Identifying Safety-Critical Requirement Defects Using a Tool-Based, Iterative Process

Hongyu Sun
Department of Computer Science
Iowa State University
sun@iastate.edu

Robyn R. Lutz
Department of Computer Science
Iowa State University
and Jet Propulsion Laboratory/Caltech
rlutz@cs.iastate.edu

1. Introduction

Deriving accurate system requirements from a natural language description is difficult. A common approach to translate formatted, natural language user requirements into accurate system requirements is through a collaboration of domain and application engineers. However, this process often leads to flawed requirements. A promising solution is to catch early requirements defects via rapid prototyping [2].

In this paper, we use an executable specification language, SpecTRM-RL, for early prototyping. Different from the traditional code-based prototyping approaches, it helps increase consistency between the prototype and the requirements and provides both formal analysis and dynamic evaluation to support verification and validation (V&V) [2]. As a result more requirements related defects can be found during the early development stage.

2. Approach

Our approach to develop natural-language-based requirements into more accurate system requirements is a tool-supported waterfall process. Figure 1 shows the steps in the associated, iterative V&V process. Since changes made to the prototype as a result of the V&V steps may introduce new defects into the prototype, the steps must be iterated. Thus, we introduce iterations of V&V to help assure that the requirements prototype is valid and verified throughout the specification process.

The V&V iterations begin when translating the prototype into SpecTRM-RL format. SpecTRM-RL is a state chart based tabular specification language [1]. SpecTRM can generate an executable, visualized model from SpecTRM-RL for simulation.

Two formal analyses, Determinism Analysis and Robustness Analysis, examine the overlapping conditions to get rid of non-determinism and search for missing conditions to guarantee that the specified conditions form a tautology [4]. SpecTRM automatically enumerates overlapping and missing conditions, and we use manual inspections to identify the harmful ones.

The iterations finish when no modification takes in one V&V iteration. Related documents and definitions are then updated to incorporate the improvements.

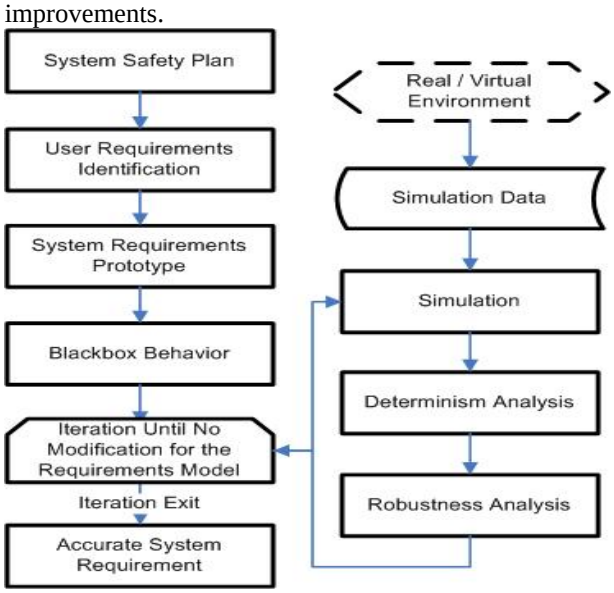

Figure 1. Overview of the Process

3. CMMMCS Application

The Coal-Mine Methane Monitoring and Control System (CMMMCS), as a subsystem of mine ventilation system, monitors the percentage of methane in mine shafts and provides feedback to the ventilation system to maintain a safe level of methane for miners [5].

Two hazards were identified by means of a Preliminary Hazard Analysis (PHA) [4]: The first was, "No emergency warning is sounded when the level of methane in the mine shaft reaches an explosive level." The second was, "A mine shaft with an explosive level of methane is not isolated (by closing emergency doors)."

We then identified the requirements the customer needed from the system, specifying them in natural language and distinguishing routine functions and safety-critical functions. Routine functions included automatically adjusting the air speed in mine shafts to keep the methane percentage within the safe range. Safety-critical functions included emergency handling (in response to an explosive hazard) and maintenance

handling (in response to a broken sensor).

By describing the behavioral requirements in state charts, we derived an initial system requirement prototype consisting of five system modes. To do this, we had to make assumptions such as initialization conditions and the limits on the capabilities of the system to handle incomplete, overlapping and unclear conditions.

