The Evolution of Tropos: Contexts, Commitments and Adaptivity

Raian Ali, Amit K. Chopra, Fabiano Dalpiaz,
Paolo Giorgini, John Mylopoulos, and Vitor E. Silva Souza
Department of Information Engineering and Computer Science
University of Trento – Italy
{ali, chopra, dalpiaz, paolo.giorgini, jm, vitorsouza}@disi.unitn.it

Abstract. Software evolution is the main research focus of the Tropos group at University of Trento (UniTN): how do we build systems that are aware of their requirements, and are able to dynamically reconfigure themselves in response to changes in context (the environment within which they operate) and requirements. The purpose of this report is to offer an overview of ongoing work at UniTN. In particular, the report presents ideas and results of four lines of research: contextual requirements modeling and reasoning, commitments and goal models, developing self-reconfigurable systems, and requirements awareness.

1 Introduction

At the University of Trento (UniTN), research on Tropos is conducted within the Software Engineering and Formal Methods research program[1]. Currently, our main research challenge is facilitating software evolution so that systems may be able to evolve in response to changes in their operational environment and, more pertinently, in their requirements themselves. We are addressing this challenge by formalizing high-level concepts, and developing tools, techniques, and methodologies around these concepts. Our approach is to support evolution via design-time models that are made available at runtime. These models capture stakeholder intentions and commitments, social interactions, business processes, and organizational goals.

Evolution can be automatic (self-adaptation), or manual, or something in between. When evolution is automatic, design-time models determine what is to be monitored, what are the possible ways to adapt the behavior of the system when it deviates from its intended purposes, and how to evolve the system at runtime. When evolution is manual, these models are used as support for human activities. They offer a comprehensive view of the requirements and traceability links between elements of these models and the software code.

The rest of the report describes our current research objectives and activities, our latest results, and future work.

[1] http://www.troposproject.org

2 Objectives and Scientific Contributions

Our activities in the area of software evolution may be broadly divided into the following topics: contextual requirements modeling and reasoning, commitments and goal models, architectures for self-reconfigurable systems, and requirements awareness. The following elaborates on each.

Modeling and reasoning about contextual requirements. Advances in computing, sensors, and communication technology have given rise to new computing paradigms such as ambient, ubiquitous and pervasive computing. These paradigms weave computing systems with human living environments to transparently meet human needs. Context, a core element of these paradigms, can be defined as the reification of the environment, and includes whatever provides a surrounding within which the system operates [11]. Before influencing the behavior of software, context influences the behavior of users. It influences user goals and their choices in determining how to reach these goals. Capturing this latest influence is an essential step towards software developed to meet user requirements in different contexts.

In our research, we are interested in modeling and reasoning techniques for developing software systems expected to operate in varying contexts. We extend the Tropos goal modeling framework [1, 2] with context and allow the designer to capture the relation between the space of variants of a goal model and the context in which each variant is applicable. The framework defines a set of modeling constructs to analyze and discover relevant information the system needs at runtime to identify and characterize the context in which it is operating. We also propose various reasoning techniques to support the analysis. Particularly, we are interested in (i) checking the consistency of contextual goal models, (ii) detecting harmful interplays between tasks of a goal model originating from conflicting changes over the context, (iii) deriving goal model variants that comply with certain context and users' priorities, and (iv) deriving a subset of executable tasks that can satisfy at the minimal cost users' goals in all analyzed contexts. A prototype tool has been implemented to support reasoning about contextual goal models. The modeling and reasoning framework has been applied on two systems scenarios: a smart home for people with dementia, and a museum-guide to support museum visitors.

Social commitments and goal models. Requirements modeling for open settings such as for service-oriented and sociotechnical systems pose new challenges due to the autonomy and heterogeneity of the participants, that is, *agents*. Such settings are also highly dynamic—agents may not even know each others' identities before runtime [3].

The *i** approach was influential in emphasizing the social nature of requirements fulfillment—agents often *depend* on others to achieve their goals. An *i** dependency involves one actor wanting something, and another being able and committed to delivering that something. However, *i** does not does not achieve a clean separation between an agent's internals and its social relationships with others. For example, the formalization of dependencies refers to the ability of the dependee, that is, its internal routines. As a result, *i** has limited applications in open settings.

