SCENARIO-BASED APPROACH TO RAPID PROTOTYPING OF HUMAN-MACHINE SYSTEMS

Nikolai N. Mansurov [a], Dmitri Vasura [b]

[a] *Department for CASE tools, Institute for System Programming, Russian Academy of Sciences*
B.Kommunisticheskaya 25, Moscow 109004 Russia; nick@ispras.ru
Current: KLOCwork Solutions Corp.,
1 Antares Drive, Ottawa K2E 8C4, Ontario Canada
[b] *Department for CASE tools, Institute for System Programming, Russian Academy of Sciences*
B.Kommunisticheskaya 25, Moscow 109004 Russia; vasura@ispras.ru

Abstract: This paper discuses a scenario-based approach to rapid prototyping of Human-Machine Systems, specifically targeted towards the early phases of embedded software development. The key concept of our approach is so-called co-design of the user interface prototype and the black-box behavior of the new system. Our approach, together with the corresponding tool - the MOST Use Case Studio improves collaboration between the members of the design team, facilitates involvement of business people, customers, problem domain experts and other non-technical stakeholders into capturing and validating requirements models and accelerates requirements definition cycle.

Keywords: Requirements Analysis, Design, Prototyping, Event Sequence, Simulation, Interface

1. INTRODUCTION

Lately, time-to-market has become the dominating factor for industrial success, placing enormous pressure on manufacturers and developers to find much more efficient means of producing high-quality software for their products. At the same time today's embedded applications are becoming more complex, especially in the automotive industry, where in-vehicle displays, instrument clusters, brake and suspension logic, air bag and engine controllers are becoming more sophisticated.

According to (IDC, 2000), traditional development methodologies and tools do not produce applications fast enough to keep up with customer demands, higher turnover of software developers and the competitive pressures from more agile companies. Therefore, there is an increasing demand for new methodologies and tools that can speed up the time it takes to design, construct, deploy and maintain applications.

Rapid capturing and validation of requirements early in the life cycle is one of the key isues in accelerating development of Human-Machine Systems. Failure to understand and validate requirements can result in frequent and expensive re-work at later phases. Significant savings can result from reducing the cost of finding and fixing defects, including field errors. According to a recent study done by the University of West Virginia and the US Air Force, (Monkevich, 1999), most defects can be traced back to the early phases of software development process (see Fig. 1). The cause of 36% of defects was incorrect requirements translation, 5% of defects happened because of incomplete requirements, while another 28% of defects were attributed to logical design faults. Only 31% of defects were attributed to all other causes combined.

Specific challenge facing today's embedded software engineers is that design requirements for embedded applications must be communicated accurately and effectively to all members of the design team, both internally and externally, between manufacturer and supplier. Additionally, product design iterations must be accomplished faster than before to decrease product cycle times in the competitive marketspace.

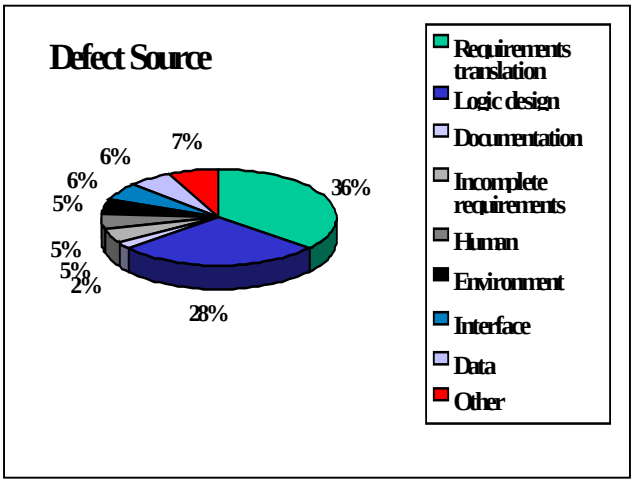

Fig. 1. Causes of defects

Prototyping has been identified as an important approach to early requirements validation. Prototyping exposes functional and behavioral aspects of the system as well as implementation considerations, thereby increasing the accuracy of requirements and helping to control their volatility during development (Wood, Kang, 1992).

