

Requirement analysis, design and verification

RADV, 2IW20

Jan Friso Groote (J.F.Groote@tue.nl, HG 6.75, 247 5010) and Jaco van de Pol (J.C.v.d.Pol@tue.nl, HG 6.88, 247 2963).

<http://www.win.tue.nl/~jfg/educ/2IW20/2005/overzicht.html>

The purpose of this course is to learn means to specify behaviour of systems and to experience the design of a system, starting from behavioural requirements which must be proven to hold on the design.

The first five weeks of the trimester will be used to learn the specification language μ CRL and to use it as a manual verification tool. After these five weeks there will be an examination. There is one possibility to retry this exam, at the end of the second trimester. The next five weeks are dedicated to an assignment. Here the goal is to apply the techniques and tools to the design of a small distributed and/or embedded system. The purpose is to design this system such that it is proven to comply with the requirements.

The marks for the exam and the assignment both contribute equally to the final score.

Literature

The course material consists of

- W.J. Fokkink, J.F. Groote and M.R. Reniers. Modelling Distributed Systems. Lecture notes. 2003.
- A.G. Wouters. Manual for the μ CRL toolset (version 2.8.2). Technical Report SEN-R0130, CWI, 2001.

The lecture notes by Fokkink, Groote and Reniers and the manual by Wouters can be obtained at the office of the secretary (Tineke van de Bosch, 247 5010, HG 6.74). The documents can also be downloaded from the website. The exam will cover the lecture notes on modelling distributed systems as well as everything said during the lectures. The manual for the μ CRL toolset will not be a part of the exam.

Assignment

The assignment consists of designing a controller for a small distributed and/or embedded system. Below a suggestion for such a system can be found, but, in consultation with the teachers, also another system can be chosen. The assignment can be carried out in groups of one to four persons.

Carrying out the assignment consists of the following steps:

1. Identify in words global requirements on the whole system. Typical requirements are ‘two robots arms never access the same critical area at the same time’. These requirements can be described in natural language.
2. Identify the interactions of the control system to the whole system. Describe clearly but compactly the meaning of each interaction in words.

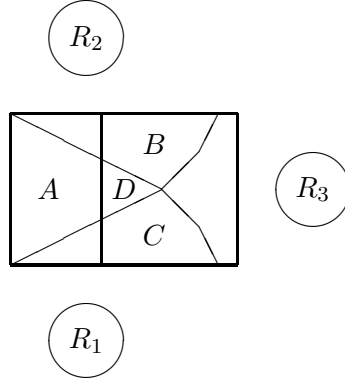


Figure 1: A production unit

3. Translate the global requirements in terms of these interactions.
4. Describe a compact architecture of the structure of the control system.
5. Describe the behaviour of all components in the architecture in terms of μ CRL.
6. Verify using the toolset that all requirements given in item 3 above are valid on the design in μ CRL.

The assignment 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 the 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.

The default assignment is the following, inspired by an assignment of an internee at Philips CFT who had to design the safety layer for a dual pick and place machine. Such machines are used to put electrical components on printed circuit boards. We rephrase the assignment to contain three robots, instead of two.

We assume a product assembly platform with three robots (see Figure 1). Initially the platform is empty, and the robots are in a safe initial resting position. The product enters the platform autonomously and will communicate that it has entered the platform. In the mean time instructions are issued to each robot, that the robots must execute. A correctly executed instruction is positively acknowledged. If the robots have executed their commands, they are instructed to go back to their resting positions, and after having done so, a signal is given that the product can be removed. After this the platform is ready to receive another product.

A robot may have to enter areas where other robots have to carry out work as well. Robot R_1 can access common areas A , B and D , robot R_2 can access common areas A , C and D and robot R_3 can access such areas B , C and D . Collisions between robots must be avoided.

Each robot can enter a faulty mode. Whenever a robot detects an internal software inconsistency, faulty hardware, or has any other problem it indicates an error, and it cannot guarantee anymore in which area the robot arm resides. In this case access to all areas reachable by the faulty robot is forbidden. Moreover, all instructions that cannot be executed anymore must be refused by the system.

The assignment is to design a safety layer, that accepts the instructions and forwards these to the robots, whenever these can safely execute them. A complication is that the manufacturer does not want to spend a separate computer for the safety layer, and therefore it must be implemented on the three computers that are already in the system controlling the robots. The computers are connected by an asynchronous point to point network.

This assignment is underspecified. This means that in certain cases you have the freedom to make your own design decisions. Note that a difficult aspect of this assignment is the asynchronicity of the network and the robots. If a robot indicates failure, to which extent can other robots be prevented from doing harm? There is a trade-off between optimizing production and allowing a risk for a collision.

Typical problems that can be expected are:

- Robots start to operate before a product has entered the platform.
- Robots can collide in normal operation.
- A robot has indicated faulty behaviour, but the system still accepts and tries to execute instructions that may cause a collision between the faulty and a correctly operating robot.

A model of the control system must be made, that guarantees that problems will not occur, while guaranteeing that the machine can produce at maximal speed. If potentially hazardous situations are still allowed because of maximal production, or other reasons, this should be very clearly be stated in the report. The model must be such that qualified programmers can easily implement it.

Tool sets

See the web page of this course for information about the tools and the toolset.

Global time schedule

Below a global time schedule is indicated. Furthermore, there are exercises indicated, which will be treated during the lecture. Students are supposed to make these exercises before the lectures, and compare their answers with those of the lecturer. The exercises can be found in the lecture notes on Modelling Distributed Systems.

1-12-2004. Abstract data types.

2-12-2004. Elementary process algebra, RSP, bisimulation. Exercises 2.5, 2.12, 2.14.

8-12-2004. Data in actions and processes, sum operator and conditionals. Exercises 3.3, 3.7, 3.10, 3.13, 4.3.

- 9-12-2004. Linear process equations, CL-RSP, Invariants (chapter 8). Exercises 3.17, 3.20, 3.21.
- 15-12-2004. Hidden actions. Exercises 8.8, 8.18.
- 16-12-2004. Protocol specifications. Exercises 5.1, 5.7, 5.11, 5.12.
- 22-12-2004. Linearisation of processes. Exercises 6.8, 6.11.
- 23-12-2004. Confluence and τ -priorisation. As exercises linearise the processes $X = a \cdot X \cdot X$, and $X = Y \cdot \delta$, $Y = a \cdot Y \cdot Y + c$.
- 12-1-2005. Cones and foci theorem. Exercises 7.9, 7.10 and 9.7.
- 13-1-2005. Questions and reserve. Exercises 9.4 and 9.5.
- 18-1-2005. Exam.
- 27-1-2005. Assignment, μ CRL toolset.
- 3-2-2005. Tools such as parelm, constelm, structelm, etc.
- 10-2-2005. Instantiator, confluence reduction.
- 17-2-2005. Prover. Modal logic. Muccheck.
- 14-3-2005. Exam (retry).
- 18-3-2005. Latest possible date to hand in the assignment.