Emergency Department
Follow-up Treatment

J.T van Boggelen
ARGESIM comparison C 6: Emergency Department - Follow-up Treatment

Subject
ARGESIM is a European non profit working group providing the infrastructure and administration for dissemination of information on modelling and simulation in Europe. The ARGESIM working group has defined several modeling and simulation cases for the purpose of evaluating the capabilities of modeling and simulation languages. These cases, and their simulation models defined in many different languages, are available at http://www.argesim.org/index.php?id=68.

Assignment
Model and simulate ARGESIM comparison number 6: the emergency department with follow-up treatment. This example addresses discrete simulation languages and tests features for modelling, concepts of availability, and complex control strategies. Compare the results with the results of other simulators as specified at the ARGESIM web site and discuss the differences. Use the new CIF 3 language with visualization tools, see http://cif.se.wtb.tue.nl.
Emergency Department
Follow-Up Treatment
ARGESIM comparison 6

J.T. van Boggelen

SE-420729

Bachelor Final Project

Supervisor: dr. ir. D.A. van Beek
Advisor: ir. L. Swartjes

Eindhoven University of Technology
Department of Mechanical Engineering
Manufacturing Networks Group

Eindhoven, July 3, 2013
Summary

To compare different programming languages, the TU of Vienna created a program called ARGESIM. Several benchmarks are used for comparison. In this report, one of these benchmarks is used to compare the programming language CIF, with other programming languages.

The benchmark is about the follow-up treatment at an Emergency Department. The patients have to go through the treatments rooms, depending on the patient type. The Emergency Department is simulated with a model by using CIF as the programming language.

Three strategies are tested and compared. The normal strategy, the Doctor Exchange Strategy, where the doctors of the wards change, and the Priority Ranking Strategy, where the patient that enter the ward for the second time get a priority. The use of the strategies has a different effect on the average waiting time of the patients.

The verification of the model is done by several means. The throughput of a single patient is tested. Also the total number of patients that generated in the model should be the same as the number of patients leaving the model. An option in CIF is to always choice the first or last transition or choice a random transition. As the model is a deterministic model, there should be no difference in the results between the different options. This is also tested. These three methods can be combined with the visualization of the model to get out the errors in the model.

The results of the CIF mode are compared with the other programming languages. Due to different assumptions that were used with the different programming languages, the results can vary. Nevertheless the difference in the results is small. The results of CIF are comparable with the results of the other programming languages. Only the results of the Priority Ranking Strategy show some difference, because of the assumptions.
# Table of Content

1. Introduction ......................................................................................................................... 1
   1.1. ARGESIM .................................................................................................................... 1
   1.2. CIF ............................................................................................................................... 1
   1.3. Emergency Department ............................................................................................... 1
   1.4. Outline ......................................................................................................................... 2
2. Emergency Department ...................................................................................................... 3
   2.1. System overview .......................................................................................................... 3
   2.2. Assumptions ................................................................................................................ 5
   2.3. Task A: Overall Treatment Time .................................................................................. 5
   2.4. Task B: Doctor Exchange strategy ............................................................................... 5
   2.5. Task C: Priority Ranking ............................................................................................ 6
   2.6. Control strategy ........................................................................................................... 6
3. Model .................................................................................................................................. 7
   3.1. Patient ......................................................................................................................... 7
   3.2. Arrival process ............................................................................................................. 7
   3.3. Queues ........................................................................................................................ 7
   3.4. Registration .................................................................................................................. 7
   3.5. Wards .......................................................................................................................... 7
   3.6. Plaster ............................................................................................................................ 8
   3.7. X-ray ............................................................................................................................ 8
4. Results ................................................................................................................................ 9
   4.1. Task A: Overall Treatment Time .................................................................................. 9
   4.2. Task B: Doctor Exchange strategy ............................................................................... 9
   4.3. Task C: Priority Ranking .............................................................................................10
   4.4. Comparison ................................................................................................................11
5. Validation ...........................................................................................................................13
6. Verification .........................................................................................................................15
   6.1. Assumptions .................................................................................................................15
   6.2. Comparison ................................................................................................................17
7. Conclusion and Recommendation .....................................................................................21
Bibliography ..............................................................................................................................23
1. Introduction

This project is based on an ARGESIM benchmark. This chapter starts with a brief introduction to ARGESIM. After that the simulation language CIF will be explained. Next, a short explanation the emergency department of benchmark will be given. This chapter ends with the outline of the following chapters in this report.

1.1. ARGESIM

ARGESIM is an abbreviation of ARbeitsGEmeinschaft SIMulation. ARGESIM is an establishment of ARGE Simulation news which is a nonprofit working group of the TU in Vienna [1]. ARGESIM is set up to compare different simulation software. To compare the different software a number of benchmarks were set up. These benchmarks are simple, comprehensible models. Based on these models special features of modeling and experimentation within simulation languages, also with respect to an application area, are compared. Features are, for instance: modeling technique, event handling, numerical integration, steady-state calculation, distribution fitting, parameter sweep, output analysis, animation, complex logic strategies, sub models, macros, and statistical features.

1.2. CIF

The simulation software used for this benchmark is CIF. CIF stand for Compositional Interchange Format. “CIF is an automata-based modeling language for specification of discrete event, timed, and hybrid systems. The CIF 3 tooling will support the entire development process of controllers, including specification, supervisory controller synthesis, simulation-based validation and visualization, verification, real-time testing, code generation.”[2] The project was simulated with CIF 3 the latest version of CIF. CIF 3 was created and is currently developed by the Systems Engineering group of the Mechanical Engineering department, at the Eindhoven University of Technology (TU/e).

1.3. Emergency Department

The benchmark of ARGESIM that is used in this report is about an emergency department for dressing of wounds. See appendix F. Broken limbs are put in plaster. After a few days a follow-up examination must be performed to monitor the healing process. If necessary, additional treatment will be administered.
1.4. Outline

In chapter 2 the Emergency Department is described in more detail. The model is explained in chapter 3. In chapter 4, the results of the simulations are given. The validation of the model will be discussed in chapter 5. Next, in chapter 6 the verification of the model is discussed. Finally, the conclusions and recommendations are given in chapter 7.
2. Emergency Department

2.1. System overview

Figure 2.1: Schematic plan of the emergency department with the treatment rooms.

Figure 2.1: Schematic plan of the emergency department with the treatment rooms.
The emergency department is divided into the following facilities:

- **Registration**: Casualties are assigned to casualty wards 1 or 2.
- **Wards (CW1 and CW2)**: Patients get treatment in the ward by one of the two doctors which are available in each ward.
- **X-ray**: Consists of two X-ray units with a combined waiting line.
- **Plaster**: Room where plaster casts can be removed or applied. Only one doctor will work here, with a single waiting queue.

During registration, the patients are diagnosed with the necessity of further treatment. The patients are then given a patient type to define the follow-up treatment. Four types of patients are defined:

1. Patients requiring X-raying: Patients are first examined in the casualty ward, and then sent to the X-ray room. Before they leave, their X-ray photographs are examined once again in the casualty ward.
2. Removal of plaster casts: Patients enter a casualty ward, are sent to the plastering room, and then leave the department.
3. Plaster casts requiring X-raying and renewal: Patients enter the casualty ward, are sent to the X-ray room and are given new plaster casts. After checking the new plasters by X-raying, afterwards patients are readmitted to the casualty ward, they leave the department.
4. Changing wound dressings: Patients are admitted to a casualty ward, and then leave the department.

The percentile distribution of the patients over the four groups is as follows:

1: 35%, 2: 20%, 3: 5%, 4: 40%.

The emergency department of the hospital opens at 7.30 a.m. The patients start to arrive from the beginning of the day. The doctors start to work at 8.00 a.m. The time between the arrivals of patients is exponentially distributed with a mean of 0.3 minutes.

When the right treatment is diagnosed for the patient, the patient is sent to ward one or ward two. The percentile distribution is 60% to 40% respectively.

The treatment time of the patients in minutes in each of the treatment rooms are triangular distribution (minimal value / most likely value / maximum value).

- **Registration**: 0.2 / 0.5 / 1.0
- **CW1**: 1.5 / 3.2 / 5.0
- **CW2**: 2.8 / 4.1 / 6.3
- **X-ray**: 2.0 / 2.8 / 4.1
- **Plaster**: 3.0 / 3.8 / 4.7
2.2. Assumptions

Some assumptions had to be made to model the system. Namely, based on the specification, some properties remained undefined. When a patient travels between a waiting queue and a treating point or from a treating point to a queue of another treatment, no traveling time or delay is modeled.

To justify neglecting of the travel time, we make an estimation of the travel time. We take 0.2 minutes as estimation. For all the treatment rooms the travel time is in proportion smaller than 8 % of the process time. So the travel time is negligible at the treatment rooms. At the registration the travel time is of the same magnitude as the process time. However, because the doctor arrives 30 minutes later, the queues for the wards are already filled with patients, independent of the travel time of the patients. Also the queues of the wards will never be empty as new patients coming keep on arriving. So travel time can be neglected.

Even though the doctors start to work at 8.00 a.m., the registration starts to work at 7.30 a.m. when the patients start to arrive. Another assumption that is made is when there is a change between the doctors in the different wards, both doctors have to be available. When both doctors are busy and one of the doctors finishes his treatment, the doctor has to wait until the other doctor is finished with his treatment before the doctors can switch between the wards.

2.3. Task A: Overall Treatment Time

To see how the follow up treatment works a, model is made and tested with various set-ups. The first test, Task A, was to determine both the overall treatment time for 250 patients and the overall treatment time for each patients type. The model was made in CIF 3.

2.4. Task B: Doctor Exchange strategy

When the treatment time of the system is calculated, other strategies can be implemented in the system to see if this gives a better result for the overall treatment time. In task B the Doctors Exchange strategy is used. If the waiting queue in CW2 excesses 20 patients, one of the experienced doctors in CW1 replaces one of the inexperienced doctors in CW2. The doctor from CW2 will go to CW1, but due the more complex cases he has to deal with; the working time of the doctor will increase with 20 %. As soon as the waiting queue for CW2 is down to five people and the doctor of CW1 is still in CW2, the doctors exchange back to their normal casualty ward.
2.5. **Task C: Priority Ranking**

Another strategy which is implemented is the priority Ranking strategy. This means when a patient is entering a treatment room for the second time, this patient has priority over patients who come in for the first time. This only happens with patient types 1 and 3 in the wards and in the X-ray.

2.6. **Control strategy**

To control the system, some control strategies have to be applied at certain treatment rooms. When a patient is leaving the registration, a decision has to be made to which ward the patient is going. The decision is made on a percentile distribution. So the decision can be made locally based on the local variables of the percentile distribution.

The same decision has to be made if a patient leaves the X-ray and has to go one of the wards. This decision is made locally in the X-ray based on the local variables of the percentile distribution. This decision is only made when there is no priority involved in the system.

When there is priority in the system, the patients in the priority queue of the ward have to be treated first. So the decision to treat the priority patients first is made locally based on the state variables of both the priority queue and the normal queue.

When the doctors need to be changed in the Doctors Exchange Strategy the decision to switch is based on the state variables in both wards. The doctors in both wards are checked and it is verified whether both doctors they are available.
3. Model

The simulation model is built in CIF 3.0. This simulation tool is developed by the Manufacturing Networks Group at the TU/e. CIF 3.0 is a plug-in for the Java IDE Eclipse environment with Modeling Tools. The model simulation can be found in appendix B.

3.1. Patient

The patient is modeled as a tuple with three variables; the arrival time, the number of times it passes the X-ray, and the patient type. The variables of the patient type and the number of times passing the X-ray are necessary to send the patient to the right treatment rooms in the right order. The arrival time is necessary to calculate the overall treatment time and standard deviation on that.

3.2. Arrival process

The arrival process is simulated in the plant generator. Here patients are generated with an inter arrival time that is distributed exponentially with a mean of 0.3 minutes. The generator generates 250 patients in total.

