mRUBiS: An Exemplar for Model-Based Architectural Self-Healing and Self-Optimization (Artifact)

Thomas Vogel
Department of Computer Science, Humboldt-Universität zu Berlin, Germany
thomas.vogel@cs.hu-berlin.de
0000-0002-7127-352X

Abstract

Self-adaptive software systems are often structured into an adaptation engine that manages an adaptable software by operating on a runtime model that represents the architecture of the software (model-based architectural self-adaptation). Despite the popularity of such approaches, existing exemplars provide application programming interfaces but no runtime model to develop adaptation engines. Consequently, there does not exist any exemplar that supports developing, evaluating, and comparing model-based self-adaptation off the shelf. Therefore, we present mRUBiS, an extensible exemplar for model-based architectural self-healing and self-optimization. mRUBiS simulates the adaptable software and therefore provides and maintains an architectural runtime model of the software, which can be directly used by adaptation engines to realize and perform self-adaptation. Particularly, mRUBiS supports injecting issues into the model, which should be handled by self-adaptation, and validating the model to assess the self-adaptation. For this purpose, the exemplar provides two case studies of self-healing and self-optimization. Finally, mRUBiS allows developers to explore variants of adaptation engines (e.g., event-driven self-adaptation) and to evaluate the effectiveness, efficiency, and scalability of the engines.

2012 ACM Subject Classification Software and its engineering → Development frameworks and environments, Software and its engineering → Software development techniques

Keywords and phrases Self-adaptation, architecture, runtime models, simulator

Digital Object Identifier 10.4230/DARTS.4.1.1

Related Article Thomas Vogel, "mRUBiS: An Exemplar for Model-Based Architectural Self-Healing and Self-Optimization", in Proceedings of the 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2018), ACM, 2018.
https://doi.org/10.1145/3194133.3194161

Related Conference 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2018), May 28-29, 2018, Gothenburg, Sweden

1 Scope

The mRUBiS artifact targets self-adaptive software that is split into an *adaptation engine* implementing a feedback loop for self-adaptation and an *adaptable software* realizing the domain logic while the engine manages the software. mRUBiS helps researchers in developing, evaluating, and comparing adaptation engines that perform *model-based architectural self-adaptation*. Thus, the engine uses an architectural runtime model of the adaptable software as a basis for self-adaptation.

Despite the popularity of model-based architectural self-adaptation, none of the existing SEAMS artifacts particularly addresses this kind of self-adaptation by providing an architectural runtime model of the adaptable software. In contrast, the existing artifacts provide application programming interfaces (APIs) to manage the adaptable software. Consequently, using these exemplars for model-based architectural self-adaptation requires from developers to implement a runtime model and a causal connection between the model and the APIs. This is challenging

since developers have to assure the synchronization and fidelity of the runtime model with the running software. Thus, existing SEAMS exemplars do not support developing, evaluating, and comparing model-based architectural self-adaptation off the shelf.

Therefore, we present mRUBiS, an extensible exemplar for model-based architectural self-adaptation. It simulates the mRUBiS system as the adaptable software and provides as well as maintains an architectural runtime model of the system. This model serves as the interface for adaptation engines to realize and perform architectural adaptation of mRUBiS. Thus, model-based architectural self-adaptation is supported off the shelf. Developers are relieved from implementing a runtime model and a causal connection to the adaptable software as well as setting up a corresponding runtime infrastructure. Instead, they can focus on designing, implementing, and evaluating the adaptation logic on top of the provided runtime model.

The simulation performed by the exemplar consists of a predefined number of iterations over the following three steps:

1. According to a *scenario*, the simulator injects *issues* into the runtime model and thus to the mRUBiS architecture, which should be handled by self-adaptation.
2. The adaptation engine developed by the user of the exemplar is triggered to analyze and adapt the mRUBiS architecture described in the model. The adaptation aims at resolving the injected issues and thus at satisfying the goals of mRUBiS.
3. According to a set of *validators*, the simulator validates the adaptation and runtime model to check whether issues are remaining in the mRUBiS architecture. It further evaluates the self-adaptation by computing the utility of the current architecture based on a *utility function* and by measuring the execution time of the self-adaptation (i.e., of the 2nd step). This data is summarized at the simulation end.

For the mRUBiS architecture, we provide scenarios, issues, validators, and utility functions for a self-healing and a self-optimization case study. For both case studies, the artifacts further provides sample solutions. Furthermore, each of these elements can be replaced or extended by developers to address other case studies. Even mRUBiS as the adaptable software described by the runtime model can be replaced or extended. The language to express the runtime model is generic and supports modeling arbitrary component-based architectures and properties.

Moreover, the mRUBiS artifact does not restrict the adaptation engines developed on top of it. In contrast, it even encourages developers to explore variants of engines, for instance, by optionally using the provided change events to drive the model-based self-adaptation. Developers can use their favorite technologies to implement the engines such as code (Java) or model-based rules (e.g., expressed with OCL and Story Diagrams) that operate on the runtime model. Finally, the exemplar allows developers to evaluate the effectiveness (in terms of the utility of the adaptable software), efficiency (in terms of execution time), and scalability of self-adaptation by scaling the size of the architectural model and the number of injected issues per simulation round.

