Project Guide
System Validation 2016
TU Delft

Jeroen J.A. Keiren

September 6, 2016

1 Project Description

The project concerns designing a controller for a small distributed embedded system. This year's project is described in more detail below. However, it is allowed to design any embedded controller or distributed algorithm, provided you obtain approval by the lecturer of the course. The assignment has to be carried out in groups of four students.

Register your project group on Blackboard!

1.1 Background

The Netherlands is a country with a dense railway system. Our railways have lots of crossings with roads, bicycle lanes, etc. In order to keep traffic safe, most railway crossings today are guarded by bells, lights, and barriers. In this project, you are designing, modelling and verifying such a railway crossing.

1.2 Description

We study a simple model of a railway crossing, consisting of a road along which cars can cross in two directions, towards the north and south. Trains run on two tracks (track 1 and track 2). Trains can arrive in both directs , travelling east or west, on both tracks. Similarly, there are sensors $S_{1,e}, S_{1,w}, S_{2,e}, S_{2,w}$ in the railroad tracks that detect trains that arrive at the railway crossing, and that leave the railway crossing (so, two sensors for each track). An sensors $S_{1,train}$ and $S_{2,train}$ detect whether a train is currently on the crossing. The crossing is guarded by (at least) two barriers Ba_n, Ba_s, two lights L_n, L_s, and a bell, *Bell*. When a train arrives, the lights go on, followed by the bell, and subsequently the barriers close.

1.3 Architecture

In this simple model, trains can arrive not only when the crossing is empty, but also when a train is in the crossing on the other track. Also, they can come from two directions. Once a detector has been triggered, the system needs to make sure trains can cross safely. This means the lights need to be turned on, the bell needs to start ringing, and the barriers need to close. When a train leaves, the barriers can only be opened and the lights and the bell can only be turned off whenever it is safe to do so.

You are required to design a system consisting of at least 4 parallel components, and there should be no component that knows the state of the entire crossing.

If you work on this assignment, you will find out that the description, although quite precise at first sight, still leaves quite a number of aspects undetermined. This is quite common, and by itself already a reason to make a formal description of the behaviour of the controllers. In cases where the behaviour is not well described in this text, you must make your own choice regarding the desired behaviour. You should be able to explain why your decision is sensible. When you feel that deviating from this text would yield a system with a nicer behaviour you are allowed to do so, provided you can defend your choice.

1.4 Extensions

The description of the system as presented above still is a simplification of reality. For instance, there is no way to know whether a car is still on the crossing when the barriers are closing; also, we do not specify any timing information, so we cannot express properties such as: the barriers are closed within 10 seconds after a train arrives.

You are encouraged to extend your system. Some suggestions for extensions are:

- add a lane for bicycles;
- introduce (a) sensor(s) that detect whether cars are still on the crossing;
- instead of having only two barriers (one on each side), have two barriers on each side, so you can split the barriers for traffic entering/leaving the crossing. This allows cars to "escape" from the crossing when barriers are closing;
- introduce signals for the trains to be able to stop the train if the crossing is not yet safe. This is a mechanism that is, for instance, used for crossings near railway stations, where the speed of the trains is low.

Note that correct extensions may be rewarded positively in the grading, and simplifications might affect your grade negatively. However, complete (and successful) verification of a simpler system is generally rewarded higher than incomplete (or unsuccessful) verification of a complex system. It is therefore recommended that you first model and verify the system without any extensions and report on this, and only then describe your extensions and their verification.

2 Project Plan

2.1 Steps

The project comprises carrying out the following steps:

1. Identify in words global requirements for the whole system. Typical requirements are ‘a train may not be on the crossing when the barriers are not closed’. These requirements are initially to be described in natural language.
2. Identify the interactions that are relevant for your system. Describe clearly but compactly the meaning of each interaction in words.
3. Describe a compact architecture of the structure of the system **consisting of at least 4 parallel components**.
4. Translate the global requirements in terms of these interactions.
5. Describe the behaviour of all controllers in the architecture using mCRL2.
6. Verify using the tools that all requirements given in item 4 above are valid for the design in mCRL2.

The project must be documented in a technical report that covers all items above. This report must be a concise technical account of the system and must be written such that from it the requirements, action interface, architecture and behavioural design can be easily understood. It must also be clear how the requirements are verified, in such a way that this can easily be redone without consulting any of the authors of the report.[1] Sample reports from earlier assignments can be downloaded from the course web page.[2]

2.2 Project Groups

You are required to form groups of 4 and register the group composition on **blackboard** before **Monday September 12, 2016 at 17:00**. If you have assembled a group of 4, you can still add yourself to a group that does not have 4 people yet.

