Dynamic Displays

In some cases, the user interface may require objects to be automatically updated on the display as opposed to being manipulated by end user interaction. RIP also supports the creation of real-time dynamic displays. Real-time dynamic displays are based on events occurring over time. For example, objects can automatically move, disappear, change color or flash by specifying the time at which such events are to occur. To make an object move, the requirements analyst graphically creates an animation path for the object to follow. A menu is then used to specify the object's start time and speed. Other dynamic changes to objects such as flashing and changing color are specified graphically using time events. A time event captures the date, time and type of activity which is to occur at the given date and time. RIP provides a simulator which executes object animation paths and time events. The end user is able to interact with the prototype while the RIP simulator is executing dynamic activities.

2.1.5 Executing the RIP Prototype

When the prototype is complete the final users of the system are brought in and asked for an assessment. The prototype is executed through user interaction. That is, the end user selects an object on the display to trigger off a scenario event. If dynamic activities are defined, then the simulator is executed. Typical comments that end users may provide are that objects need to be a different color, objects need to be relocated on the display or additional functionality needs to be built into the user interface. Using RIP, the prototype can quickly be updated to reflect the end users comments. For instance, graphical attributes such as color and location can be changed in front of end users in a matter of minutes. The RIP tool provides graphical facilities for creating, modifying and validating prototypes. It is even envisioned that the end users of a system can very quickly learn how to use RIP and develop displays.
2.2 Proto

Proto [4,5] supports the prototyping of functional requirements. A functional prototype can model and execute the system's operational capabilities to check the completeness and correctness of information as it flows through the system. This type of prototyping is typically performed on critical or complex applications. The Proto tool not only supports the functional prototyping of requirements for single processor environments but provides facilities for prototyping requirements targeted to parallel or distributed architecture's as well. Course and medium-grained parallelism can be explicitly identified and validated by requirements analysts. This capability is unique amongst current Parallel computer aided software engineering (CASE) tools since it addresses high level architecture trade-offs and software to hardware mapping trade-offs during the requirements and early design stages of development [6].

The remainder of this section presents an overview of the Proto capabilities along with an associated methodology for using the tool. When performing functional prototyping the requirements analyst first obtains functional specifications for the application. Such functional specifications represent precise descriptions of what the system does.

2.2.1 Creating a Proto Software Model

Next, the requirements analyst uses Proto to graphically construct a software model (logical model) which captures the functions in the application being prototyped. During this process, the analyst creates multiple refined data-flow-like diagrams to identify the various functions in the algorithm and their corresponding relationships. An object-oriented editor is used to create all of the data objects required by the application. The details of the various functions of the requirements are captured in a high level data-flow-like language which was explicitly developed for the tool. In addition, a reusable library of functions is available to expedite model development. Collectively, the data-flow-like diagrams, data objects, language statements and library functions make up a Proto software model. Figure 4 represents a Proto software model.

2.2.2 Creating a Proto Hardware Model

If the software is targeted for a parallel or distributed architecture environment or the analyst wishes to identify and validate potential parallelism in the application being prototyped then a hardware model can be graphically created and associated with the software model.

The analyst uses an object-oriented approach to create a hardware model which consists of processors, buses, memories and their associated hardware connections. Each of these hardware primitives have associated properties which get input into templates by the analyst. Such parameters are specified in simulation time units and include processor execution speed, memory read/write access time and bus delay time. In Proto time is relative as opposed to absolute. However, relative time comparisons are useful for assessing various high level architecture trade-offs early in the lifecycle. The analyst can also attach utilization displays to any of the hardware components. Utilization displays are digital and analog gauges which dynamically display resource utilization during the prototype interpretation. Figure 5 provides an example of a Proto hardware model.

Figure 4 - A Proto Software Model.

Figure 5 - A Proto Hardware Model.

191
Proto provides already generated multiple-instruction, multiple-data (MIMD) parallel processing hardware configurations for the Encore Multimax and the Intel Hypercube. Using the hardware primitives identified above, the analyst can create various distributed and/or parallel processing architecture configurations.

2.2.3 Mapping the Software Model to the Hardware Model

By examining a Proto software model, the analyst can determine which software functions are potential candidates for parallel execution. The dataflow paradigm lends itself to identifying parallelism. Good candidates are those functions which do not rely on one another for data. Parallelism is expressed by allocating different software functions to different hardware components.

Once the hardware model is defined, the analyst can graphically associate the logical software functions with the physical hardware components or allow the system to automatically generate software-to-hardware mappings using heuristics. Such heuristics are based on load balancing, software connectivity and hardware routing algorithms. More than one such software-to-hardware mapping can be associated with a hardware model. Moreover, more than one hardware model can be associated with a functional prototype.