3.3. Queues

The queues are simulated as buffers with a first in, first out (FIFO) policy. The buffers contain an array with all patients in the waiting queue. A new incoming patient is added to the array at the end. When a patient leaves the buffer the first patient from the array is sent to the treatment room and the remaining patient are retained in the array.

3.4. Registration

In the registration, the patient are indentified and given a patient type. The patient type is assigned with a percentile distribution. The percentile distribution is simulated by a uniform distribution and four guards. When one of the guards is true, the corresponding patient type is put in the patient tuple. It is not possible that multiple guards are correct, so every patient has only one patient type. After the registration the patients are send to one of the casualty wards. To decide which ward the patient is send, a uniform distribution with 2 guards is used.

3.5. Wards

In the ward each of the doctors is modeled as a plant. In the initial state the doctor has to wait 30 minutes before they begin. After that, the doctor can receive a patient from the queue. The time the doctor needs for the treatment is triangular distributed. The triangular distribution also depends on whether the doctor is experienced or inexperienced. When the patient is treated
and leaves the ward, the patient can go to 3 treatment rooms and the exit. When the patient is of type 2 it goes to the plaster. When the patient leaves the ward for the first time and is of type 1 or 3 the patient goes to the X-ray. When it is the second time the patient is treated in the ward, it leaves to the exit. If the patient is of type 4 it leaves the system.

When the Doctor Exchange Strategy is implemented in the model (Appendix C), the doctors have to check whether they are allowed to switch, and when they have to switch back. When, in the initial position, the doctors are in their corresponding ward, a doctor is finished with the treatment of a patient, the queue for CW2 is checked, whether it is longer than 20 patients. If it is shorter the doctor receives a new patient and starts his treatment. When the waiting queue is longer, the doctor is checked whether is busy with a treatment. If so, the doctor will wait until the doctor is finished with the treatment. If not, the doctors switch ward. Now the doctors have changed, they have to check whether the waiting queue for CW2 is under 5 patients. To check this, the same control strategy is used as to check the waiting queue in the initial position.

When the Priority Ranking Strategy is used, the doctor has to treat the patients coming in for the second time. The patients who come in first are in the normal waiting queue and the patients who come in for the second time are in the priority waiting queue. There is only one priority queue for all 4 doctors. So first it is checked if there are any patients in the priority buffer. If so the first patient in the priority buffer gets treated. If not the first patient in the corresponding queue of that ward gets treated. The simulation of the implementation of the Priority Ranking Strategy can be found in appendix D.

### 3.6. Plaster

The plaster is a room where plaster casts are applied or removed. The plaster room receives patients of type 2 and 3. Patients of type 2 come in from the ward. Patients of type 3 are coming from the X-ray. The treatment time of the doctor is triangular distributed. When a patient leaves the plaster it goes to the X-ray or exit depending on the patient type. Patients with type 2 go to the exit. Patients with type 3 go to the X-ray.

### 3.7. X-ray

In the X-ray the doctor checks if the broken limbs are in the correct position. Patients with type 1 and 3 are treated in the X-ray. Depending on the type of the patient, type 1 or 3, the patient has to be treated respectively once or twice. When the patient enters the X-ray the patient information is updated so the number of times the patient was treated in the X-ray is correct. When a patient with type 1 is treated it is send back to one of the wards for the examining of the X-ray photographs. When a patient with type 3 is treated, the patient information is checked to see how many times the patient was treated in the X-ray. When it is the first time the patient was treated the patient leaves to the plaster. When it was the second time, the patient leaves to the one of the wards.
4. Results

The results were calculated by averaging the results of 20 simulations. The simulations were made with random transitions with a random seed. In this chapter the results of each task are discussed. The results of each simulation and the combined results can be checked in appendix C. Finally, the results of each task are compared with each other.

4.1. Task A: Overall Treatment Time

The results of the simulations can be found in appendix A, Table A.1. The main results are given in Table 4.1. After $386 \pm 22$ [min] all patients were treated, and left the emergency department. The average treatment time of patient type 1 and 3 is much larger than of the patients with type 2 and 4. This is because patient 1 and 3 have to go to the ward twice. When they come back at the ward for the second time the waiting queue is expanded, so the waiting time of these patients is increased.

Table 4.1: Results of task A. The treatment time and the standard deviation of the treatment time of the patients are plotted against the different patient types and the combined results of the patients.

<table>
<thead>
<tr>
<th>Task A</th>
<th>treatment time [min]</th>
<th>standard deviation [min]</th>
<th>Count [-]</th>
</tr>
</thead>
<tbody>
<tr>
<td>type 1</td>
<td>236.3</td>
<td>53.8</td>
<td>88.9</td>
</tr>
<tr>
<td>type 2</td>
<td>140.5</td>
<td>31.8</td>
<td>50.5</td>
</tr>
<tr>
<td>type 3</td>
<td>252.5</td>
<td>21.3</td>
<td>12.7</td>
</tr>
<tr>
<td>type 4</td>
<td>127.9</td>
<td>47.6</td>
<td>98.0</td>
</tr>
<tr>
<td>overall</td>
<td>175.5</td>
<td>81.6</td>
<td>250</td>
</tr>
</tbody>
</table>

4.2. Task B: Doctor Exchange strategy

The results of the simulations can be found in appendix A, Table A.2. The results of task B are comparable with the results of task A. The results of task B also have a higher treatment time for patient type 1 and 3. The main results of task B are given in Table 4.2. After $414 \pm 16$ [min] all patients were treated, and left the emergency department.
Table 4.2: Results of task B. The treatment time and the standard deviation of the treatment time of the patients are plotted against the different patient types and the combined results of the patients.

<table>
<thead>
<tr>
<th>Task B</th>
<th>treatment time [min]</th>
<th>standard deviation [min]</th>
<th>Count [-]</th>
</tr>
</thead>
<tbody>
<tr>
<td>type 1</td>
<td>242.3</td>
<td>58.8</td>
<td>88.9</td>
</tr>
<tr>
<td>type 2</td>
<td>140.2</td>
<td>34.5</td>
<td>50.3</td>
</tr>
<tr>
<td>type 3</td>
<td>260.7</td>
<td>24.0</td>
<td>12.9</td>
</tr>
<tr>
<td>type 4</td>
<td>128.7</td>
<td>52.5</td>
<td>98.0</td>
</tr>
<tr>
<td>overall</td>
<td>178.4</td>
<td>89.5</td>
<td>250</td>
</tr>
</tbody>
</table>

4.3. Task C: Priority Ranking

The results of the simulations can be found in appendix A, Table A.3. The treatment time of patients at task C differs from task A and B. This is because of the priority implemented in the model at the ward. After \(369 \pm 11\) [min] all patients were treated, and left the emergency department. The main results of task C are shown in Table 4.3.

Table 4.3: Results of task C. The treatment time and the standard deviation of the treatment time of the patients are plotted against the different patient types and the combined results of the patients.

<table>
<thead>
<tr>
<th>Task C</th>
<th>treatment time [min]</th>
<th>standard deviation [min]</th>
<th>Count [-]</th>
</tr>
</thead>
<tbody>
<tr>
<td>type 1</td>
<td>162.6</td>
<td>44.4</td>
<td>88.6</td>
</tr>
<tr>
<td>type 2</td>
<td>165.0</td>
<td>33.8</td>
<td>50.5</td>
</tr>
<tr>
<td>type 3</td>
<td>170.6</td>
<td>16.7</td>
<td>13.4</td>
</tr>
<tr>
<td>type 4</td>
<td>156.7</td>
<td>46.1</td>
<td>97.6</td>
</tr>
<tr>
<td>overall</td>
<td>161.1</td>
<td>74.4</td>
<td>250</td>
</tr>
</tbody>
</table>
4.4. Comparison

To compare the different strategies with each other, the results of the simulation are combined in Figure 4.1. In the figure the average treatment time is plotted against the patient types, and the overall average treatment time of all the patients.

![Treatment Time Chart](image)

**Figure 4.1: Average treatment time plotted against patient type.**

When the normal control strategy of task A and the Doctors Exchange strategy of task B are compared, it shows that the average treatment time is longer with the Doctors Exchange strategy. The average waiting time is longer with every patient type, so none of the patients benefit from the Doctors Exchange strategy. The average treatment time is longer because the doctors have to change. When one of the doctors is still busy, the other doctor has to wait, so the utilization of the doctor will be lower. Another cause is the 20 % increase of the process time of the doctor, who changed from CW2 to CW1, gets by working with more complex cases.

If we compare the Priority Ranking Strategy from task C to the normal control strategy of task A, the difference is clearly visible in Figure 4.1. The average treatment time of patients’ type 1 and 3 are decreased because of the priority given in CW1 and CW2. The average treatment time of the patients with type 2 and 4 increases, because these patients have to wait longer at the ward. However the overall average treatment time decreases. As a result of the Priority Ranking Strategy the difference between the patient types is smaller than with the normal control Strategy.

The deviation of the patients’ treatment time can be shown by calculating the standard deviation of the treatment time. The result for every patient’s type is shown in Figure 4.2.
Figure 4.2: Standard deviation on the average treatment time plotted against patient type.

If the standard deviation of the Doctor Exchange Strategy is compared to the normal control strategy it shows that the Doctor Exchange Strategy has a larger spread of the treatment time of the patients. The reason for this difference is the longer average treatment time of the patients. At the beginning of working day the difference in treatment time between the two strategies is minimal. But because the overall treatment time increases with the Doctor Exchange Strategy, the treatment time of the patient at the end of the day must be longer, so the standard deviation of the treatment time of the patients increases.

The standard deviation of the treatment time of the Priority Ranking Strategy is smaller compared to the normal control strategy. The spread is smaller because of two reasons. The spread of patient type 1 and 3 is decreased because of the priority given by these patient’s types when they enter the ward for the second time, so the treatment time of the patient at the beginning of the day and the end of the day are less than with the normal control strategy. Also the difference between the different patient types is less so the overall spread of the patients will be less.
5. Validation

To validate the model, several methods are used. Model validation is used to check whether the simulation model is a proper abstraction of the reality. The first step was to run the model with only one patient with a pre-determined patient type. This is done with each patient type. This is done to check if the patient follows the right path through the treatment rooms and to check if the combined process time corresponds with the throughput time.

The next step is to check if the number of patients that come in the model is the same as the patients that leaving the model. So the patients are counted when they are generated and when the patients leave at the exit the patients get counted. These two numbers should be the same when all the patients have been treated.

It is possible that at a certain point of time multiple transitions can occur. CIF gives a list of the possible transitions. An option in CIF is to choose which transition should be performed first. CIF can choose to perform the first, the last or a random transition every time. If a model is deterministic, there should be no difference between these three options. Several simulations were made with the three options, with no differences in the results between the options.

The last verification method is to use visualization. Visualization gives a clear view on the system and lets you see what is going on in the system without interfering it. An example of visualization is visible at figure 5.1.
Figure 5.1: Example of visualization at time 33.6[min]. The four treatment rooms and the registration have the same build. Top left gives the treatment room name. Bottom left gives the number of patients in the waiting queue. The bars at the bottom right give information if the doctor is busy or idle. The remaining green bars give information about the remaining process time in ratio with the total process time.

The real advantage is when the visualization gets combined with the other verification methods. When an error occurs in the model which has been found by, for instance, simulation with first and last transitions, the error can be detected by visualization. The connecting code, between the model and the visualization, can be found in appendix E.
6. Verification

6.1. Assumptions

One of the main targets of this assignment is to compare the results, provided by CIF, with other programming languages and programming software. One of the main reasons for differences between the results must be due to the assumptions made. With the definition of the benchmark of ARGESIM, several other solutions with programming languages were given. Before the results of CIF can be compared with these results, the assumptions made in the other languages must be compared. The two main assumptions were whether the registration was manned by a doctor or by a person not defined as a doctor, for instance a receptionist. This assumption has influence on the starting time of the registration. The other main assumption is in which way the priority ranking was integrated in the model.

Table 6.1: Assumptions made by the different program languages.

