Applying Product-Line Fault Tree Analysis to Build Safer Multi-Agent Systems

Josh Dehlinger
Computer Science Department
Iowa State University
dehlinge@cs.iastate.edu

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

Abstract

Software fault tree analysis (SFTA) is a widely used technique for safety analysis of a system as it helps identify the causal relationship between failures and hazards. Yet, SFTA is difficult to apply to dynamic systems with shifting run-time configurations (eg., autonomous multi-agent systems (MAS)). This paper illustrates the application and use of a product-line approach to SFTA applied to a highly-autonomous, distributed MAS. Using this approach, engineers can evaluate safety properties and the dependability of a system early in the development lifecycle. Further, this approach utilizes a technique in which the assets produced are reusable (i.e., applicable towards all configurations of the system) with little additional work.

Keywords*:* Multi-agent systems, software fault tree analysis, product-line engineering, safety analysis.

1. Introduction

Software fault tree analysis (SFTA) is a common technique for safety analysis for single, mission-critical, software systems [4]. However, SFTA incurs tremendous redundant work for engineers on systems where members differ only slightly from each other because it requires a SFTA to be manually produced for each different member [5]. For example, a distributed, multi-agent system (MAS) in which each agent contains a core set of common functionality but differ in a small set of features would necessitate the manual creation of several SFTAs since there would be several different instantiations (depending on the set of features in any member) found within the distributed system. Adapting a product-line approach for such a system, however, would alleviate the redundant work and create a reusable SFTA asset.

In [3], we have presented a product-line SFTA technique that documents the SFTA for all members into a reusable asset. A tool, PLFaultCAT [2], was built to automatically derive a member's SFTA from a product-line SFTA. In [1], we have shown how product-line concepts can be incorporated into the development of a MAS so that requirements specifications can be used as a reusable asset and so a product-line SFTA can be used as a safety analysis technique for a MAS. A MAS can be viewed as a product line by keying the different levels of intelligence capable by a role as its variation points.

In this paper, we further illustrate how SFTA can be used in MAS as a safety analysis technique to produce more dependable and reliable systems. We describe how to identify potential failure points of a MAS and mitigate them by introducing additional safety requirements.

2. Software Fault Tree Specification and Derivation

A software fault tree analysis (SFTA) is derived from requirements and engineer expertise. A product-line SFTA additionally uses a Commonality and Variability Analysis for its creation [2, 3].

The creation of a product-line SFTA for a multi-agent system (MAS) entails associating one or more commonalities and/or variation points of the MAS with each leaf node of a SFTA. Full details can be found in [3]. A representative product-line SFTA for a distributed MAS (an autonomous satellite cluster) is given in Figure 1.

To derive an unique agent's SFTA from the product-line SFTA, PLFaultCAT [2] can automatically generate and display the SFTA based upon the selection of variation points (ie., its capable levels of intelligence) present within the agent, shown in Figure 2.

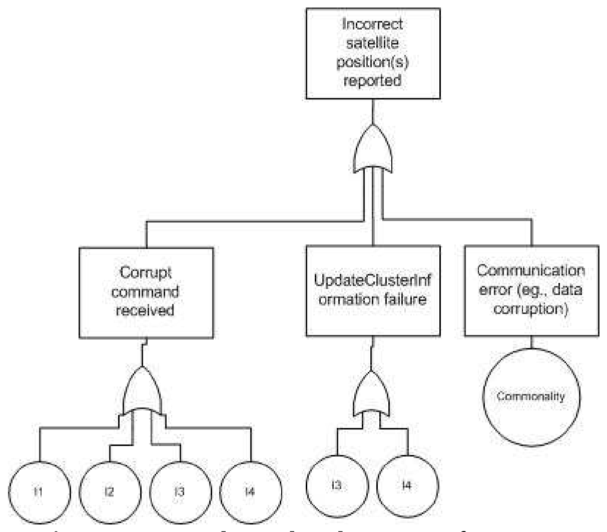
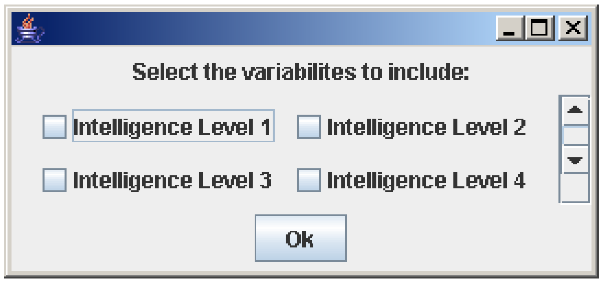

