Journal of Network and Computer Applications 69 (2016) 134–151

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

A model-driven approach for engineering trust and reputation into software services

Francisco Moyano *, Carmen Fernandez-Gago, Javier Lopez

Network, Information and Computer Security Lab, University of Malaga, 29071 Malaga, Spain

ARTICLE INFO

Article history:
Received 21 May 2015
Received in revised form
3 March 2016
Accepted 16 April 2016
Available online 27 April 2016

Keywords:
Trust
Reputation
Model-driven Engineering
Self-adaptation

ABSTRACT

The ever-increasing complex, dynamic and distributed nature of current systems demands model-driven techniques that allow working with abstractions and self-adaptive software in order to cope with unforeseeable changes. Models@run.time is a promising model-driven approach that supports the runtime adaptation of distributed, heterogeneous systems. Yet, frameworks that accommodate this paradigm have limited support to address security concerns, hindering their usage in real scenarios. We address this challenge by enhancing models@run.time with the notions of trust and reputation. Trust improves decision-making processes under risk and uncertainty and constitutes a distributed and flexible mechanism that does not entail heavyweight administration. This paper presents a trust and reputation framework that is integrated into a distributed component-model that implements the models@run.time paradigm, thus allowing the system to include trust in their reasoning process. The framework is illustrated in a chat application by implementing several state-of-the-art trust and reputation models. We show that the framework entails negligible computational overhead and that it requires a minimal amount of work for developers.

1. Introduction

Two important changes are coming to the Information and Communication Technology (ICT) world. On the one hand, the service-oriented vision enables on-the-fly improvements upon the functionality available to users. Applications are more dynamic and call for rapid adaptation strategies in order to meet new requirements and to respond to their changing environment. On the other hand, the boundaries between physical and virtual worlds are vanishing with the emergence of the Internet of Things (IoT), where sensors and actuators are embedded in daily life objects and are linked through networks capable of producing vast amount of data. The aforementioned reasons blur boundaries between design and runtime (Ghezzi, 2011) as they prevent designers from envisioning all possible circumstances that might appear during the execution of an application.

The widespread adoption of these systems requires addressing three main concerns: complexity, dynamicity and security. The software engineering community is developing methods for addressing the first two concerns. In particular, model-driven engineering tames the complexity of systems by working with high-level models of them (Schmidt). *Models@run.time* keeps abstract representations of running systems in order to reason about changes and drive dynamic reconfigurations (Blair et al., 2009), which tackle dynamicity.

One way to address security in these systems is by not taking for granted trust relationships among users, components, and system environments. These relationships must be explicitly declared, monitored and changed according to the system evolution. The contribution of this paper is the design and implementation of a trust and reputation development framework, together with its integration into a platform for self-adaptive, distributed component-based systems. The advantage of this integration is that reconfiguration decisions can be reasoned in terms of trust relationships and reputation information. As a result of such integration, developers can rely on a development framework that allows them to build highly dynamic, self-adaptive and trust-aware systems.

A remarkable issue about trust and reputation models today is that they often present high coupling with the application context, as they are designed as ad hoc mechanisms that are plugged into existing applications, which in turn limits their reusability (Farmer and Glass, 2010). Therefore, one of the goals of our approach is allowing developers to implement different types of trust models. We achieve this by identifying high level concepts that form trust and reputation metamodels, and which abstract away from concrete instances. Then, developers can use these concepts as

* Corresponding author.
E-mail addresses: moyano@lcc.uma.es (F. Moyano), mcgago@lcc.uma.es (C. Fernandez-Gago), jlm@lcc.uma.es (J. Lopez).

http://dx.doi.org/10.1016/j.jnca.2016.04.018
1084-8045/

building blocks for their trust and reputation models.

The work presented here is an extension over our previous work (Moyano et al., 2013). The extensions include an implementation, considerations of reputation, and validation. This is done in two steps: first, we present a trust and reputation framework that allows developers implement a broad range of models; second, as we implement this framework on top of a self-adaptive platform, developers can use the output of the models in order to reconfigure the system at runtime.

The paper is structured as follows. Section 2 presents some works that are related to ours. An introduction to a models@run.time platform called Kevoree is given in Section 3. A brief discussion on trust and reputation concepts is presented in Section 4, whereas Section 5 describes the implementation of the framework. Section 6 presents our approach for allowing trust- and reputation-based reconfigurations of the system. A case study that illustrates the use of the framework in a chat application is described in Section 7. Section 8 yields experimental results as for the overhead and the amount of work that the development of such application requires, as well as the limitations of the framework and technical challenges that we faced. Finally, Section 9 concludes the paper by presenting some research challenges and lines for future work.

