System Validation (IN4387)
Project Guide
September 2012

Mohammad Mousavi
Eindhoven University of Technology

1 Project Description

The project concern designing a controller for a small distributed and/or embedded system. Below a suggestion for such a system can be found. However, it is possible 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.

The default project of this year consists of a small packet storage system (see figure 1). The assignment is inspired by the distributed controller of an operational product manufactured by a Dutch company called Vanderlande, which is one of the world market leaders in luggage handling-, packet storage- and parcel distribution systems. Vanderlande Industries has installed luggage handling systems in more than 600 airports all around the world and is among the top 5 companies in the world in the area of mechanized distribution centers. The company is located in Veghel (appr. 110 km east of Delft and 20 km north of Eindhoven, see: `www.vanderlande.nl`).

Packets arrive at the left (at C_1). Using a unidirectional conveyor belt packets are moved to the elevators (C_3 and C_4). With the two elevators they can be moved up and down and be stored in the rack (C_5). The lowest elevator can move to the level below the conveyor belt, but it cannot reach the highest rack. The upper elevator can reach the highest rack, but cannot reach the lowest position. The elevators are mounted above each other, which means that they can not reach the same position. Self-evidently, they cannot pass each other either. Each platform of the elevators can store at most one packet.

Each level at the rack can contain exactly one packet. When a platform reaches the right level, instructions must be given to the rack and the elevator to move a packet from or onto the elevator platform. This also holds for the conveyors. When a platform is at the same level as the conveyors, the platform and the conveyor hardware must be instructed to move the packet.

If a packet arrives at this mini warehouse (C_1), the system can decide to accept the packet, but it can also decide to leave it at the entrance if there is no space to store it. However, it is appreciated that an as large as possible number of packets can be stored, where if possible also the elevator platforms can be used as storage positions.

At the right (C_2) packets can be ordered which will then be delivered in the same order as they have been requested. It can be assumed that packets can always be accepted by whatever device requested the packets.

There are five controllers C_1, C_2, C_3, C_4 and C_5 that govern the behaviour of the system. The assignment is to model the behaviour of these controllers and prove that they cooperate well. Each controller is only allowed to have knowledge of its own device. So, only C_5 is allowed to know which levels of the rack are filled with which packets. And similarly, only C_3 knows to which level the upper elevator is travelling. The purpose of this is to make each device with its controller modular and therefore reusable in other contexts.

This means that if a packet is ordered at the controller C_2, it must request all the other controllers to find out whether a packet is present and where it is located. Subsequently, a plan

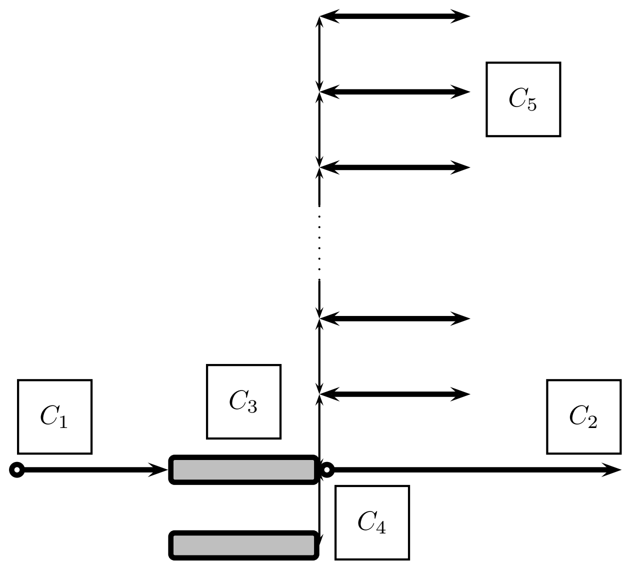

Figure 1: A temporary storage with two elevators

must be negotiated to move the packet to the exit. Note that at the same time packets can enter the system, which must be stored simultaneously.

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.

What is important is that if packets are present in the system, the system will always deliver them. Under no circumstance the system may deadlock or be able to only deliver the packets in predetermined orders. Furthermore, packets may not bump into each other. This also holds for the elevators.

In a realistic system, one may also have to deal with failing motors or other mishap, meaning that an elevator gets stuck or a storage level cannot be used. Sometimes, it is even the case that packets are removed from the storage rack or drop to the floor. In such cases the system must inform the outside world that something has gone wrong. It is not necessary to deal with these situations in this assignment. It is however important to realise that the current system is quite a simplification of those systems that occur in reality.

2 Project Plan

Steps. Carrying out the project consists of executing the following steps:

1. Identify in words global requirements for the whole system. Typical requirements are 'packets are never stored at places that are occupied'. 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. Translate the global requirements in terms of these interactions.

4. Describe a compact architecture of the structure of the system.

5. Describe the behaviour of all controllers in the architecture using mCRL2.
6. Verify using the toolset that all requirements given in item 3 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. Sample reports from the last year assignment can be downloaded from the course web page.

Project Groups. The students are asked to form groups of 4 and send their group composition to `smrmousavi@gmail.com`, before **Friday September 14, 2012 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 comosition of the groups will be announced on the course page.

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.

Deliverables and Deadlines. Three deliverables are planned for the project:

First deliverable October 5, a report (in the pdf format) containing: introduction, requirements, interactions and architecture
Feedback discussed during the meeting on October 10

Second deliverable October 19 ,the full report (in the pdf format).
Feedback discussed during the meeting on October 24

Final deliverable The final report (in the pdf format), a zip file including all source files for models and properties with a short readme text file explainig 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 `smrmousavi@gmail.com`.