END EFFECTOR MONITORING SYSTEM: AN ILLUSTRATED CASE OF OPERATIONAL PROTOTYPING

Jane T. Malin and Sherry A. Land
Intelligent Systems Branch, ER2
NASA Johnson Space Center
Houston, TX 77058

Carroll Thronesbery
The MITRE Corporation
1120 NASA Road 1
Houston, TX 77058

ABSTRACT. This paper introduces operational prototyping, to help developers apply software innovations to real-world problems, to help users articulate requirements, and to help develop more usable software. Operational prototyping has been applied to an expert system development project. The expert system supports fault detection and management during grappling operations of the Space Shuttle payload bay arm. The dynamic exchanges among operational prototyping team members are illustrated in a specific prototyping session. We discuss the requirements for operational prototyping technology, types of projects for which operational prototyping is best suited and when it should be applied to those projects.

1. INTRODUCTION

We developed the concept of operational prototyping during a case study of NASA intelligent systems [3], which was conducted to identify useful types of human-intelligent system interaction. During that case study, we observed that developers supporting a large flight controller group were using highly successful methods, although (or perhaps because) they were not following standard software engineering practices. Though their methods were informal, they developed fieldable software rapidly, satisfied their user customers, and avoided the pitfalls of wandering development, lack of documentation and excessive nonstandard custom software. Their methods interested us, not only because of their success, but because they seemed to also address problems that human-computer interaction designers have in common with users. That is, usability evaluation and human factors concerns often have too little effect on delivered software. These methods emphasized the user and usability, and achieved success at lower development cost, by using rapid prototyping methods and by exploiting the advanced graphics technology and automated object-oriented programming capabilities available in expert system development environments.

Traditional software engineering has weaknesses in the areas of helping users to articulate system requirements, assisting technology insertion, and providing innovative task support to users. Consequences of these weaknesses include slow and late deliveries, and products that require costly rework to satisfy customers. The practices we observed were strong where traditional software engineering was weak. Since we could expect that advanced software development tools would become more widely available to software developers in the future, we analyzed these methods and began to describe and codify them. We refined them to integrate with software engineering concepts and human-computer interaction concerns [1].

We call these methods operational prototyping. Operational prototyping is an approach to aid the development of innovative software applications for complex tasks, such as those found in aerospace operations. Innovative applications are those for which important aspects are not well understood by software developers. For instance, the operational prototyping approach introduced in this paper is based on applications of artificial intelligence technology to real-time fault management problems.

Operational prototypes are called "operational" because they can be used in an operational setting to demonstrate how a new approach solves a specific problem and supports user task performance. Since the prototyping is user-driven, these systems are more likely to be operations-scenario-driven than systems from many traditional, requirements-driven projects. They are called "prototypes" because of their informal, iterative development. Although the development schedule is informal, the addressing of system requirements is rigorous. This must be so, since the stakes are high: these prototypes are fielded for side-by-side use and evaluation in the operational setting. They are intended to stand in until the "software engineered" versions are developed, and can be used to validate them.

2. EXAMPLE APPLICATION

Recently, we have had the opportunity to further evaluate and refine these methods by doing our own operational prototyping. We have been using these methods while developing a monitoring and fault detection expert system for flight controllers, the DEcision Support SYstem (DESSY) End Effector application. DESSY is an object-oriented, rule-based, decision support system to assist ground-based flight-controllers in monitoring and managing faults in the Shuttle payload bay arm. It is being developed incrementally: the end effector module is the second of several modules being developed for arm subsystems. The End Effector module helps to monitor the grappling device at the end of the arm, which makes a secure connection with the payload so that it can be manipulated by the arm.

On the one hand, the DESSY End Effector development approach resembles traditional software development because there are similar development activities, which are initiated in about the same order as a traditional waterfall development approach. A requirements analysis drove some initial designs for displays, object hierarchies and structures, rule organizations, and testing scenarios.