This paper discusses our scenario-based approach for rapid prototyping of Human-Machine Systems, specifically targeted towards the early phases of embeded software development. The key to our approach is so-called *co-design* of the prototype user interface of the system and the definition of significant behaviors of the system. We suggest how to use the emerging user interface to capture scenarios as well as to animate available scenarios of the emerging behavior specification. Both processes reinforce each other and thus reduce the requirements definition cycle. Our methodology supported by the corresponding Web-based tool facilitates involvement of business people, customers, problem domain experts and other stakeholders into capturing and validating formal requirements models, and improves communication between the members of the design team.

The following are the main business objectives of our approach:

- Upfront definition and validation of the product with all stakeholders – to stop the snowball of incorrect decisions driven by incomplete or incorrectly understood requirements
- Improved communication between members of the design team, both internally and externally, by up-front combined visualization of the prototype user interface and use case scenariosAchieving acceleration of requirements validation cycle by using tools to automatically perform model-to-model transformations, high-yield validation, as well as generate interface specifications, code and tests for deployable components from scenarios

The rest of the paper has the following organization. In section 2 we provide an overview of our requirements engineering methodology. In Section 3 we outline the key ideas of the scenario-based co-design of the user interface and the behavior specification. In section 4 we outline the Use Case Studio toolkit, which implements the suggested approach. In section 5 we provide a brief comparison with related approaches.

2. OVERVIEW OF OUR SCENARIO-BASED PROTOTYPING

Our scenario-based prototyping approach consists of several steps, which can be performed iteratively.

- Capture the set of external actors and use cases as UML use case diagrams (Booch, 1998);
- Design prototype user interfaces for all external actors using the Interface Editor tool, which automatically generates a user-friendly interface of the formal scenario model;
- Interactively capture desired behavior for each use case using the generated user interface;
- Specify complete behavior of the system by identifying *episodes* (short interaction sequences, potentially down to individual *operations*) and arranging them into a UML Activity Diagram;
- Validate functional requirements using a combination of visual and formal techniques:
  - Animate validation scenarios against the generated user interface
  - Automatically synthesize an executable model using our Event Automata approach, run model checking tool on the synthesized model, replay problematic sequences against the generated user interface (Mansurov, *et. al.*, 2000).

When the prototype of the system is completed and validated, several products can be automatically generated from the scenario model: 1) a draft design specification of the system in SDL; 2) test cases in standard TTCN language; 3) source code for components in Java, or C++; 4) interface definitions for components in IDL (Mansurov, *et al*, 1999).

3. CO-DESIGN OF THE USER INTERFACE AND BEHAVIOR IN SCENARIO-BASED PROTOTYPING

In this section we outline the key idea of our scenario-based prototyping - co-design of prototype user interfaces and use case scenario models.

The objective of the co-design approach is to capture requirements and create a *business black-box model* of the system, consisting of the prototype user interface of the system and the set of the use cases. Each use case is captured as a scenario, showing interactions between the system and its environment. Such model can be effectively understood and validated by the stakeholders of the future system.

The prototype user interface to the system is designed using our Interface Editor tool. This can be used to model graphical user interface of a software system or front panels of a Human-Machine Interface of a hardware device. User interface elements are associated with *events*, representing interaction between the system and its environment. This association allows using the prototype user interface as a front-end to the formal scenario model.

Desired behaviors of the system are captured by directly activating the elements of the user interface. Associations with events allow recording activations of the elements of the user interface as sequences of events in the form of a Message Sequence Chart (ITU-T, 2000) or UML Sequence Diagram. Captured scenarios can be refined using the Scenario Editor tool.

Animating scenarios against the prototype user interface can validate the system definition. At this step, additional scenarios can be added to the use case. If any problems of inconsistencies are discovered either in user interface, or in the behavior description, one should go back to the previous steps and change the appropriate parts of the use case model.