<table>
<thead>
<tr>
<th>Programming software/language</th>
<th>Registration: Doctor or no doctor</th>
<th>Priority Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSIM</td>
<td>No doctor</td>
<td>2\textsuperscript{nd} entry Ward and X-ray</td>
</tr>
<tr>
<td>Matlab/DEVS</td>
<td>Unknown</td>
<td>2\textsuperscript{nd} entry Ward and X-ray</td>
</tr>
<tr>
<td>FLEXSIM</td>
<td>Unknown</td>
<td>2\textsuperscript{nd} entry Ward and X-ray</td>
</tr>
<tr>
<td>Anylogic</td>
<td>Unknown</td>
<td>2\textsuperscript{nd} entry Ward and X-ray</td>
</tr>
<tr>
<td>Taylor/ED</td>
<td>No doctor</td>
<td>2\textsuperscript{nd} entry Ward and X-ray</td>
</tr>
<tr>
<td>SLX</td>
<td>Doctor</td>
<td>Unknown</td>
</tr>
<tr>
<td>Arena</td>
<td>Doctor</td>
<td>Patient type 2</td>
</tr>
<tr>
<td>CIF</td>
<td>No Doctor</td>
<td>2\textsuperscript{nd} entry Ward</td>
</tr>
</tbody>
</table>

The results generated by Arena are not included in the comparison. The results were off scale with the rest of the program languages and with respect to the Priority Ranking Strategy another assumption was made.

As seen in Table 6.1 the assumption made about the doctor in the registration, is different for some of the program languages and software. So to see what the difference is, an extra set of simulations was done with a doctor the registration. The results of the simulation, with and without a doctor in the reception area are visible in Figure 6.1
If the two simulations are compared, the difference in the overall average treatment time is relatively small to the overall treatment time of the two simulations. Also the difference is smaller than the standard deviation of the treatment time which is around 8 [min]. Although the overall average treatment time stays the same, the average treatment time of the different patient types varies. Patient type 1 and 3 get a shorter average treatment time, while patient type 2 and 4 get a longer average treatment time. When the first patients of type 1 and 3 get to the ward for the second time, the queue is not as long as when there is no doctor in the registration, because the registration had less time to classify the patients and send them to the ward. Because of the same reason the patients of type 2 and 4 that come in late, have to wait longer because more patients of type 1 and 3 are getting treated in the ward for the second time when they come in, so the queue is longer for those patients.

Figure 6.1: Results of the simulation with and without a doctor in the registration.
6.2. Comparison

The different programming languages and software can be compared. The results of the three tasks are plotted. The results of task A are plotted in Figure 6.1. The results of the Doctors Exchange Strategy can be seen in Figure 6.2.

![Figure 6.2: Results of various programming languages and programming software for task A. Average treatment time is plotted against the patient types and all patients.](image)

The results of CIF are comparable with the results of other programming languages. Most of the results are within the boundaries of the standard deviation of the treatment time of the CIF results.
In the results of the Doctors Exchange Strategy, more variations than the normal control strategy are visible. The reliability of some of the results in this task is questionable. For instance the average treatment time of patients with type 3 is shorter than type 1, with FLEXSIM and Taylor/ED. This is a surprising result as the patients of type 3 have the same path as type 1, but have 2 treatment rooms more to go to.

The results of the Priority Ranking Strategy are plotted in figure 5.3.
Figure 6.4: Results of various programming languages and programming software for task C. Average treatment time are plotted against the patient types and all patients.

The overall average treatment time of the various software languages are comparable. But in the results of the patient types, the differences between the languages are a lot bigger. This effect can be caused by the combination of different assumptions. The consequences of the combination of assumptions were not possible to examine, as much of the information about the assumptions is not available.
7. Conclusion and Recommendation

An objective of this project was to compare the results of CIF 3 with the results of other simulators. The results simulated with CIF, were comparable to the result of other simulation languages. The results differ by the assumptions made in the model. One of the main assumptions made, was whether de registration was manned by a doctor or not. The assumption was made that there was no doctor working there. This assumption was made differently in some of the other simulators, so another simulation was made, with a doctor in the registration area, to see what the difference was. The overall average treatment time was didn’t change, but the average treatment time of the patients did. This was a consequence of the shorter queue for the wards at the beginning of the working day. The results of task A and task B were visually in the same range as the results of other simulators. The average treatment time of the different patient types matches most of the results of the other simulators. Because of the different assumptions made in priority, the results of task C differ a lot more than the results of tasks A and B. The overall average treatment time was in the same range, but the average treatment time of each type of patient, had a greater variation. This is a consequence of the combination of assumptions. Due to the lack of information about these assumptions, it was not possible to make any conclusions on the validation of the results of task C.

Another objective was so compare the different strategies and analyze the differences. The Doctor Exchange Strategy had a negative effect on the average treatment time of the patients, compared to the normal control strategy. The average treatment time was longer because of the lost working time of the doctors, when they needed to switch. Also the needed treatment time of the inexperienced doctor, that had to work in CW2 was increased. The Priority Ranking Strategy gave a better result compared to the normal control strategy. The overall average treatment time decreased. The average treatment time of patient type 1 and 3, becomes shorter because the patients do not have to wait as long for the second treatment in the ward, because they have priority. The patients of type 2 and 4 have a longer average treatment time, as result of the longer wait for the ward.

As a final addition, during the BFP I needed to learn to work with CIF. When the project started, I had no experience with CIF. So the project started with making some simple models with CIF. The first models had to be written in CIF3 but were simulated by means of an automatic translation for CIF2. During the project the new CIF3 simulator was developed. The new simulator had some great advantages, including the implementation of arrays. With the CIF2 simulator, a simulation with buffers of 15 patients took about 5 to 6 hours. With the new CIF3 simulator and the implemented arrays the simulation was done in 6 seconds. The function to send information over channels is not available yet, so an alternative solution was made to send the patient information to the next treatment room. To make all the necessary information accessible in the trajectory data, a few extra variables had to be created. This was necessary, because it is not yet possible to write arrays into the trajectory data. The language of CIF is easily understandable, with was useful as there was little documentation about CIF, so most off the information was extracted from a few samples.

A further recommendation is to implement a global control strategy to control the distribution of the patients over the queues of the ward. This strategy keeps the estimated remaining process time of the patients in the waiting queue in mind, so that both wards are about ready at the same time. Also the combinations of assumptions that are made in the other models can be researched.
Bibliography


## Appendix A

Table A.1: Results of simulations of task A.

<table>
<thead>
<tr>
<th></th>
<th>average treatment time</th>
<th>Number of patients</th>
<th>finaltime</th>
<th>standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>type 1</td>
<td>type 2</td>
<td>type 3</td>
<td>type 4</td>
</tr>
<tr>
<td>241.9</td>
<td>132.7</td>
<td>233.1</td>
<td>131.3</td>
<td>180.7</td>
</tr>
<tr>
<td>241.8</td>
<td>133.3</td>
<td>260.2</td>
<td>122.7</td>
<td>177.3</td>
</tr>
<tr>
<td>250.1</td>
<td>150.2</td>
<td>246.2</td>
<td>135.2</td>
<td>189.8</td>
</tr>
<tr>
<td>234.7</td>
<td>140.3</td>
<td>223.0</td>
<td>134.0</td>
<td>175.1</td>
</tr>
<tr>
<td>230.7</td>
<td>148.9</td>
<td>239.0</td>
<td>126.4</td>
<td>173.0</td>
</tr>
<tr>
<td>219.7</td>
<td>137.4</td>
<td>234.0</td>
<td>119.7</td>
<td>164.8</td>
</tr>
<tr>
<td>241.1</td>
<td>144.9</td>
<td>254.2</td>
<td>126.5</td>
<td>175.7</td>
</tr>
<tr>
<td>237.6</td>
<td>137.0</td>
<td>261.8</td>
<td>132.5</td>
<td>178.3</td>
</tr>
<tr>
<td>220.3</td>
<td>143.1</td>
<td>262.5</td>
<td>118.6</td>
<td>163.9</td>
</tr>
<tr>
<td>215.6</td>
<td>164.2</td>
<td>275.6</td>
<td>126.3</td>
<td>168.5</td>
</tr>
<tr>
<td>247.9</td>
<td>124.9</td>
<td>287.5</td>
<td>140.2</td>
<td>190.8</td>
</tr>
<tr>
<td>244.5</td>
<td>138.0</td>
<td>232.6</td>
<td>130.0</td>
<td>175.8</td>
</tr>
<tr>
<td>240.1</td>
<td>147.1</td>
<td>248.2</td>
<td>129.3</td>
<td>179.0</td>
</tr>
<tr>
<td>237.7</td>
<td>127.0</td>
<td>256.5</td>
<td>114.6</td>
<td>172.9</td>
</tr>
<tr>
<td>249.8</td>
<td>145.1</td>
<td>285.3</td>
<td>142.2</td>
<td>189.7</td>
</tr>
<tr>
<td>243.7</td>
<td>124.9</td>
<td>248.0</td>
<td>123.6</td>
<td>168.2</td>
</tr>
<tr>
<td>219.1</td>
<td>136.1</td>
<td>252.3</td>
<td>116.0</td>
<td>159.2</td>
</tr>
<tr>
<td>240.6</td>
<td>142.4</td>
<td>253.2</td>
<td>130.4</td>
<td>183.5</td>
</tr>
<tr>
<td>228.4</td>
<td>152.5</td>
<td>247.1</td>
<td>134.7</td>
<td>172.8</td>
</tr>
<tr>
<td>239.5</td>
<td>140.1</td>
<td>248.8</td>
<td>124.5</td>
<td>171.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>average treatment time</th>
<th>Number of patients</th>
<th>finaltime</th>
<th>standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>type 1</td>
<td>type 2</td>
<td>type 3</td>
<td>type 4</td>
</tr>
<tr>
<td>236.3</td>
<td>140.5</td>
<td>252.5</td>
<td>127.9</td>
<td>175.5</td>
</tr>
<tr>
<td>10.6</td>
<td>9.7</td>
<td>16.8</td>
<td>7.5</td>
<td>8.6</td>
</tr>
</tbody>
</table>

25
Table A.2: Results of the simulations of task B, the Doctor Exchange Strategy

<table>
<thead>
<tr>
<th>average treatment time</th>
<th>Number of patients</th>
<th>standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>type 1</td>
<td>type 2</td>
<td>type 3</td>
</tr>
<tr>
<td>235.5</td>
<td>143.8</td>
<td>244.3</td>
</tr>
<tr>
<td>237.4</td>
<td>118.6</td>
<td>259.1</td>
</tr>
<tr>
<td>246.0</td>
<td>134.6</td>
<td>274.2</td>
</tr>
<tr>
<td>240.5</td>
<td>143.5</td>
<td>300.4</td>
</tr>
<tr>
<td>231.5</td>
<td>137.2</td>
<td>268.6</td>
</tr>
<tr>
<td>233.8</td>
<td>149.0</td>
<td>239.4</td>
</tr>
<tr>
<td>236.1</td>
<td>151.4</td>
<td>292.8</td>
</tr>
<tr>
<td>238.7</td>
<td>119.4</td>
<td>268.8</td>
</tr>
<tr>
<td>237.4</td>
<td>134.8</td>
<td>271.3</td>
</tr>
<tr>
<td>249.7</td>
<td>148.7</td>
<td>259.9</td>
</tr>
<tr>
<td>235.7</td>
<td>148.3</td>
<td>244.1</td>
</tr>
<tr>
<td>252.7</td>
<td>132.6</td>
<td>253.2</td>
</tr>
<tr>
<td>237.6</td>
<td>128.7</td>
<td>269.4</td>
</tr>
<tr>
<td>258.8</td>
<td>146.3</td>
<td>235.1</td>
</tr>
<tr>
<td>240.8</td>
<td>144.7</td>
<td>264.3</td>
</tr>
<tr>
<td>259.0</td>
<td>139.0</td>
<td>277.7</td>
</tr>
<tr>
<td>234.2</td>
<td>148.8</td>
<td>261.7</td>
</tr>
<tr>
<td>258.9</td>
<td>154.7</td>
<td>251.0</td>
</tr>
<tr>
<td>248.8</td>
<td>121.6</td>
<td>252.9</td>
</tr>
<tr>
<td>233.6</td>
<td>157.5</td>
<td>227.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>average treatment time</th>
<th>Number of patients</th>
<th>standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>type 1</td>
<td>type 2</td>
<td>type 3</td>
</tr>
<tr>
<td>242.3</td>
<td>140.2</td>
<td>260.7</td>
</tr>
<tr>
<td>9.2</td>
<td>11.5</td>
<td>18.3</td>
</tr>
</tbody>
</table>
Table A.3: Results of the simulations of task C, the priority Ranking Strategy.

