Evaluating the Reusability of Product-Line Software Fault Tree Analysis Assets for a Safety-Critical System

Josh Dehlinger[1] and Robyn R. Lutz[2]

[1] Department of Computer and Information Sciences, Towson University, 7800 York Road, Towson, Maryland, USA 21252
jdehlinger@towson.edu
[2] Department of Computer Science, Iowa State University, 226 Atanasoff Hall, Ames, Iowa, USA 50011 and Jet Propulsion Laboratory / Caltech
rlutz@cs.iastate.edu

Abstract. The reuse of product-line assets enables efficiencies in development time and cost. Safety analysis techniques for Software Product-Line Engineering (SPLE) construct safety-related, non-code artifacts with the aim of reusing these assets for new product-line members. In this paper we describe results from the construction and reuse of a key safety-analysis technique, Product-line Software Fault Tree Analysis (PL-SFTA), and its supporting tool, PLFaultCAT. The main contribution of this work is the evaluation of PL-SFTA and PLFaultCAT for the reuse of safety analysis assets in a product line. The context is a safety-critical product line of spacecraft developed as a multi-agent system.

Keywords: reusable safety analysis assets, safety aspects of reuse, product-line software fault tree analysis, multi-agent system product lines.

1 Introduction

Software product-line engineering (SPLE) is a key enabling technology for reuse. It provides a proactive and systematic approach for the design and development of systems to create a set of similar products, a product line, from reusable assets. A *software product line* (SPL) is defined as "a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way" [1]. SPLE supports reusability by developing a set of products that share core commonalities and differ via a set of managed variabilities [13] and has been shown to be able to reduce the design, development and production time and cost of systems through the reuse of code and non-code artifacts [1].

Engineering for safety-critical systems, such as cardiac pacemakers [10] and medical imaging systems [11], has adopted a SPLE approach to take advantage of reusable artifacts during design and development. To maintain the safety properties requisite for critical systems, safety analysis techniques and tools specific to SPLE had to be created to accommodate the variability inherent in a SPL while also producing reusable artifacts that can be used for all product-line members [10].

S.H. Edwards and G. Kulczycki (Eds.): ICSR 2009, LNCS 5791, pp. 160–169, 2009.

Product-Line Software Fault Tree Analysis (PL-SFTA) [2] [6] extends traditional Software Fault Tree Analysis (SFTA) [8] by incorporating SPLE to produce reusable safety analysis assets. Unlike traditional SFTA, PL-SFTA develops fault trees that incorporate the variabilities amongst the SPL members to provide reusable safety analysis assets for the entire SPL. PL-SFTA is supported by a tool, PLFaultCAT [6], that enables this reuse by automatically producing the fault tree for each product-line member from the PL-SFTA. The goal is to support the safety analysis needed on a new product-line member with a less costly and more automated process.

This paper furthers the argument that PL-SFTA and PLFaultCAT supply a beneficial safety analysis technique and tool. Specifically, the contribution of this paper is an evaluation of the degree to which their generated reusable safety analysis assets can be directly applied to new product-line members within SPLE, including:

- A description of how PL-SFTA and PLFaultCAT systematically capture and reuse non-code safety analysis assets for safety-critical product lines
- An evaluation of the degree to which the safety analysis assets developed using PL-SFTA can be reused toward new product-line members using a significant case study based on a NASA-proposed multi-agent system product line (MAS-PL)
- An assessment of the degree to which the PLFaultCAT tool reduces the effort needed to construct the safety analysis products for a new member from the reusable product-line safety analysis assets
- A discussion of the implications of the evaluation of PL-SFTA and PLFaultCAT on the degree to which software engineering artifacts can be reused for MAS-PLs compared to traditional SPLs

This work is a part of a larger effort that investigates how safety-critical SPLs can be designed and developed with the support of SPL-specific, reusable safety analysis techniques. The long-term goal is to provide safety analysis assets for new product-line members in a timely and cost-efficient manner.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 summarizes the PAM MAS-PL case study used here for evaluation. Section 4 describes PL-SFTA in the context of SPLE. Section 5 provides an analysis of our empirical results regarding the reusability of the PL-SFTA safety analysis assets. Section 6 discusses the results from our experimental evaluation and their implications for MAS-PLs and SPLs. Finally, Section 7 provides concluding remarks.