It is important to engage stakeholders into the process of prototyping the system, especially into the design of the user interface. Research into requirement elicitation indicates that customer participation in sketching the high-level prototype user interface of the future system might be one of the most effective strategies for discovering hidden requirements and implicit expectations of customers (Shipman, et. al, 2000).

4. THE MOST USE CASE STUDIO TOOLKIT

We developed tool support for our scenario-based approach to rapid prototyping - the MOST Use Case Studio. The MOST Use Case Studio implements the scenario-based co-design of user interface and behavior.

The MOST Use Case Studio consists of a set of visualization tools and Validation and Code Generation Kernel, described in (Mansurov, 1999). The Kernel uses automatic synthesis technique to produce executable code from scenarios. The Kernel is based on our MOST-SDL tool (Mansurov, 1999).

Visualization tools of the MOST Use Case Studio toolkit include the following:

- Interface Editor - an interactive tool to create and edit prototype user interfaces;
- Scenario Recorder – uses the generated user interface to capture or animate scenarios. The Scenario Recorder tool uses the Video Camera metaphor with "Play", "Record", "Fast Forward" and "Fast Backward" buttons (see Fig. 2);
- Scenario Editor - a visual editor for scenarios represented as UML Sequence Diagrams or ITU-T Message Sequence Charts;
- Episode Editor - a visual editor for so-called episode sequences represented as UML Activity Diagrams or ITU-T High-Level Message Sequence Charts;
- Episode Simulator, - visual interface to simulation of UML Activity Diagrams;
- Use Case Editor - a visual editor for UML use cases
- Model Navigator - visual access to the repository.
- Web interface – presents scenario models through the Internet. Our Web interface allows collaboration of design team members by publishing models on the Web. The Web interface also allows capturing and validating scenarios through the Web (similar to the Scenario Recorder).

4.1. Interface Editor

Interface Editor of the MOST Use Case Studio allows to visually and intuitively prototype a user interface, without having any experience in programming such interfaces for industry applications.

The editor looks like a graphical editor (see Fig. 2). The prototype user interface is defined as a set of *panels*. Each panel has a *background* (which can be imported from a digital camera, or from a scanner). The background has several *active areas*, the so-called user interface (UI) elements. Active areas can be visually arranged on the screen to create the desired layout. Input elements can be associated with *an activation effect*, which creates an illusion of a virtual user activating this element during animation of scenarios (see Fig. 3).

Generated user interface is used to capture scenarios, as well as to present traces for validation. Generated user interface is controlled by the Scenario Recorder tool.

Fig. 2. Interface Editor, updating selected active area

The association between UI elements and events has dual use: in "record" mode this association is used to create (edit) a Sequence Diagram; in "playback" mode this association is used to animate a Sequence Diagram.

4.2. Scenario Recorder

When the first draft user interface for a certain use case is available, scenarios for this use case can be captured using our Scenario Recorder Tool. This supports our concept of co-design of the user interface and behavior specification.

Scenario Recorder tool allows to record sequences of events, corresponding to the interactions between the system and its environment by visually activating (i.e. pressing, selecting, switching etc.) UI elements of user interface panels, created by the Interface Editor. Sequences of events activated through UI elements can be simultaneously displayed in the Sequence Diagram Editor.

Scenario Recorder panel looks like a Video Camera control panel (see Fig. 3). Using "back" and "fast back" buttons from the panel the user may undo some steps and then start recording again or replay the sequence of actions using buttons "forward" and "fast forward". Both back and forward operations are reflected on the Sequence Diagram.

During simulation all events traversed by the Scenario Recorder are visualized against the user interface panels as dynamically changing states of the UI elements. Activation effects create an illusion of activating the UI elements by a virtual user.

Scenarios can be created or edited manually using the Scenario Editor tool of the MOST Use Case Studio (see Fig. 4). Currently, our Scenario Editor Tool uses the Message Sequence Chart notation to represent scenarios.

