Designing and Evolving Distributed Architecture using Kevoree

(*)François Fouquet, (**)Erwan Daubert, (*)Grégory Nain, (***)Brice Morin, (**)Johann Bourcier, and (**)Olivier Barais

(*)Interdisciplinary Center for Security Reliability and Trust (SnT) University of Luxembourg
(**)University of Rennes 1, IRISA, INRIA Centre Rennes, France
(***)SINTEF ICT, Oslo, Norway

Abstract. Modern software applications are distributed and often operate in dynamic contexts, where requirements, assumptions about the environment, and usage profiles continuously change. These changes are difficult to predict and to anticipate at design time. The running software system should thus be able to react on its own, by dynamically adapting its behavior, in order to sustain a required quality of service. A key challenge is to provide the system with the necessary flexibility to perform self-adaptation, without compromising dependability. Models@Runtime is an emerging paradigm aiming at transferring traditional modeling activities (focusing on quality, verification, and so on) performed by humans, to the running system. In this trend, Kevoree provides a models@ runtime platform to design heterogeneous, distributed and adaptive applications based on the component based software engineering paradigm. At the end of this tutorial, applicants will be able to develop and assemble new components and communication channel to design complex self- adaptable distributed architectures by reusing existing piece of code.

1 Presenter

François Fouquet holds a Master degree in Software Engineering at the University of Rennes 1 in 2009. He obtained his PhD thesis from the Triskell research group in Rennes in 2013. He currently holds a postdoc position at Interdisciplinary Center for Security Reliability and Trust (SnT) in Luxembourg. His topics of interest include software engineering, modeling at design and runtime, and distributed systems. During his PhD work, he notably developed Kevoree. François Fouquet has co-authored 4 international peer-reviewed conference papers and 1 peer-reviewed journal article and several workshop papers.

Olivier Barais (`http://goo.gl/fQOF7`) is an Associate Professor at the University of Rennes 1, member of the Triskell INRIA research team. He received an engineering degree from the Ecole des Mines de Douai, France in 2002 and a PhD in computer science from the University of Lille 1, France in 2005. After having been a PhD student in the Jacquard INRIA research team, he is currently associate professor at University of Rennes 1 and a member of the Triskell INRIA group. His research interests include Component Based Software Design, Model-Driven Engineering and Aspect Oriented Modeling. Olivier Barais has co-authored 8 journals, 36 international conference papers, 2 book chapters and 26 workshop papers in conferences and journals such as SoSyM, IEEE Computer, ICSE, MoDELS, SPLC and CBSE.

Johann Bourcier is an Associate Professor at the University of Rennes 1, where he is a member of the Triskell INRIA research team. He received his Ph.D. degree in 2008 from the University of Grenoble 1. He is a former member of the LIG Adele research group (Grenoble) and later the Distributed Software Engineering Section in the Department of Computing of Imperial College London. His research interests include the use of software engineering to simplify the development of highly dynamic pervasive applications. Johann Bourcier has co-authored 9 international peer-reviewed conference papers and 1 peer-reviewed journal article and 2 book chapters and several workshop papers.

2 Duration and Room Requirements

The workshop duration is half a day (4 hours). The type of tutorial will be a hands-on tutorial. This tutorial requires a room with wireless access for participants. We expect 15-20 participants.

3 Scope

The intended audience is both software engineers/researchers and PhD students. The attendants must have a laptop with Virtual Box installed or a laptop with a JDK 1.7 and a recent Java IDE. The attendants must be familiar with Java or any Object Oriented Technology. Academics and practitioners alike will benefit from the tutorial. We expect that the presentation is of particular interest for builders of distributed tools and algorithms, both academics and practitioners, who would like to have abstractions to deal with heterogeneous and distributed adaptive applications.

4 Goal and Objectives

The goal of this tutorial is to provide the fundamentals of the Kevoree framework language for designing heterogeneous and distributed adaptive applications.

The goal of this tutorial is to present Kevoree through a practical session to a wide number of Software Engineering practitioners. This event would be a great venue for people to learn-by-example about dynamic adaptations, distribution and continuous design in a reliable environment. This tutorial is also intended to be a privileged moment to collect comments and feedback on our approach and realizations.