<table>
<thead>
<tr>
<th>average treatment time</th>
<th>Number of patients</th>
<th>standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>type 1</td>
<td>type 1</td>
</tr>
<tr>
<td>type 1</td>
<td>type 2</td>
<td>type 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>159.</td>
<td>1</td>
<td>165.</td>
</tr>
<tr>
<td>168.</td>
<td>1</td>
<td>156.</td>
</tr>
<tr>
<td>160.</td>
<td>0</td>
<td>176.</td>
</tr>
<tr>
<td>158.</td>
<td>1</td>
<td>143.</td>
</tr>
<tr>
<td>155.</td>
<td>5</td>
<td>159.</td>
</tr>
<tr>
<td>154.</td>
<td>0</td>
<td>139.</td>
</tr>
<tr>
<td>157.</td>
<td>8</td>
<td>157.</td>
</tr>
<tr>
<td>163.</td>
<td>0</td>
<td>171.</td>
</tr>
<tr>
<td>164.</td>
<td>0</td>
<td>177.</td>
</tr>
<tr>
<td>174.</td>
<td>2</td>
<td>178.</td>
</tr>
<tr>
<td>158.</td>
<td>6</td>
<td>183.</td>
</tr>
<tr>
<td>164.</td>
<td>1</td>
<td>163.</td>
</tr>
<tr>
<td>167.</td>
<td>6</td>
<td>168.</td>
</tr>
<tr>
<td>158.</td>
<td>1</td>
<td>171.</td>
</tr>
<tr>
<td>163.</td>
<td>7</td>
<td>171.</td>
</tr>
<tr>
<td>167.</td>
<td>8</td>
<td>150.</td>
</tr>
<tr>
<td>165.</td>
<td>9</td>
<td>178.</td>
</tr>
<tr>
<td>163.</td>
<td>7</td>
<td>162.</td>
</tr>
<tr>
<td>158.</td>
<td>5</td>
<td>155.</td>
</tr>
</tbody>
</table>

Table A.4: Results of the simulations of task C, the priority Ranking Strategy.

<table>
<thead>
<tr>
<th>average treatment time</th>
<th>Number of patients</th>
<th>standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>type 1</td>
<td>type 1</td>
</tr>
<tr>
<td>type 1</td>
<td>type 2</td>
<td>type 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>162.</td>
<td>6</td>
<td>165.</td>
</tr>
<tr>
<td>5.3</td>
<td>12.1</td>
<td>20.4</td>
</tr>
</tbody>
</table>
Appendix B

// patient: a list of elements. Contains element id: # of times in X-ray +1, typ: type of patient (1,2,3,4). tb: time patient was generated

```
type patient = tuple(int id; int typ; real tb);
```

// sup is a substitution. sup is created to make sure that the algebraic variable p1, p2, ... always exits, even when array x is empty.

```
alg patient sup1 = if bufferreg.bufsize=0 : (0,0,0.0)
  else bufferreg.x[0]
end;

alg patient sup2 = if bufferward1.bufsize=0 : (0,0,0.0)
  else bufferward1.x[0]
end;

alg patient sup3 = if bufferward2.bufsize=0 : (0,0,0.0)
  else bufferward2.x[0]
end;

alg patient sup4 = if bufferxray.bufsize=0 : (0,0,0.0)
  else bufferxray.x[0]
end;

alg patient sup5 = if bufferplaster.bufsize=0 : (0,0,0.0)
  else bufferplaster.x[0]
end;
```

generator: Generator (0.3);
bufferreg: BufferReg (generator.send, generator.pat);
registration: Registration (0.2, 0.5, 1.0, bufferreg.send, sup1);
bufferward1: Buffer_3i_2o (registration.send1, xray1.send1, xray2.send1, registration.pat, xray1.pat, xray2.pat);
ward1d1: Doctor (1.5, 3.2, 5.0, bufferward1.send1, sup2);
ward1d2: Doctor (1.5, 3.2, 5.0, bufferward1.send2, sup2);
bufferward2: Buffer_3i_2o (registration.send2, xray1.send2, xray2.send2, registration.pat, xray1.pat, xray2.pat);
ward2d3: Doctor (2.8, 4.1, 6.3, bufferward2.send1, sup3);
ward2d4: Doctor (2.8, 4.1, 6.3, bufferward2.send2, sup3);
bufferplaster: Buffer_6i_1o (ward1d1.send2, ward1d2.send2, ward2d3.send2, ward2d4.send2, xray1.send3, xray2.send3, ward1d1.pat, ward1d2.pat, ward2d3.pat, ward2d4.pat, xray1.pat, xray2.pat);
plaster: Plaster (3.0, 3.8, 4.7, bufferplaster.send, sup5);
bufferxray: Buffer_5i_2o (plaster.send2, ward1d1.send1, ward1d2.send1, ward2d3.send1, ward2d4.send1, plaster.pat, ward1d1.pat, ward1d2.pat, ward2d3.pat, ward2d4.pat);
xray1: Xray (2.0, 2.8, 4.1, bufferxray.send1, sup4);
xray2: Xray (2.0, 2.8, 4.1, bufferxray.send2, sup4);
exit: Exit (ward1d1.send3, ward1d2.send3, ward2d3.send3, ward2d4.send3, plaster.send1, ward1d1.pat, ward1d2.pat, ward2d3.pat, ward2d4.pat, plaster.pat);
// Generator generates the incoming patients with exponential distribution with mean 0.3 until 250 patient have been generated

def Generator(alg real mean):
    event send;
    disc patient pat = (1, 0, 0.0);
    cont t = 0.0 der 1.0;
    cont tb = 0.0 der 1.0;
    disc int count=0, ch_gen_buf=0;
    disc dist real expo=exponential(mean);
    disc real generation_time = 0.0;
    cont gentime= 0.0;

    location sending:
        initial;
        equation gentime'=0.0;
        event send! do t := 0.0, count:=count+1, generation_time:= (sample expo)[0], expo := (sample expo)[1], gentime:= (sample expo)[0], ch_gen_buf:=0 goto wait;

    location wait:
        equation gentime'=-1.0;
        when t >= generation_time and count < 250 do pat:=(1,0,tb), ch_gen_buf:=1 goto sending;

end

// Buffer with 6 incoming channels and 1 outgoing channel. Buffer contains array x with patients. Patients send with FIFO

def Buffer_6i_1o(event receive1, receive2, receive3, receive4, receive5, receive6; alg patient p1, p2, p3, p4, p5, p6):
    event send;
    disc list patient x;
    disc int bufsize=0;

    location buffer:
        initial;

        event receive1? do x:=x+[p1], bufsize:=bufsize+1;
        event receive2? do x:=x+[p2], bufsize:=bufsize+1;
        event receive3? do x:=x+[p3], bufsize:=bufsize+1;
        event receive4? do x:=x+[p4], bufsize:=bufsize+1;
        event receive5? do x:=x+[p5], bufsize:=bufsize+1;
        event receive6? do x:=x+[p6], bufsize:=bufsize+1;
        event send! when bufsize>0 do x:=x[1:], bufsize:=bufsize-1;

end

// Buffer with 5 incoming channels and 2 outgoing channel. Buffer contains array x with patients. Patients send with FIFO.
// Patient send with parallel composition.

def Buffer_5i_2o(event receive1, receive2, receive3, receive4, receive5; alg patient p1, p2, p3, p4, p5):
    event send1, send2;
    disc list patient x;
    disc int bufsize=0;
location buffer:
    initial;
    
    event receive1? do x:=x+[p1], bufsize:=bufsize+1;
    event receive2? do x:=x+[p2], bufsize:=bufsize+1;
    event receive3? do x:=x+[p3], bufsize:=bufsize+1;
    event receive4? do x:=x+[p4], bufsize:=bufsize+1;
    event receive5? do x:=x+[p5], bufsize:=bufsize+1;
    event send1!, send2! when bufsize>0 do x:=x[1:], bufsize:=bufsize-1;

end

// Buffer with 3 incoming channels and 2 outgoing channel. Buffer contains array x with patients. Patients send with FIFO.
// Patient send with parallel composition.
def Buffer_3i_2o(event receive1, receive2, receive3; alg patient p1, p2, p3):
    event send1, send2;
    disc list patient x;
    disc int bufsize=0;

location buffer:
    initial;
    
    event receive1? do x:=x+[p1], bufsize:=bufsize+1;
    event receive2? do x:=x+[p2], bufsize:=bufsize+1;
    event receive3? do x:=x+[p3], bufsize:=bufsize+1;
    event send1!, send2! when bufsize>0 do x:=x[1:], bufsize:=bufsize-1;

end

// Buffer with 1 incoming channels and 1 outgoing channel. Buffer contains array x with patients. Patients send with FIFO.
def BufferReg(event receive; alg patient p1):
    event send;
    disc list patient x;
    disc int bufsize=0;

location buffer:
    initial;
    
    event receive? do x:=x+[p1], bufsize:=bufsize+1;
    event send! when bufsize>0 do x:=x[1:], bufsize:=bufsize-1;

end

// registration gives the patient a patient type and sends the patient to one of the wards.
// typedef and cwchoice are uniform distribution, process_time and pel take a integer in the range of the distribution
// in location processing the typ of the patient is defined, depending on the value pel taken from the uniform distribution

31
def Registration (alg real low, mean, high; event receive; alg patient p1):
    event send1, send2;
    disc patient pat = (0,0,0.0);
    disc dist int typedef = uniform(1, 100);
    disc dist int cwchoice= uniform(1, 100);
    disc dist real tria=triangle(low,mean,high);
    cont t = 0.0 der 1.0;
    disc real tb=0.0, process_time=0.0;
    disc int typ=0, pel=0, id=0,cw=0;
    cont protime=0.0;

location receiving:
    initial;
    equation protime'=0.0;
    event receive? do id:=p1[id], tb:=p1[tb], pat:=p1, t := 0.0, process_time:= (sample tria)[0], tria :=
                        (sample tria)[1], pel:= (sample typedef)[0], typedef := (sample typedef)[1],cw:= (sample
                        cwchoice)[0], cwchoice := (sample cwchoice)[1], protime:=(sample tria)[0]
                        goto    processing;

location processing:
    equation protime'=-1.0;
    when pel<=35       do typ:=1 goto defining;
    when pel>35 and pel<=55 do typ:=2 goto defining;
    when pel>55 and pel<=60 do typ:=3 goto defining;
    when pel>60               do typ:=4 goto defining;

location defining:
    equation protime'=-1.0;
    do pat:=(id,typ, tb) goto sending;

location sending:
    equation protime'=-1.0;
    event send1! when t >= process_time, cw>  40 do pel:=0, protime:=0.0 goto receiving;
    event send2! when t >= process_time, cw<=40 do pel:=0, protime:=0.0 goto receiving;
end

// Doctor treats patient and send them to the next treatment depending on patient type
// process_time takes a real number out of the triangular distribution tria

def Doctor (alg real low, mean, high; event receive; alg patient p1):
    event send1, send2, send3;
    disc patient pat = (0,0,0.0);
    disc dist real tria=triangle(low,mean,high);
    cont t = 0.0 der 1.0;
    disc real tb=0.0, process_time=0.0;
    disc int typ=0, id=0, busy=0;
    cont protime=0.0;

// delay until doctors start to work. 8.00 a.m.
location arriving:
    initial;
    equation protime'=0.0;
when t>30 goto receiving;

location receiving:
equation protime'=0.0;
event receive? do pat:=p1, t := 0.0, id:=p1[id], tb:=p1[tb], typ:=p1[typ], process_time:= (sample tria)[0], tria := (sample tria)[1], busy:=1, protime:= (sample tria)[0] goto processing;