4.3. Episode Editor And Episode Simulator

When all scenarios for a certain use case are recorded in Scenario Recorder, the Episode Editor of the MOST Use Case Studio can be used to combine these scenarios into an Activity Diagram, which specifies complete behaviour of the system in the given use case. Activity Diagram consists of the references to individual episodes (or sub-scenarios) and flow lines, including alternatives and repetitions (see Fig. 5).

When the Activity Diagram is completed, the use case can be simulated using the Episode Simulator tool of the Use Case Studio.

Fig. 3. Scenario Recorder with the generated user interface in animation mode

The Episode Simulator also has a Video Camera control panel. Step-by-step simulation of an Activity Diagram can be performed using “forward”, “play” and “back” buttons on the Episode Simulator panel. When the “play” button is pressed, simulator steps through the Activity Diagram, i.e. makes a transition from the currently selected symbol to the next symbol by the flow line. If there are several possible transitions from the currently selected symbol, the Episode Simulator highlights all possible symbols, to which the transition may be done, and asks the user to make a choice.

When a reference symbol is reached, the corresponding Sequence Diagram is loaded into the Sequence Diagram Editor and simulation is continued in the Scenario Recorder.

5. RELATED WORK

Visualization of formal specifications for non-technical stakeholders is becoming an active research field (Visual, 1999). For example, AMBER is a visual formal notation for describing and analysing business processes models (Luttinghuis, 1999). The TestBed Studio provides a non-technical GUI for visualizing, creating and simulating the model.

Several groups explore user-friendly animation of scenarios aimed at presentation to the stakeholders. Prof. Jeff Magee uses the so-called SceneBeans to build custom animations for scenarios (Magee, *et. al.*, 2000). In contrast, our approach emphasizes very simple means of building prototype user interfaces, in order to engage customers into this activity. In our approach, a sketch of the user interface, provided directly by a customer, can be scanned and imported into the tool. This however results in lesser "degree of realism" in animations.

The major difference of our approach, compared to visualization of state-machine traces, is that we suggest using the prototype interface not just for requirements elitication purposes and visualization purposes, but also for capturing scenarios.

6. CONCLUSIONS

In this paper we presented our scenario-based approach to rapid prototyping of Human-Machine Systems. Our approach is targeted towards the early phases of embedded software development. The key concept of our approach is so-called co-design of the user interface prototype and the black-box behavior of the new system. We have selected very simple means of creating prototype user interfaces, in order to facilitate involvement of non-technical stakeholders in requirements definition process. Our scenario-based approach makes it possible to use the prototype user interface for capturing the desired behaviors of the system, as well as for animating these