2 Related Work

SPLE supports the systematic planning, design and development of a family of software systems through understanding, controlling and managing their common characteristics and differences. SPLE develops a family of products and relies on the analysis of the commonalities and variabilities of the product-line members prior to their development. Following Weiss and Lai [13], we specify a SPL using a Commonality and Variability Analysis (CVA) to document the SPL's *commonalities* (i.e., requirements of the entire product line), *variabilities* (i.e., specific requirements not contained in every member of the product line) and *dependencies* (i.e., constraints amongst selection of the variable features).

In previous work [2] [3], we have integrated the Family-Oriented Abstraction, Specification and Translation (FAST) SPLE methodology [13] into Agent-Oriented Software Engineering (AOSE) to enable the analysis and design of MAS-PL. In the domain engineering phase, the MAS-PL's requirements are defined and specified using our Gaia-PL methodology [2]. The application engineering phase then reuses the software engineering artifacts developed in the domain engineering phase to build new product-line members, in this case software agents within the MAS-PL.

There has been little related work to date specifically in safety-critical SPLs. Prior work in software safety [8] and in reuse for safety-critical systems [9] has not dealt directly with SPLE safety analyses. On the other hand, prior work in SPLE has not addressed the additional, high-assurance needs of safety-critical SPLs [1] [13]. To provide safety assurances for critical SPLs, we developed PL-SFTA and its tool support, PLFaultCAT [5] [6], to allow for the creation of fault trees for a SPL while supporting the reuse inherent in SPLE. A *fault tree* is a directed AND/OR graph that represents a hazard and its contributing causes. Each node is an event or condition that can contribute to the occurrence of the hazard [8]. In addition to supporting the development and reuse of safety analysis artifacts for critical SPLs, PLFaultCAT provides additional, automated safety analyses to identify failure points and safety-critical requirements [5]. These safety analysis results, when developed during the domain engineering phase, can be applied to all product-line members.

Other SPLE-specific safety analysis techniques, including [4] [7] [10], have provided systematic approaches for conducting safety analyses on critical SPLs without studying the extent to which the resulting artifacts can be reused.

3 The Prospecting Asteroid Mission

To evaluate this work, we used requirements based on the Prospecting Asteroid Mission (PAM), a NASA-proposed concept mission based on the Autonomous Nano-Technology Swarm (ANTS) technology to explore the asteroid belt [12]. This mission will consist of up to 1,000 spacecraft that can autonomously form subswarms to investigate asteroids of interest. Except for a spacecraft's scientific instrumentation specialties, each PAM spacecraft has identical hardware.

Each PAM spacecraft will be designated as a *leader*, a *messenger* or a *worker* [12]. A *leader* will determine the types of asteroids and data to pursue and will coordinate the efforts of *worker* spacecraft. A *messenger* will coordinate communication among spacecraft and with the Earth. Each *worker* will perform scientific investigation using its specialized equipment (e.g., a spectrometer). Within the PAM swarm, there is significant redundancy for each spacecraft type since 60-70% of the PAM spacecraft could be lost over the duration of the mission due to failures, collisions, etc. To preserve mission-critical requirements, additional capabilities (i.e., product-line variabilities) are given to some spacecraft to achieve redundancy at the swarm level through reconfiguration and adaptation. For example, some spacecraft may be able to switch at runtime from *messenger* to *leader* in response to loss or failure of other spacecraft, or to be tasked with monitoring for an impending solar storm (an optional variability).

The design and development of the PAM spacecraft as a MAS-PL would allow for the reuse of software engineering assets during design and development. In the

context of SPLE, the safety-related, commonality requirements include the navigation and guidance capabilities, collision avoidance, solar storm protection, etc. Additionally, each of the PAM spacecraft will have similar requirements related to self-coordination, self-healing and self-optimization behaviors (i.e., requirements).