On the other hand, our operational prototyping methods differ in important ways from traditional development methods. First, the development activities are being addressed iteratively. No attempt has been made to state all the requirements in detail before exploring the implications of those requirements by design, implementation, and evaluation of a prototype. Since DESSY is an innovative application, it is not possible to understand the requirements fully before beginning the design. The second major difference is use of the prototype in operational scenarios as a means for involving the customer in analysis, design, implementation, and evaluation. Our flight controller representative offers suggestions for solving development problems rather than just passing judgment on the decisions we have already made. Finally, we refine the requirements and design based on our experience with the prototype. This is where the magic of operational prototyping happens.

2.1. Example Operational Prototyping Session

To convey the sense of operational prototyping in a concrete fashion, we present an example of a specific operational prototyping session early in the development of the DESSY End Effector module. In this session, we made improvements in the screen layout and discovered new requirements for the intelligent system.

Figure 1 shows an original screen layout sketched on the Macintosh before the prototype was developed. The end effector consists of two mechanisms: a snare and a rigidizer. The snare has wires which wrap around a pin protruding from the payload, thereby capturing the payload. The rigidizer pulls the snare into the interior of the end effector until

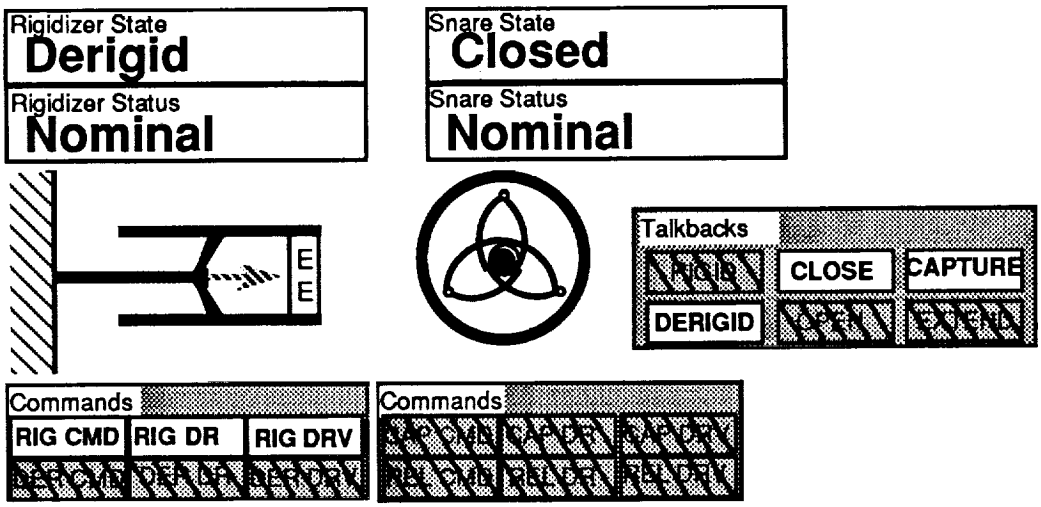

Figure 1. Before Operational Prototyping Session

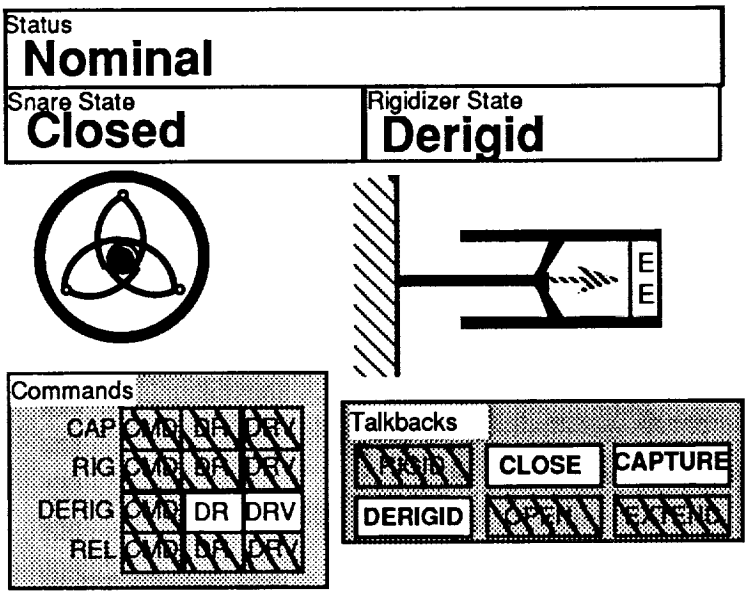

Figure 2. After Operational Prototyping Session

the end effector and the wall of the payload press against one another, making rigid the connection between the arm and the payload. These two mechanisms are represented iconically in the central portion of the screen, and these icons move to portray the current state of the each mechanism. Most of the remainder of the screen is organized around the icons, with rigidizer information on the left and snare information on the right. Because this design is neat, orderly, and logical, all the team members, including the user, were initially satisfied before the prototyping began.

During the first prototyping session, however, it became apparent that with the original screen layout, events in nominal capture and release sequences were difficult to follow in real time. Some end effector events last only a second and can be accompanied by seven display changes. With the design in Figure 1, none of us (not even our expert flight

controller) could adequately monitor a nominal sequence of events in real time. As a group, we arrived at the solution depicted in Figure 2.

