(Relazione da consegnare al Direttore del Dipartimento almeno un mese prima della scadenza annuale del contratto)

Relazione Scientifica Annuale sull’attività svolta nell’ambito dell’assegno di ricerca

Nominativo dell’assegnista di ricerca: Camilli Matteo

Titolo dell’assegno di ricerca: Analisi di sistemi software complessi in ambito distribuito

Specificare se si tratta di assegno di ricerca di tipo A o di tipo B: Tipo A

Docente referente: Bellettini Carlo

Durata del contratto da 01/09/2017 a 31/08/2019

Periodo di riferimento della relazione da 01/09/2017 a 31/08/2018

Objectives

Formal specification and verification of Self-Adaptive systems:
We want to support formal specification and quantitative verification of (real-time) self-adaptive systems in a distributed fashion (i.e., having a decentralized adaptation control). Self-adaptation [1, 2] has been widely recognized as an important research topic in many different areas such as software engineering and formal methods [3]. Moreover, facing the complexity deriving from time-dependent and distributed self-adaptive systems through design-time specification and verification has been recognized among the major challenges in the field of self-adaptation [1]. Thus, formal methods providing the ability to reason on such a complexity are highly demanded. In particular, formal design models and approaches to validation and verification at design-time are of extreme importance in engineering distributed (and real-time) self-adaptive systems.

Formal specification and verification of Microservices-based applications:
Our main objective is to support the design and the analysis of microservices-based systems. In fact, formal methods based on well-known techniques seem to be a promising starting point for tackling the issue of writing correct microservice systems [4]. This starting point gives us solid footing for addressing problems like the description of complex process flows or coordination patterns among services. It is still unclear, how exactly formal methods can be leveraged to naturally capture the practical scenarios in real-world microservices-based applications.
Uncertainty Quantification:
Modern software systems are required to operate in a highly uncertain and changing environment. They have to control the satisfaction of their requirements at run-time, and possibly adapt and cope with situations that have not been completely addressed at design-time. Software engineering methods and techniques are, more than ever, forced to deal with change and uncertainty (lack of knowledge) explicitly [5]. Our objective is to tackle the challenge posed by uncertainty in delivering more reliable systems by supporting the software development lifecycle with formal approaches able to mitigate the uncertainty before the deployment of the release builds.

Results

We currently provide support for design-time modeling and analysis of real-time self-adaptive systems. We developed both the fundamentals and a working implementation of our modeling approaches and design-time verification techniques. Specifically, we contribute with the following key aspects: (i) definition of a formalism for modeling self-adaptive behavior with real-time constraints; (ii) modular and incremental modeling and verification process due to separation in zones (e.g., adaptation/functional logics); (iii) verification technique for checking adaptation requirements satisfaction, including timed adaptation; (iv) approach able to map zones defined upon the model into regions of the system state space, thus allowing the identification of different behaviors (normal behavior, undesired behavior, adaptive behavior, etc.) upon the state space structure, and the verification of properties of interest with respect to Self-adaptation.

To support formal reasoning on distributed self-adaptive systems with decentralized adaptation control we developed a formal framework based on High-Level Petri nets (HLPN) to specify in a natural way structural changes that are likely to occur in adaptable and evolvable distributed applications. In this field, we contribute with the following key aspects: (i) We defined a modeling formalism for self-adaptive systems with decentralized adaptation control (with multiple feedback loop components); (ii) we discuss the major complexity issues of our approach deriving from complex models; (iii) we show how all the native HLPN analysis techniques can be used on our modeling framework; (iv) we studied the applicability and the benefits of our approach in the context of self-healing and self-optimizing systems.

In the field of formal modeling and analysis of microservices-based applications, we introduced a number of advances to demonstrate how formal methods can be leveraged to naturally capture and reason on the complexity of real-world microservices-based applications. Namely, we contributed in the field with the following key points: (i) we propose to use Time Basic Petri nets (i.e., a particular extension of Petri nets) supporting the definition of temporal constraints, to analyze the properties of microservice-oriented applications orchestrated by the Netflix Conductor engine; (ii) we defined a formal semantics for the workflow language in which microservices are black-box described by Petri net modules; (iii) we provided the ability to apply computer aided verification by using off-the-shelf model checking software tools; (iv) we introduced an approach to formal design-to run-time verification of microservice-based process flows built on top of Conductor.
To tackle the challenge posed by uncertainty in delivering more reliable systems we provided support for the software development lifecycle with formal approaches able to mitigate the uncertainty before the deployment of the release builds. In particular, (i) we introduced and formalized a novel online MBT technique that complements classic test case generation with an uncertainty-aware sampling strategy; (ii) we described how online MBT can be used to feed a Bayesian inference calibrator that continuously learns from test data to perform IUQ; (iii) we presented the evaluation of both the accuracy and the efficiency of our IUQ process depending on the selected sampling strategy, thus showing the effectiveness of our uncertainty-aware technique.