4 Product-Line Software Fault Tree Analysis Using PLFaultCAT

This section describes the creation, analysis and reuse of a PL-SFTA for the safety-critical MAS-PL described in Section 3. The creation of the PL-SFTA occurs during the domain engineering phase of SPLE and constructs the safety analysis artifacts reused for specific product-line members in the application engineering phase.

4.1 Domain Engineering – Development of Reusable Safety Analysis Artifacts

The development of the PL-SFTA using PLFaultCAT consists of three steps:

Step 1. *Identify a root node hazard(s) and develop intermediate node tree*. The root node of a fault tree represents a potential hazard the system design and implementation should mitigate. This hazard may be known from an existing Preliminary Hazard Analysis [8] or from a product-line Software Failure Modes, Effects and Criticality Analysis (SFMECA) [4]. From the root node, a backward search is done to find causal, contributing events. Through gathering the causal events, an intermediate node tree is constructed to establish the cause-event hierarchy. The intermediate node tree, while not necessary in the construction of a PL-SFTA, aids in jump-starting the organization and analysis of the PL-SFTA and serves as the input to PLFaultCAT. Essentially, the intermediate node tree represents a typical fault tree without the Boolean logic gate relationships between causal events and effects.

Step 2. *Refine intermediate node tree and document in PLFaultCAT*. The intermediate node tree may contain nodes that do not reflect the level of detail needed. Thus, domain expertise, and/or the use of other safety analysis artifacts like SFMECA [4], may be needed to analyze the tree for completeness, capture additional events leading to a failure (e.g., events from the environment) and refine nodes. The domain expertise is additionally needed to determine the necessary logical combination of the children nodes to cause the parent node, as is done in traditional fault tree analysis [8]. As with any FTA, the reliance on domain expertise means that the PL-SFTA is only as good as the accuracy and completeness of the information used to create it.

Step 3. *Consider the influence of commonality and variability requirements on all leaf nodes*. This step employs a bottom-up approach to analyze each leaf node of the intermediate SFT and determines which commonality and/or variability requirements contribute to causing the root node event to occur. In doing this, we associate the range of commonality and variability choices for any individual product-line member with how it might influence a particular hazard. Not every commonality or variability will have an influence or appear within any given fault tree. However, every leaf event node should have an associated commonality, variability, and/or basic (primary) event (e.g., an environment or user input). Considering the influence of a present or

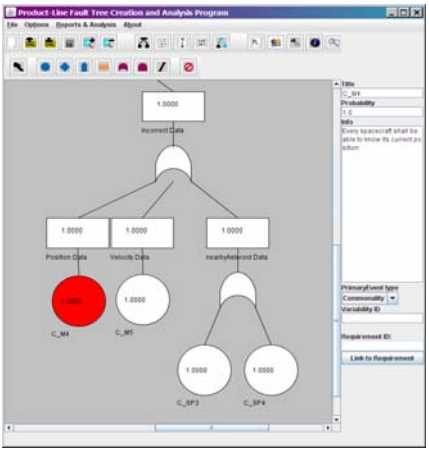

Fig. 1. Associating SPL requirements to the leaf nodes of a PL-SFTA in PLFaultCAT

absent variability on an event is straightforward; we analyze the influence of the variability being present within the product and not functioning as designed. If, however, the node relates to a commonality rather than a variability, we link the fault tree's leaf node with the appropriate commonality. Using PLFaultCAT makes associating a commonality and/or variability with a failure node straightforward. The PLFaultCAT interface, shown in Figure 1, allows labels for "Basic Event" nodes, depicted as circles, as a Commonality or Variability as well as defining a label or ID for the variability (i.e., the textbox under the heading "Variability ID").

The application of these three steps, using the SPL requirements documented in a CVA and a list of hazards to be mitigated, constructs a PL-SFTA that encapsulates the safety analyses of all product-line members. Figure 2 provides the PL-SFTA for the PAM MAS-PL for the hazard "Spacecraft to Asteroid Collision" depicting the failure nodes and the associated SPL requirement leaf nodes.

