System Validation (IN4387) Project Guide

Jeroen J.A. Keiren
VU University Amsterdam

November 2013

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 can be carried out in groups of one to four students.

This year's project consists of a safety system for a bridge. The assignment is inspired by a project carried out at Rijkswaterstaat, which is a Dutch governmental organisation that is responsible for the core infrastructure in the Netherlands. This infrastructure includes roads, waterways, dikes and tunnels. Safety regulations and past experiences indicate that active components, such as bridges and sluices, need to have safety control layers. These layers must ensure that unsafe situations do not occur, but on the other hand they should not unnecessarily, negatively impact the flow of road traffic.

The direct cause for studying bridges is a series of incidents that happened at the "Ketelbrug", which is a bridge in the highway A6 between the Dutch cities Emmeloord and Lelystad. In the first incident and inexperienced operator was controlling the bridge. By then it had become standard to operate the bridge using emergency controls due to problems with the standard controls, e.g. in case of traffic jams. Using the emergency controls caused the bridge to open while the barriers were not closed, and as a result a car drove into the bridge,

Figure 1: Accident at the Ketelbrug. Picture by Harry Otten.

see Figure 1 [1]. This incident lead to the creation of an additional safety layer in the bridge's control system. Some time after its introduction, this system kicked in when a sensor broke, leaving the closing of the bridge undetected; as a result, the barriers were left in their closed position, blocking the road traffic for hours, causing impatient drivers to turn around and drive in the opposite direction on the highway. This indicates that safety requirements in a system like this can be very subtle, and not thinking about progress requirements (the cars should get a chance to cross the bridge) can have adverse effects on safety.

We study a simple model of a bridge, see Figure 2, consisting of two lanes, four barriers $B_1, \dots B_4$, (two on either side), and two sets of signs on either side of the bridge (pre-signs $P_1, \dots P_4$ and stop-signs $S_1, \dots S_4$). The pre-signs signal that the bridge is going to be opened, and new traffic should not enter the bridge (think of a yellow traffic light), and the stop-signs indicate that all traffic should stop (think of a red traffic light), note that the pre-signs come on before the stop signs. The bridge is opened and closed using a motor that can be switched on and off. The status of the motor can be checked (moving up, moving down, stopped, broken). Each of the barriers is equipped with (at least) two sensors, one detecting whether the barrier is up, the other detecting whether it is down. In a similar way, there are at least four sensors in the bridge; two sensors detecting whether the bridge is down, and two that detect whether it is up. Note that you cannot assume that the sensors give consistent information. The stop signs and the pre-signs both consist of two lamps (showing the same information) that can be broken. If none of the lights switch on, the barriers are not allowed to close.

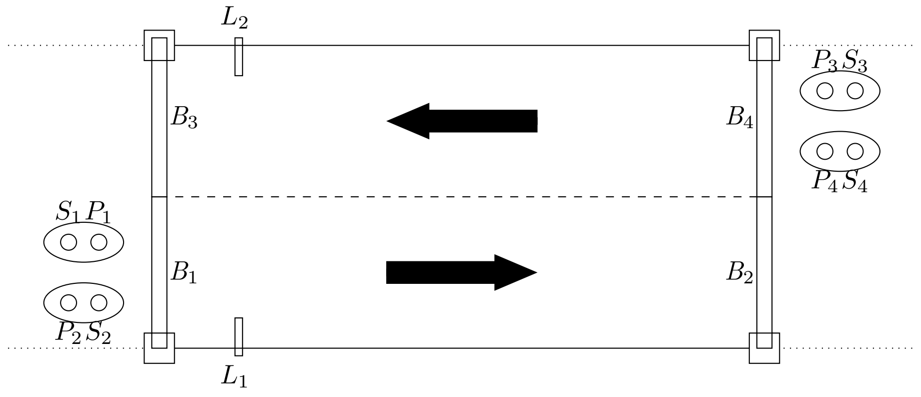

Figure 2: Situational sketch of the bridge

For safety reasons, when the bridge is down it is held in place using two locking pins L_1 and L_2. Each locking pin has a sensor detecting its position. Road traffic may not cross the bridge if the locking pins are not engaged. Also, the bridge has a brake that needs to be applied whenever it is not supposed to be moving.

In reality, the control system of a bridge consists of two parts. The interface controller, which is the system with which the bridge operator communicates. This sends the commands to, e.g, open and close the bridge to the safety controller. If the requested operation is safe, the safety controller forwards the request, and operates the hardware. In this assignment, we only focus on the safety controller, and we ignore the interface controller.

The safety controller can receive commands that operate on the bridge. Its responsibility is to block any command that is unsafe. When an unsafe command is received, an error is sent to the interface controller. Likewise, hardware problems or other error conditions are signalled to the interface controller. While avoiding any dangerous situations, the flow of traffic should only be hampered if this cannot be avoided.

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. 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.

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 'the bridge may not open if 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 3 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 the last year assignment can be downloaded from the course web page.

[1]When using tools, this means that you also include the exact versions of the tools you have used.

2.2 Project Groups

The students are asked to form groups of 4 and send their group composition to `j.j.a.keiren@vu.nl`, before **Monday November 18, 2013 at 17:00**. You can also send a request to the aforementioned email address and the lecturer will place you in a group and email your group composition back to you. The composition of the groups will be announced on the course page.

2.3 Progress Meetings

Progress meetings will be held on Wednesdays and each group will be assigned a time-slot of 15 minutes to present their work. 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 December 8, a report (in the pdf format) containing: introduction, requirements, interactions and architecture.
Feedback discussed during the meeting on October 10.

Second deliverable January 10, the full report (in the pdf format).
Feedback discussed during the meeting on October 24.

Final deliverable January 24, 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 emailed by each and every member separately to `j.j.a.keiren@vu.nl`.

References

[1] J. de Rooij, *Softwarefout veroorzaakte ongeluk Ketelbrug*, Computable, 2011, `http://www.computable.nl/artikel/nieuws/security/3814774/1276896/softwarefout-veroorzaakte-ongeluk-ketelbrug.html`