Incorrect satellite position(s) reported
Corrupt command received
UpdateClusterInformation failure
Communication error (eg., data corruption)
Commonality
I1
I2
I3
I4
I3
I4

Figure 1. Example product-line SFTA for a MAS


Select the variabilites to include:

Intelligence Level 1 | Intelligence Level 2

Intelligence Level 3 | Intelligence Level 4

Figure 2. Variability selection to derive an agent's product-line SFTA in PLFaultCAT

3. Safety Analysis of a Multi-Agent System

This section discusses the value to engineers of the product-line software fault tree analysis (SFTA) for a multi-agent system (MAS) in identifying potential failure points of the system and recognizing combinations of variation points within an agent that may cause possible hazards.

3.1 Identifying Failure Points

An advantage of a SFTA as a safety analysis technique is the ability to quickly determine the presence of single-point failures of a single system (a root node in the SFTA followed by a logical OR gate). A product-line SFTA allows the quick identification of single-point failures over the entire product-line. An example product-line single-point failure for a MAS is shown in Figure 1. In the case of a MAS adopting product-line concepts [1], this means the ability to pinpoint possible single-point failures for every possible instantiation of an agent (adhering to the commonalities and variation points allowed within the system). This is advantageous over the traditional application of SFTA to a MAS because it is not necessary to manually create each SFTA for each possible instantiation of an agent and then manually inspect each SFTA for a single-point failure.

Using a product-line SFTA for a MAS also allows for the identification of variation points (intelligence levels) or combinations of variation points that disproportionately contribute to hazards. Scanning the leaf nodes (where commonalities and variation points are associated to low-level hazards) of the product-line SFTA could lead to the discovery that a particular variability or set of variation points can contribute to high number of hazards within the set of SFTAs produced. This information proves valuable if engineers determine that the hazard risk of leaving the MAS design unchanged is unacceptable from a safety, dependability and/or reliability standpoint.

3.2 Mitigating Failure Points

Upon the identification of failure points within a MAS, engineers have the opportunity to take mitigating steps to improve the safety, dependability and/or reliability of the system. Additionally, since SFTA is constructed early in the development lifecycle, mitigating steps can be taken quite early in MAS development.

The mitigation of a single-point failure within the product-line SFTA can be done by introducing additional requirements, architectural components, guard conditions, operating rules or other counteractions into the design of the MAS. A mitigation strategy for combinations of variation points can be to add dependencies (restricting the selection of sets of variation points) into the instantiation agents in a MAS. Alternatively, a similar strategy for mitigating failure points (described above) can be adopted.

4. Concluding Remarks

This paper illustrates the application of a product-line software fault tree analysis (SFTA) technique to the requirements specifications of multi-agent systems (MAS). This work further integrates product-line concepts into agent-oriented software engineering and provides a tool-supported safety analysis technique that partially automates the reuse of software safety assets. The identification of possible failure points using the product-line SFTA in a MAS decreases engineer effort during early safety analysis. It is hoped that this approach will provide the means to support the safe reuse of requirements specifications and dependable designs in mission-critical, distributed MAS.

Acknowledgements

This research was supported by the National Science Foundation under grants 0204139 and 0205588 and by the Iowa Space Consortium.

References

[1] J. Dehlinger and R. R. Lutz, "A Product-Line Approach to Safe Reuse in Multi-Agent Systems", In *ICSE 2005 4th International Workshop on Software Engineering for Large-Scale Multi-Agent Systems* (SELMAS'05), St. Louis, MO, pp. 83-89, 2005.

[2] J. Dehlinger and R. R. Lutz, "PLFaultCAT: A Product-Line Software Fault Tree Analysis Tool", In *The Automated Software Engineering Journal*, to appear.

[3] J. Dehlinger and R. R. Lutz, "Software Fault Tree Analysis for Product Lines", In *Proceedings 8th International Symposium High Assurance Systems Engineering* (HASE'04), Tampa, FL, pp.12-21, 2004.

[4] N. G. Leveson, *Safeware: System Safety and Computers*, Addison-Wesley, Reading, MA, 1995.

[5] R. R. Lutz, "Software Engineering for Safety: A Roadmap" In *Proceedings Conference The Future of Software Engineering*, Limerick, Ireland, pp. 213-226, 2000.