Our recent work on modeling agents and social relations among them replaces dependencies with interaction protocols and social commitments [3, 4]. Social commitments are brought about and manipulated solely by interaction among agents [7]. The protocols serve as specifications of convention. An agent's social commitments cleanly capture an agent's external relationships with others without referring to any agent's internals. Given an agent's goal model and capabilities—the specification of its internals—one can reason if a particular protocol *supports* the agent's goals. Specifically, support for an agent's goal may be determined objectively without referring to the agent's beliefs about others. By contrast, in *i**, an agent's belief about the workability of dependencies must be justified.

An agent's beliefs about another's ability or intentions with respect to a certain goal may be important in arriving at certain decisions. However, it is also important to systematically understand and separate the internal from the external—this enables us to build agent reasoning in a modular fashion. For example, an agent may first determine if a protocol is suitable for its goals, and then select with whom to interact within that protocol based on its internal model of others.

Social commitments are more expressive than dependencies in *i**. Social commitments are conditional, thus enabling capturing reciprocity among agents—that if one agent brings about some condition, then the other bring about another condition. Moreover, social commitments also refer to the contextual setting—these are often important in contractual settings. Formal reasoning for social commitments is also well-developed [12].

Architectures for self-reconfigurable sociotechnical systems. A sociotechnical ystem (STS) is an interplay of humans, organizations and technical systems. STSs are distributed systems where a number of autonomous and intentional actors interact in order to achieve their respective objectives. STSs are characterized by dynamism, unpredictability and weak controllability. The operational environment is subject to sudden and unexpected changes, actors may join and leave the system at will, social dependencies between actors are at risk because of actors' autonomy, and actors may fail in achieving their goals. The interests of the actors can be supported technologically by a software architecture that (i) monitors the actors' behavior, (ii) diagnoses failures against correct behavioral models, and (iii) reacts to failures via compensation actions. We have proposed an architecture based on this cycle in [5].
Our architecture becomes an integral component of an STS. The correct behavior of actors is specified by their respective goal model. The architecture observes the actions performed by participating actors, compares the monitored data against goal models, and enacts reconfigurations in response to failures. The implemented algorithms are based on the Belief-Desire-Intention paradigm [6]. Indeed, an actor participating in an STS behaves correctly if, whenever a goal is activated, it selects and executes a plan that eventually will lead to the achievement of that goal. Failures occur if the actor does not carry out the plan correctly, doesn't perform any action, or if a dependee does not bring about the dependum for the depender. Reconfiguration actions take into account the autonomy and uncontrollability of the participants: the architecture can (i) perform real actions by controlling actuators; (ii) remind or suggest the actors what to do; and (iii) assign some responsibilities to external agents.

The architecture has been applied to a smart-home case study, where the mission of the system is to support a patient in his everyday activities.

Requirements awareness. Lately there has been growing interest in systems that can adapt to changes in their environment or requirements during run-time. This kind of adaptive system generally uses some kind of feedback loop to monitor, diagnose and compensate these adverse situations. We're interested in studying the requirements that lead to this feedback loop functionality and we propose a new class of requirements, called Awareness Requirements (AwReqs). AwReqs are requirements that refer to other requirements, quality constraints or domain assumptions, and their success or failure. As a simple example, consider the requirements for a meeting scheduler. To schedule a meeting, one should know about the agenda of the participants of the meeting, arrange the meeting (set date/time, book room), and finally notify all participants about it. As a requirement for adaptation, we may want to say that the goal of notifying the participants should never fail, or that booking a room should succeed 90% of the times over any given month. To these AwReqs, the requirements engineer can attach compensation actions that would get the system back to normal operation. AwReqs can also refer to quality constraints (QCs) and domain assumptions (DAs). If there was a QC stating that meetings should have 75% attendance, an AwReq could say that this quality constraint should succeed 90% over every week. AwReqs for DAs are analogous. And since AwReqs are requirements themselves, one could create an AwReq that refers to the success of another AwReq (a meta-AwReq). Our research on this topic is detailed in [9], where we also propose: (a) a formalization using OCL; (b) elicitation techniques for AwReqs; (c) patterns for AwReqs; (d) graphical notation; and (e) a systematic process to go from AwReqs to feedback loops.

