Ph.D. Proposal

Static analysis and runtime monitoring to ensure the consistency of dynamic applications

Supervisors

Walter Rudametkin (Associate professor) <Walter.Rudametkin@polytech-lille.fr>
Lionel Seinturier (Professor) <Lionel.Seinturier@univ-lille1.fr>

Research team

Spirals Research Group
Inria Lille - Nord Europe
Parc Scientifique de la Haute Borne
40, avenue Halley - Bat. B, Park Plaza
59650 Villeneuve d'Ascq – FRANCE

Resumé

Les approches de génie logiciel modernes permettent la construction d'applications complexes qui peuvent changer leur structure en cours d'exécution sans s'arrêter. Ces applications sont connues sous le nom d'Applications Dynamiques. Cependant, les changements durant l'exécution dans les applications dynamiques peuvent introduire des incohérences qui mènent à des défaillances. Cette thèse se focalisera sur la construction d'outils pour l'analyse et la vérification d'applications dynamiques pour assurer leur cohérence et bon fonctionnement malgré des changements dynamiques.

Summary

Modern software engineering techniques allow building applications that change at runtime, without stopping. We call these applications *Dynamic Applications*. However, changing applications at runtime to cause inconsistencies that lead to failure. This Ph.D thesis will focus on state-of-the-art tools to analyse and verify that applications remain consistent and function properly despite dynamic change.

Context

Modern software applications are large and complex. To tame their complexity, many software engineering approaches have been invented and refined over the years. Of them, we can mention Modular Programming, Object-Oriented Programming, Component-Based Software Engineering, Service-Oriented Computing, and more recently, Service-Oriented Components and Micro-Services. Yet, at the same time as software has become more complex, there is an ever increasing need to make it more flexible and nimble in order to quickly adapt to changes. For example, software requires being patched, updated, new features added, old features removed, and in general, resilient to recover rapidly from changes. To minimize the downtime suffered by applications, these changes should be applied at runtime.

Dynamic applications—where software components can be added, removed and substituted during execution—allow software to adapt and adjust to changing environments, and to accommodate evolving features or to recover from faults. Unfortunately, dynamic applications raise design and development issues that have yet to be fully addressed.

Ph.D. Project

Positioned well within the context of developing and executing dynamic applications, this Ph.D. project will focus on the numerous issues caused by dynamic changes at runtime that introduce inconsistencies into the application.

The objective of this Ph.D. is to ensure dynamic applications remain consistent and function properly despite dynamic changes such as component updates, patches, removals or failures.

In order to do so, we propose the following strategy:

- **Evaluate the state-of-the-art software engineering practices for building dynamic applications.**
  There exists much work in the area of Component-Based Software Engineering [1], Dynamic Software Reconfiguration [2] and Dynamic Software Updating [3] that is related. This work should particularly focus on frameworks built using modern object-oriented languages such as those that run on the Java Virtual Machine. Of particular interest are the OSGi framework [4] and the iPOJO component model [5,6].
- **Build a taxonomy of inconsistencies caused by dynamic changes.**
  Issues such as ensuring that memory leaks do not occur after a change or that the old versions of components are no longer used must be identified. Some inconsistencies come from components that are not decoupled enough [7] or component services that are incorrectly defined [7], or improper shutdown of components and their threads [8][9].

- **Use static analysis techniques to detect inconsistencies.**
  Static analysis can be used to identify possible inconsistencies at the source-code and bytecode levels. These analyses should ensure that the issues in the taxonomy of inconsistencies are checked. They should assist developers in building dynamic applications by identifying these issues while the system is under construction.
- **Introduce active monitoring into dynamic applications to detect inconsistencies otherwise only visible at runtime.**
  Because static analysis is insufficient to ensure all cases are properly identified, runtime monitoring and dynamic runtime analysis will be required to identify causes of inconsistencies or malfunctioning components [10]. These analyses may include detecting component coupling at runtime [7], properly stopping components, impact analysis, locating failed components, locating misused pointers [11], finding memory leaks, etc.
- **Propose programming practices and/or changes to component frameworks to correct the identified issues.**
  These include but are not limited to how to properly design and program dynamic components, how to package components and their implementation classes, how to define component interfaces and services.

The outcome of this work is a programming model with proper tooling that assists developers in building dynamic applications all the while ensuring consistency and proper functioning throughout the application's lifecycle.

Skills Summary

The Ph.D. candidate will develop her/his skills in Java SE, OSGi, iPOJO, Spoon, JVM TI, Python, Linux, Docker, Git, among many others.

As is a common practice in the Spirals research team, all source code will be open sourced using either the GPL or the Apache License. It is expected that the student participates in related open source communities. This should also assist in the technological transfer from academic prototype to industrial-ready tools.

Experimentations to demonstrate the effectiveness of developed tools should be performed on large and complex modular software systems such as the Eclipse IDE, the Jitsi video conferencing tool and the Glassfish Java EE server.

References

[1] Szyperski, C., Gruntz, D. and Murer, S., 2002. Component software: beyond object-oriented programming 2nd ed., Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

[2] Cheng, B. H., De Lemos, R., Giese, H., Inverardi, P., Magee, J., Andersson, J., ... & Whittle, J. (2009). Software engineering for self-adaptive systems: A research roadmap. In Software engineering for self-adaptive systems (pp. 1-26). Springer Berlin Heidelberg.
[3] Hicks, M. and Nettles, S., 2005. Dynamic software updating. ACM Transactions on Programming Languages and Systems, 27(6), pp.1049–1096.
[4] http://www.osgi.org
[5] http://felix.apache.org/documentation/subprojects/apache-felix-ipojo.html
[6] Escoffier, Clément, Richard S. Hall, and Philippe Lalanda. "iPOJO: An extensible service-oriented component framework." Services Computing, 2007. SCC 2007. IEEE International Conference on. IEEE, 2007.
[7] Walter Rudametkin, Ph.D. thesis, Robusta : an approach to building dynamic applications, supervised by Jacky Estublier. Defended February 21, 2013 in Grenoble, France.
[8] Kramer, J. and Magee, J., 1990. The evolving philosophers problem: dynamic change management. IEEE Transactions on Software Engineering, 16(11), pp.1293–1306.
[9] Vandewoude, Yves, Ebraert, P., Berbers, Yolande and D'Hondt, T., 2007. Tranquility: A Low Disruptive Alternative to Quiescence for Ensuring Safe Dynamic Updates. IEEE Transactions on Software Engineering, 33(12), pp.856–868.
[10] Gonzalez-Herrera, Inti, et al. "Scapegoat: an Adaptive monitoring framework for Component-based systems." Working IEEE/IFIP Conference on Software Architecture. 2014.
[11] Gama, Kiev, and Didier Donsez. "Service coroner: A diagnostic tool for locating OSGI stale references." Software Engineering and Advanced Applications, 2008. SEAA'08. 34th Euromicro Conference. IEEE, 2008.