4.2 Application Engineering – Derivation and Reuse of Safety Analysis Artifacts

The development of a PL-SFTA, described in Section 4.1, enables the reuse of this safety analysis artifact to derive the SFT for a specific product-line member during the application engineering phase. We describe this process in this section.

Step 1. *Select the variabilities for a new product-line member.* A new product-line member is defined through the selection of which variabilities or features to include from the CVA. A product-line member is created by selecting the variabilities that it will contain and defining the values of the variabilities. PLFaultCAT supports the selection of a new product-line member's variability requirements by providing a checkbox window that presents all possible variabilities for the SPL. PL-SFTA does not itself enforce or check the dependencies prescribed in the CVA. Other tools, such as DECIMAL [5], are capable of enforcing the dependencies and constraints detailed in the CVA for large, complex SPLs. PLFaultCAT is used after the choice of variabilities has been determined to be legal.

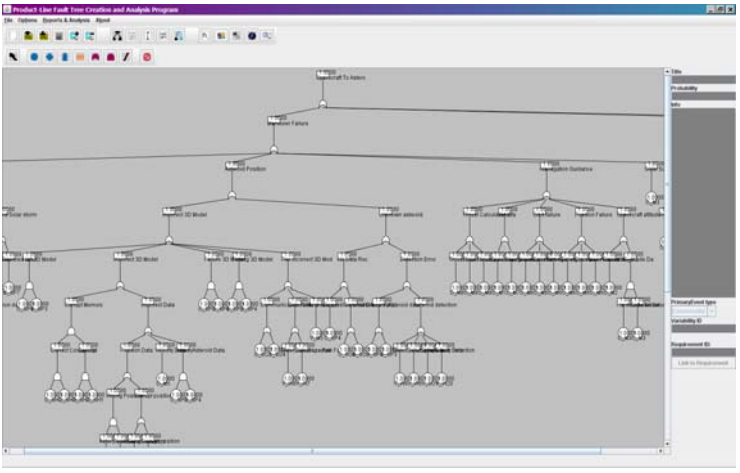

Fig. 2. PL-SFTA for the PAM MAS-PL hazard "Spacecraft to Asteroid Collision"

Step 2. *Generate the product-line member fault tree using PLFaultCAT*. After establishing and verifying a product-line member, we prune the product-line SFTA to create a baseline SFTA for the new system. The pruning process, described fully in [2] [6], first uses a depth-first search to automatically remove the subtrees that have no impact on the product-line member being considered and then relies on a small amount of domain knowledge to further collapse and prune the SFTA. A subtree within the PL-SFTA will have no impact if all the leaf nodes of that subtree contain variability requirements not included for the specific product-line member. The pruning errs on the side of caution to derive the product-line member's SFT from the PL-SFTA since it only marks the subtrees that can be removed without review and does not actually do any pruning. This is advantageous from a safety perspective because it simply indicates those subtrees where neither commonalities nor selected variabilities can be found in the subsequent children nodes. This algorithm then defers the actual pruning to the domain experts, as described in the next step. Figure 3 shows a portion of the resulting product-line member SFT derived from the PL-SFTA shown in Figure 2 for the PAM MAS-PL for the hazard "Spacecraft to Asteroid Collision.

Step 3. *Apply domain knowledge*. After removing the subtrees that had no bearing on the product-line member under consideration in Step 2, the fault tree may be able to be further pruned and/or collapsed within PLFaultCAT. However, this step requires domain knowledge and illustrates the limit to fully-automated PL-SFTA reuse. Removal of subtrees will often lead to orphaned logic gates or other opportunities to safely simplify the fault trees of a new product-line member. If removing orphaned OR gates when there is only one causal event remaining, we collapse the lower event into the parent event. If there is only one commonality or variability leaf node remaining, we attach it to the parent event and remove the OR gate. When AND gates are involved, more caution is required. Intuitively, if at least one input line to an AND gate is removed, the output event is impossible. However, it was found that this is not always the case, so each removal of an AND gate warrants further scrutiny. The

