Mechatronics 24 (2014) 231–240

Contents lists available at ScienceDirect

Mechatronics

journal homepage: www.elsevier.com/locate/mechatronics

Safety analysis of mechatronic product lines

Seppo Sierla [a], Bryan M. O'Halloran [b], Heikki Nikula [a], Nikolaos Papakonstantinou [a,*], Irem Y. Tumer [b]

[a] Department of Automation & Systems Technology, Aalto University, FI-00076 AALTO, Espoo, Finland
[b] School of Mechanical, Industrial, & Manufacturing Engineering, Oregon State University, 204 Rogers Hall, Corvallis, OR 97331, USA

ARTICLE INFO

Article history:
Received 13 December 2012
Accepted 3 February 2014
Available online 3 March 2014

Keywords:
Functional Failure Identification and Propagation
Safety
Risk analysis
Product line
Product platform

ABSTRACT

Most methodologies for the design and analysis of mechatronic systems target a single product. From a business perspective, successful product development requires shortening development times, reducing engineering costs and offering a greater variety of product options for customers. In software engineering, the software product line (SPL) technology has been developed to meet these conflicting goals, and several major companies have reported success stories resulting from SPL adoption. In mechanical engineering, similar methodologies have been developed under the name of product platforms. Methodologies for analyzing product qualities such as safety or reliability have been introduced for both SPL and product platforms. The problem with these methodologies is that they consider either software or mechanical product design, so they do not guide developers to find the best balance between the controller and the equipment to be controlled. Several system properties of a mechatronic product line should be investigated with mechatronic analysis methodologies before the development process branches to software, electronic and mechanical design. In particular, safety is one system property that can only be analyzed by considering both the equipment and its controller, so mechatronic methodologies early in the design are advantageous for discovering safety-related design constraints before costly design commitments are made. This paper extends the Functional Failure Identification and Propagation (FFIP) framework to the safety analysis of a mechatronic product line with options in software signal connections and equipment. The result of applying FFIP is that unsafe combinations of options are removed from the product line.

1. Introduction

A fundamental discovery in the field of mechatronics is that conceptual designs should be analyzed against criteria such as cost, performance, safety or weight before the design process separates to unidisciplinary branches such as software, electronic and mechanical engineering. Numerous modeling and architectural approaches have been proposed for the purpose of finding the optimal design through interdisciplinary collaboration [1,2] and for the purpose of shortening the time to market [3]. However, such approaches are challenged by the product line technology that is establishing itself in single discipline engineering, especially for software. Several years ago, the software product line (SPL) technology became a breakthrough solution to the problem of providing a broader range of customer options with decreased development cost and time [4]. In SPL, the architectural focus has been on explicitly specifying which modules are common to all products and which modules realize customer options. Technology for generating executable software products based on a chosen set of customer options has been in place for several years and has enabled successful adopters of SPL to seize market dominance [5]. Organizations developing software-intensive mechatronic products are thus forced to choose between the benefits of either a mechatronic architecture or a SPL architecture, as the product line paradigm has not yet established itself in mechatronic research.

If developers of software-intensive safety-critical products decide to adopt a product line approach, available methodologies focus on safety issues arising from the software itself, such as ensuring that software features do not interact to bring the system to an unsafe state [6] or exhaustive verification of product instances for certain required properties [7]. The need for verification of software requirements against system safety requirements is recognized and addressed by applying safety analysis methods such as Software Fault Tree Analysis (SFTA) and software failure mode, effects and criticality analysis (SFMECA) on the SPL [8], but this avoids the more fundamental question of how to perform safety analysis of the product line at the system level in order to decide what functionality is implemented in software, what are

* Corresponding author. Tel.: +358 465669948.
E-mail addresses: nikolaos.papakonstantinou@aalto.fi (N. Papakonstantinou), irem.tumer@oregonstate.edu (I.Y. Tumer).

http://dx.doi.org/10.1016/j.mechatronics.2014.02.003
0957-4158/

the potentially hazardous interactions between this software and the rest of the system, and how this knowledge is used to define constraints on valid combinations of features in the SPL.

Feature modeling is an established technique in SPL to describe commonality and variability as a tree structure, usually only for software, but sometimes also for other system parts such as sensors [9]. It is here posited that the feature model should be defined for a system consisting of software, electronic and mechanical parts, and that this model should be subjected to a safety analysis, so that unsafe feature combinations are eliminated. The resulting feature model can then be handed to software developers, and the safety of the software design and implementation, even in the face of evolution and maintenance pressures, can be addressed with the existing body of research in SPL. A recent advance toward a mechatronic direction has been the integration of AADL (Architecture Analysis & Design Language) to Fault Tree Analysis (FTA) [10], but this is still limited to the software and execution hardware. Methods such as FTA, SFTA and SFMECA, which have been applied in the context of safety critical SPL [6,8,10], are limited by the human user's ability to identify hazardous fault propagation paths, and model-based frameworks are needed for the safety analysis of increasingly complex systems [11].

