Computer Standards & Interfaces 48 (2016) 71–79

Contents lists available at ScienceDirect

Computer Standards & Interfaces

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

Standards-based metamodel for the management of goals, risks and evidences in critical systems development

Xabier Larrucea [a,*], Cesar Gonzalez-Perez [b], Tom McBride [c], Brian Henderson-Sellers [1]

[a] Tecnalia Research & Innovation, Parque Tecnológico de Zamudio, Ibaizabal Bidea, edificio 202, 48170 Zamudio, Bizkaia, Spain
[b] Institute of Heritage Sciences (Incipit), Spanish National Research Council (CSIC), San Roque, 2, 15704 Santiago de Compostela, Spain
[c] Faculty of Engineering and Information Technology, University of Technology Sydney, PO Box 123, Broadway, NSW 2007, Australia

ARTICLE INFO

Article history:
Received 28 January 2016
Received in revised form 11 April 2016
Accepted 24 April 2016
Available online 26 April 2016

Keywords:
ISO/IEC 24744
ISO/IEC 15026
ISO 26262

ABSTRACT

Safety critical system development includes a wide set of techniques, methods and tools for assuring system safety. The concept of evidence is one of the key notions used to provide safety confidence to stakeholders. Safety goals must be identified during safety analysis. In addition, risks should also be considered and managed, and linked to the achievement of safety goals. This paper proposes an extension of the ISO/IEC 24744 metamodel for development methodologies in order to integrate the management of goals, risks and evidence into system development lifecycles in an ISO/IEC 15026-compliant manner that is related to the approach of assurance cases. The proposed extension is illustrated through a real-life scenario in the automotive domain where the system being developed must comply with ISO 26262, a standard in this domain. By using the proposed approach, the management of goals, risks and evidence in critical systems development is formalized and harmonized with different ISO/IEC standards, resulting in a more robust and systematic treatment of these crucial aspects.

1. Introduction

Nowadays, software systems have an increasingly relevant role in safety critical scenarios [1], with most of these systems depending on the careful engineering of its software elements [2,3]. A widely adopted approach to ensure that a software-intensive system is safe is to seek evidence that demonstrates that the system, as a whole, reaches a specific confidence in terms of safety [4]. In this regard, safety analysis is crucial as part of requirements analysis [3] in order to identify safety goals and understand the risks and hazards involved [5].

Safety analysis, usually linked to the strategic, functional and non-functional goals of the system, is performed by considering the relevant evidence [6] that may or may not support each goal. Proving the very functionality of the system is often evidence-based [7,8], and industrial experiences such as [7] highlight the relevance of evidence indicating that "*Product-based evidence should show that the system has the required safe behaviour.*" Hence, evidence constitutes the cornerstone to show that a system conforms to a particular standard [4]; this is the case of the automotive industry where software plays an increasingly important role in subsystems such as braking, parking assistance or fuel level estimation. Indeed, ISO 26262 [9] was released in 2011 to cover a complete safety-driven development life cycle for road vehicle construction. This standard comprises 10 parts covering systems and software development; part 9, specifically, defines the objectives of safety analysis, which aims to minimize "*the consequences of faults and failures on the functions, behaviour and design of items, and elements*" and contributes to the "*identification of new functional or non-functional hazards not previously considered during hazard analysis and risk assessment*" [9]. In fact, hazard analysis and risk assessment methods are used to identify and categorize hazardous events and specify safety goals [10]. Safety goals, in turn, are usually characterized in standards such as ISO 26262. Safety and risk analysis, goals and evidence are also used in domains other than automotive such as aerospace [11,12]. ISO/IEC 15026 [13] is concerned with software and systems assurance generally, taking an approach that assurance is provided through a claim that has been or will be achieved with some level of certainty based on evidence. Part 4 of ISO/IEC 15026 defines a systems assurance process view for which the purpose is "to achieve the assurance claims regarding the system properties selected for special attention and to provide a body of information showing the achievement of those claims." The systems assurance view adds some requirements to existing systems and software development processes to ensure that some specific activities are performed and relevant evidence is made available. Finally, it is worth saying that, as stated in [5], safety-related activities such as those described here are part of project management, and therefore we argue that these

* Corresponding author.
E-mail addresses: xabier.larrucea@tecnalia.com (X. Larrucea), cesar.gonzalez-perez@incipit.csic.es (C. Gonzalez-Perez), Tom.McBride@uts.edu.au (T. McBride), brianhs@me.com (B. Henderson-Sellers).
[1] Retired.

http://dx.doi.org/10.1016/j.csi.2016.04.004
0920-5489/© 2016 Elsevier B.V. All rights reserved.

activities should not be considered in isolation but in close connection to other project management and technical activities.

Also in the same context, assessment [2] and certification [14] activities of safety critical systems not only provide a means to demonstrate that a system is compliant with a set of standards [4], but also confer a certain degree of confidence to stakeholders. This has been shown to be the case, in particular, for the aviation [15] and automotive [16] domains; in the latter, the term "assessment" is preferred instead of "certification" with the same meaning. We also note that a domain-specific Automotive SPICE [17] has been developed. Thus, assessments as defined by ISO/IEC 15504 (i.e. SPICE) [18] have been used in these safety contexts but incorporating domain-specific variations [19]. More recently, technical report ISO/IEC TR 33014 [20] provides a guide for process improvement, and mentions objective evidence as result of the implementation of processes, goals as part of the different levels defined in this technical report and risks associated with the application of processes.