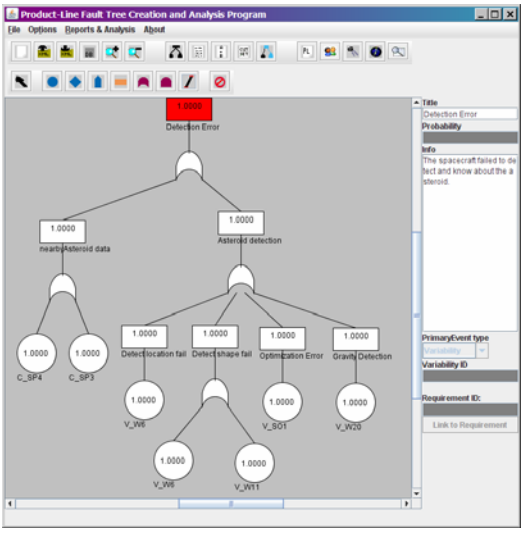

Fig. 3. Derived SFT from the PL-SFTA for a specific spacecraft in the PAM MAS-PL

clean-up of the derived SFTs for the new product-line member(s) presented in this step is a manual process and must be pursued with care. Enough information should be retained within the product-line member's fault tree for future hazard analysis and mitigation strategies. The application of domain knowledge helps in the derivation of the SFTs for a new product-line member by removing extraneous nodes and focusing attention on nodes that may contribute to failures in a specific product-line member. This provides additional assurance for the reused safety analysis asset.

The following section evaluates the claim that the application of these three steps, which reuse the PL-SFTA to derive specific safety analysis assets for a product-line member, provides safety analyses more efficiently and at a reduced cost.

5 Evaluation of Safety Analysis Asset Reuse

The application of the PL-SFTA to the PAM case study developed four fault trees for the hazards: 1. Spacecraft to Spacecraft Collision; 2. Spacecraft to Asteroid Collision; 3. Spacecraft Solar Storm Damage; and, 4. Failure to Detect Impending Solar Storm. The PL-SFTA associated 85.7% of the commonalities and 72.5% of the variabilities with at least one of the leaf nodes i the set of fault trees.

In the domain engineering phase, PLFaultCAT did not provide significant advantages over other tools in PAM beyond additional opportunity to embed textual hazard analysis information in the fault tree. This allowed a cross-check of the information provided in the fault tree to previously derived safety requirements.

In the application engineering phase, PLFaultCAT provided significant reuse advantages by exercising the pruning method outlined in Section 4. In the case study, approximately 54% of the failure nodes in the PL-SFTA were found to be common to

Table 1. Results of the Application of PL-SFTA to the PAM Case Study

Hazard	Total Failure Nodes	Common Failure Nodes	% Commonality Requirements	Core Reuse	PLFaultCAT Automation
Spacecraft to Asteroid Collision	82	64	88.5%	78.0%	72.2%
Spacecraft to Spacecraft Collision	84	61	63.8%	72.0%	82.1%
Spacecraft Solar Storm Damage	87	52	60.0%	59.8%	72.2%
Failure to Detect Solar Storm	91	13	6.7%	14.3%	93.8%

all 160 unique spacecraft of the PAM MAS-PL. That is, the minimum expected reuse of the PL-SFTA safety analysis asset for any given PAM spacecraft would be 54%.

Table 1 provides the results for each of the hazards examined using PL-SFTA for the PAM MAS-PL case study. The "Hazard" column describes the root node hazard of a fault tree; the "Total Failure Nodes" column shows the total number of failure nodes of a fault tree from Steps 1-3 described in Section 4; the "Common Failure Nodes" column gives the number of failure nodes that will be common to all product-line members of the PAM MAS-PL; the "% Commonality Requirements" is the percentage of the requirements associated to the leaf nodes of a fault tree that are commonality requirements; the "Core Reuse" column is the percentage of the failure nodes that are common to all product-line members of the PAM MAS-PL; and, finally, the "PLFaultCAT Automation" column shows the percentage of the nodes that could be safely and automatically pruned from the PL-SFTA using PLFaultCAT.

