D.6.7 - ADVANCE TRAINING MATERIAL II

ADVANCE

Grant Agreement:	*287563*
Date:	*30/11/2014*
Pages:	*10*
Status:	*Final*
Authors:	*Andy Edmunds University of Southampton*
Reference:	*D6.7*
Issue:	*1*

Partners / Clients:

FP7 Framework Programme	***European Union***

Consortium Members:

University of Southampton	***Critical Software Technologies***	***Alstom Transport***	***Systerel***	***Heinrich Heine Universität***	***Selex ES***

1 Introduction

This document describes the learning materials that are available for Event-B and Rodin. There are slides and papers from tutorial presentations in Section 2, and web-based learning resources in Section 3. Each section contains brief descriptions of the learning materials, and related links.

2 Papers and Slides

2.1 An Introduction to using Event-B for Cyber-Physical System Specification and Design

Slides providing an overview of the Advance process, including Event-B/Rodin tool features; the development approach; and the timing model.

▷ Link to Slides

2.2 A Tutorial on the Advance Safety Analysis Process

Using a case study involving aircraft landing gear, we provide an overview of the ADVANCE process flow, combining formal modelling with requirements analysis and safety analysis, a process which has now been applied to both industrial case studies. We present a tutorial to guide the user through the ADVANCE process flow covering STPA-based safety analysis, formal modelling and refinement in Event-B, verification and testing.

▷ Link To Report

2.3 Mastering System Analysis and Design through Abstraction and Refinement

Training material based on lectures given at 2012 Marktoberdorf International Summer School on Engineering Dependable Software Systems by Michael Butler, University of Southampton.

The complexity of requirements and complexity of operating environments make error detection in early stages of software system development difficult. This paper makes an argument for the use of formal modelling and verification in early stages of system development to identify and eliminate errors in a timely fashion. Precision is key to eliminating errors in requirements while abstraction is key to mastering requirements complexity.

The paper outlines the way in which precision and abstraction may be achieved through modelling and how refinement allows the complexity to be managed through layering. The role of model validation and model verification in improving the quality of formal models and in improving the quality of the requirements is also outlined. The formalism used throughout is Event-B supported by the Rodin tool-set.

▷ Presentation and Slides

Using a System Theoretic Process Analsyis (STPA) we can use hazard analysis techniques to derive safety constraints that can be modelled formally in Event-B. In the following video the fundamentals of the approach are explained.

▷ Link to Video

2.4 Guidelines for Decomposition of Control System Models in Event-B

The guidelines describe how a system-level Event-B model, of a system with multiple physical devices under some coordinated control, may be decomposed into sub-models. These sub-models represent separate architectural components which include devices, and controllers, and the signalling mechanisms between them.

▷ Link to Report

2.5 Practical Theory Extension - Tutorial

Describes the theory extension capabilities (with a demo) and covers inductive definitions, proofs, and theory deployment.

▷ Link to Slides

2.6 ADVANCE Discrete Event Simulation Training

- Introduction to Discrete Event Simulation
- Introduction to Discrete Event Simulation II

A two part introduction to discrete event simulation for post graduates which covers the fundamental principles of discrete event simulation with particular emphasis on how concurrency is managed and non-determinism is avoided. It will then look at some aspects of simulator performance and explore how a multi-threaded simulator implementation can achieve performance gains for some applications. It will conclude by comparing discrete event simulation with continuous simulation and briefly explore mixed discrete/continuous (hybrid) simulation.

▷ Link To Materials

2.7 ADVANCE Industrial Training

A Practical Introduction to using Event-B for Complex Hardware and Embedded System Specification and Design. Event-B is a proof-based modelling language and method that enables the systematic development of specifications using a formal notion of refinement. The Rodin platform is the Eclipse-based IDE that provides automated support for Event-B modelling, refinement and mathematical proof. This tutorial will show how an abstract, untimed specification can be represented in Event-B and then systematically refined to a timed, architectural representation from which a software or hardware description can be derived for each architectural component. We will show how, at each refinement stage, the proofs needed to verify that the concrete model meets its abstract specification are generated automatically by Rodin. Having proved that the refinement is correct, the Rodin animator and model checker, ProB, is then used to execute the model, verify invariants and temporal properties and show absence of deadlock. We will also show how the Property Specification Language (PSL) or SystemVerilog assertions can be derived from the invariants and temporal properties developed in Rodin for use in a traditional, simulation-based verification flow.

