The Journal of Systems and Software 107 (2015) 50–70

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Modeling and verification of Functional and Non-Functional Requirements of ambient Self-Adaptive Systems

Manzoor Ahmad [a], Nicolas Belloir [a,*], Jean-Michel Bruel [b]

[a] University of Pau and the Pays of the Adour, LIUPPA, 64000 Cedex, France
[b] University of Toulouse, CNRS/IRIT, F-31062 Toulouse Université Cedex, France

ARTICLE INFO

Article history:
Received 23 May 2014
Revised 14 May 2015
Accepted 15 May 2015
Available online 27 May 2015

Keywords:
Non Functional Requirements
Model Driven Engineering
Relax
Dynamic Adaptive Systems
Properties verification
Goal Oriented Requirements Engineering

ABSTRACT

Self-Adaptive Systems modify their behavior at run-time in response to changing environmental conditions. For these systems, Non-Functional Requirements play an important role, and one has to identify as early as possible the requirements that are adaptable. We propose an integrated approach for modeling and verifying the requirements of Self-Adaptive Systems using Model Driven Engineering techniques. For this, we use Relax, which is a Requirements Engineering language which introduces flexibility in Non-Functional Requirements. We then use the concepts of Goal-Oriented Requirements Engineering for eliciting and modeling the requirements of Self-Adaptive Systems. For properties verification, we use OMEGA2/IFx profile and toolset. We illustrate our proposed approach by applying it on an academic case study.

1. Introduction

As applications continue to grow in size, complexity, and heterogeneity, it becomes increasingly necessary for computing-based systems to dynamically self-adapt to changing environmental conditions. These systems are called Dynamically-Adaptive Systems (DASs) (Whittle et al., 2009). Example applications that require DASs capabilities include automotive systems, telecommunication systems, environmental monitoring, and power grid management systems. In this context, an adaptive system is a set of interacting or interdependent entities, real or abstract, forming an integrated whole that together are able to respond to environmental changes or changes in the interacting parts. Self Adaptive Systems (SAS) like other systems, have goals that must be satisfied and, whether these goals are explicitly identified or not, system requirements should be formulated to guarantee goal satisfaction. This fundamental principle has served systems development well for several decades but is founded on an assumption that goals are fixed. In general, goals can remain fixed if the environment in which the system operates is stable (Whittle et al., 2008). The distributed nature of SAS and changing environmental factors (including human interaction) makes it difficult to anticipate all the explicit states in which the system will be during its lifetime.

It is generally accepted that errors in requirements are very costly to fix (Lutz, 1993). The avoidance of erroneous requirements is particularly important for the emerging class of systems that need to adapt dynamically to changes in their environment. Many such DASs are being conceived for applications that require a high degree of assurance (Kasten et al., 2003), in which an erroneous requirement may result in a failure at run-time that has serious consequences. The requirement for high assurance is not unique to DASs, but the requirement for dynamic adaptation introduces complexity of a kind not seen in conventional systems where adaptation, if it is needed at all, can be done offline. The consequent dynamic adaptation complexity is manifested at all levels, from the services offered by the run-time platform, to the analytical tools needed to understand the environment in which the DASs must operate.

Requirements Engineering (RE) is concerned with what a system ought to do and within which constraints it must do it. RE for SAS, therefore, must address what adaptations are possible and how those adaptations are carried out. In particular, questions to be addressed include: what aspects of the environment are relevant for adaptation? Which requirements are allowed to vary or evolve at run-time and which must always be maintained? In short, RE for SAS must deal with uncertainty because the expectations on the environment frequently vary over time. We identify the uncertainty in requirements of these systems and show how to verify it.

We are of the view that, on one hand, requirements for SAS should consider the notion of uncertainty while defining it; on the other hand, there should be a way to verify these requirements as early

* Corresponding author. Tel.: +33559407571; fax: +33559407654.
E-mail addresses: manzoor.ahmad@univ-pau.fr (M. Ahmad), nicolas.belloir@univ-pau.fr, nbelloir@gmail.com (N. Belloir), bruel@irit.fr (J.-M. Bruel).

http://dx.doi.org/10.1016/j.jss.2015.05.028
0164-1212/

as possible, even before the development of these systems starts. In order to handle the notion of uncertainty in SAS, RE languages for these systems should include explicit constructs for identifying the point of flexibility in its requirements (Whittle et al., 2009). In this context, we provide an integrated approach to achieve this objective. We have used two approaches for defining and modeling requirements, i.e., Goal-Oriented Requirements Engineering (GORE) techniques are used to define and model the requirements of SAS (Goldsby et al., 2008; Lapouchnian et al., 2005; Yu et al., 2008; 2004) and SysML is used to specify the system and to provide a link with the requirements.

