ENGSCI 355 Labs

Thomas Adams

2024-08-29

Table of contents

Preface

These are an online version of the Labs for ENGSCI 355. The topics covered are: a hands-on simulation of a manufacturing process; conceptual modelling using HCCM; implementing HCCM models in Jaamsim; and missing data imputaton.

Part I

Practical Lab

1 Operations System in Practice

The goal of this lab is to give you some hands-on experience with an operations system, the type of system that we will be focussing on simulating. Hopefully this will give you some idea of what is needed to simulate a system in terms of:

- the components of the system and how they interact with each other (entities and their behaviour);
- the type and amount of information/data that is needed, both for activity durations and control policies;
- the types of experiments that can be performed and how the system can be redesigned.

1.1 Making Paper Cars

The system that we will use as an example is making a car out of paper. You will each be given a piece of paper with the net of paper car on it as in Figure 1.1.

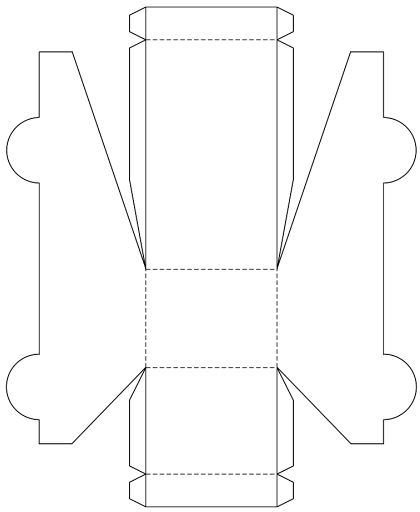

Figure 1.1: The Net Used to Make Paper Cars

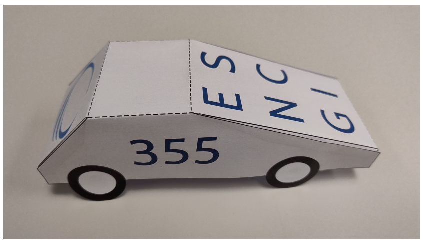

Figure 1.2: A Completed Car

You will also get a pair of scissors, some tape, and blank pieces of paper. To make the car:

1. Trace the net onto a new piece of paper.
2. Cut the new net out.
3. Fold the paper and tape the edges shut placing the tabs on the inside.

Figure 1.2 shows an example of a completed car.

First everyone should make one car by themselves. Once you have, show one of the instructors to get signed off. Then, discuss with you group how you can work together to make paper cars. You might want to experiment with different setups/policies and try making a few cars to see how long it takes and gather some data.

There will be a competition to see which group can make the most cars in 10 minutes. Before the time starts each group must submit an estimate of how many cars they believe they will be able to make. The score for each group will then be comprised of the following elements:

- 1 point for each car completed up to and including the estimated number.
- 0.25 points for each car completed above the estimated number.
- -0.75 points for each car not completed in the estimated number.

Additionally, the following rules must be followed:

1. Each car must be traced and cut individually.
2. Cars must be the same shape as the original template, including tabs.
3. You can have as many stencils as you like.
4. All final cars must have started as a blank, unfolded piece of paper.
5. You may not have any pre-cut tape or nets.
6. All cars must have been made only by members of your group.
7. All cars must be folded and taped neatly to count. The lecturer has final say on whether a car meets the required neatness.

1.2 Reflections

Now that you have attempted to make as many cars as you can you may wish to reflect on the process by asking yourself the following questions:

- Did your group have any traced/cut out cars left at the end?
- What was the bottleneck/slowest part of the system?
- Did you collect any data/do any experiments? If so, did they help? Would you do more/different ones now?
- What would you do differently next time?

The process that we considered was relatively simple. Cinsider how would your group's strategy change if any of the following additional conditions were added:

- Blank pieces of paper for you to trace onto only become available one at a time every two minutes;
- You have to make different styles of cars on demand;
- There is a limit to how many traced nets/cut pieces of tape you can have at any point (buffer limit);
- Each time a pair of scissors is stopped being used there is a cooldown time of 1 minute.

Part II

Conceptual Modelling Labs

2 HCCM Framework

This chapter describes the Hierarchical Control Conceptual Modelling (HCCM) framework which is used to build a conceptual model, aligned with the HCCM standard from lectures, that represents the practical activity, i.e., making paper cars, from Chapter 1.

Working in the same groups as for the practical activity and using this chapter as guidance, over the next two labs you will work through the phases for HCCM modelling shown in Figure 2.1 and complete templates for those steps. In the next lab you will complete phases 1, 2, 3, and start phase 4. The remainder of phase 4 will be completed in the lab after that. Chapter 1 provides a partially completed conceptual model of the car making system that you can use as a starting point.