[1]When using tools, this means that you also include the exact versions of the tools, and even the platform, that you have used to do the verification.

[2]The reports are not necessarily excellent, and it is likely that there are ways to improve upon them.

2.3 Progress Meetings

Progress meetings will be held on Fridays and groups will be assigned time slots to present and discuss their work on a regular basis. It is strongly advised that the groups prepare well before the meetings to use their meeting time efficiently. Also it is advised to prepare a short report of each project step and review it during each progress meeting. The time-slots for the project meeting will also be announced on the course page.

2.4 Deliverables and Deadlines

Three deliverables are planned for the project:

First deliverable **Monday September 19, 23:59 (start of week 3)**, a report (in the pdf format) containing: introduction, requirements, interactions and architecture.
Feedback discussed during the meeting on September 23.

Second deliverable **Monday October 10, 23:59 (start of week 6)**, the full report (in the pdf format), and a zip file including all source files for models and properties with a short readme text file explaining the content of each file.
Feedback discussed during the meeting on October 14.

Final deliverable **Friday October 28, 23:59 (end of week 8)**, The final report (in the pdf format), a zip file including all source files for models and properties with a short readme text file explaining the content of each file. A short **reflection report** (less than a page) explaining how the project went and the contribution of each member of the group to the final deliverable (a percentage for each member) has to be e-mailed by each and every member separately to `j.j.a.keiren@tudelft.nl`.

3 Assessment

The project consists of 6 parts, that together constitute the design of a verified model. Grading will take into account, but will not necessarily be limited to, the aspects that are described below for each step.

1. **Global requirements of the system.**

    - All listed requirements should be SMART (Specific, Measurable, Attainable, Relevant, Time bound). Focus in grading is on the first four.
    - Requirements should cover all basic functionality described in the problem description.

2. **Interactions with their meaning.** Should cover (at least) the interactions between the system and the outside world. All interactions are explained, and used consistently in later phases.

3. **Architecture.** The architecture consists of at least four parallel components. These components with their connections (channels) need to be identified. The interactions of part 2 need to be mapped onto the links between components. All external interactions that are used in the architecture diagram are explained in the previous step. Diagrams are clearly readable.

4. **Requirements in terms of interactions.** All requirements of 1 need to be made specific in terms of (1) natural language, with reference to the interactions of 2, these formulations need to be precise and non-ambiguous. If the μ-calculus is used in the next step, than the requirements should also be described in terms of the modal μ-calculus (or regular HML, which is a proper subset). If extensions of the system are considered, than the formal requirements should be adapted accordingly.

   Hint: in general it can pay off to split a single requirement formulated at stage 1 into multiple μ-calculus formulas, or to use quantifiers if you want to express properties that are symmetric for all properties. At least, be precise in your formulation!

5. **Behaviour of all controllers in the architecture in mCRL2 .** The mCRL2 model should:

   - contain the interactions of 2 as actions;
   - allow these actions;
   - contain a parallel component in the initialisation for each of the components identified in the architecture;
   - have communications and interactions consistent with the architecture at 3.

   The model should be such that all requirements are satisfied. The document should at least contain:

   - A basic description of the mCRL2 model.
   - Highlight interesting parts of the formalisation.
   - Contain the initial process, and its motivation.

   Note that there is no need to explicitly model the user! You can simply allow the actions in the controller that communicate with the user, without specifying any communication for them. A "User" process does not count towards the four parallel components, and is preferably left out.

6. **Verification of all requirements in 4.** For each of the requirement in 4 a detailed description should be provided for their verification. There are at least two acceptable approaches:

   - Verification of a μ-calculus formula representing the property; it should be explained that the property does not hold vacuously.
   - A visualisation based verification; generate the state space of the system in which all action immaterial to the correctness of the property have been hidden, then the state space is reduced using an appropriate behavioural equivalence (motivate your choice), and of the reduced state space it is argued that the property obviously holds. Note that in this case the property really needs to hold straightforwardly to be acceptable. In general this approach only works if the state space of the reduced system is extremely small.

   The description of the verification process should:

   - Contain a description of *how* the verification is performed.
   - Contain the precise commands and versions of tools used.

General remarks

- Please check language, in particular spelling, before you hand in!
- Someone else (think a student taking the course next year) should be able to redo the verification using your report with the files.
- Make sure the document is consistent! So, use action names consistently throughout the document. Also make sure that the properties you verify are the properties that are described in the document! The document should also be consistent with you mCRL2 specification.
- Include your group number in the document.