// decision to which treatment point the patient is send
// decision based on patient type and id (number of times through X-ray)
location processing:
equation protime'=-1.0;
event send1! when t >= process_time, typ=1, id=1 do busy:=0, protime:=0.0 goto receiving;
event send3! when t >= process_time, typ=1, id=2 do busy:=0, protime:=0.0 goto receiving;
event send2! when t >= process_time, typ=2 do busy:=0, protime:=0.0 goto receiving;
event send1! when t >= process_time, typ=3, id=1 do busy:=0, protime:=0.0 goto receiving;
event send3! when t >= process_time, typ=3, id=2 do busy:=0, protime:=0.0 goto receiving;
event send3! when t >= process_time, typ=4 do busy:=0, protime:=0.0 goto receiving;

end

def Xray(alg real low, mean, high; event receive; alg patient p1):
event send1, send2, send3;
disc patient pat = (0,0,0,0);
disc dist real tria=triangle(low,mean,high);
disc dist int cwchoice= uniform(1, 100);
cont t = 0.0 der 1.0;
disc real process_time=0.0;
disc int typ=0, id=0, cw=0, busy=0;
cont protime=0.0;

// when patient is received, id is changed.

location receiving:
initial;
equation protime'=0.0;
event receive? do pat:=(p1[id]+1,p1[typ],p1[tb]), typ:=p1[typ], id:=p1[id]+1, t := 0.0, process_time:= (sample tria)[0], tria := (sample tria)[1], cw:= (sample cwchoice)[0], cwchoice := (sample cwchoice)[1], busy:=1, protime:= (sample tria)[0] goto defining;

// decision to which treatment room the patient is send
// decision based on patient type and id (number of times through X-ray)
location defining:
equation protime'=-1.0;
when typ=1 goto sending1;
when typ=3, id=2 goto sending2;
when typ=3, id=3 goto sending1;

// decision to which ward patient is send, is based on percentile distribution
location sending1:
equation protime'=-1.0;
event send1! when cw<=40, t >= process_time do busy:=0, protime:=0.0 goto receiving;
event send2! when cw>  40, t >= process_time do busy:=0, protime:=0.0 goto receiving;

location sending2:
equation protime'=-1.0;
event send3! when t >= process_time do busy:=0, protime:=0.0 goto receiving;
end

def Plaster(alg real low, mean, high; event receive; alg patient p1):
event send1, send2;
disc patient pat = (0,0,0.0);
disc dist real tria=triangle(low,mean,high);
cont t = 0.0 der 1.0;
disc real process_time=0.0;
disc int typ=0, busy=0;
cont protime=0.0;

location receiving:
initial;
equation protime'=0.0;
event receive? do pat:=p1, typ:=p1[typ], t := 0.0, process_time:=(sample tria)[0], tria := (sample tria)[1], busy:=1, protime:=(sample tria)[0] goto processing;

// decision to which treatment room the patient is send
// decision based on patient type
location processing:
equation protime'=-1.0;
event send1! when t >= process_time, typ=2 do busy:=0, protime:=0.0 goto receiving;
event send2! when t >= process_time, typ=3 do busy:=0, protime:=0.0 goto receiving;
end

// in Exit all the calculations of average treatment time and standard deviations
def Exit(event receive1, receive2, receive3, receive4, receive5; alg patient p1, p2, p3, p4, p5):
disc patient pat=(0,0,0.0);
disc list real through;
disc list int typlist;
disc int  count=0, count1=0, count2=0, count3=0, count4=0, typ=0, nr=0;
disc real tb=0.0, sum=0.0, sum1=0.0, sum2=0.0, sum3=0.0, sum4=0.0, avgtime=0.0, avg1time=0.0, avg2time=0.0, avg3time=0.0, avg4time=0.0, throughtime=0.0, throughsum1=0.0, throughsum2=0.0, throughsum3=0.0, throughsum4=0.0, finaltime=0.0, throughsum=0.0, stddev=0.0, stddev1=0.0, stddev2=0.0, stddev3=0.0, stddev4=0.0;
cont t=0.0 der 1.0;

location exit:
initial;
event receive1? do count := count + 1, pat:=p1, tb:=p1[tb], typ:=p1[typ], typlist:=typlist+[p1[typ]]. finaltime:=t goto summation;
when count>=250 goto avg1;

location summation:
  when typ=1 do sum:=sum+(t-tb), sum1:=sum1+(t-tb), count1:=count1+1, throughtime:=t-tb goto calc1;
  when typ=2 do sum:=sum+(t-tb), sum2:=sum2+(t-tb), count2:=count2+1, throughtime:=t-tb goto calc2;
  when typ=3 do sum:=sum+(t-tb), sum3:=sum3+(t-tb), count3:=count3+1, throughtime:=t-tb goto calc3;
  when typ=4 do sum:=sum+(t-tb), sum4:=sum4+(t-tb), count4:=count4+1, throughtime:=t-tb goto calc4;

location calc1:
  do avgtime:=sum/count, avg1time:=sum1/count1, through:=through+throughtime goto exit;

location calc2:
  do avgtime:=sum/count, avg2time:=sum2/count2, through:=through+throughtime goto exit;

location calc3:
  do avgtime:=sum/count, avg3time:=sum3/count3, through:=through+throughtime goto exit;

location calc4:
  do avgtime:=sum/count, avg4time:=sum4/count4, through:=through+throughtime goto exit;

location avg1:
  when nr< 250, typlist[nr]=1 do throughsum:=throughsum+(through[nr]-avgtime)*(through[nr]-avgtime),
  throughsum1:=throughsum1+(through[nr]-avg1time)*(through[nr]-avg1time) goto avg2;
  when nr< 250, typlist[nr]=2 do throughsum:=throughsum+(through[nr]-avgtime)*(through[nr]-avgtime),
  throughsum2:=throughsum2+(through[nr]-avgtime)*(through[nr]-avgtime) goto avg2;
  when nr< 250, typlist[nr]=3 do throughsum:=throughsum+(through[nr]-avgtime)*(through[nr]-avgtime),
  throughsum3:=throughsum3+(through[nr]-avgtime)*(through[nr]-avgtime) goto avg2;
  when nr< 250, typlist[nr]=4 do throughsum:=throughsum+(through[nr]-avgtime)*(through[nr]-avgtime),
  throughsum4:=throughsum4+(through[nr]-avgtime)*(through[nr]-avgtime) goto avg2;
  when nr>=250 goto ending;

location avg2:
  when typlist[nr]=1 do stdev:=sqrt((1/(count-1))*throughsum), stdev1:=sqrt((1/(count-1))*throughsum1),
  nr:=nr+1 goto avg1;
  when typlist[nr]=2 do stdev:=sqrt((1/(count-1))*throughsum), stdev2:=sqrt((1/(count-1))*throughsum2),
  nr:=nr+1 goto avg1;
  when typlist[nr]=3 do stdev:=sqrt((1/(count-1))*throughsum), stdev3:=sqrt((1/(count-1))*throughsum3),
  nr:=nr+1 goto avg1;
  when typlist[nr]=4 do stdev:=sqrt((1/(count-1))*throughsum), stdev4:=sqrt((1/(count-1))*throughsum4),
  nr:=nr+1 goto avg1;

location ending:
  event receive1?;

end
Appendix C

generator: Generator (0.3);
bufferreg: BufferReg (generator.send, generator.pat);
registration: Registration (0.2, 0.5, 1.0, bufferreg.send, sup1);
bufferward1: Buffer_3i_2o (registration.send1, xray1.send1, xray2.send1,
registration.pat, xray1.pat, xray2.pat);
ward1d1: Doctor (1.5, 3.2, 5.0, bufferward1.send1, sup2);
ward1d2: Doctor_D2 (1.5, 3.2, 5.0, 2.8, 4.1, 6.3, bufferward1.send2, sup2);
bufferward2: Buffer_3i_2o (registration.send2, xray1.send2, xray2.send2,
registration.pat, xray1.pat, xray2.pat);
ward2d3: Doctor_D3 (1.5, 3.2, 5.0, 2.8, 4.1, 6.3, bufferward2.send1, sup3);
ward2d4: Doctor (2.8, 4.1, 6.3, bufferward2.send2, sup3);
bufferplaster: Buffer_6i_1o (ward1d1.send2, ward1d2.send2, ward2d3.send2, ward2d4.send2,
xray1.send3, xray2.send3, ward1d1.pat, ward1d2.pat, ward2d3.pat,
ward2d4.pat, xray1.pat, xray2.pat);
plaster: Plaster (3.0, 3.8, 4.7, bufferplaster.send, sup5);
bufferxray: Buffer_5i_2o (plaster.send2, ward1d1.send1, ward1d2.send1, ward2d3.send1,
ward2d4.send1, plaster.pat, ward1d1.pat, ward1d2.pat, ward2d3.pat,
ward2d4.pat);
xray1: Xray (2.0, 2.8, 4.1, bufferxray.send1, sup4);
xray2: Xray (2.0, 2.8, 4.1, bufferxray.send2, sup4);
exit: Exit (ward1d1.send3, ward1d2.send3, ward2d3.send3, ward2d4.send3,
plaster.send1, ward1d1.pat, ward1d2.pat, ward2d3.pat, ward2d4.pat,
plaster.pat);

def Doctor_D3(alg real low_d2, mean_d2, high_d2, low_d3, mean_d3, high_d3; event receive; alg patient p1):
  event send1, send2, send3;
  disc patient pat = (0,0,0.0);
  disc dist real tria_d2=triangle(low_d2,mean_d2,high_d2), tria_d3=triangle(low_d3,mean_d3,high_d3);
  cont t = 0.0 der 1.0;
  disc real tb=0.0, process_time_d3=0.0, process_time_d2=0.0;
  disc int typ=0, id=0, busy_d2=0, busy_d3=0, ts=0, ch_ward_xray=0, ch_ward_plaster=0,
               ch_ward_exit=0;
  cont protime=0.0;

  location arriving:
    equation protime'=0.0;
    initial;
    when t>30 goto receiving_d3;

  location check_bufsize_d3:
    equation protime'=0.0;
    when bufferward2.bufsize> 20, ward1d2.busy_d2=0 do busy_d2:=0 goto receiving_d2;
    when bufferward2.bufsize<=20 do busy_d3:=0 goto receiving_d3;

  location check_bufsize_d2:
    equation protime'=0.0;
    when bufferward2.bufsize> 5 do busy_d2:=0 goto receiving_d2;
    when bufferward2.bufsize<=5, ward1d2.busy_d3=0 do busy_d3:=0 goto receiving_d3;

  location receiving_d3:
equation protime' = 0.0;
event receive? do pat:=p1, busy_d3:=1, t := 0.0, id:=p1[id], tb:=p1[tb], typ:=p1[typ],
  process_time_d3:= (sample tria_d3)[0], tria_d3 := (sample tria_d3)[1],
  protime:= (sample tria_d3)[0] goto channel_sending_d3;
  when t>0.2, ts=0 do pat:=p1, busy_d3:=1, t := 0.0, id:=p1[id], tb:=p1[tb], typ:=p1[typ],
  process_time_d3:= (sample tria_d3)[0], tria_d3 := (sample tria_d3)[1],
  protime:= (sample tria_d3)[0] goto channel_sending_d3;

location receiving_d2:
equation protime' = 0.0;
event receive? do pat:=p1, busy_d2:=1, t := 0.0, id:=p1[id], tb:=p1[tb], typ:=p1[typ],
  process_time_d2:= (sample tria_d2)[0], tria_d2 := (sample tria_d2)[1],
  protime:= (sample tria_d2)[0] goto channel_sending_d2;
  when t>0.2, ts=0 do ch_ward_xray:=0, ch_ward_plaster:=0, ch_ward_exit:=0, ts:=1;

location channel_sending_d2:
equation protime' = -1.0;
when t>=0.2 do ch_ward_xray:=0, ch_ward_plaster:=0, ch_ward_exit:=0 goto processing_d2;