This overall goal can be broken down into several concrete sub-objectives:
(i) Component development Use the Kevoree Annotation API to create component types, then use a Kevoree Maven plugin to extract the components' models at compile time. **(ii) Assembly (graphical & script)** Use editor to assemble an application through a graphical DSL or through the use of a Kevoree Script. **(iii) Deployment on (single & multiple) node** Experiment the simple deployment on single vs multiple execution nodes. **(iv) Continuous Design** Experiment the continuous design facilities of Kevoree that shorten the development cycles. **(v) Development of Self-Adaptive system** Design and create a self-adaptive system, through the development of a simple ECA reactive system. **(vi) Cloud simulation** Simulate the deployment of an application in a Cloud, and perform cross-layer adaptations using the Kevoree abstraction.

5 Overview of the tutorial

The tutorial divides into four parts here described.

5.1 Part 0 - Introduction (20 min talk - 20 min Prerequisites)

Presentation This first part introduces models@runtime and the Kevoree Framework. After a bit of history, some facts, and examples of realizations, the participants prepare their machines with the necessary tools for the following parts.

Prerequisites The practical session of this part ensures that a JDK, Maven, Java/Scala development tool are available.

5.2 Part 1 - The basics (20 min talk - 30 min hands-on)

Presentation In this part, the different basic features manipulated in Kevoree are presented. Participants get familiar with the Domain Specific language used in Kevoree to deal with distributed systems.

Practical The participants discover the Kevoree Editor and During and Runtime in this practice. They have to create a first application, deploy it and run it.

Material `http://www.kevoree.org/tutorials#tutorial1`

5.3 Part 2 - Do-It-Yourself (20 min talk - 30 min hands-on)

Presentation The presentation of this session describes how to create a Kevoree Component Type. It introduces the Annotation API, used to describe the component type in the code, and the Compilation Chain to extract the component model at compile time, and create the component library.

Practical In this session, attendees complete and change a sample Producer/Consumer component code. They then go through the entire chain from development to deployment and experience the continuous design abilities of Kevoree.

Material `http://www.kevoree.org/tutorials#tutorial2`

5.4 Part 3 - Self-Adaptation@Runtime (15 min talk - 45min hands-on)

Presentation Finally, we present the principle of using Models@Runtime to develop self-adaptive systems, that can autonomously adapt at runtime.

Practical Participants have to completely develo by themselves a simple Event-Condition-Action reasoner that modifies the model of the running system and ask for an adaptation at runtime.

Material `http://www.kevoree.org/tutorials#tutorial3`

5.5 Part 4 - Kloud adaptation using Kevoree (10 min talk - 30 min hands-on))

Presentation This part presents how the abstractions provided by Kevoree can be used to drive a cross-layer cloud adaptation).

Practical Participants have to design a simple architecture to drive several system virtual machines (built on top of LXC (https://github.com/lxc/lxc)).

Material `http://www.kevoree.org/tutorials#tutorial4`

6 Resources

Resources about Kevoree are available at `http://www.kevoree.org`. The source code is available on Github. Some talks given about Kevoree are proposed online along with video tutorials.

References

1. Gordon S. Blair, Nelly Bencomo, and Robert B. France. Models@runtime. *IEEE Computer*, 42(10):22–27, 2009.
2. François Fouquet, Olivier Barais, Noël Plouzeau, Jean-Marc Jézéquel, Brice Morin, and Franck Fleurey. A Dynamic Component Model for Cyber Physical Systems. In *15th International ACM SIGSOFT Symposium on Component Based Software Engineering*, Bertinoro, Italie, July 2012.
3. Francois Fouquet, Erwan Daubert, Noel Plouzeau, Olivier Barais, Johann Bourcier, and Arnaud Blouin. Kevoree : une approche model@runtime pour les systemes ubiquitaires. In *UbiMob2012*, Anglet, France, June 2012.
4. François Fouquet, Erwan Daubert, Noël Plouzeau, Olivier Barais, Johann Bourcier, and Jean-Marc Jézéquel. Dissemination of reconfiguration policies on mesh networks. In *DAIS 2012*, Stockholm, Suède, June 2012.
5. François Fouquet, Grégory Nain, Brice Morin, Erwan Daubert, Olivier Barais, Noël Plouzeau, and Jean-Marc Jézéquel. An Eclipse Modelling Framework Alternative to Meet the Models@Runtime Requirements. In *Models 2012*, Innsbruck, Autriche, October 2012.
6. B. Morin, O. Barais, J-M. Jézéquel, F. Fleurey, and A. Solberg. Models@ run.time to support dynamic adaptation. *Computer*, 42(10):44–51, 2009.
7. Brice Morin, Olivier Barais, Gregory Nain, and Jean-Marc Jezequel. Taming Dynamically Adaptive Systems with Models and Aspects. In *ICSE'09: 31st International Conference on Software Engineering*, Vancouver, Canada, May 2009.