3 Future work

Future work on contextual requirements includes applying the framework developed so far to security requirements. The main idea is that contexts can influence security requirements and then security has to be analyzed and handled according to the context where the system operates. For example, in an emergency situation (such as fire), a person would allow the rescue team to access his personal data such as his location and his health status, while in a normal situation the same person would have more restricted security concerns. Our interest here is to extend the goal-oriented requirements engineering for security to cope with contextual security requirements introducing new constructs and different forms of reasoning specific for security.

Concerning commitments and self-reconfigurable systems, we are currently analyzing how a monitor-diagnose-compensate loop changes when we consider commitments together with goals. We will develop runtime agent reasoning for actors specified as goals, qualities and commitments. A correctness property, from an actor's perspective, would take the form of policies: achieve so and so goals but without violating so and so commitments. The key here is to formalize the notion of a variant in terms of both goals and commitments, and then understand adaptation as switching between variants — similar to the development in [10].

With respect to awareness requirements, the research is at its beginnings and there is much to be done. First and foremost, we intend to conduct case studies to assess our proposal. For that matter, we plan on developing a prototype framework that implements feedback loops from requirement models, most likely using previous experience in diagnosing frameworks [8]. Other challenges that lie ahead include analyzing the role of contexts with respect to AwReqs, implementing consistency checking for the model, and studying predictive and evolutionary features that could improve adaptability.

Acknowledgments This work has been partially funded by the EU Commission, through projects SecureChange, COMPAS, NESSOS and ANIKETOS.

References

1. R. Ali, F. Dalpiaz, and P. Giorgini. Location-Based Software Modeling and Analysis: Tropos-Based Approach. Proceedings of the 27th International Conference on Conceptual Modeling (ER'08), pages 169–182, 2008.
2. R. Ali, F. Dalpiaz, and P. Giorgini. A Goal Modeling Framework for Self-Contextualizable Software. Proceedings 14th International Conference on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD'09), LNBIP 29-0326, pages 326–338. Springer, 2009.
3. A. K. Chopra, F. Dalpiaz, P. Giorgini, and J. Mylopoulos. Modeling and Reasoning about Service-Oriented Applications via Goals and Commitments. Proceedings 22nd International Conference on Advanced Information Systems Engineering (CAiSE'10), 2010. to appear.
4. A. K. Chopra, F. Dalpiaz, P. Giorgini, and J. Mylopoulos. Reasoning about Agents and Protocols via Goals and Commitments. Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10), 2010. to appear.
5. F. Dalpiaz, P. Giorgini, and J. Mylopoulos. An Architecture for Requirements-Driven Self-Reconfiguration. Proceedings 21st International Conference on Advanced Information Systems Engineering (CAiSE'09), LNCS 5565, pages 246–260. Springer, 2009.
6. F. Dalpiaz, P. Giorgini, and J. Mylopoulos. Software Self-Reconfiguration: a BDI-Based Approach. Proceedings 8th International Conference on Autonomous Agents and Multiagent Systems, pages 1159–1160. IFAAMAS, 2009.
7. M. P. Singh. Agent communication languages: Rethinking the principles. IEEE Computer, 31(12):40–47, Dec. 1998.
8. V. E. S. Souza and J. Mylopoulos. Monitoring and Diagnosing Malicious Attacks with Autonomic Software. Proceedings 28th International Conference on Conceptual Modeling (ER'09), pages 84–98, Gramado, Brazil, 2009. Springer.
9. A. Lapouchnian, V. E. S. Souza, and J. Mylopoulos. Awareness Requirements for Adaptive Systems. Submitted for review, 2010.
10. Ji Zhang and B. H. C. Cheng. Model-Based Development of Dynamically Adaptive Software. Proceedings 28th International Conference on Software Engineering (ICSE), pages 371–380, 2006.
11. A. Finkelstein, A. Savigni. A Framework for Requirements Engineering for Context-Aware Services. Proceedings of STRAW'01, 2001.
12. A. K. Chopra and M. P. Singh. Multiagent Commitment Alignment. Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems, 2009, pages 937—944