location channel_sending_d3:
equation protime' = -1.0;
when t>=0.2 do ch_ward_xray:=0, ch_ward_plaster:=0, ch_ward_exit:=0 goto processing_d3;

location processing_d3:
equation protime' = -1.0;
event send1! when t >= process_time_d3, typ=1, id=1 do busy_d3:=0, protime:=0.0,
  process_time_d3:=0.0, t:=0.0, ts:=0, ch_ward_xray:=1 goto check bufsize_d3;
  event send3! when t >= process_time_d3, typ=1, id=2 do busy_d3:=0, protime:=0.0,
  process_time_d3:=0.0, t:=0.0, ts:=0, ch_ward_exit:=1 goto check bufsize_d3;
  event send2! when t >= process_time_d3, typ=2 do busy_d3:=0, protime:=0.0,
  process_time_d3:=0.0, t:=0.0, ts:=0, ch_ward_plaster:=1 goto check bufsize_d3;
  event send1! when t >= process_time_d3, typ=3, id=1 do busy_d3:=0, protime:=0.0,
  process_time_d3:=0.0, t:=0.0, ts:=0, ch_ward_xray:=1 goto check bufsize_d3;
  event send3! when t >= process_time_d3, typ=3, id=3 do busy_d3:=0, protime:=0.0,
  process_time_d3:=0.0, t:=0.0, ts:=0, ch_ward_exit:=1 goto check bufsize_d3;
  event send3! when t >= process_time_d3, typ=4 do busy_d3:=0, protime:=0.0,
  process_time_d3:=0.0, t:=0.0, ts:=0, ch_ward_exit:=1 goto check bufsize_d3;

location processing_d2:
equation protime' = -1.0;
event send1! when t >= process_time_d2, typ=1, id=1 do busy_d2:=0, protime:=0.0,
  process_time_d2:=0.0, t:=0.0, ts:=0, ch_ward_xray:=1 goto check bufsize_d2;
  event send3! when t >= process_time_d2, typ=1, id=2 do busy_d2:=0,
event send2! when t >= process_time_d2, typ=2 do busy_d2:=0, protime:=0.0, process_time_d2:=0.0, t:=0.0, ts:=0, ch_ward_exit:=1
    goto check_bufsize_d2;

event send1! when t >= process_time_d2, typ=3, id=1 do busy_d2:=0, protime:=0.0, process_time_d2:=0.0, t:=0.0, ts:=0, ch_ward_xray:=1
    goto check_bufsize_d2;

event send3! when t >= process_time_d2, typ=3, id=3 do busy_d2:=0, protime:=0.0, process_time_d2:=0.0, t:=0.0, ts:=0, ch_ward_exit:=1
    goto check_bufsize_d2;

end

def Doctor_D2(alg real low_d2, mean_d2, high_d2, low_d3, mean_d3, high_d3; event receive; alg patient p1):
    event send1, send2, send3;
    disc patient pat = (0,0,0);
    disc dist real tria_d2=triangle(low_d2,mean_d2,high_d2), tria_d3=triangle(low_d3,mean_d3,high_d3);
    cont t = 0.0 der 1.0;
    disc real tb=0.0, process_time_d3=0.0, process_time_d2=0.0 ;
    disc int typ=0, id=0, busy_d2=0, busy_d3=0, ts=0, ch_ward_xray=0, ch_ward_plaster=0, ch_ward_exit=0;
    cont protime=0.0;

location arriving:
    initial;
    equation protime'=0.0;
    when t>30 goto check_bufsize_d2;

location check_bufsize_d2:
    equation protime'=0.0;
    when bufferward2.bufsize> 20, ward2d3.busy_d3=0 do busy_d2:=0 goto receiving_d3;
    when bufferward2.bufsize<=20 do busy_d2:=0 goto receiving_d2;

location check_bufsize_d3:
    equation protime'=0.0;
    when bufferward2.bufsize> 5 do busy_d3:=0 goto receiving_d3;
    when bufferward2.bufsize<=5, ward2d3.busy_d2=0 do busy_d2:=0 goto receiving_d2;

location receiving_d3:
    equation protime'=0.0;
    event receive? do pat:=p1, busy_d3:=1, t := 0.0, id:=p1[id], tb:=p1[tb], typ:=p1[typ],
                    process_time_d3:=1.2* (sample tria_d3)[0], tria_d3 := (sample tria_d3)[1],
                    protime:= 1.2* (sample tria_d3)[0] goto channel_sending_d3;
    when t>0.2, ts=0 do ch_ward_xray:=0, ch_ward_plaster:=0, ch_ward_exit:=0, ts:=1;
location receiving_d2:
  equation protime'=0.0;
  event receive? do pat:=p1, busy_d2:=1, t := 0.0, id:=p1[id], tb:=p1[tb], typ:=p1[typ],
  process_time_d2:= (sample tria_d2)[0], tria_d2 := (sample tria_d2)[1],
  protime:= (sample tria_d2)[0] goto channel_sending_d2;
  when t>0.2, ts=0 do ch_ward_xray:=0, ch_ward_plaster:=0, ch_ward_exit:=0, ts:=1;

location channel_sending_d2:
  equation protime'=-1.0;
  when t>=0.2 do ch_ward_xray:=0, ch_ward_plaster:=0, ch_ward_exit:=0 goto processing_d2;

location channel_sending_d3:
  equation protime'=-1.0;
  when t>=0.2 do ch_ward_xray:=0, ch_ward_plaster:=0, ch_ward_exit:=0 goto processing_d3;

location processing_d3:
  equation protime'=-1.0;
  event send1!  when t >= process_time_d3, typ=1, id=1 do busy_d3:=0, protime:=0.0,
  process_time_d2:=0.0, process_time_d3:=0.0, t:=0.0, ts:=0, ch_ward_xray:=1 goto check_bufsize_d3;
  event send3!  when t >= process_time_d3, typ=1, id=2 do busy_d3:=0, protime:=0.0,
  process_time_d2:=0.0, process_time_d3:=0.0, t:=0.0, ts:=0, ch_ward_exit:=1 goto check_bufsize_d3;
  event send2!  when t >= process_time_d3, typ=2 do busy_d3:=0, protime:=0.0,
  process_time_d2:=0.0, process_time_d3:=0.0, t:=0.0, ts:=0, ch_ward_plaster:=1 goto check_bufsize_d3;
  event send1!  when t >= process_time_d3, typ=3, id=1 do busy_d3:=0, protime:=0.0,
  process_time_d2:=0.0, process_time_d3:=0.0, t:=0.0, ts:=0, ch_ward_xray:=1 goto check_bufsize_d3;
  event send3!  when t >= process_time_d3, typ=3, id=3 do busy_d3:=0, protime:=0.0,
  process_time_d2:=0.0, process_time_d3:=0.0, t:=0.0, ts:=0, ch_ward_exit:=1 goto check_bufsize_d3;
  event send3!  when t >= process_time_d3, typ=4 do busy_d3:=0, protime:=0.0,
  process_time_d2:=0.0, process_time_d3:=0.0, t:=0.0, ts:=0, ch_ward_exit:=1 goto check_bufsize_d3;

location processing_d2:
  equation protime'=-1.0;
  event send1!  when t >= process_time_d2, typ=1, id=1 do busy_d2:=0, protime:=0.0,
  process_time_d2:=0.0, process_time_d3:=0.0, t:=0.0, ts:=0, ch_ward_xray:=1 goto check_bufsize_d2;
  event send3!  when t >= process_time_d2, typ=1, id=2 do busy_d2:=0, protime:=0.0,
  process_time_d2:=0.0, process_time_d3:=0.0, t:=0.0, ts:=0, ch_ward_exit:=1 goto check_bufsize_d2;
  event send2!  when t >= process_time_d2, typ=2 do busy_d2:=0, protime:=0.0,
  process_time_d2:=0.0, process_time_d3:=0.0, t:=0.0, ts:=0, ch_ward_plaster:=1 goto check_bufsize_d2;

40
event send1! when t >= process_time_d2, typ=3, id=1 do busy_d2:=0, protime:=0.0, process_time_d2:=0.0, process_time_d3:=0.0, t:=0.0, ts:=0, ch_ward_xray:=1 goto check_bufsize_d2;

event send3! when t >= process_time_d2, typ=3, id=3 do busy_d2:=0, protime:=0.0, process_time_d2:=0.0, process_time_d3:=0.0, t:=0.0, ts:=0, ch_ward_exit:=1 goto check_bufsize_d2;

event send3! when t >= process_time_d2, typ=4 do busy_d2:=0, protime:=0.0, process_time_d2:=0.0, process_time_d3:=0.0, t:=0.0, ts:=0, ch_ward_exit:=1 goto check_bufsize_d2;

end
Appendix D

generator: Generator (0.3);
bufferreg: BufferReg (generator.send, generator.pat);
registration: Registration(0.2, 0.5, 1.0, bufferreg.send, sup1);
bufferprio: Buffer_2i_4o (xray1.send1, xray2.send1, xray1.pat, xray2.pat);
bufferward1: Buffer_1i_2o (registration.send1, registration.pat);
ward1d1: Doctor (1.5, 3.2, 5.0, bufferward1.send1, bufferprio.send1, sup2, sup6);
ward1d2: Doctor (1.5, 3.2, 5.0, bufferward1.send2, bufferprio.send2, sup2, sup6);
ward2d3: Doctor (2.8, 4.1, 6.3, bufferward2.send1, bufferprio.send3, sup3, sup6);
ward2d4: Doctor (2.8, 4.1, 6.3, bufferward2.send2, bufferprio.send4, sup3, sup6);
bufferplaster: Buffer_6i_1o (ward1d1.send2, ward1d2.send2, ward2d3.send2, ward2d4.send2, xray1.send2, xray2.send2, ward1d1.pat, ward1d2.pat, ward2d3.pat, ward2d4.pat, xray1.pat, xray2.pat);
plaster: Plaster(3.0, 3.8, 4.7, bufferplaster.send, sup5);
bufferxray: Buffer_5i_2o (plaster.send2, ward1d1.send1, ward1d2.send1, ward2d3.send1, ward2d4.send1, plaster.send1, xray1.send1, xray2.send1, xray1.pat, xray2.pat);
xray1: Xray (2.0, 2.8, 4.1, bufferxray.send1, sup4);
xray2: Xray (2.0, 2.8, 4.1, bufferxray.send2, sup4);
exit: Exit (ward1d1.send3, ward1d2.send3, ward2d3.send3, ward2d4.send3, plaster.send1, ward1d1.pat, ward1d2.pat, ward2d3.pat, ward2d4.pat, plaster.pat);

def Registration(alg real low, mean, high; event receive; alg patient p1):
event send1, send2;
disc patient pat = (0,0,0.0);
disc dist int typedef = uniform(1, 100);
disc dist int cwchoice= uniform(1, 100);
disc dist real tria=triangle(low,mean,high);
cont t = 0.0 der 1.0;
disc real tb=0.0, process_time=0.0;
disc int typ=0, pel=0, id=0,cw=0;
cont protime=0.0;

location receiv:
initial;
equation protime'=0.0;
event receive? do id:=p1[id], tb:=p1[tb], pat:=p1, t := 0.0, process_time:=(sample tria)[0], tria:=(sample tria)[1], pel:=(sample typedef)[0], typedef:=(sample typedef)[1],cw:=(sample cwchoice)[0],
cwchoice:=(sample cwchoice)[1], protime:=(sample tria)[0] goto processing;

location processing:
equation protime'=-1.0;
when pel<=35 do typ:=1 goto defining;
when pel>35 and pel<=55 do typ:=2 goto defining;
when pel>55 and pel<=60 do typ:=3 goto defining;
when pel>60 do typ:=4 goto defining;

location defining:
equation protime'=-1.0;
do pat:=(id,typ, tb) goto sending;
location sending:
  equation protime' = -1.0;
  event send1! when t >= process_time, cw > 40 do pel := 0, protime := 0.0 goto receiving;
  event send2! when t >= process_time, cw <= 40 do pel := 0, protime := 0.0 goto receiving;
end

