Computers in Industry 82 (2016) 1–18

Contents lists available at ScienceDirect

Computers in Industry

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

Writing and verifying interoperability requirements: Application to collaborative processes

N. Daclin[a,*], S. Mallek Daclin[a], V. Chapurlat[a], B. Vallespir[b,c]

[a] LGI2P—Laboratoire de Génie Informatique et d'Ingénierie de Production Site de l'Ecole des Mines d'Alès, Parc Scientifique G. Besse, 30035 Nîmes cedex 1, France
[b] Univ. Bordeaux, IMS, UMR 5218, F-33400 Talence, France
[c] CNRS, IMS, UMR 5218, F-33400 Talence, France

ARTICLE INFO

Article history:
Received 10 April 2015
Received in revised form 5 April 2016
Accepted 11 April 2016
Available online xxx

Keywords:
Interoperability requirements
Repository for interoperability requirements
Requirements verification
Domain Specific Language

ABSTRACT

Interoperability analysis is highly correlated with interoperability requirements, the ability to grasp, structure, author and verify such requirements has become fundamental to the analytical process. To this end, requirements must be: (1) properly submitted in a suitable and usable repository; (2) written correctly by stakeholders with relevance to the studied domain; and (3) as easily verifiable as possible on various models of the system for which interoperability capabilities are being requested. The purpose of this article is to present both a structured repository for interoperability requirements and a Domain Specific Language to write and verify interoperability requirements – within a collaborative process model – using formal verification techniques. The interoperability requirements repository, which serves to structure interoperability requirements and make them available, is itself structured through abstraction levels, views and interoperability life cycle dimensions. Additional parameters detailing the requested information and the known impacts of requirements on behavior of the studied system have also been included. The Domain Specific Language provides the means for writing interoperability requirements. Afterwards, these requirements – more specifically the temporal requirements – are re-written into properties by transforming the temporal logic TCTL to allow for their effective verification by using the model checker UPPAAL. The overall approach is illustrated in a case study based on a collaborative drug circulation process. The article also draws conclusions and offers an outlook for future research and application efforts

1. Introduction

In the field of Systems Engineering (SE) [1], like in any specialized engineering field (mechanical, Information Systems, mechatronics, etc.), requirements engineering is a critical activity dedicated to ensuring that a given system meets all expressed requirements (i.e. original requirements corresponding to stakeholder[1] expectations and prescriptions, as well as requirements induced by technical choices and decisions throughout the system design phase).

For one thing, a requirement assigns, without ambiguity and in a coherent manner, designers' tasks and constraints when devising a solution. A requirement can be described using standards [2], reference models and vocabularies [3]. However, the existing vocabulary and requirement checklists commonly adopted in the SE field tend to be understandable, yet perfectible and abstract when taking a particular category of requirements into account (e.g. non-functional requirements such as interoperability) or a particular system (e.g. enterprises involved in collaborative processes). For another thing, the interoperability concept [4] remains a key factor of success for enterprises that share and exchange processes, service data and resources in a collaborative context; moreover, a number of existing works have sought to characterize [5], implement [6] and assess this very concept [7–9]. Nevertheless, compliance with interoperability requirements within a partnership is neglected and constitutes a challenge that can provide: (1) structure to interoperability requirements, (2) a

* Corresponding author.
E-mail addresses: surname.name@mines-ales.fr (N. Daclin), bruno.vallespir@ims-bordeaux.fr (B. Vallespir).

[1] We have adopted the definition in [3] that defines a stakeholder as a "*party having a right, share or claim in a system or in its possession of characteristics that meets said party's needs and expectations*".

http://dx.doi.org/10.1016/j.compind.2016.04.001
0166-3615/

means for expressing requirements, and (3) the tools needed to detect possible interoperability flaws.

The paper's focus is threefold: grasping and structuring interoperability requirements, identifying the means of writing such requirements, and performing verifications with formal techniques [10]. The definition and verification of requirements involves a collaborative process model for the purpose of highlighting interoperability issues that may lead to dysfunctions and interfacing problems from technical, organizational (including human) and conceptual points of view. Following this brief introduction, the problem statement and expected outcomes will be presented. The relevant research work will be provided and discussed in Section 3. Section 4 will then lay out the proposed repository for interoperability requirements, featuring its dimensions, their relationships and the steps allowing for its utilization. Section 5 will offer a case study to illustrate the value of such an approach; lastly, Section 6 will assess the approach and its possible enhancements.