Although the overall reuse of the PL-SFTA in this study is approximately 54%, in most cases the reuse potential of a fault tree in the PL-SFTA was even higher, in the 60-80% range. The only exception was the "Failure to Detect a Solar Storm" fault tree which had a 14% minimum reuse potential. The reason is that the major contributing factor to a lower reuse potential for the safety analysis assets of the PL-SFTA is its relation to the requirements. The fault trees for "Spacecraft to Spacecraft Collision", "Spacecraft to Asteroid Collision" and "Spacecraft Solar Storm Damage" all had root nodes directly related to product-line commonalities. Since each of these requirements necessitates a PAM spacecraft to prevent the hazards outlined in these fault trees, all PAM spacecraft will be equipped with the functionality to prevent those hazards. As a result, a large portion of the PL-SFTA could be reused regardless of the specific configuration of the spacecraft. However, for a hazard that stems from a product-line variability, such as the "Failure to Detect Solar Storm" hazard, the reuse potential is much less since the failure of the product-line variability to mitigate against the hazard is only found in a subset of the product line's members.

This case study also found that, of failure nodes that could be safely pruned for a new product-line member, PLFaultCAT was able to automatically perform a minimum of 72% of the trimming without losing necessary information. Thus, 28% of the work was left to be done manually. This metric reflects the effort saved in reuse of the PL-SFTA safety analysis assets. The automation that PLFaultCAT does when pruning for a specific member(s) is sensitive to the number of Boolean AND gates in the fault tree. As a result of the conservative pruning approach, PLFaultCAT will not automatically remove the AND gates as a safety precaution. Thus, PLFaultCAT does more automated pruning for fault trees with fewer AND gates. Despite this, in the PAM case study the automation to manual effort was at least a 3:1 ratio.

These results compare favorably to those of an initial case study we performed in [6] on the Floating Weather Station (FWS) product line [13]. Unlike the study presented here, the FWS case study was a smaller, traditional SPL (i.e., not agent-based). In the FWS study, we found that a smaller portion of the PL-SFTA nodes, 45%, was common to all products of the product line. However, like the PAM case study, the FWS study found that PLFaultCAT was able to automatically prune 70% of the nodes that could safely be pruned. The difference in the percentage of common failure nodes in the two studies (45% for the FWS; 54% for PAM) is likely due to the difference in the types of applications in the case studies. That is, the results reported in the FWS study reflect the application of PL-SFTA to a single fault tree for a case study consisting of fewer than 20 requirements. More importantly however, is that the product-line members of the FWS did not share the same safety concerns as they did in the PAM study. Shared safety concerns seem to result in more reuse.

6 Discussion and Implications

The agent characteristics of the PAM case study as well as the types of variabilities that were present had a significant impact on our results. The PAM spacecraft will have the responsibility for protecting and healing itself from the possible risks of space exploration. For this reason, each spacecraft is to be equipped with the behavior to protect itself from the types of hazards modeled in the PL-SFTA. Further, the variabilities of the PAM MAS-PL reflected the differing types of scientific investigation possible on the spacecraft and had only a minor impact on the leaf nodes.

The implication of this result is that a PL-SFTA may best suit a MAS-PL rather than a traditional SPL since the agents of a MAS-PL will typically also include self-protecting and self-healing characteristics as commonalities and may have variabilities that are less likely to be safety-critical. For traditional SPLs that have few variabilities that will impact the safety of a system, a PL-SFTA can be applied to achieve comparable results found in this study. For traditional SPLs with a large number of variabilities that can impact the safety of a system, such as the FWS, the reuse of the PL-SFTA safety analysis assets is far more efficient, especially for large SPLs, than to serially construct SFTAs for each of the desired product-line members.