The artifact has been developed with the Eclipse Modeling Framework (EMF) and it is available as a plug-in for Eclipse. Thus, any adaptation engine developed on top of this artifact can use any EMF-compatible technique to process, analyze, and change the architectural runtime model to perform self-adaptation.

2 Content

The artifact package includes a snapshot of the Git repository `https://github.com/thomas-vogel/mRUBiS` (March 16th, 2018; last commit: 962edbcbe476cf8d9b6a6cfb50c1b1bb304d35ee). The package contains the following sub-packages (each sub-package is an Eclipse plug-in project that can be imported as a project into Eclipse to continue development and to extend the artifact):

- `de.mdelab.comparch`
  This package defines the metamodel for the CompArch (short for component architecture) modeling language that is used to express the architectural runtime model.
- `de.mdelab.comparch.edit`
  This package defines edit operations for CompArch models. It is completely and automatically generated from the CompArch metamodel by EMF and used by the generated tree-based editor for CompArch models (see package `de.mdelab.comparch.editor`).
- `de.mdelab.comparch.editor`
  This package implements a tree-based editor for CompArch models that uses the generated edit operations (see `de.mdelab.comparch.edit`). It is completely and automatically generated from the CompArch metamodel by EMF.
- `de.mdelab.comparch.editor.graphical`
  This package implements the graphical CompArch modeling editor and thus, defines the notation for CompArch models. The editor is based on Eclipse Sirius and only depends on the CompArch metamodel (see package `de.mdelab.comparch`).
- `de.mdelab.simulator`
  This package implements the core of the simulator. This core is generic and does not any contain aspects that are specific to a concrete adaptable software such as mRUBiS. Consequently, generic validators are implemented in this package. This package only depends on the CompArch metamodel (see package `de.mdelab.comparch`).
- `de.mdelab.simulator.mrubis`
  This package extends the simulator core and thus, the `de.mdelab.simulator` package by implementing aspects that are specific to mRUBiS such as certain validators, the injectors for the issues, and the utility functions.
- `de.mdelab.simulator.mrubis.examples`
  This package just bundles the two packages
  - `de.mdelab.simulator.mrubis.examples.selfhealing`
  - `de.mdelab.simulator.mrubis.examples.selfoptimization`

  to provide them as Eclipse example projects.
- `de.mdelab.simulator.mrubis.examples.selfhealing`
  This package implements example solutions for adaptation engines for the self-healing case study of mRUBiS. It is provided as Eclipse Example Project, in which CompArch models can be generated (see package `de.mdelab.simulator.mrubis.ui`) and the simulator (see package `de.mdelab.simulator.mrubis`) is used to develop, test, and evaluate the solutions.
- `de.mdelab.simulator.mrubis.examples.selfoptimization`
  This package implements the example solution for an adaptation engine for the self-optimization case study of mRUBiS. It is provided as Eclipse Example Project, in which CompArch models can be generated (see package `de.mdelab.simulator.mrubis.ui`) and the simulator (see package `de.mdelab.simulator.mrubis`) is used to develop, test, and evaluate the solution.
- `de.mdelab.simulator.mrubis.feature`
  This package is a so called Eclipse feature project that just bundles all other sub-packages to the feature called *mRUBiS Exemplar*. Such a feature eases generating a release of the exemplar for the Eclipse update site from which (a release of) the mRUBiS exemplar can be installed to Eclipse.
- `de.mdelab.simulator.mrubis.ui`
  This package just implements the UI for generating CompArch models for mRUBiS. It uses mRUBiS-specific knowledge defined in the `de.mdelab.simulator.mrubis` package to generate a CompArch model of a user-defined size.

The overall package further includes two folders and two files:

- `docs`
  This folder contains the Eclipse update site project and the update site itself for the mRUBiS exemplar. The mRUBiS exemplar can be installed from the update site to Eclipse. The folder further contains the generated Java documentation (Javadoc) of the exemplar.
- `wiki`
  This folder contains a snapshot of the wiki with comprehensive documentation of the mRUBiS exemplar. The snapshot is a set of Markdown files.
- `LICENCE`
  The Eclipse Public License 1.0 file.
- `README.md`
  A high-level description of the mRUBiS exemplar.

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). In addition, the artifact is also available at: `https://github.com/thomas-vogel/mRUBiS`.

4 Tested platforms

mRUBiS is available as a plug-in for the Eclipse Oxygen.2 (4.7.2) Release with the Eclipse Modeling Tools package. This package provides Eclipse including the Eclipse Modeling Framework (EMF). The artifact has been developed and tested with Java 8 on Ubuntu 16.04 while only basic tests have been made with Windows.

5 License

The artifact is available under Eclipse Public License 1.0.

6 MD5 sum of the artifact

1e9de8c6a61c2850006c5f631dcc32a3

7 Size of the artifact

5896049 bytes (5.9MB on disk)