2. Interoperability requirements engineering

2.1. Problem statement and expected outcomes

Nowadays, a technical problem involving interoperability in the fields of computer science and Information and Communication Technologies [11] basically includes organizational aspects (i.e. are the organization and its personnel able to collaborate efficiently?) and conceptual aspects (do the data being exchanged share a common semantics?) [12–14]. While initially focused exclusively on data exchange and sharing, topics such as process interoperability [15] are also receiving consideration at present. Moreover, interoperability encompasses other aspects, like for instance the interfacing issue, which extends to the autonomy and reversibility of partners involved in the collaborative venture [16]. Interoperability therefore is an important and mandatory capability to ensure effectiveness in terms of: exchanging and sharing information, products and resources; aligning and orchestrating collaborative processes; and establishing decisions or policies. Among the numerous relationships concepts (collaboration, cooperation, coalition . . .) and their associated time scale, interoperability is for instance necessary in organizations such as Collaborative Networked Organizations [61] which rely on collaborative business processes (which can be coordinated, orchestrated or else, synchronized) which themselves rely on interoperable activities, resources, applications (through collaboration points [62]) which themselves, for instance, exchange data. However, the better the understanding and definition of interoperability, the more complex its implementation, monitoring, control and improvement. This statement actually leads to considering interoperability requirements and their verification from a more suitable perspective [17].

Requirements engineering practices must consider two major aspects [18], namely the requirements management (access, versioning, change, traceability, etc.) and the actual engineering steps (elicitation, writing, refinement,). Requirements must be checked, throughout a system's life cycle from design phase to execution phase via the corresponding components and subsystems development phase [2], in order to prove expectations have been satisfied and avoid problems (e.g. drift from expected objectives, cancellation in worst cases). Similarly, some requirements must be verified during the actual operations phase and until the system is decommissioned (or at least partially redesigned). Requirements engineering therefore plays a major role in the success or failure of a project [19,58], yet it is often neglected by actors [20,59,60].

The requirements engineering process continues to be considered as time- and resource-consuming and without clear added value. Stakeholders however should always keep in mind that as more errors or omissions are carried to the upstream engineering phases, the remedial costs in downstream phases will increase (modification to the existing system) [21] (Fig. 1). On this figure, an important aspect is the “cost to extract defects” in relation to the different steps of development which shows that the more a defect is identified belatedly, the more the correction cost is important. The requirements engineering belongs to the field of the definition of the problem so, it is beneficial to spend time defining clearly what is expected in order to avoid (as much as possible) problems in the later phases of development. Interoperability is a non-functional requirement (NFR) to be incorporated throughout the system life cycle [22] that affects both the functioning and quality of system service yet that has remained neglected [23]. Hence, interoperability requirements engineering is a key to handling, improving and ensuring that interoperability capabilities are being properly controlled.

First of all, a simple requirements baseline is unsuitable. Requirements need to be combined into a structured reference, i.e. a repository. The overall objective is to structure requirements for them to be easily: (1) traceable throughout the system life cycle (defined, verified, allocated, satisfied), (2) modifiable/removable/addable, (3) usable for determining relevant solutions, and (4)

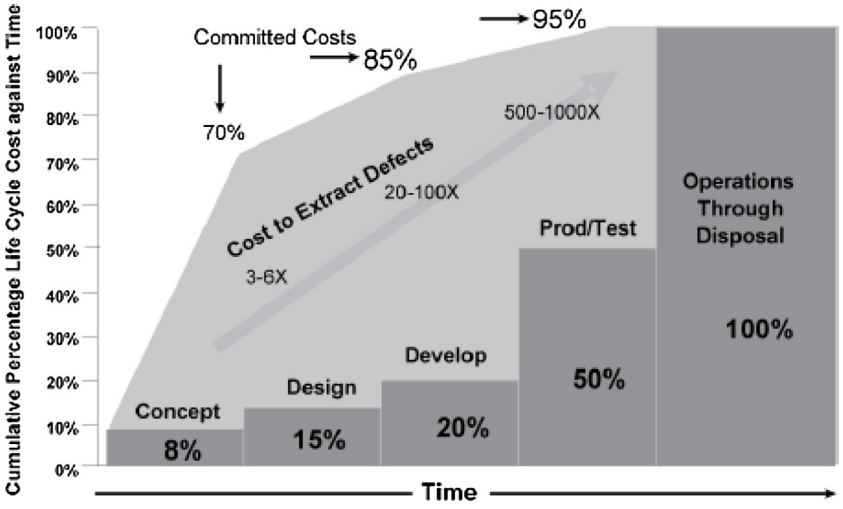

Fig. 1. Commited Life-cycle cost against time (extracted from [21] from Defense Acquisition University 1993).