A concern for performing safety analysis on critical SPLs is whether the technique is scalable as the SPL grows more complex by incorporating more variabilities and product-line members. From the experience of applying the PL-SFTA to the PAM case study, it appears that our method and tool will scale adequately. This is because most of the added complexity in a large SPL lies in the domain engineering phase when the PL-SFTA is constructed. In [4], we provide a structured process to construct the SFMECA for a MAS-PL from the requirement specifications of the Gaia-PL methodology [2] [3]. Since the construction of the PL-SFTA relies heavily on the aid of a SFMECA, the scalability is at least as robust as that of the SFMECA.

7 Concluding Remarks

This paper described and evaluated the structured reuse of SFTA documents for a product line of collaborating spacecraft developed as a multi-agent system. Results

showed that reuse of the PL-SFTA assets across this safety-critical product line reduced effort over serial construction of SFTAs for each spacecraft, with 54% of the failure nodes common to the 160 unique spacecraft. The implication of this for other safety-critical product lines is that SPLs built as multi-agent systems, which often incorporate adaptive and self-healing behaviors as commonalities, fit especially well with PL-SFTA reuse. More generally, the PL-SFTA evaluation recorded advantages in expanded reuse of safety-analysis assets across a safety-critical SPL.

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

References

[1] Clements, P., Northrop, L.: Software Product Lines. Addison-Wesley, Boston (2002)
[2] Dehlinger, J.: Incorporating Product-Line Engineering Techniques into Agent-Oriented Software Engineering for Efficiently Building Safety Critical Multi-Agent Systems, Ph.D. Thesis. Iowa State University (2007)
[3] Dehlinger, J., Lutz, R.R.: A Product-Line Approach to Promote Asset Reuse in Multi-Agent Systems. In: Garcia, A., Choren, R., Lucena, C., Giorgini, P., Holvoet, T., Romanovsky, A. (eds.) SELMAS 2005. LNCS, vol. 3914, pp. 161–178. Springer, Heidelberg (2006)
[4] Dehlinger, J., Lutz, R.R.: Bi-Directional Safety Analysis for Product-Line, Multi-Agent Systems. In: ACM SIGBED Review: Special Issues on Workshop Innovative Techniques for Certification of Embedded Systems, vol. 3(4) (2006)
[5] Dehlinger, J., Humphrey, M., Padmanabahn, P., Lutz, R.R.: Decimal and PLFaultCAT: From Product-Line Requirements to Product-Line Member Software Fault Trees. In: 29th International Conference on Software Engineering Companion, Minneapolis, MN, pp. 49–50 (2007)
[6] Dehlinger, J., Lutz, R.R.: PLFaultCAT: A Product-Line Software Fault Tree Analysis Tool. Automated Software Engineering Journal 13(1), 169–193 (2006)
[7] Feng, Q., Lutz, R.R.: Bi-Directional Safety Analysis of Product Lines. Journal of Systems and Software 78(2), 111–127 (2005)
[8] Leveson, N.G.: Safeware: System Safety and Computers. Addison-Wesley, Boston (1995)
[9] Leveson, N.G., Weiss, K.A.: Making Embedded Software Reuse Practical and Safe. In: ACM SIGSOFT Software Engineering Notes, pp. 171–178 (2004)
[10] Liu, J., Dehlinger, J., Lutz, R.: Safety Analysis of Software Product Lines Using State-Based Modeling. Journal of Systems and Software 80(11), 1879–1892 (2007)
[11] Schwanke, R., Lutz, R.: Experience with the Architectural Design of a Modest Product Family. Journal of Software Practice and Experience 34(13), 1273–1296 (2004)
[12] Sterritt, R., Rouff, C., Rash, J., Truszkowski, W., Hinchey, M.: Self-* Properties in NASA Missions. In: Proceedings International Conference on Software Engineering Research and Practice, Las Vegas, NV, pp. 66–72 (2005)
[13] Weiss, D.M., Lai, C.T.R.: Software Product Line Engineering: A Family-Based Software Development Process. Addison-Wesley, Boston (1999)