

Automated Event Tree Analysis Based-on Scenario Specifications

Wei-Tek Tsai, Ray Paul*, Chun Fan, Lian Yu
Department of Computer Science and Engineering
Arizona State University
Tempe AZ, 85287

*Department of Defense
Washington, DC

Abstract

Event tree analysis is widely used for reliability and safety analyses as it can be used to specify how the system behaves under various failure conditions. Even though event tree analysis is automated, the construction of event trees is unfortunately manual, and thus error-prone. This paper proposes an automated technique to generate event trees from scenario specifications even before system design and implementation. Thus, an engineer can examine the event tree to evaluate system attributes such as reliability and safety during analysis. This technique starts by specifying system scenarios using the ACDATE model (Actors, Conditions, Data, Actions, Timing and Events), and then the model is executed to produce event trees based on various failure models. The tree generated can be used to generate the effect-cause diagram, and these two models can be used together to pinpoint the failed components in case of system failures.

Key words: Event tree analysis, scenario analysis and specifications, simulation.

1. Introduction

Event tree analysis is a popular technique for safety analysis. Even though the analysis can be automated, the construction of event trees is done manually, which can be costly and error-prone. Event tree analysis belongs to a collection of techniques related to *events*, and other related techniques include event-driven programming commonly used in embedded systems [1] as well as UML such as sequence diagrams, GUI-based systems, in design patterns such as Observer [1], distributed applications [3, 4], and simulation such as discrete-event simulation (DES). An event tree shows the events that are generated giving a specific condition, such as a failure condition. By tracing the event tree, one can see the possible effects of the condition. This paper proposes an automated event tree analysis based on scenario specifications.

2. Scenario Specification

Scenarios are derived from system requirements and formalized by the ACDATE (Actors, Conditions, Data, Actions, Timing, and Events) model. The semantics of ACDATE can be represented as a state transition (Figure 1): if an *actor* is in the *pre-condition*, and it receives a triggering *event* that satisfies the guard

condition, it will perform an *action* and change its state to the *post-condition*. The *action* performed may generate *events* that can be sent to other *actors*. This is illustrated in Figure 1.

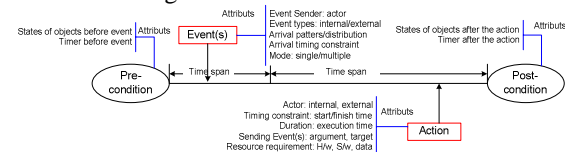


Figure 1. ACDATE Model

Once scenarios are formalized in this manner, it is possible to perform a variety of analyses, such as dependency analysis, completeness and consistency analysis, timing analysis, and reachability analysis [6]. The scenario specified can also be verified using model checking techniques using temporal logic, and test scripts can be generated to the system once it is implemented [6]. This technique can be used to specify and test embedded systems as well as Web services [7]. The ACDATE model can also be simulated using our tool without any programming [8].

3. Automated Event Tree Analysis

3.1 Event Tree Generation

Once system scenarios are specified using the ACDATE model, one can generate their corresponding event trees using the following algorithm:

Algorithm 1: generating an event tree for event E

1. Generate the list of the actors involved in processing event E.
2. Generate all the possible combinations of the states of the involved actors, say,
 $C_i, i = 0, 1, \dots, m-1$.
3. For each C_i
 - a. Generate the actions performed, and events generated.
 - b. Added the events identified into the event tree.
 - c. Repeat this process until no more events or actions.
 - d. If same events are generated during the process, a flag is generated.

We have developed a simulation tool that can generate the event tree automatically from the ACDATE model. Note that potentially many trees can be generated for a

large system, and each tree can be huge if the system keeps on responding to events generated to handle previous events. If the system was specified incorrectly, the tree generated can never be terminated because a circular list of events will be generated to handle an external event. This can be detected by examining the red flag generated during the process, and once a cycle is detected, a failure report will be generated to warn the system analyst of possible incorrect specifications.