def Doctor (alg real low, mean, high; event receive1, receive2; alg patient p1, p2):
  event send1, send2, send3;
  disc patient pat = (0,0,0.0);
  disc dist real tria = triangle(low, mean, high);
  cont t = 0.0 der 1.0;
  disc real tb = 0.0, process_time = 0.0;
  disc int typ = 0, id = 0, busy = 0;
  cont protime = 0.0;

location arriving:
  initial;
  equation protime' = 0.0;
  when t > 30 goto receiving;

location receiving:
  equation protime' = 0.0;
  event receive1? do pat := p1, t := 0.0, id := p1[id], tb := p1[tb], typ := p1[typ], process_time := (sample tria)[0], tria := (sample tria)[1], busy := 1, protime := (sample tria)[0] goto processing;
  event receive2? do pat := p2, t := 0.0, id := p2[id], tb := p2[tb], typ := p2[typ], process_time := (sample tria)[0], tria := (sample tria)[1], busy := 1, protime := (sample tria)[0] goto processing;

location processing:
  equation protime' = -1.0;
  event send1! when t >= process_time, typ = 1, id = 1 do busy := 0, protime := 0.0 goto receiving;
  event send3! when t >= process_time, typ = 1, id = 2 do busy := 0, protime := 0.0 goto receiving;
  event send2! when t >= process_time, typ = 2 do busy := 0, protime := 0.0 goto receiving;
  event send1! when t >= process_time, typ = 3, id = 1 do busy := 0, protime := 0.0 goto receiving;
  event send3! when t >= process_time, typ = 3, id = 3 do busy := 0, protime := 0.0 goto receiving;
  event send3! when t >= process_time, typ = 4 do busy := 0, protime := 0.0 goto receiving;
end

def Xray(alg real low, mean, high; event receive; alg patient p1):
  event send1, send2;
  disc patient pat = (0,0,0.0);
  disc dist real tria = triangle(low, mean, high);
  disc dist int cwchoice = uniform(1, 100);
  cont t = 0.0 der 1.0;
  disc real process_time = 0.0;
  disc int typ = 0, id = 0, cw = 0, busy = 0;
  cont protime = 0.0;

location receiving:
  initial;
  equation protime' = 0.0;
event receive? do pat:=(p1[id]+1,p1[typ],p1[tb]), typ:=p1[typ], id:=p1[id]+1, t := 0.0, process_time:= (sample tria)[0], tria := (sample tria)[1], cw:= (sample cwchoice)[0], cwchoice := (sample cwchoice)[1], busy:=1, protime:= (sample tria)[0] goto defining;

location defining:
  equation protime'=-1.0;
  event send1! when typ=1 do busy:=0, protime:=0.0 goto receiving;
  event send2! when typ=3, id=2 do busy:=0, protime:=0.0 goto receiving;
  event send1! when typ=3, id=3 do busy:=0, protime:=0.0 goto receiving;
end

def Plaster(alg real low, mean, high; event receive; alg patient p1):
  event send1, send2;
  disc patient pat = (0,0,0.0);
  disc dist real tria=triangle(low,mean,high);
  cont t = 0.0 der 1.0;
  disc real process_time=0.0;
  disc int typ=0, busy=0;
  cont protime=0.0;

location receiving:
  initial;
  equation protime'=0.0;
  event receive? do pat:=p1, typ:=p1[typ], t := 0.0, process_time:= (sample tria)[0], tria := (sample tria)[1], busy:=1, protime:=(sample tria)[0] goto processing;

location processing:
  equation protime'=-1.0;
  event send1! when t >= process_time, typ=2 do busy:=0, protime:=0.0 goto receiving;
  event send2! when t >= process_time, typ=3 do busy:=0, protime:=0.0 goto receiving;
end

Appendix E

ciffile "BEPdeelA.cif";
svgfile "BEPdeelA.svg";

// Generator text.

output id gen_gentime_txt text:
  value generator.gentime;
  pattern "Gentime: %.1f";
end

output id Gen_count_txt text:
  value generator.count;
  pattern "%s";
// Registration text
output id Reg_bufsize_txt text:
  value bufferreg.bufsize;
  pattern "%s";
end

output id protime_reg_txt text:
  value registration.protime;
  pattern "%.1f";
end

// Ward1 text
output id Ward1_bufsize_txt text:
  value bufferward1.bufsize;
  pattern "%s";
end

output id protime_ward1_txt text:
  value ward1d1.protime;
  pattern "%.1f";
end

output id protime_ward2_2_txt text:
  value ward1d2.protime;
  pattern "%.1f";
end

// Ward2 text
output id Ward2_bufsize_txt text:
  value bufferward2.bufsize;
  pattern "%s";
end

output id protime_ward3_3_txt text:
  value ward2d3.protime;
  pattern "%.1f";
end

output id protime_ward4_txt text:
  value ward2d4.protime;
  pattern "%.1f";
end

// Plaster text
output id Pla_bufsize_txt text:
  value bufferplaster.bufsize;
  pattern "%s";
end

output id protime_plaster_txt text:
  value plaster.protime;
  pattern "%.1f";
end
// X-ray text
output id Xray_bufsize_txt text:
    value bufferxray.bufsize;
    pattern "%s";
end

output id protime_xray1_txt text:
    value xray1.protime;
    pattern "%.1f";
end

output id protime_xray2_txt text:
    value xray2.protime;
    pattern "%.1f";
end

// exit text
output id Exit_time1_txt text:
    value exit.avg1time;
    pattern "avgtime1: %.1f";
end

output id Exit_time2_txt text:
    value exit.avg2time;
    pattern "avgtime2: %.1f";
end

output id Exit_time3_txt text:
    value exit.avg3time;
    pattern "avgtime3: %.1f";
end

output id Exit_time4_txt text:
    value exit.avg4time;
    pattern "avgtime4: %.1f";
end

output id Exit_avgtime_txt text:
    value exit.avgtime;
    pattern "avgtime: %.1f";
end

output id Exit_time_txt text:
    value exit.t;
    pattern "time: %.1f";
end

output id Exit_count1_txt text:
    value exit.count1;
    pattern "count1: %s";
end

output id Exit_count2_txt text:
    value exit.count2;
    pattern "count2: %s";
output id Exit_count3_txt text:
  value exit.count3;
  pattern "count3: %s";
end

output id Exit_count4_txt text:
  value exit.count4;
  pattern "count4: %s";
end

output id Exit_count_txt text:
  value exit.count;
  pattern "count: %s";
end

// Server statuses.

output id busy_reg attr fill:
  value registration.busy:
    1 : "#ff4040";
    0 : "#40ff40";
end
end

output id busy_plaster attr fill:
  value plaster.busy:
    1 : "#ff4040";
    0 : "#40ff40";
end
end

output id busy_xray1 attr fill:
  value xray1.busy:
    1 : "#ff4040";
    0 : "#40ff40";
end
end

output id busy_xray2 attr fill:
  value xray2.busy:
    1 : "#ff4040";
    0 : "#40ff40";
end
end

output id busy_ward1_d1 attr fill:
  value ward1d1.busy:
    1 : "#ff4040";
    0 : "#40ff40";
end
end

output id busy_ward1_d2_2 attr fill:
value ward1d2.busy:
  1 : "#ff4040";
  0 : "#40ff40";
end
end

output id busy_ward2_d3_attr fill:
  value ward2d3.busy:
  1 : "#ff4040";
  0 : "#40ff40";
end
end

output id busy_ward2_d4_attr fill:
  value ward2d4.busy:
  1 : "#ff4040";
  0 : "#40ff40";
end
end

// channel color

output id registration_out attr fill:
  value if registration.ch_reg_bufward1=1 : 1
    else if registration.ch_reg_bufward2=1 : 2
      else 0
        end
    end:
  2 : "#0000ff";
  1 : "#17a000";
  0 : "#000000";
end
end

output id ch_reg_out attr stroke:
  value if registration.ch_reg_bufward1=1 : 1
    else if registration.ch_reg_bufward2=1 : 2
      else 0
        end
    end:
  2 : "#0000ff";
  1 : "#17a000";
  0 : "#000000";
end
end

output id ward1_in attr fill:
  value if registration.ch_reg_bufward1=1 : 1
    else if xray1.ch_xray_ward1=1 : 2
      else if xray2.ch_xray_ward1=1 : 2
        else 0
          end
        end
    end:
  2 : "#ffff00";
  1 : "#17a000";
output id ch_ward1_in attr stroke:
  value if registration.ch_reg_bufward1=1 : 1
    else if xray1.ch_xray_ward1=1 : 2
    else if xray2.ch_xray_ward1=1 : 2
      else 0
    end
  end
end

output id ward2_in attr fill:
  value if registration.ch_reg_bufward2=1 : 1
    else if xray1.ch_xray_ward2=1 : 2
    else if xray2.ch_xray_ward2=1 : 2
      else 0
    end
  end
end

output id ch_ward2_in attr stroke:
  value if registration.ch_reg_bufward2=1 : 1
    else if xray1.ch_xray_ward2=1 : 2
    else if xray2.ch_xray_ward2=1 : 2
      else 0
    end
  end
end

output id xray_out attr fill:
  value if xray1.ch_xray_ward1=1 : 1
    else if xray2.ch_xray_ward1=1 : 1
    else if xray1.ch_xray_ward2=1 : 2
    else if xray2.ch_xray_ward2=1 : 2
    else if xray1.ch_xray_plaster=1 : 3
    else if xray2.ch_xray_plaster=1 : 3
      else 0
    end
  end
end
else if plaster.ch_plaster_xray=1 : 3
else 0
end
end
end
end:
3 : "#828300",
2 : "#ff8b00",
1 : "#ff8bff",
0 : "#000000";
end
end

output id ward1_out attr fill:
value if ward1d1.ch_ward_xray=1 : 1
else if ward1d2.ch_ward_xray=1 : 1
else if ward1d1.ch_ward_plaster=1 : 2
else if ward1d2.ch_ward_plaster=1 : 2
else if ward1d1.ch_ward_exit=1 : 3
else if ward1d2.ch_ward_exit=1 : 3
else 0
end
end
end
end:
3 : "#bd344b",
2 : "#00a4ff",
1 : "#ff8b00",
0 : "#000000";
end
end

output id ch_ward1_out attr stroke:
value if ward1d1.ch_ward_xray=1 : 1
else if ward1d2.ch_ward_xray=1 : 1
else if ward1d1.ch_ward_plaster=1 : 2
else if ward1d2.ch_ward_plaster=1 : 2
else if ward1d1.ch_ward_exit=1 : 3
else if ward1d2.ch_ward_exit=1 : 3
else 0
end
end
end
end:
3 : "#bd344b",
2 : "#00a4ff",
1 : "#ff8b00",
0 : "#000000";
end
end
output id ch_ward1_out attr stroke_width:
    value if ward1d1.ch_ward_xray=1 : 1
    else if ward1d2.ch_ward_xray=1 : 1
    else if ward1d1.ch_ward_plaster=1 : 1
    else if ward1d2.ch_ward_plaster=1 : 1
    else if ward1d1.ch_ward_exit=1 : 1
    else if ward1d2.ch_ward_exit=1 : 1
    else 0
    end
end

1 : 7.0;
0 : 5.0;
end
end

output id ward2_out attr fill:
    value if ward2d3.ch_ward_xray=1 : 1
    else if ward2d4.ch_ward_xray=1 : 1
    else if ward2d3.ch_ward_plaster=1 : 2
    else if ward2d4.ch_ward_plaster=1 : 2
    else if ward2d3.ch_ward_exit=1 : 3
    else if ward2d4.ch_ward_exit=1 : 3
    else 0
    end
end