The black-box behavior model was then constructed by representing the state chart in SpecTRM-Runtime Language tabular format. The prototype subsequently evolved into a more accurate specification through the iterations of simulation and analyses described below. A visualized executable model was auto-generated from the prototype specifications by SpecTRM at each iteration.

In the simulation step (see Fig.1), to improve the completeness of the model and to test the validity of the assumptions made previously, we used realistic test cases to mimic rare events. Through inspection of the simulated behaviors, three safety-critical defects were found and eliminated. One safety critical defect was that, even if the system was in emergency-handling mode, when a sensor was reported as broken it incorrectly transitioned to maintenance-handling mode. This violated the safety requirement. In this case, emergency handling should take priority over the maintenance.

Another safety-critical defect was an incomplete requirement, caused by an incorrect initialization assumption. When the system was initialized, the environment was assumed to be safe. However, this turned out to be untrue in our virtual environment. The third defect was also an incomplete requirement caused by an assumption. In the scenario where the system was in the maintenance-handling mode, manual inspection occurred, resulting in the need for a manual restart after repair. However, the system in this mode became insensitive to the presence of an excess of methane (a hazard). We updated the prototype to eliminate the identified defects and continued to next step once no more defects were found via simulation.

In the next step, Determinism Analysis (see Fig. 1), overlapping conditions were systematically examined to eliminate ambiguity and confusion. Three cases of potential non-determinism were found in the CMMMCS application. For example, one defect, caused by unexpected requirement interactions, was that in one state the conditions could be met to transition either to a routine function (next state) or to an emergency function (next state). Thus, the state chart was nondeterministic. We corrected the three defects by changing the conditions.

In the third step (see Fig. 1), Robustness Analysis, we identified two missing requirements that could be hazardous and eliminated them. For example, if the airflow in the mine shaft was abnormal and the methane level became high while in a specific state of a routine function, the system did not go to the emergency-handling mode as needed.

After updating the prototype at the end of each formal analysis step, the revised prototype was again verified and validated. Iterations of V&V must take place until no additional modifications occur. Thus, a second iteration of the V&V process was necessary.

In the second iteration, one safety critical defect was found that was caused by the first iteration's modification. The new behavior (to handle an emergency while in the maintenance mode) introduced non-determinism. We corrected this flaw by adding constraints. Also, one slightly inaccurate assumption was found and improved. The V&V process was then repeated. In the third iteration, no defects were found. The iterative steps of analysis and modifications resulted in clearer, more correct, and safer requirements.

4. Conclusion

This work can help eliminate safety-critical hazards by providing a way to assist in deriving improved system requirements. This paper demonstrated a safety-centered process to refine requirements written in natural language into a more accurate and detailed specification of system requirements. A simple requirements model of a safety-critical coal mine methane-monitoring subsystem was introduced and used to illustrate the approach. It did this through an iterative verification and validation of a prototype specified in SpecTRM. Further work is planned to address environment-model interaction for improved dynamic evaluation of the prototype. A human control component also needs to be introduced into the model.

Acknowledgements

This research was supported by the National Science Foundation grants 0204139, 0205588 and 0541163.

References

[1] M.K. Zimmerman, K. Lundqvist, N.G. Leveson, "Investigating the Readability of State-Based Formal Requirements Specification Languages", Proceedings of the 24th ICSE, 2002

[2] J. Thompson and M. Heimdahl, "An Integrated Development Environment for Prototyping Safety Critical Systems", 10th IEEE International Workshop on Rapid System Prototyping (RSP'99), pp172

[3] R. Lutz and C. Mikulski, "Empirical Analysis of Safety-Critical Anomalies During Operations", IEEE Transactions on Software Engineering , vol. 30, no. 3, March, 2004, pp172-180.

[4] N. G. Leveson, *Safeware: system safety and computers*, Addison-Wesley Publishing Company, 1995, pp295-300

[5] W. Mironowicz and S. Wasilewski, "Monitoring and Control of Ventilation to Improve Work Safety in Mines", *Mine Environment and Ventilation*, 2001, pp273-282.