The layout in Figure 2 is more compact, and the commands have been re-organized according to the direction of movement rather than mechanism. For example, the capture and rigidize commands are represented contiguously because both commands are used when capturing a payload. By the same token, the derigidize and release commands are represented contiguously because they are both used when releasing a payload. With this new display layout, we can verify that critical events take place for nominal capture and release sequences. We discovered the need for improvement by interacting with the prototype, and we arrived at an acceptable solution by interacting with one another and experimenting with the prototype.

Operational prototyping is not just for improving user interfaces. Another result of this prototyping session was correction of the logic for inferring end effector states from telemetry data. While interacting with the prototype, the flight controller identified an error in our understanding of the sequence of command telemetry. This, in turn, affected the application logic for inferring end effector states. We had all looked at this information before, but its inaccuracies were evident only when the experienced flight controller was observing the events unfolding before him.

Improvements from this session occurred in the screen layout, and in understanding of operational sequences and their implications for the logic in the application. Because this session occurred early in DESSY End Effector development, the primary improvements were in our understanding of system requirements. Improvements in later sessions have also emphasized design and implementation. It is hard to imagine how these improvements could have been achieved in such a timely way by text-based requirements analysis or by human factors evaluation later in the development process.

2.2. Analysis of Operational Prototyping Benefits

What makes the magic of operational prototyping happen? One contributor to the magic is the critical expertise included in the development team. On the DESSY End Effector development team are (1) a flight controller, (2) a computer scientist, specializing in intelligent systems, and (3) a human factors engineer, specializing in systems analysis, software engineering and intelligent systems. The flight controller contributes subject matter expertise based on his engineering background and years of experience in monitoring, detecting failures, and managing failures in the end effector. This expertise helps to ensure the quality of our intelligent system application. The flight controller contributes equally valuable expertise as a user. This user member of the team knows the most about the tasks to be supported by the new software application, and its context. This is critical if the new application is to successfully improve productivity, reliability and safety of operations tasks.

Another major contributor to operational prototyping magic is that the whole team interacts with the prototype and experiments with ideas to improve the design. Because we interact with a prototype in an operational scenario and change the screen designs during the session, each person is able to see the implications of design decisions. What You See Is What You Get, or WYSIWYG, is a descriptor for applications which immediately show the full implications of user input. Prototyping is the WYSIWYG of analysis and design.