2.2.4 Interpreting a Proto Model

The prototype is executed using the Proto Interpreter facility. The Interpreter facility provides an animated, highly interactive approach to executing the functional prototype. Once interpretation commences, each of the functions in the software model highlight as they are being executed. The analyst can set breakpoints and view data values. It is also possible to step through each of the language statements for a single function one at a time. End users can provide feedback on the correctness and completeness of the application.

Statistics relating to the prototype performance are available after an interpretation session. The statistics available not only provide information for the entire prototype interpretation session but for each individual software function and hardware element as well. Measurements for the entire interpretation session include the total time the prototype executed and the number of simulation events that were interpreted. Statistics relating to the hardware elements include percentage of time the hardware element was busy/idle and the number of times it was used.

If a parallel or distributed application is being prototyped the analyst may wish to execute the software numerous times using various hardware models and/or mappings. The various performance statistics which are available after an interpretation session aid the analyst in determining the most suitable software-to-hardware mapping and hardware architecture for a given application.

Finally, the Interpreter facility also provides the mechanism for interacting with final users. Its animation, data viewing and statistics capabilities provide a complementary environment for validating functional prototypes.

2.3 The Analyst

The Analyst [7], a proprietary tool developed by Systems Designers, Inc. and enhanced by Rome Laboratory, is a requirements elicitation and analysis tool which is based on the Controlled Requirements Expression (CORE) methodology [8]. One of the major strengths of the CORE methodology is that it supports the notion that requirements originate from several, possibly diverse viewpoints on how the same required capability will be used. CORE organizes these viewpoints as a functional hierarchy, where each viewpoint is further elaborated using data flow techniques. Logical data relationships are described using a notation similar to that used in the Jackson System Design [9]. 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.

The Analyst 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 such as describing project objectives and system constraints. Rome Laboratory has enhanced the Analyst by incorporating an expert system which provides intelligent "help facilities" to understand project progress in terms of CORE objects (e.g. diagrams and descriptions) and strategically guides its user towards project completion in accordance with method rules. Rome Laboratory has also enhanced the Analyst by providing support for automated requirements "walkthroughs". Such a capability supports the requirements engineer in conducting a "walkthrough" of system transactions with the end users to validate its behavior.

3.0 Data Sharing Between REE Tools

The REE's common database allows its tools to share data. The RIP and Proto tools are integrated such that there is two way data sharing between the tools. First, a Proto model can be used to drive a RIP user interface prototype. The Proto tool provides a set of predefined functions that facilitate the manipulation of
RIP objects. Such manipulations include removing and displaying objects, changing color of objects, setting data values and moving objects to new locations. No programming is required to connect a Proto model with a RIP user interface. The requirements analyst simply selects the RIP object which is to be manipulated by Proto, provides a name for the object in the REE database and references the named object in the desired Proto functions. The result is the prototyping of user interfaces reacting to real world stimuli.

The second type of data sharing between RIP and Proto allows data values specified in RIP to be read and manipulated by a Proto model. Consequently, the data required for a Proto model can be interactively entered using custom made, user friendly interfaces. Once again, non-programming techniques are used to specify the connection between the two tools.

The Analyst is also able to share its data with Proto. The Analyst tool is predominately a requirements solicitation and analysis tool with limited support for prototyping. The requirements analyst may decide that the requirements captured in the Analyst tool warrant further study using prototyping techniques. Since both of the methods supported by the Analyst and Proto tools are based on the data flow paradigm, their associated diagrams are similar. Consequently, the REE provides a facility which automatically converts the Analyst diagrams into Proto equivalent data flow graphs. The requirements analyst can then complete the Proto model without having to recreate the data flow graphs from scratch.

4.0 Conclusions and Future Work

The REE provides a graphically-oriented, non-programming suite of tools for requirements elicitation, specification and validation, including support for software and system prototyping. Such requirements techniques and automated support are aimed at reducing deficiencies and errors in requirements.

The current REE tools are based on conventional requirements methodologies. Future work at Rome Laboratory will focus on more formal approaches to requirements engineering which take advantage of mature artificial intelligence technologies. Since the early 1980's, Rome Laboratory has been developing the Knowledge Based Requirements Assistant (KBRA) tool which provides a formal, knowledge-based approach to requirements engineering. Future work will integrate the KBRA with the REE [10].

5.0 Acknowledgments

The Rapid Interface Prototyping System (RIP) was developed by International Software Systems Inc. (ISSI) for Rome Laboratory under Contract No. F30602-89-C-0129. The predecessor to RIP, the Rapid Prototyping System (RPS) was developed by Martin Marietta for Rome Laboratory under Contract No. F30602-86-C-009.

The Proto tool was developed by International Software Systems Inc. (ISSI) for Rome Laboratory under Contract No. F3062-89-C-0129.

The Analyst is a proprietary tool developed by Systems Designers Inc. and was enhanced for Rome Laboratory under Contract No. F49620-85-C-0132.

The Requirements Engineering Environment (REE) was developed for Rome Laboratory by International Software Systems Inc. under Contract No. F30602-89-C-0200.

6.0 References