3 : "#afff88";
2 : "#9cff1b";
1 : "#ff8bff";
0 : "#000000";
end
end

output id ch_ward2_out attr stroke:
    value if ward2d3.ch_ward_xray=1 : 1
    else if ward2d4.ch_ward_xray=1 : 1
    else if ward2d3.ch_ward_plaster=1 : 2
    else if ward2d4.ch_ward_plaster=1 : 2
    else if ward2d3.ch_ward_exit=1 : 3
    else if ward2d4.ch_ward_exit=1 : 3
    else 0
    end
end
end
output id plaster_in attr fill:
value if xray1.ch_xray_plaster=1 : 1
    else if xray2.ch_xray_plaster=1 : 1
    else if ward1d1.ch_ward_plaster=1 : 2
    else if ward1d2.ch_ward_plaster=1 : 2
    else if ward2d3.ch_ward_plaster=1 : 3
    else if ward2d4.ch_ward_plaster=1 : 3
else 0
end
end
end
end : 3 : "#9c01b";
2 : "#0044ff";
1 : "#828300";
0 : "#000000";
end
end
end

output id ch_plaster_in attr stroke:
value if xray1.ch_xray_plaster=1 : 1
    else if xray2.ch_xray_plaster=1 : 1
    else if ward1d1.ch_ward_plaster=1 : 2
    else if ward1d2.ch_ward_plaster=1 : 2
    else if ward2d3.ch_ward_plaster=1 : 3
    else if ward2d4.ch_ward_plaster=1 : 3
else 0
end
end
end
end : 3 : "#9c01b";
2 : "#0044ff";
1 : "#828300";
0 : "#000000";
end
end
end

output id plaster_out attr fill:
value if plaster.ch_plaster_xray=1 : 1
    else if plaster.ch_plaster_exit=1 : 2
else 0
end
end : 2 : "#8ac078";
null
value if registration.ch_reg_buffer1=1 : 1
else if registration.ch_reg_buffer2=1 : 2
else if xray1.ch_xray_ward1=1 : 3
else if xray1.ch_xray_ward2=1 : 4
else if xray2.ch_xray_ward1=1 : 3
else if xray2.ch_xray_ward2=1 : 4
else 0
end
end
end
end
end
end

4 : "#7b009a";
3 : "#ffffff";
2 : "#0000ff";
1 : "#17a000";
0 : "#000000";
end
end

output id ch_ward_to_rest attr stroke:
value if ward1d1.ch_ward_xray=1 : 1
else if ward1d2.ch_ward_xray=1 : 1
else if ward2d3.ch_ward_xray=1 : 2
else if ward2d4.ch_ward_xray=1 : 2
else if ward1d1.ch_ward_plaster=1 : 3
else if ward1d2.ch_ward_plaster=1 : 3
else if ward2d3.ch_ward_plaster=1 : 4
else if ward2d4.ch_ward_plaster=1 : 4
else if ward1d1.ch_ward_exit=1 : 5
else if ward1d2.ch_ward_exit=1 : 5
else if ward2d3.ch_ward_exit=1 : 6
else if ward2d4.ch_ward_exit=1 : 6
else 0
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end

6 : "#afff88";
5 : "#bd344b";
4 : "#9cff1b";
3 : "#00a4ff";
2 : "#ff8bff";
1 : "#ff8b00";
0 : "#000000";
end
end

output id ch_ward_exit_plaster attr stroke:
value if 
  ward1d1.ch_ward_xray=1 : 1
  else if ward1d2.ch_ward_xray=1 : 1
  else if ward2d3.ch_ward_xray=1 : 2
  else if ward2d4.ch_ward_xray=1 : 2
  else if ward1d1.ch_ward_plaster=1 : 3
  else if ward1d2.ch_ward_plaster=1 : 3
  else if ward2d3.ch_ward_plaster=1 : 4
  else if ward2d4.ch_ward_plaster=1 : 4
  else if plaster.ch_plaster_exit=1 : 5
  else 0
end
end

5 : "#8ac078";
4 : "#9cff1b";
3 : "#00a4ff";
2 : "#ff8bff";
1 : "#ff8b00";
0 : "#000000";
end
end

output id ch_ward_plaster_xray attr stroke:
value if 
  ward1d1.ch_ward_xray=1 : 1
  else if ward1d2.ch_ward_xray=1 : 1
  else if ward2d3.ch_ward_xray=1 : 2
  else if ward2d4.ch_ward_xray=1 : 2
  else if ward1d1.ch_ward_plaster=1 : 3
  else if ward1d2.ch_ward_plaster=1 : 3
  else if ward2d3.ch_ward_plaster=1 : 4
  else if ward2d4.ch_ward_plaster=1 : 4
  else if plaster.ch_plaster_xray=1 : 5
  else 0
end
end

5 : "#828300";
4 : "#9cff1b";
3 : "#00a4ff";
2 : "#ff8bff";
1 : "#ff8b00";
0 : "#000000";
end
end

output id ch_xray_to_ward attr stroke:
value if 
\texttt{xray1.ch\_xray\_ward1}=1 : 3 
  
else if \texttt{xray1.ch\_xray\_ward2}=1 : 4 
else if \texttt{xray2.ch\_xray\_ward1}=1 : 3 
else if \texttt{xray2.ch\_xray\_ward2}=1 : 4 
else 0 
end 
end 
end : 
4 : "#7b009a";
3 : "#ffff00";
0 : "#000000";
end
end

output id ch_xray_to_plaster attr stroke:
value if 
\texttt{xray1.ch\_xray\_plaster}=1 : 1 
else if \texttt{xray2.ch\_xray\_plaster}=1 : 1 
else 0 
end 
end 
end : 
1 : "#828300";
0 : "#000000";
end
end

output id ch_xray_plaster attr stroke:
value if 
\texttt{ward1d1.ch\_ward\_xray}=1 : 1 
else if \texttt{ward1d2.ch\_ward\_xray}=1 : 1 
else if \texttt{ward2d3.ch\_ward\_xray}=1 : 2 
else if \texttt{ward2d4.ch\_ward\_xray}=1 : 2 
else if \texttt{plaster.ch\_plaster\_xray}=1 : 3 
else if \texttt{xray1.ch\_xray\_plaster}=1 : 4 
else if \texttt{xray2.ch\_xray\_plaster}=1 : 4 
else 0 
end 
end 
end 
end 
end : 
4 : "#828300";
3 : "#828300";
2 : "#ff8bff";
1 : "#ff8b00";
0 : "#000000";
output id protime_reg attr width:
  value registration.protime:
    inmin 0;
    inmax registration.process_time+0.0001;
    outmin 0;
    outmax 55;
end
end

output id protime_plaster attr width:
  value plaster.protime:
    inmin 0;
    inmax plaster.process_time+0.0001;
    outmin 0;
    outmax 55;
end
end

output id protime_ward1 attr width:
  value ward1d1.protime:
    inmin 0;
    inmax ward1d1.process_time+0.0001;
    outmin 0;
    outmax 55;
end
end

output id protime_ward2_2 attr width:
  value if ward1d2.process_time>0: ward1d2.protime
    else 0.0
    end:
    inmin 0;
    inmax ward1d2.process_time+0.0001;
    outmin 0;
    outmax 55;
end
end

output id protime_ward3_3 attr width:
  value if ward2d3.process_time>0: ward2d3.protime
    else 0.0
    end:
    inmin 0;
    inmax ward2d3.process_time+0.0001;
    outmin 0;
    outmax 55;
end
end
output id protime_ward4 attr width:
  value ward2d4.protime:
    inmin 0;
    inmax ward2d4.process_time+0.0001;
    outmin 0;
    outmax 55;
  end
end

output id protime_xray1 attr width:
  value xray1.protime:
    inmin 0;
    inmax xray1.process_time+0.0001;
    outmin 0;
    outmax 55;
  end
end

output id protime_xray2 attr width:
  value xray2.protime:
    inmin 0;
    inmax xray2.process_time+0.0001;
    outmin 0;
    outmax 55;
  end
end
Appendix F

The description of the ARGESIM comparison #6.
Comparison 6 - Definition

Casualties from accidents are admitted to an emergency department for dressing of wounds. Broken limbs are put in plaster. After a few days a follow-up examination must be performed to monitor the healing process. If necessary, additional treatment will be administered.

Follow-up treatment in the emergency department of a hospital is the discrete process to be investigated in this comparison.

The emergency department comprises the following facilities for follow-up treatment:

- Registration (one person): casualties are assigned to casualty wards 1 or 2; the necessity of further treatment is established.
- Waiting area (people waiting to enter casualty wards 1 and 2).
- Two casualty wards (CW1, CW2; with two doctors each but CW2 staffed only by inexperienced doctors for attention to simple cases).
- X-ray room with two X-ray units (but all people waiting in one single queue).
- A room where plaster casts are applied or removed (one person).

Patients start arriving at 7.30 a.m. and queue for registration. Doctors start work at 8.00 a.m. They attend to four types of patients:

1. Patients requiring X-raying. Patients are first examined in the casualty ward, then sent to the X-ray room. Before they leave their X-ray photographs are examined once again in the casualty ward.
2. Removal of plaster casts. Patients enter a casualty ward, are sent to the plastering room, then leave the department.
3. Plaster casts requiring X-raying and renewal. Patients enter the casualty ward, are sent to the X-ray room and given new plaster casts. After checking of the new plasters by X-raying again patients are readmitted to the casualty ward. They then leave the department.
4. Changing wound dressings. Patients are admitted to a casualty ward, then leave the department.

The statistical parameters are as follows:

- The time between arrivals of patients is distributed exponentially with parameter 0.3 minutes.
- The percent distribution of patients over the four groups described above is as follows: 1: 35%, 2: 20%, 3: 5%, 4: 40%.
- 60% of patients waiting for admission to a casualty ward are admitted to ward CW1, 40% to CW2.

The parameters of the single treatment points show a triangular distribution (minimum value / mode = most likely value / maximum value):

- Registration: 0.2 / 0.5 / 1.0 (min)
- CW1: 1.5 / 3.2 / 5.0 (min)
- CW2: 2.8 / 4.1 / 6.3 (min)
- X-ray: 2.0 / 2.8 / 4.1 (min)
- Plaster: 3.0 / 3.8 / 4.7 (min)

Patients wait in queues before every treatment point (only one queue for X-raying!).

The following experiments should be performed:

a. Determine average overall treatment time for 250 patients and classify these patients by types 1) to 4).
b. Assume that a doctor from CW1 (experienced) replaces one of the inexperienced doctors in CW2 as soon as the queue for CW2 is in excess of 20 patients. Note that the working time of the doctor from CW2 now working in CW1 is increased by 20% due to the more complex cases he/she has to deal with. As soon as queue for CW2 is down to five people the inexperienced doctor still working in
CW1 is returned to CW2. Perform Task 1) on this assumption.

Try to minimize the standard deviation of overall treatment time by introducing a priority ranking. One option: patients entering one of the treatment points for the second time (type 1 patients, type 3 patients) rank higher in priority than all other patients. Other priority rankings are of course also conceivable. F. Breitenecker, Technical University of Vienna
The following tasks had to be performed:

- **Task a:** There are four different types of patients: 1) Patients requiring X-ray. 2) Removal of plaster casts. 3) Plaster casts requiring X-ray and renewal. 4) Changing wound dressings. Average treatment time should be determined, both overall and for each of the types of patients.
- **Task b:** There are experienced doctors in CW1 and inexperienced doctors in CW2. When the queue for CW2 reaches 20, one doctor from CW1 replaces a doctor in CW2. As soon as the queue is down to 5, the doctors change back. The new treatment times should be determined on this assumption.
- **Task c:** Different strategies to minimize the standard deviation of the overall treatment time should be tested, such as a priority ranking of the patients.

Criteria used:

- **General Criteria:**
  - Model description
  - Control: local/global
- **Criteria - Task a:**
  - Average overall treatment time for 250 patients / close hour
  - Classification of these patients by types 1) / 2) / 3) / 4)
- **Criteria - Task b:**
  - Average overall treatment time for 250 patients / close hour
  - Classification of these patients by types 1) / 2) / 3) / 4)
- **Criteria - Task c:**
  - a) Patients entering for the second time rank higher in priority
  - b) Other priority ranking:
    - Average overall treatment time for 250 patients / close hour
    - Classification of these patients by types 1) / 2) / 3) / 4)