It is therefore clear, from the abundant literature, that *evidence*, *goals* and *risks* are three major concepts that need to be considered in safety critical software-intensive system development, regardless of the specific domain or standards being adopted. In this regard, it would be desirable that a common understanding be reached about what these central concepts mean, how they relate to each other and to other concepts, and how they can be applied to specific domains in order to comply with specific standards. The problem of "harmonization plus refinement" has been already raised [21], and a tentative solution proposed [22] that allows for the creation of an abstract ontology that is refined in a per-domain and per-standard basis. Currently, efforts are being carried out within a special working group of ISO/IEC JTC1 SC7 where two of the authors are the co-chairs of this group, to develop an ontological infrastructure for the whole SC7, organized as follows. At the most abstract level, a Definitional Elements Ontology (DEO) will contain a minimal set of core and abstract concepts that are common to all the relevant domains and situations. From this DEO, a number of Configured Definitional Ontologies (CDOs) can be obtained; these are specific configurations of the DEO, e.g. by adding specialized concepts or removing unwanted ones. Finally, Standard Domain Ontologies (SDOs) can be created by instantiating the CDOs; an SDO usually constitutes a specific standard or domain- and situation-specific approach. Although this work is being carried out within ISO/IEC JTC1 SC7, the general approach of the solution is not tied to any particular domain, and we propose here to adopt it for the definition and further refinement of the above identified core concepts of evidence, goal and risk.

Preliminary results of the SC7 ontology working group show that the major inputs for said ontology will be ISO/IEC 24744 [23] and ISO/IEC/IEEE 24765 (SEVOCAB) [24]. Neither of these includes concepts such as evidence, goal and risk; this means that these concepts would need to be incorporated into the DEO, in one form or another, if they were to be used in software development efforts. For this reason, we propose in this paper an extension to ISO/IEC 24744 that incorporates the necessary constructs to incorporate evidence, goals and risks into any software development effort, and to do this in a fashion that is integrated with other aspects of the development endeavour. Also, this extension is proposed in a way such that it could be easily adopted as part of the future SC7 DEO and refined as needed into CDOs and SDOs for e.g. the automotive domain through ISO 26262.

The remainder of this paper is structured as follows. Section 2 provides some background on the ISO/IEC 24744 international standard, which is adopted as a conceptual framework for our work, as well as previous works on goals, risks and evidence. Section 3 describes the proposed solution, namely a detailed specification of the ISO/IEC 24744 extension for evidence, goals and risks. Section 4 validates this solution by constructing first a sample CDO for the automotive domain, then instantiating this CDO into a subset of ISO 26262, and then simulating a safety assessment process on top of a trace of the resulting SDO. Finally, Section 5 presents a discussion of the benefits of the proposed approach as well as its limitations, and suggests some future improvements.

2. Background

2.1. Standards

2.1.1. ISO/IEC 24744

ISO/IEC 24744 "Software Engineering — Metamodel for Development Methodologies" [23] is an international standard created from the software and systems engineering fields. It includes a UML-expressed metamodel that defines the following key concepts as part of any methodology:

- *Work units*. "*A work unit is a job performed, or intended to be performed, within an endeavour*" [23]. There are different types of work units: *processes* (*large-grained work units that operate within a given area of expertise* [23], such as "Quality Assurance" or "Requirements Engineering"), *tasks* (*small-grained work units that focus on what must be done in order to achieve a given purpose* [23], such as "Prepare unit tests" or "Validate requirements against stakeholders"), and *techniques (small-grained work units that focus on how the given purpose may be achieved* [23], such as "CRC Cards", or "Focus Groups").
- *Work products*. A work product is a thing of interest for the endeavour, either because it is used as an input to the process (such as a bid document), or because it is created as an interim or final result (such as a requirements specification or the software itself). There are different types of work products, the most relevant ones being *documents* (durable depictions of a fragment of reality [23]) and *models* (abstract representations of some subject that acts as the subject's surrogate for some well-defined purpose [23]).
- *Producers*. A producer is *an agent that has the responsibility to execute work units* [23]. There are different types of producers, the major ones being *people* (*individual human beings involved in the endeavour* [23]); *teams* (*organized sets of producers that collectively focus on common work units* [23], such as "the quality assurance team"); and *tools* (*instruments that help other producers to execute their responsibilities in an automated way* [23], such as "the bug-tracking system").
- *Stages*. A stage is a *managed time frame within an endeavour* [23]. Work units only specify what is supposed to be done, but they do not say when; stages, on the contrary, establish the temporal framework for work units. There are different types of stages, the most relevant ones being *phases* (managed intervals of time during which the endeavour changes levels of abstraction, such as "Inception", "Development" or "Maintenance"); and *milestones* (managed points in time that *mark a significant event within the endeavour* [23], such as "Visual Freeze" or "Requirements Signed Off").

Also, ISO/IEC 24744 defines the concept of *action* as the mechanism by which tasks cause effects on work products, and that these effects may be of several types: create, modify, read only or delete.

Finally, it is worth mentioning that ISO/IEC 24744 departs from the usual OMG-centric "strict metamodelling" approach and instead employs a more powerful architecture based on powertypes to represent both the methodology and the enacted endeavour at the same time within a single metamodel [25,26]. By using this approach, extension guidelines are provided as part of the standard so that customized variants can be constructed by combining regular object-oriented specialization and instantiation of classes in the metamodel [27].

It is beyond the scope of this paper to make a complete description of ISO/IEC 24744; please see [23] for additional information.