2. Related work

There is an increasing interest in considering notions of trust in self-adapting systems in order to leverage reconfiguration decisions, especially in the areas of multi-agent, component-based and service-oriented systems. In some cases, trust is considered in its hard variant, where trust is seen as an aggregation of quality of service (QoS) and security properties. In other works, the soft variant of trust is used, which means that social aspects such as reputation or preferences are taken into account (Rasmusson and Jansson, 1996).

As mentioned earlier, trust is seen as a powerful tool to leverage decision-making even with partial information. This fact is especially remarked in STRATUS (Robertson et al., 2013), a set of technologies that aim at predicting and responding to complex cyber attacks. When it detects an attack, the platform switches to back-up components and finds alternative pathways of communication. The trust model that supports this platform (Robertson and Laddaga, 2012) is based on conditional trust, that is, trust in certain capabilities of a system. The authors argue that experience-based trust is not useful because configurations in cyber attacks change frequently, laying statistical analysis useless. They propose ways to make the most out of the little information available, and introduce concepts like contagion that allows formalizing trust in a host based on the distance from an infected host. Even when the underlying idea is the same, they define a model, whereas in our work the model is only defined by the developer. Also, our framework is more general purpose and includes the option of implementing reputation models, whereas STRATUS is more specialized in cyber defence and does not tackle reputation.

A classical scenario of application of trust is multi-agent systems (Ramchurn et al., 2005), where Vu et al. (2011) propose trust-based mechanisms as a way to self-organize the agents in case of deceitful information. In particular, the trust value of an agent towards another one is an aggregate of direct experiences and testimonies. The use of artificial intelligence, and concretely, machine learning together with trust in order to adapt the behavior of agents is proposed in the work by Klejnowski et al. (2010). They propose an architecture where there is an *observer* component that gathers information about the agent and presents it to the *controller* in two views: a long-term and a short-term one. The *controller* finds a suitable behavior according to this information. Given that new unexpected situations might arise, agents must be able to try out new strategies and learn which ones provided the best results.

These works differ from ours in several aspects. They are framed within multi-agent systems, which means that each agent takes their own reconfiguration decisions. In our case, the system as a whole acts as a controller of its evolution. We do not consider machine learning techniques and we tackle both, trust and reputation. The most important difference is that whereas these works propose their own models, our framework is model-agnostic, meaning that it delegates to developers the responsibility of implementing trust or reputation models.

Given the highly open and distributed nature of service-oriented environments, the traditional use of trust is for either protecting providers from potentially malicious clients or for shielding clients against potentially malicious providers (e.g. providers that publish a higher Quality of Service (QoS) than offered). As an example of the first situation, Conner et al. (2009) present a feedback-based reputation framework to help service providers to determine trust in incoming requests from clients. As an example of the second approach, Crapanzano et al. (2010) propose a hierarchical architecture for SOA where there is a so-called super node overlay that acts as a trusting authority when a service consumer looks for a service provider.

In both, component- and service-oriented systems, an important research area is determining the level of trust, or the trustworthiness, of the system as a whole, or of individual subsystems (i.e. services or components). In case that the trust value is too low, a reconfiguration takes place in order to try to improve it. In this direction, Hanen and Bourcier (2012) present a runtime architecture that allows a service-oriented system to meet a

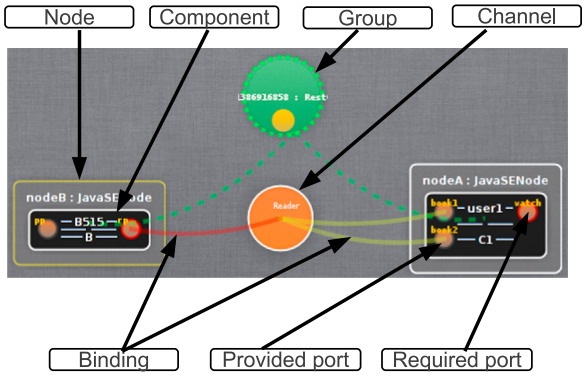

Fig. 1. Kevoree architectural elements.