Project Guide
System Validation 2018-2019
TU Delft[*]

Jeroen J.A. Keiren (`j.j.a.keiren@tudelft.nl`)

September 10, 2018

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 Brightspace!** If you already participated in the project in a previous year, please contact the lecturer.

1.1 Description

Transfer systems are commonly used as a part of industrial production machines and plants. One area where such systems are used is in the handling and production of (silicon) wafers. This plays a key role in the production of microchips.

In this project you are to specify and verify a (simplified) machine that prepares wafers for the production of microchips (such machines are, for instance, produced at ASML).

A sketch of the system is shown in Figure 1. The machine consists of a UV lamp (L) that is to project a design onto a wafer exactly once per wafer. This is done in a vacuum chamber, in which wafers can be inserted through two airlocks (A_1 and A_2). Inside the vacuum chamber, there is a robot (R_3) that can move to the projector (L) and the airlocks (A_1 and A_2), and that can pick up or drop a wafer at these locations.

Outside the vacuum chamber, there are robots (R_1 and R_2). Here, R_i can pick wafers up from input stack I_i and place them in the corresponding airlock (A_i). When a wafer is ready, a robot R_i can retrieve it from the airlock, and put in on the corresponding output stack O_i.

[*]The project description has been adapted from a project described by Sjoerd Cranen.

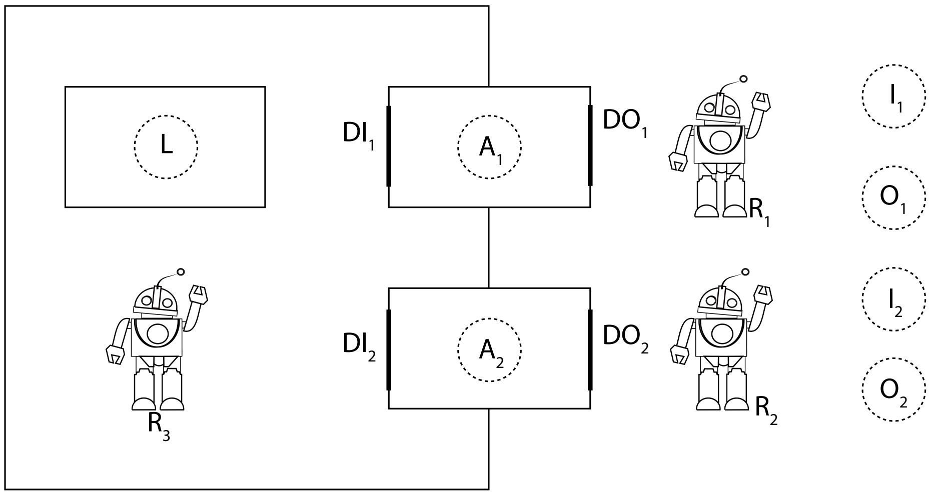

Figure 1: Schematic depiction of an extended machine.

Of course, a robot can only reach A_i when the correct door (DI_i or DO_i) is open.

Robots can be ordered to move to a position, pick up a wafer or drop a wafer. The lamp can be instructed to project the design onto a wafer. Doors can be instructed to open and close. Actuators send a reply when a command has executed correctly. Doors may get stuck, and the input stacks may become empty, or the output stacks may become full. As long as a wafer can be moving though the production process, the machine should ensure that it does so. Assume that all locations marked by dotted circles have a sensor that tells whether a wafer can be picked up or dropped on that location.

1.2 Architecture

You are required to design, model, and verify a control system for the system consisting of at least 4 autonomously operating controllers (parallel components) that are communicating when needed, and there should be no component that knows the state of the entire machine.

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.

1.3 Simplifications and extensions

The description of the system as presented already is very much a simplification of reality; however, it may be wise to start modelling with an ever simpler version (e.g. only consider a single airlock). You are also encouraged to think about extensions.

Note that correct extensions can positively affect the grade, and simplifications might affect your grade negatively. However, complete (and successful) verification of the basic 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 as set of meaningful (functional) requirements for the whole system. Typical requirements are ‘except for the stacks, there are never two wafers in the same location’, which can be described in more detail by considering the specific layout of the system. These requirements are initially to be described in natural language.

   Hint: Your requirements should contain both *safety* requirements and *liveness* requirements.