Because we all interact with the prototype, we are able to consider different design concerns and viewpoints simultaneously. The implications of a proposed change can be evaluated from engineering, intelligent systems, users, and human factors perspectives

because the team has expertise in all these areas. Because we can consider these viewpoints simultaneously, we are able to make dynamic tradeoffs. For example, if a new change seems ideal except for implementation feasibility (e.g., a performance or a development time impact), the expert in that area can voice an objection along with the reason for it. At that point, a brainstorming session can ensue, in which all participants can propose potential alternatives. In this situation, with all experts present to help solve the problem, a solution that satisfies the concerns of each is more likely to be developed. In fact, this is the motivation behind the concurrent engineering impetus in systems development [2]. Finally, operational prototyping magic happens when the team focus is on generating improvement options, rather than on evaluating the prototype for acceptance or rejection.

What is the magic? Quite simply, the magic of operational prototyping is a quickly improved design. The application design is improved by concurrent and cooperative design changes, and thus meets requirements and design constraints better than could have been achieved by team members working separately. It is achieved more quickly because team members with each type of critical expertise meet around the prototype, exploring the implications of design decisions from each perspective.

3. INTEGRATING SYSTEM DESIGN WITH OPERATIONAL PROTOTYPING

3.1. What Projects are Best for Operational Prototyping?

The strengths of operational prototyping appear to be complementary with traditional software engineering. Traditional software engineering practices are reasonably adequate for projects with stable requirements; whereas operational prototyping is well-suited to projects whose requirements are initially unstable, or where continuous improvement is a goal.

Two types of development projects have initially unstable requirements: those for innovative applications of new technology and those for innovative support of user tasks. Innovative applications of technology to specific problems, by their innovative nature, are unproven and unfamiliar. Unproven designs make managers of project resources uneasy. Prototypes can demonstrate the feasibility of an innovative application to ease the concerns of those managers. Because of the abbreviated development cycle, prototypes allow the evaluation of multiple innovative alternatives so that the best one can be selected. Thus, innovative applications of technology to specific problems are good candidates for operational prototyping.

Another type of project whose requirements are inherently unstable are those which provide innovative support of user tasks. The first contribution of operational prototyping is to give users a proper medium in which to express their requirements. Operational prototyping allows users to explore the implications of their expressed requirements before expending considerable resources for a definitive requirements document. Operational prototypes can help to clarify hidden tasks and requirements. For example, user requirements for support for supervising, overriding and updating intelligent systems have not been immediately evident in most expert systems projects. Another contribution is to allow initial requirements to change *early* in project development, when those changes are the least costly. The requirements for a software application change when the application is introduced into the workplace. An operational prototype can help to drive out many of those changes earlier, before the first delivery, when the design can be changed more easily and cheaply. The operational prototype can also remain available to support further design for continuous improvement by the customer after the system is delivered.

In summary, the best projects for operational prototyping are those whose requirements are unstable. Development projects which fit this description are those which involve innovative applications of new technology, those which involve innovative support of user tasks, and those where continuous improvement is a customer goal.

3.2. When Is the Best Time to Apply Operational Prototyping?

The best time for operational prototyping is at early stages of projects. Operational prototyping is a requirements articulation process which accepts pre-requirement inputs and produces a working prototype and a set of stabilized requirements to support full-scale development. From that point, the full-scale development and integration of the new system can proceed according to traditional software engineering approaches. Since operational prototyping will have stabilized the system requirements, the traditional approaches should be even more efficient and less susceptible to unexpected delays. Thus, operational prototyping can bring initially risky projects to a sufficient level of maturity for traditional development.

The operational prototype can provide additional benefits. If measures are taken to overcome potential safety and reliability concerns, the operational prototype can be used to provide interim user task support during full-scale development. It can also be maintained to help users articulate the inevitable changes which will occur in the future as tasks change and experience is gained with new systems.

These roles for operational prototyping imply sophisticated software support, both for prototyping and for side-by-side operation in control centers. Current commercial technology appears to be prepared for this challenge.

