Reliability Engineering and System Safety 132 (2014) 20–35

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

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

Spacecraft early design validation using formal methods

Marco Bozzano [a], Alessandro Cimatti [a], Joost-Pieter Katoen [b], Panagiotis Katsaros [d], Konstantinos Mokos [c,2], Viet Yen Nguyen [c,*,3], Thomas Noll [b], Bart Postma [c,1], Marco Roveri [a]

[a] Fondazione Bruno Kessler, Via Sommarive 18, Povo, 38123 Trento, Italy
[b] RWTH Aachen University, Software Modeling and Verification Group, 52056 Aachen, Germany
[c] European Space Agency, Postbus 299, 2200 AG Noordwijk, The Netherlands
[d] Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece

ARTICLE INFO

Article history:
Received 19 December 2013
Received in revised form
29 April 2014
Accepted 3 July 2014
Available online 17 July 2014

Keywords:
Safety and dependability analysis
Performance analysis
Model checking
AADL modeling language
Spacecraft

ABSTRACT

The size and complexity of software in spacecraft is increasing exponentially, and this trend complicates its validation within the context of the overall spacecraft system. Current validation methods are labor-intensive as they rely on manual analysis, review and inspection. For future space missions, we developed – with challenging requirements from the European space industry – a novel modeling language and toolset for a (semi-)automated validation approach. Our modeling language is a dialect of AADL and enables engineers to express the system, the software, and their reliability aspects. The COMPASS toolset utilizes state-of-the-art model checking techniques, both qualitative and probabilistic, for the analysis of requirements related to functional correctness, safety, dependability and performance. Several pilot projects have been performed by industry, with two of them having focused on the system-level of a satellite platform in development. Our efforts resulted in a significant advancement of validating spacecraft designs from several perspectives, using a single integrated system model. The associated technology readiness level increased from level 1 (basic concepts and ideas) to early level 4 (laboratory-tested).

1. Introduction

A spacecraft is a machine that fulfills mission objectives outside of Earth. Spacecraft design involves a vast body of natural sciences and many engineering fields, including, but not completely covering, materials, optics, power, propulsion, performance, reliability and security engineering. These disciplines are integrated into an interdisciplinary field known as *systems engineering* that addresses the design of systems and the management of complex engineering projects over their life-cycle. Space systems engineering [1] is an evolving field and its current state of practice is strongly influenced by a relatively new engineering discipline, namely that of software development.

Spacecraft in the early space age included software whose size was at most a few dozens of lines of code. The advent of digital interfaces of parts and equipment, and the flexibility of software-based control over analogue interfaces and electrical/mechanical control led to an exponential growth of the size of the deployed software [2]. Nowadays, the latter is compiled from millions lines of code. Hence, software dictates the overall spacecraft behavior to an ever-increasing degree. This is also reflected within the space systems engineering life-cycle. More emphasis is now given to the system–software perspective that encompasses the interaction between the software and the remainder of the system, typically perceived by the software engineers as hardware.

The COMPASS project [3] advances the system–software perspective by providing means for its validation in the early design phases, such that system architecture, software architecture, and their interfacing requirements are aligned with the overall functional intents and risk tolerances. Validation in the current practice is labor-intensive and consists mostly of manual analysis, review and inspection. We improve upon this by adopting a *model-based approach* using formal methods. In COMPASS, the system, the software and its reliability models are expressed in a single modeling language. This language originated from the need for a language with a rigorous formal semantics, and it is a dialect of the Architecture Analysis & Design Language (AADL). Models expressed in our AADL dialect are processed by the COMPASS toolset that automates analyses which are currently done manually. The automated analyses allow studying functional correctness

* Corresponding author.
E-mail address: vietyen.nguyen@iese.fraunhofer.de (V.Y. Nguyen).
[1] Present affiliation: Sogeti, Netherlands.
[2] Present affiliation: Terma A/S, Denmark.
[3] Present affiliation: Fraunhofer IESE, Germany.

http://dx.doi.org/10.1016/j.ress.2014.07.003
0951-8320/

of discrete, real-time and hybrid aspects under degraded modes of operation, generating safety and dependability validation artifacts, performing probabilistic risk assessments, and evaluating effectiveness of fault management. The analyses are mapped onto discrete, symbolic and probabilistic model checkers, but all of them are completely hidden away from the user by appropriate model-transformations. The COMPASS toolset is thus providing an easy-to-use push-button analysis technology.