We have experimented several home security embedded systems, and in most cases, an external event will generate at most 3 to 4 internal events. For other types of systems, the number of events generated can be significantly larger. For example, military command and control systems often generate several hundred internal events for an external event, furthermore sometimes several same events will be generated during the handling of an external events. Thus, a red flag generated may not indicate an incorrect specification but a cause for concern.

Figure 2 shows an example event tree, where the numbers within parentheses represent the probability of the branch. The outcomes are the product of probability along the paths. The probability of events can be determined if the operational profile is available and used during analysis.

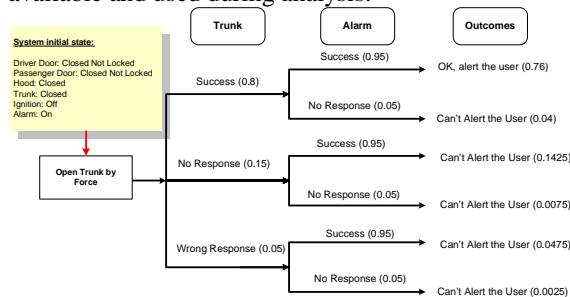


Figure 2. An Example Event Tree

Using the generated event tree, one can identify the direct and indirect effects by tracing along the event-tree path until reaching leaf nodes. One may also trace from the effects back to the cause by tracing the event tree backward from the leaf nodes to the root nodes. The process can be automated.

3.2 Effect-Cause Analysis

Fault tree analysis is a common technique for safety analysis [5], and it can be used to determine the cause-effect relationships. Unfortunately, often a fault tree is related to system design because an engineer often needs to trace the effect to the causes, and the process often starts from the environment to the internal design of the system. Thus fault trees can be constructed until the system design is available. This is different for event trees because an event tree can be generated

based on system scenario specifications even before system design. Furthermore, by examining all the event trees related to a system, one can generate the effect-cause diagram that links from an effect to its possible causes, and this construction can be automated. Furthermore, it is possible to assign the probability of a cause if an operational profile and failure distributions are available. Effect-cause diagram is similar to fault tree except that the elements on the diagram are analysis entities.

3.3 Pinpointing Failures

Recall that an event tree traces from a cause to its possible effects, while the effect-cause diagram is exactly opposite as it traces from an effect to its possible causes. By combining these two, one can pinpoint which system components that failed during failure analysis. For example, if the system has an effect *A*, which may have three possible causes (*C1*, *C2*, *C3*). But using event tree analysis, one can find what the effects of these three causes are. If the effects of *C1* and *C3* are missing while the effects of *C2* are visible, one can conclude that *C2* is the real cause of the effect *A*.

4. Conclusion

This paper proposes an automated technique to generate event trees commonly used in safety analysis. The trees are generated based on a formalized scenario specification model ACDATE. Once the event trees are generated, one can also generate the effect-cause diagram which can be used to pinpoint the components that actually failed during failure analysis. Because the proposed techniques are based on scenario specifications only, they can be used during any system development stage including analysis before any system design.

References

- [1] B. P. Douglass, *Doing Hard Time: Develop Real-Time Systems with UML, Objects, Frameworks, and Patterns*, Addison Wesley, 1999.
- [2] D. Luckham, *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*, Addison-Wesley, 2002.
- [3] D. C. Schmidt, M. Stal, H. Rohnert, and F. Buschmann, *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects*, Wiley & Sons, 2000.
- [4] N. Storey, *Safety-Critical Computer Systems*, Addison Wesley, Reading, MA, 1996.
- [5] W. T. Tsai, L. Yu, X. Liu, A. Saimi, and Y. Xiao, "Scenario-based Test Generation Tool for Embedded Systems", in Proc. of IEEE IPCCC 2003, pp.335-342.
- [6] W. T. Tsai, R. Paul, and L. Yu, A. Saimi, and Z. Cao, "Scenario-Based Web Service Testing with Distributed Agents", accepted to publish by IEICE Transaction, June, 2003.
- [7] W. T. Tsai, L. Yu, R. Paul, F. Chan, and X. Liu, "Rapid Scenario-Based Simulation and Model Checking for Embedded Systems", to appear in Proc. of SEA, 2003.