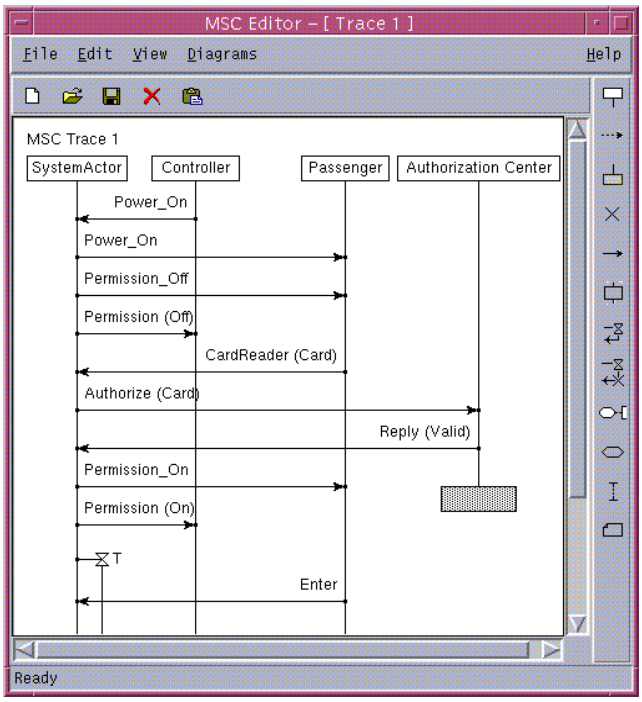

Fig. 4. Scenario Editor

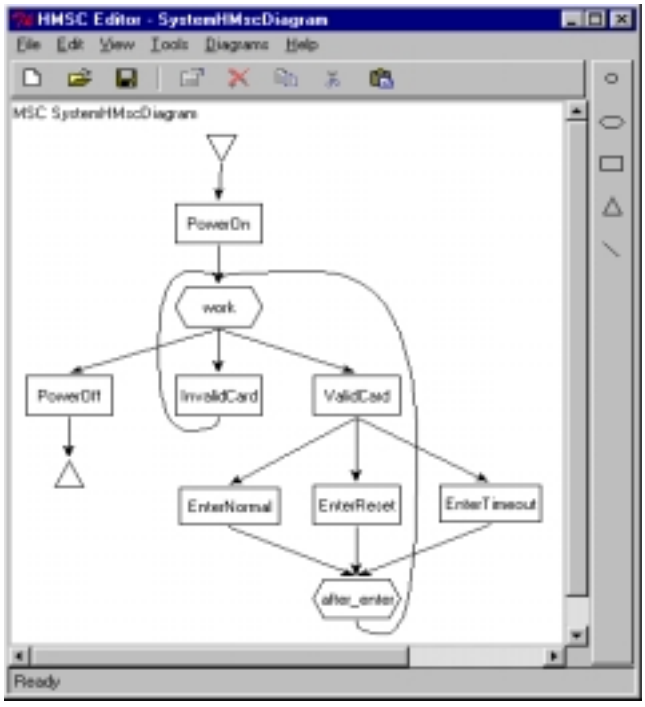

Fig. 5. Episode Sequence Editor

behaviors for validation purposes. Our approach, together with the corresponding tool - the MOST Use Case Studio - improves collaboration between the members of the design team, facilitates involvement of business people, customers, problem domain experts and other non-technical stakeholders into capturing and validating requirements models and accelerates requirements definition cycle

We believe, that the suggested approach together with a set of analysis, transformation and validation tools produces a comprehensive rapid prototyping and requirements definition environment, which can be used by non-technical experts.

REFERENCES

Booch, G., Rumbaugh, J., Jacobson, I., (1998), *The Unified Modeling Language User Guide*, Addison-Wesley, 1998

Mauw, S. (1999), 1st Int. Symposium on Visual Formal Methods (VFM'99), Ed. S. Mauw, *et.al.*, Eindhoven, The Netherlands, 1999, Computing Science Reports, Department of Mathematics and Computing Science, Eindhoven University of Technology, report 99-08

ITU-T, Rec. Z.120, Message Sequence Charts (MSC), Geneva, 1996

Mansurov, N (1999), Automatic Synthesis of SDL from MSC in Forward and Reverse Engineering, In: (Mauw, 1999), pp. 44-64

Mansurov, N, Vasura, D. (2000), Approximation of (H)MSC semantics by Event Automata, In: Proc. SAM'2000 workshop, Grenoble, France, 2000

Monkevich, O., (1999), SDL-based Specification and Testing Strategy for Communication Network Protocols, In: Proc. 9th SDL Forum, Montreal, Canada, June 21-26, 1999

IDC, 2000, Application Design and Construction tools market forecast and analysis, 2000-2004, May, 2000.

Luttinghuis, P., Visualizing Business Processes, In: (Mauw, 1999), pp 82-113

Magee, J., Pryce, N., Giannakopoulou, D., Kramer, J. (2000), Graphical Animation of Behavior Models, http://www-dse.doc.ic.uk/Software/SceneBeans

Wood, D., Kang, K. (1992), A Classification and Bibliography of Software Prototyping, Technical report, CMU/SEI-92-TR-13, 1992

Shipman, F., Moore, Comparison of Questionnaire-based and GUI-based Requirements Gathering, In: Proc. ASE'2000, Grenoble, France, 2000