The first ideas and concepts for the development of the COMPASS toolset emerged in 2007, due to a series of significant advances in model checking [4], and especially in its probabilistic counterpart [5]. These advances opened prospects for an integrated model-based approach towards system–software correctness validation, safety and dependability assessment and performance evaluation during the design phase. Its technology readiness level was estimated at level 1, i.e. basic principles were observed and reported. The European Space Agency (ESA) issued a statement of work to improve system–software co-engineering and this was commissioned to the COMPASS consortium consisting of RWTH Aachen University, Fondazione Bruno Kessler and Thales Alenia Space. Development started soon after, and in 2009 a COMPASS toolset prototype was delivered to the European space industry. Maturation was followed by subsystem-level case studies performed by Thales Alenia Space [6]. As of 2012, two large pilot projects took place in ESA for a spacecraft in development. This marked the maturation of the COMPASS toolset to early level 4, namely laboratory-tested. The novel contribution of this paper is the report on the second pilot project. This paper furthermore summarizes the background work and the first pilot project, whose results were published elsewhere [7]. Altogether, it describes the current state of the art in system–software spacecraft co-engineering, ranging from the used techniques, to the tools and the conducted industrial projects.

This paper is organized as follows. A brief overview of space systems engineering is given in Section 2, which is followed by an introduction to the developed modeling language (Section 3), the toolset (Section 4) and its analyses. The spacecraft platform is described in Section 5, and the pilot projects are presented in Sections 6 and 7. The paper wraps up with the related work (Section 8) and the conclusions (Section 9).

2. Space systems engineering

The European tradition and practice of spacecraft engineering is codified in the ECSS standards [9] issued by the European Space Agency. The spacecraft system life-cycle is depicted in Fig. 1. It starts with mission analysis in phase 0. In phase A, management and engineering plans are set up and functional aspects and feasibility are investigated. During phase B, a preliminary system design is drafted and reviewed. In the phases that follow, the system design is refined to its implementation and the system is verified, launched and operated. In the early phases and most notably in phase B, the system is decomposed into its constituent parts, including the thermal, power, attitude control subsystems and the software. Experts on the respective engineering discipline further refine the subsystems, while system engineers ensure coherency among them.

Increasing software functionality. Software plays an increasing and vital role in the overall system that is evident from the exponential growth of the source code sizes in modern spacecraft [2]. Today, software accompanies modern microprocessors in order to provide unprecedented functionality for an ever increasing range of mission demands. For example, navigation systems are equipped with software that delivers strict orbit control for improved precision. Also, on-board software handles enormous mission data sets generated by the high-resolution sensors used in Earth observation satellites. Dedicated software implements functionality that addresses key reliability and autonomy requirements. Especially for deep space missions, where communication windows are short and delays are long, autonomy in terms of survivability is essential to mission success. This is mainly achieved by a *fault management system* [10], the major part of which is realized through software. Supported functions include monitoring, detection, isolation and mitigation of spacecraft faults that may occur due to the harsh space conditions (mechanical stress, wear and radiation). Functional correctness, safety, and

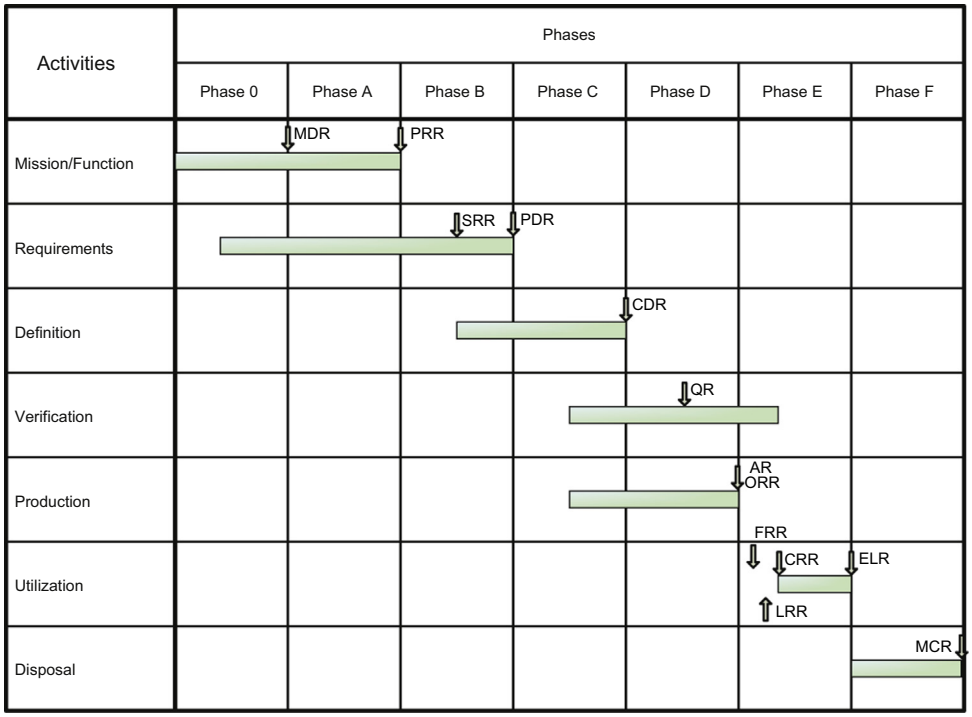

Fig. 1. Space systems engineering life-cycle of European missions. *Source*: ECSS Standard on Project Planning and Implementation [8].