3.3. Roles for Human Factors Personnel in Operational Prototyping

Operational prototyping also implies changes in the roles of system developers. Human factors personnel have much to gain from concurrent engineering methods such as this. There is hope that their roles can change from ones of frustrated guidelines enforcers and evaluators on the sidelines, to positive active developers. Human factors personnel can become valuable team members, helping with development of operational test scenarios, and facilitating prototyping sessions while they represent usability and human factors style concerns.

We are currently developing a feasibility prototype that includes reusable and modifiable software library elements in the prototyping system. These library elements can enforce the concerns of several types of system developers, including human factors personnel. For example, library elements can embody human factors guidelines in designs that will be the first things the prototyping teams use. Instead of refining guidelines documents and style guides that are difficult to use, human factors personnel can participate in design of application elements that conform to their concerns, and that can be used immediately in prototyping.

4. RELATED WORK AND CONCLUSION

A useful source for refining the concept of operational prototyping has been Boehm's spiral model [4]. The spiral model shows how to place operational prototyping and traditional software engineering in the same project development life cycle. It shows how to accommodate iterative development while maintaining a sense of direction for the project. It also shows how to accommodate changing risk priorities through periodic evaluations, giving the early project development a risk-driven, rather than a document-driven source of

direction. Finally, it helps to identify what risks to address within each prototyping iteration as the project matures.

Operational prototyping has two primary objectives: (1) to demonstrate the feasibility of new application, (2) provide interim operation during full-scale development. It is intended to be complementary to traditional software engineering approaches rather than a replacement for them. It complements them by addressing the risk which is most disruptive to traditional approaches: unstable requirements. Software engineering organizations can complete full-scale development for successful operational prototypes, while the operational prototype provides interim task support to its user. After full-scale development is completed, the operational prototype can serve as a test-bed for future enhancements. Because operational prototyping is a requirements articulation process, the primary emphasis is on analysis, though the quality of that analysis is evaluated by designing, building, implementing and using the prototype.

Operational prototyping helps developers to apply new technology to real-world problems by helping software developers, subject-matter experts, and human factors experts and users to quickly explore the implications of requirements, design, and implementation decisions. Operational prototyping helps users to articulate their requirements for systems that use new technology, and it helps developers to manage the risks associated with new technologies. Consequently, it can help reduce the costs of both development and operations, while improving the quality of both.

AKNOWLEDGEMENT

We would like to acknowledge the contributions of the Remote Manipulator System (RMS) flight controller section at Johnson Space Center/NASA (DF44), especially those of Salvator A. Ferrara. These contributions have added significantly to the success of the DESSY project described in this article. The members of the RMS section have contributed to both the strategic direction of DESSY planning and the technical knowledge concerning user tasks and the engineering of the RMS.

REFERENCES

1. Malin, J.T., & Thronesbery, C.G., "Analysis and Design of Intelligent Systems: The Role of Operational Prototyping," presented at JSC Training Seminar, HUMAN-COMPUTER INTERACTION DESIGN: MAKING INTELLIGENT SYSTEMS TEAM PLAYERS, NASA, Lyndon B. Johnson Space Center, Houston, TX, 1992.

2. Erb, D.M., "COMPUTER-AIDED SOFTWARE ENGINEERING (CASE): A 15-YEAR VISION AND RECOMMENDATIONS," MTR 92W0000104, The MITRE Corporation, McLean, VA, 1992.

3. Malin, J.T., Schreckenghost, D.L., Woods, D.D., Potter, S.S., Johannesen, L., Holloway, M., & Forbus, D., "Making Intelligent Systems Team Players: Case Studies and Design Issues." NASA TECHNICAL MEMORANDUM 104738, NASA, Lyndon B. Johnson Space Center, Houston, TX, 1991.

4. Boehm, B.W., "A Spiral Model of Software Development and Enhancement," IEEE COMPUTER, Vol. 21, No. 5, 1988, pp. 61-72.