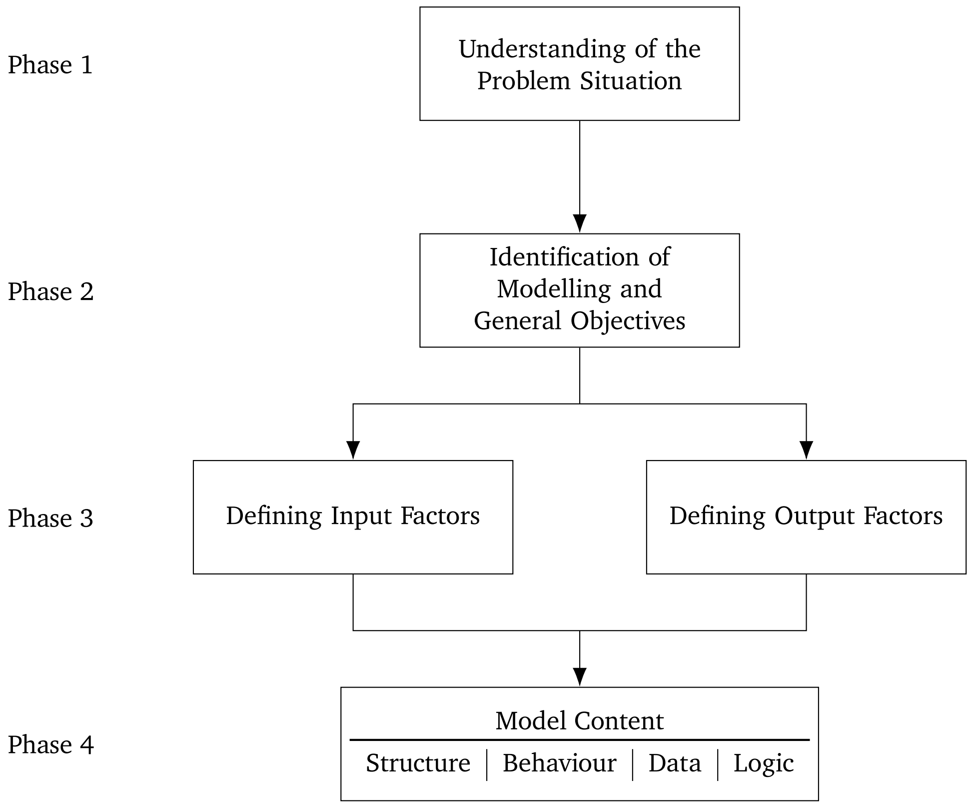

Figure 2.1: Conceptual Modelling Phases

2.1 Understanding of the Problem Situation

In Phase 1, in order to understand the problem situation, you need to summarise what is happening in a concise way. There is no strict rule for the best way to do this. One good approach is listening to the problem "holder", i.e., person/people who have the problem such as a client, then reflecting what you have heard in a couple of paragraphs with lists of key details and questions. You can then work through one or more iterations of feedback and refinement to get a final, agreed upon problem description.

2.2 Identification of Modelling and General Objectives

For Phase 2, as described in lectures, there are two types of objectives to consider when developing a simulation:

> "The second step deals with the determination of the objectives. According to Robinson [26] they drive all aspects of the modeling process and are a subset of an organization's aims. Further, objectives can be classified into modeling and general objectives, where the latter are concerned with the flexibility, run-speed, visual-display and model/component reuse."

For the modelling objective you may like to think about what you trying to discover using simulation, and what level of performance you are trying to achieve in which areas/metrics.

2.3 Defining Output Responses

Phase 3 includes defining both the output responses and input factors. You can do these in either order, but it can often be useful to define the output responses first, as it may help you think about what inputs will influence the outputs.

Output responses are things that can be measured and compared to understand how a system has behaved/performed. They are the metrics used to compare different simulation scenarios. The output responses should let you know whether the modelling objectives have been achieved and why or how. You may also want to consider how this will be reported (tables, graphs, etc.).

2.4 Defining Input Factors

Input factors are things that can be changed and may modify how a system behaves/performs. They are often defined to create multiple different scenarios to compare via simulation. They are also what you can change to try and achieve the modelling objectives.

2.5 Model Content

In Phase 4 the model content is defined. There is no strict order in which you need to complete the four components (structre, behaviour, data, and logic). A possible approach, that we will take in this lab, is to:

1. Identify the entities;
2. Draw the behavioural paths;
3. Define the data;
4. Define the structure (including the entities again);
5. Define the logic.

Using this approach you may still find yourself deciding to add/remove parts that you have already defined. This is a normal part of the conceptual modelling process, and you need to go back to the part of the process you want to change – for example adding and entity or activity – and then update the rest of the CM.

For the model content definition of our conceptual model we will follow the new HCCM standard. This standard is presented in an academic article (currently under review) that is available on Canvas under `Files > Lectures > Conceptual Modelling` in the file hccm-standard.pdf

2.5.1 Identifying Entities

Before formally defining entities it is often useful to identify entities in the system and whether they are active, i.e., have behaviour like a doctor or patient, or passive, i.e., are part of the system that should be modelled but that don't have explicit behaviour like a waiting room with a given capacity, but that doesn't actually have defined actions.