Activities

Formal specification and verification of Self-Adaptive systems:
We introduced a formal framework for specifying and verifying the behavior of real-time self-adaptive systems which leverages a particular temporal extension of Petri nets. Our proposal combines a zone-based specification approach with timed adaptation models. These last have been defined by extending the adaptation models presented in [6] with temporal constraints and different temporal semantics for modeling both mandatory and optional timed events. Our zone-based modeling approach allows functional aspects to be kept separated from adaptation aspects. Zones, describing different steady-state behaviors of the system, can then be used either in isolation to verify non-adaptive behavior by means of intra-zone properties, or all together, to verify global system requirements through inter-zone properties. The verification of timing requirements is supported through timed properties, able to check that both functional aspects and adaptation comply with specific temporal deadlines. In addition, the framework supports interesting (timed) robustness properties, to ensure self-healing capability that represents a very important issue when dealing with real-time or even time-critical systems. The technique is supported by a open-source Java software called Zafety (publicly available at http://camilli.di.unimi.it/zafety).

We also supported the specification and the verification of self-adaptive system with decentralized adaptation control by introducing a formal framework for modeling and analyzing self-adaptive systems with decentralized adaptation control [6,7]. The framework makes use of High-Level Petri nets which represents a sound and expressive formal model for distributed discrete-event systems. Our framework can be used to specify in a natural way structural changes that are likely to occur in adaptable and evolvable distributed applications. We also support validation and verification activities to check correctness of multiple concurrent feedback loops, described in terms of MAPE (Monitor, Analyze, Plan, and Execute) components over a shared Knowledge. The software tool supporting this proposal (publicly available at https://github.com/SELab-unimi/pnemu) is called PNEmu and it has been implemented using the Python language.

Formal specification and verification of Microservices-based applications:
We proposed a formal semantics for microservices-based process flows specified using Conductor (i.e., an open source orchestration framework in use at Netflix, Inc. [8]). Our approach mechanically produces a formal representation in terms of Petri Nets. The translation process is fully automated by means of a Java software tool called Conductor2Pn (publicly available at https://bitbucket.org/seresearch_unimi/conductor2pn). The generated formal model can be used in turn to perform model checking, simulation, model-based testing, and runtime verification by means of
powerful off-the-shelf tools. The tool has been validated on a number of benchmarking microservices-based systems, written using the Conductor language. We also provided support for runtime analysis of microservices-based applications through our envisioned approach to continuous design-to-run-time verification. Namely, we exploited the technique introduced in [9] to perform runtime verification of temporal properties on cloud applications built on top of Conductor.

Uncertainty Quantification:
We developed a IUQ methodology to quantify and mitigate the system uncertainty during the software development life cycle. The methodology makes use of a novel online Model-based Testing (MBT) [10] technique that combines test case generation guided by uncertainty-based strategies and Bayesian inference [11]. We provided an explicit model of the inherent uncertainty and we exploited this definition to provide a means to stress and observe the software product in its own uncertain parts. This way we quantify the design-time uncertainty before the deployment of a release build. To show the feasibility and the advantages of the approach, we built a prototypal toolchain and validated it by performing a large experimental activity with a number of simulated examples. The toolchain has been released as open source publicly available at https://github.com/SELab-unimi/mdp-simulator-monitored.

Future work and research directions
Towards verification driven development — Formal verification of large pieces of software is now increasingly feasible and this is paving the way to transferring these techniques from research to practice. However, a mismatch between monolithic formal verification techniques and modern agile/incremental development processes, still exists. To address this issue, verification should become agile [12], and its introduction into agile processes should be made possible. Thus there is a call for new approaches supporting verification-driven development in the same way as test-driven development is possible today. This represents a long term research we are going to conduct.

References

Publications


La presente relazione, non contiene dati sensibili e dati giudiziari di cui all’art. 4, comma 1, lettere d) ed e) del D.Lgs. 30.6.2003 n. 196.

*Si autorizza la pubblicazione della relazione annuale sul sito web del Dipartimento.*

Firmato (In Stampatello)   NOME Matteo   COGNOME Camilli

Data 09/07/2018

Il Responsabile Scientifico   L’ Assegnista di Ricerca

(Firma)   (Firma)