▷ Link To Slides

2.8 ADVANCE Project Partner Training

- The Functional Mockup Interface (FMI) 2.0
- Multi-simulation Concepts

The two presentations cover the fundamental concepts needed to implement a multi-simulation environment and a specific, industry standard interface, FMI 2.0.

▷ Link To Materials

2.9 Integrating Formal Verification and Simulation of Hybrid Systems Rodin Multi-Simulation Plug-in

▷ Link to Slides

3 Web-based Resources

3.1 The Rodin Handbook - Tutorial

The objective of the tutorial is to provide the necessary skills and theoretical knowledge to make use of Rodin and Event-B. Knowledge of logic, and some background in formal modelling, would be useful. Little or no assistance should be needed, to progress through the material. The tutorial also covers installation and configuration for Rodin; and teaches, step by step, how to model using Event-B. We also provide essential theoretical detail, and provide links to further information.

▷ Link To Tutorial

3.1.1 Educational Video

A *YouTube* channel has been created to host video tutorials. The scope of the videos is wide ranging, and covers both the Event-B method and Rodin tool. For example, there are videos giving guidelines for using the platform; modelling advice; use of plug-ins; and lessons in manual proof techniques. Contributions to the channel's content will be provided by the project members during the lifetime of the ADVANCE project.

▷ Link to the Event-B and Rodin channel

3.2 ProB

ProB is an animator and model checker for the B-Method. It allows fully automatic animation of many B specifications. It can be used to systematically check a specification for a range of errors. The constraint-solving capabilities of ProB can also be used for model finding, deadlock checking and test-case generation. ProB can now use Kodkod, a constraint solver for relational logic, as an alternate way to solve constraints.

In addition to B, ProB also supports Event-B, CSP-M, TLA+, and Z. It can be installed in Rodin, and comes with B-Motion Studio; which facilitates generation of domain specific graphical visualizations. ProB is now being used by Siemens and Alstom for data validation of complicated properties. Commercial support is provided by the spin-off company Formal Mind.

Railways Data Validation Tutorial

This tutorial introduces formal data validation principles, based on the interests of the railway industry. It exposes key concepts, the mathematical language to express properties, and the tool chain; and is applied to a number of simple examples.

▷ Link to Railway Data Validation Tutorial

User Manual

Covering a wide range of topics including installation, limitations, consistency checking, constraint-based checking, LTL, use of ProB with Event-B and other formalisms.

▷ Link to User Manual

Tutorial

A tutorial introduction to ProB, focusing on classical B. Much of this material is also applicable to other formalisms; such as CSP, Event-B and Z.

▷ Link to Tutorial

ProB Developer Manual

A document for developers wishing to understand the inner workings of the ProB tool. Includes details of the Tcl/Tk architecture, Prolog, and extending ProB.

▷ ProB Developer Manual

3.3 B Motion Studio

B-Motion Studio is a visual editor which enables the developer of a formal model to easily set up a domain specific visualization to discuss with a domain expert. B-Motion Studio allows the user to create the visualizations based on the "look and feel" principle instead of having to write code.

B-Motion Studio is based on ProB and integrated into Rodin. It supports creating visualizations for Event-B specifications. However, B-Motion Studio is also open for other formal languages.

Commercial support is provided by the spin-off company Formal Mind.

▷ Link to Materials

3.4 ProB Unit Plugin

The Tcl/Tk version of ProB now provides an option for unit analysis. It can be used to verify the usage of physical units within a B machine; which can be done by annotating variables with their physical units; or inference, for variables without annotations. It can also detect incorrect usage of units (e.g. in the addition of variables that are annotated differently). A plug-in is also available for use with Rodin and Event-B.