Major corporations developing products with embedded software have recently reported productivity increases of an order of magnitude as a result of SPL adoption [12], so the widespread use of mechatronic design and analysis techniques in such organizations requires supporting the product line paradigm. Previously, the authors have proposed a safety analysis method, Functional Failure Identification and Propagation (FFIP), for a single mechatronic product [13]. The goal of this paper is to transition FFIP to a product line approach. This paper builds directly on the authors' earlier publication [13], so that the input of the analysis is a product line expressed as a feature model, and the output is another feature model from which the unsafe product variants have been removed. The idea of using feature models to describe design alternatives that are filtered by FFIP has been presented in conferences [14,15]. This paper extends the approach with a semi-automated workflow and toolchain for processing a product line expressed as a feature model.

2. Background

2.1. Safety and reliability analysis methods

A vast set of tools and methods exist in reliability engineering used during the design process. Standard methods such as Fault Tree Analysis (FTA), Failure Modes and Effects Analysis (FMEA), Event Tree Analysis (ETA), Reliability Block Diagram (RBD), and Probabilistic Risk Assessment (PRA), require designers to have a detailed knowledge of the system and a significant amount of useful data to support each analysis. Without significant advancements, these methods are only applicable to a higher fidelity level of design [16]. Several relatable methods have been developed to FMEA to mitigate the risk of failure modes. The Function Failure Design Method (FFDM) [17–19], originally introduced as an alternative to FMEA, is a knowledge repository based approach where historical component failure modes are linked to specific function-flow pairs. This allows a designer to infer about the relative occurrence of failure modes for different function-flow pairs. Since historical failure propagation data is configuration specific, the FFDM method is limited to single failure impact analysis. The Risk in Early Design (RED) extends FFDM by presenting a method of formulating functional-failure likelihood and consequence based risk assessment classified as high-risk to low-risk function-failure combinations [20–22]. The Function Failure Rate Design Method (FFRDM) [19], a separate extension from FFDM, categorizes failure modes using likelihood. This is accomplished by implementing component failure rate data to normalize the amount of time a component is used. While these methods capture failure mode likelihood of occurrence, likelihood of detection, and severity, they lack the ability to quantify functional impact.

The motivation to understand and evaluate the impact of potential failures in early design has led to several approaches that capture the functional effects of a fault. In this paper, a fault is defined as an abnormal condition or defect of a component that may lead to a failure. Early work by Wang and Jin has focused on describing the nature of faults from the conceptual design perspective [23], leading to other methods that show how those faults may affect the performance of components in the system [24–26]. These methods take a qualitative perspective in order to be more applicable in early design. Quantitative methods use descriptions of fault probability to provide a risk assessment at the early design stage [17,21,27]. While qualitative methods define the impact of failures, quantitative methods identify the likelihood of failures and evaluate failures in terms of the product's or system's ability to perform a desired functionality. O'Halloran et al. have developed a framework to calculate a minimum, maximum, and weighted average system reliability during functional design [28]. The result of this calculation is then used to reason about how to improve low reliability values by adding redundancies and investigating new functionality. Hata et al. identify failure modes by representing functions with streams and a network structure [27]. Historical data and rules are used to determine the impact to the functional model from a defect. This method cannot handle multiple failures, dynamic behavior, and requires a large amount of input on potential failures in the design prior to the evaluation. The method presented in this paper qualitatively determines the functional effect by simulating behavior. The extended FFIP method in this paper has the specific advantage that is uses a dynamic simulation and thus can capture the impact of faults over a specified time. The dynamic simulation normalizes the functional impact for a set length of time and for a set of design alternatives. The results of a functional impact are specific to a design alternative and are used to reason about the best design.

In addition to addressing the impact of faults, a variety of fault propagation methods have been introduced in reliability engineering. These aim to understand and isolate component failures [29], mitigate faults and errors in software [29–32], and mitigate faults in both hardware and software [33,34]. Krus and Grantham present a failure propagation analysis method developed as a direct extension of the previously mentioned FFDM and RED method to capture the failure propagation using a functional model [35]. Wang and Jin take a different approach to failure propagation by using a function event network to analyze failure propagation and assign statistical reliability measure to each functional failure [23]. Huang and Jin extend this work to the Conceptual Stress and Conceptual Strength Interference Theory (CSCSIT) by relating conceptual stress to flow and conceptual strength to function [25]. While the results are qualitative, this work uses historical data to capture early reliability and does not capture emergent behavior because it is recorded for a single component failure. The extended FFIP method in this paper uses behavior for each component and simulates the design to show failure propagation paths. Table 1 is used to summarize the closely related methods in this section.

2.2. The function failure identification and propagation framework

The basic FFIP framework [36] consists of three major elements: the system representation, simulator, and reasoner. The system representation consists of a functional model and a configuration flow graph (CFG). The functional model captures design intent. A