2. Identify the (observable) interactions between components that are relevant for your system. Describe clearly but compactly the meaning of each interaction in words. An example such an interaction is *turn UV light L on*.

3. Describe a compact architecture of the structure of the system **consisting of at least 4 parallel components**.

   The idea is to simply draw boxes for the components in your system, and connect them according to the way in which they communicate. Identify the interactions described in the previous step, and indicate them in the architecture. Also identify what interactions *between* your components are needed in order to ensure correct operation of your controller.

4. Describe the behaviour of all controllers in the architecture using mCRL2.

5. Verify using the tools that all requirements given in item 1 above are valid for the design in mCRL2.

Hint: if your system is small, you might be able to verify some properties by using behavioural equivalences, and comparing some abstraction of your system to a specification that describes the desired property. It is more likely, though, that you will need to formalise your requirements using the modal μ-calculus.

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 Brightspace.[2]

2.2 Project Groups

Project groups consist of 4 students. You can choose your own group, and you should enroll in a project group on Brightspace. Make sure to enroll in **brightspace** before **Monday September 17, 2018**.

If somehow, immediately after the start of the project, you end up with a group of fewer than 4 people, please let me know, and I will merge groups.

2.3 Progress Meetings

Progress meetings will be held on Mondays during the second half of the course and groups will be assigned time slots to 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 Brightspace.

2.4 Deliverables and Deadlines

Three deliverables are planned for the project:

First deliverable **Wednesday September 19 2018, 23:59**, a report (in the pdf format) containing: introduction, requirements, interactions and architecture. Feedback will be given digitally.

Second deliverable **Wednesday October 10 2018, 23:59**, the full report (in the pdf format), and a zip file including all source files for models and

[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.

properties with a short readme text file explaining the content of each file. Feedback will be given digitally.

Final deliverable **Wednesday October 31 2018, 23:59**, 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.

Reflection report **Wednesday October 31 2018, 23:59**, A short **individual** 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 handed in by each and every member separately. The results of the reflection reports will be used to adjust the individual grades of the team members.

Deliverables (except for the reflection report) are to be handed in through Brightspace.

3 Assessment

The project consists of 5 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.

   Common problems:

   - Missing liveness requirements.
   - Referring to a safety requirements as “functional requirement”.

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

   Common problems:

   - Missing explanation.
   - Not consistently used in later phases.
   - Wrong granularity.

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 interactions that are used in the architecture diagram are explained in the previous step. Diagrams are clearly readable.

   Common problems:

   - Multiple architectures are given. Give a single architecture, namely the one that you verified!
   - Interactions in architecture diagrams are not named consistently with the interactions defined in part 2.
   - Diagrams are cluttered.
   - Internal interactions are not identified.

4. **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 report should at least contain:

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

   Common problems:

   - No description of the formalisation in mCRL2.
   - No explanation of the structure and/or interesting parts of the mCRL2 model.
   - No description of the initialisation of the process (what communications are there, what parallel components do you have).

5. **Verification of all requirements in 1.** For each of the requirements in 1 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.

For every requirement that is verified using the μ-calculus, your report should contain the μ-calculus formulas used.

Hint: in general it is a good idea to split a single requirement formulated at stage 1 into multiple, more detailed requirements, and 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!

Common problems:

- Writing $[a]\mathit{true}$. This is a mortal sin! The formula is always true. Likewise, $\langle a\rangle\mathit{false}$ is always *false*!
- Incorrect syntax, *e.g.* writing $[\overline{a*}]\phi$ instead of $[\overline{a}^*]\phi$.
- Writing functional requirements such that something only needs to happen once (instead of from every reachable state).
- Using actions that are not defined as interaction in part 2.
- Not making indices (in parameters of actions) explicit. Either instantiate them, or quantify (*e.g.* $\forall id : \mathit{SensorId}.\phi$).

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

- Your report should be *concise and to the point*, but should also be complete.
- Make sure your language, in particular spelling, is correct.
- Someone else (think a student taking the course next year) should be able to redo the verification using your report with the files.
- Your report should be consistent! So, e.g., use action names consistently throughout the document.

- Make sure that the properties you verify are the properties that are described in the document!
- The report should also be consistent with your mCRL2 specification.
- Your group number, and the names and student numbers of each of the group members should be included in the report.