▷ Link to Materials

3.5 ProB 2.0 with Groovy

ProB 2.0 is the new Java Application Programming Interface (API) for the ProB kernel. The goal of the new kernel is that it would be easy for developers to extend the software. For this reason, ProB 2.0 has integrated the groovy console into the software so that it is easy to test out features while they are being developed. It also provides new programmatic abstractions for commonly used tasks. ProB 2.0 currently supports the animation of models and the evaluation of formulas in the Classical B and Event B formalisms.

▷ Link to Materials

3.6 ProR

3.6.1 Documents

ProR is a Platform for Requirements Engineering (http://pror.org). The Requirements Modelling Framework (RMF) Wiki page contains a user guide as well as an introductory tutorial.

▷ The RMF WIki Page
▷ Link To ProR User Guide
▷ Link To ProR Tutorial
▷ Link To Rodin/ProR Integration Tutorial

3.6.2 Screen-casts

This screen-cast provides an overview of the ProR User Interface. It covers: Workspaces; Projects; creation of Requirements Interchange Format (RIF) documents; Properties and Outline View; Editing requirement content; reordering with drag & drop, copy & paste; adding new attributes to requirements; creating links between requirements

▷ Click Here To Watch

A 5-minute screen-cast to demonstrate the usage of the Diff.

▷ Click Here To Watch

In this talk, we present the Eclipse RMF and its GUI, ProR. RMF supports a generic data model that is based on the emerging ReqIF standard. ReqIF is an Object Management Group (OMG) standard that was developed by the automotive industry to aid the exchange of requirements between manufacturers and suppliers. ProR is the first clean-room implementation of a ReqIF tool. We expect most industry tools to support ReqIF by the end of 2012, and some already do. Support for the project was pledged by Airbus, Thales, MKS and many others (see http://eclipse.org/rmf). As ProR is generic, it can be used with a wide range of development processes. But the real power stems from the ability to extend it with plug-ins to provide additional functionality.

▷ Click Here To Watch

3.7 UML-B

The wiki has two different versions of UML-B (the original UML-B and the new iUML-B). A wiki page has been created with an overview of iUML-B. It describes the general methodologies and conventions of iUML-B diagram modelling and provides links to each of the separate diagram pages.

▷ iUML-B Link Page

A wiki page has been created for the new class diagrams editor. This contains an introductory manual as well as links to a class diagram tutorial.

▷ iUML-B Overview

The wiki page for the state-machines plug-in of iUML-B reflects the latest version of the plug-in.

▷ State-machine Overview

A video tutorial describing iUML-B Statemachines is available:

▷ Link to iUML-B Statemachine Video

3.8 Code Generation

This tutorial is based on the Heating Controller case study. The models used in the tutorial are available from SourceForge, via appropriately placed links. The code generation stage produces implementable Ada code, and also an Event-B model.

▷ Code Generation Tutorial

3.9 Model Decomposition

Model decomposition is a powerful technique to manage the design of large and complex systems. It allows developers to partition the system in a logical manner. Separation into modules aids reasoning at an appropriate level of abstraction, and allows models to remain tractable in terms of reasoning, and efficency of the tool. The decomposed modules can be treated as independent artefacts, so that the modeller can concentrate on each one individually, and does not have to worry about the other parts.

▷ Decomposition Overview

▷ Decomposition Plug-in User Guide

3.10 Composition

first Composition is the process by which it is possible to combine different subsystems into a larger system. Known and studied in several areas, this has the advantage of re-usability and combination of systems especially when it comes to distributed systems.

▷ Composition User Guide

3.11 Generic Instantiation

It is believed that re-usability in formal development should reduce the time and cost of formal modelling within a production environment. Along with the ability to reuse formal models, it is desirable to avoid unnecessary re-proof when reusing models. We propose a way of instantiating generic models and extending the instantiation to a chain of refinements. We define sufficient proof obligations to ensure that the proofs associated to a generic development remain valid in an instantiated development thus avoiding re-proofs.

▷ Generic Instantiation Overview

▷ Generic Instantiation Plug-in User Guide

3.12 Theory Plug-in

The Theory plug-in provides the ability to extend the Event-B language, and the proving infrastructure, in a way that should feel natural for Rodin users. The wiki page provides useful information about the plug-in and its capabilities.

▷ Link To The Wiki-Page

▷ Download User Manual