We propose a model-based requirements modeling and verification process for SAS that takes into account the uncertainty in requirements of these systems. We provide some tools to implement our approach and then apply it on an academic case study. The notion of goals is added to take into account the advantages offered by GORE. Requirements verification is done using a model checking technique.

This paper is organized as follows: In Section 2, we describe the background and the concepts which form the basis of this work, Section 3 shows the state of the art regarding RE for SAS and properties verification of these systems, Section 4 illustrates our proposed approach through an example and the tools that we have developed, Section 5 shows the case study that we used for the validation of our approach, and Section 6 concludes the paper and shows the future work.

2. Background

2.1. RELAX

Relax is an RE language for DASs in which explicit constructs are included to handle uncertainty. For example, the system might wish to temporarily Relax a non-critical requirement in order to ensure that critical requirements can still be met. The need for DASs is typically due to two key sources of uncertainty. First is the uncertainty due to changing environmental conditions, such as sensor failures, noisy networks, malicious threats, and unexpected (human) input; the term *environmental uncertainty* is used to capture this class of uncertainty. A second form of uncertainty is *behavioral uncertainty*, which refers to situations where the requirements themselves need to change. It is difficult to know all requirements changes at design time and, in particular, it may not be possible to enumerate all possible alternatives (Whittle et al., 2009).

2.1.1. RELAX vocabulary

The vocabulary of Relax is designed to enable the analysts to identify the requirements that may be Relax-ed when the environment changes. Relax addresses both types of uncertainties. Relax also outlines a process for translating traditional requirements into Relax requirements. The only focal point is for the requirement engineers to identify the point of flexibility in their requirements. Relax identifies two types of requirements: one that can be Relax-ed in favor of other ones, called variant or Relax-*ed*, and other that should never change, called *invariant*. It is important to note that the decision of whether a requirement is invariant or not is an issue for the system stakeholders, aided by the requirements engineers.

Relax takes the form of a structured natural language, including operators designed specifically to capture uncertainty (Whittle et al., 2008); their semantics is also defined. Fig. 1 shows the set of Relax operators, organized into modal, temporal, ordinal operators and uncertainty factors. The conventional modal verb *SHALL* is retained for expressing a requirement, with Relax operators providing more flexibility in how and when that functionality may be delivered. More specifically, for a requirement that contributes to the satisfaction of goals that may be temporarily left unsatisfied, the inclusion of an alternative, temporal or ordinal Relax-ation modifier, will define the requirement as Relax-able.

RELAX operator	Description
Modal Operators	
SHALL	a requirement must hold
MAY ... OR	a requirement specifies one or more alternatives
Temporal Operators	
EVENTUALLY	a requirement must hold eventually
UNTIL	a requirement must hold until a future position
BEFORE, AFTER	a requirement must hold before or after a particular event
IN	a requirement must hold during a particular time interval
AS EARLY, LATE AS POSSIBLE	a requirement specifies something that should hold as soon as possible or should be delayed as long as possible
AS CLOSE AS POSSIBLE TO [frequency]	a requirement specifies something that happens repeatedly but the frequency may be relaxed
Ordinal Operators	
AS CLOSE AS POSSIBLE TO [quantity]	a requirement specifies a countable quantity but the exact count may be relaxed
AS MANY, FEW AS POSSIBLE	a requirement specifies a countable quantity but the exact count may be relaxed
Uncertainty Factors	
ENV	defines a set of properties that define the system's environment
MON	defines a set of properties that can be monitored by the system
REL	defines the relationship between the **ENV** and **MON** properties
DEP	identifies the dependencies between the (relaxed and invariant) requirements

Fig. 1. Relax operators (Whittle et al., 2009).