System Modelling and Design

Introduction

Revision: 1.0, March 3, 2007

Ken Robinson

School of Computer Science & Engineering
The University of New South Wales, Sydney Australia

February 26, 2012

© Ken Robinson 2005-2010

mailto: k.robinson@unsw.edu.au
Outline I

Introduction

System Modelling & Design
System Modelling & Design

This course is called *system modelling and design*.

What is it about?

It is about producing designs for systems, not necessarily software system, that behave consistently with respect to the requirements for the system. We are going to proceed differently to the way you would normally proceed as a programmer.

As a programmer, you generally move from your understanding of the system that you’ve been asked to develop directly to an implementation. If you were asked to explain why the resultant system behaves according to the requirements, you would probably find it very difficult to answer that question.

Indeed it is even possible to argue that it is impossible to answer that question.

In this course we are going to build models based on the requirements, so that we are tackling the question from the other direction: We start with the required behaviour and proceed from there.
Engineering not BlackSmithing

What we are aiming for is *engineering*. Engineers should be able to explain any system they claim to have *designed*.

Blacksmiths bashes a piece of metal until it looks like what is desired.

I submit that many programmers do the same.

None of the common programming “methods”, including *Design Patterns* have any concept of *measurements*, so that, if someone asks you to show why your program does what you claim (believe) it does, you don’t have much of a chance.
We will be using a method named *Event-B*

Event-B consists of:

- **a state:** consisting of a set of variables, whose values define a state that is formally defined;
- **events:** that describe *things that can happen* to the state.

Events consist of:

- **parameters:** values that can be used to control events;
- **guards:** Boolean conditions on the state and the parameters that define the cases for which an event can *fire*;
- **actions:** the change of state that will occur if the event fires.
Events and Requirements

A careful consideration of the above description of an event will show that events are perfect for formalising a requirement:

- what is required to happen;
- the conditions under which it should happen;
- any parameters that affect the requirements.
Correct by Construction

Rodin, the tool that is used to present and develop Event-B models is used to formally verify Event-B models, but very importantly we will be promoting a procedure called

*Correct by Construction*

in which we develop our models from the bottom up in a sequence of *refinement* steps, very carefully done so it is always clear that the model correctly captures the design we are proposing.

Discharging proof obligations then confirms what we believe is already inherent in the construction.

In order to use this plan we generally go through many refinement stages to enable informal inferencing at each refinement step.