The goal is to identify everything that is involved in a meaningful way in all of the activities that are important to the system. Thinking about the inputs and outputs can also be useful. Clearly the entities must be influenced in some way by the inputs, and they must themselves influence the outputs. You may also consider that an activity does not have a significant influence on the performance of the system, and decide to exclude it – and therefore any entities that are involved only in that activity. Likewise the participation of a particular entity in an activity might be deemed inconsequential and therefore excluded. Although it is possible to revisit and add/remove entities later, at this stage you want to consider the whole system carefully, as it is easier to include/exclude an entity now than to change it later.

2.5.2 Drawing Behavioural Paths

Once preliminary identification of entities has been done, behavioural paths for each of the active entities should be drawn. These are essentially flowcharts with a special structure. Circles represent events, usually used when entities are arriving and leaving. Rectangles represent activities, including when entities have to wait for another activity. Red squares at the top left of an activity (or sometimes an event) let us know that some logic is triggered when the activity starts. This generally occurs at the start of "wait" activities and is used to check whether the conditions that mean the entity can stop waiting and move on to the next activity are met.

What we are trying to do when drawing the behavioural paths is identify the activities and events that the entities participate in, the possible orders that these can occur in, and any points where some control logic needs to be used.

Both when identifying the entities and drawing the behavioural paths it is important to keep track of any assumptions and simplifications that you make.

2.5.3 Define the Data

The data for the conceptual model includes both variables, and data modules. Variables can change their value throughout the simulation and are generally used to store some information temporarily before it is required later in the simulation. Data modules contain the information that is needed to perform the simulation and can be collected beforehand. Data mocules can also represent the input/experimental factors – the things that may change between different simulation scenarios. For each data module the following information should be given:

1. The name of the data module;
2. The source of the data, where the information was obtained;
3. The way the data is modelled, is it represented by a constant, a distribution, etc.
4. Whether the output is deterministic or stochastic;
5. The inputs that the module requires;
6. The quantity that the module outputs.

When presenting a conceptual model is useful to put the data first, as it is often referenced throughout the rest of the conceptual model.

2.5.4 Define the Structure

To define the structure we start with formally defining the entities by listing them along with any attributes that they have. Some common attributes, such as ID number and the activity that the entity is currently participating in, are often omitted to avoid repitition. Attributes are usually included either to assist with the system behaviour – for example record whether a patient has had a test – or to capture the perfomance of the system – how long something has waited for.

Next we define the transitions. Each arrow on a behavioural diagram corresponds to a transition. We can collate these in a table describing: the entity that is performing the transition, and the events that the entity transitions from and to. You can simply number them starting from 1, or adopt a convention of using the entity's initial as a prefix.

Once the transitions have been defined we can define the activities and events. Usually these are presented in two tables, one for the activities and one for the events. For each event (either standalone or as part of an activity) the table should include:

1. The participant(s);
2. The type – either scheduled or controlled;
3. The state changes that occur when the event happens.

The main things that occur in state changes are:

- Schedule an end event – usually in the start event of an activity with a scheduled end event;
- Starting another activity/event – this usually happens in a scheduled end event where an entity is transitioning to another scheduled event;
- Trigger some logic – often in the start event of an activity with a controlled end event.

The simulation start event, and arrive events are often more complicated and involve scheduling the initial events and creating entities.

2.5.5 Define the Logic

The final part of the conceptual model content is the logic. Each trigger that you drew in a behvioural path (the red squares) should correspond first to a trigger statement in the state changes of an event, and a piece of logic defined here. These pieces of logic are used to determine how the system behaves – what activity an entity should do next. It is common to have logic control the behaviour when one entity needs to wait for another, as when the first entity arrives it needs to check whether the other is free to perform an activity with it. The logic is usually presented as pseudocode, alongside the entity that triggers the logic.

2.6 Assumptions and Simplifications

Throughout the four phases of the HCCM framework you should document the assumptions and simplifications that you make. Assumptions are related to uncertainties about the system being modelled, and are used to fill in gaps in the information that is required for the simulation. Simplifications are changes that are made to the model to make it easier to defined or implement.

3 Inputs, Outputs, and Behaviour

In this lab you will complete the first three Phases of the HCCM framework, and part of the fourth, with your group.

3.1 Understanding of the Problem Situation

In the box below write a problem description for making paper cars, think about what you are trying to solve/discover by simulating this activity. You may want to look at Chapter 1 again to remind yourself about the process.

3.2 Modelling Objectives

In the box below write the modelling objectives for making paper cars, i.e., what are you trying to discover using simulation?

3.3 General Objectives

In the box below write the general objectives for making paper cars, i.e., what are some of the general properties you'd like your simulation to have?

3.4 Defining Output Responses

In the box below write the output responses for making paper cars, i.e., what are you going to measure to determine the performance of the system?

3.5 Defining Input Factors

In the box below write the input factors for making paper cars, i.e., what are you going to change to achieve the modelling objectives?

3.6 Identifying Entities

In the box below list the entities for making paper cars.

3.7 Drawing Behavioural Paths

The activity diagrams for the pencil & template, and scissors are given below in Figures 3.1, and 3.2.

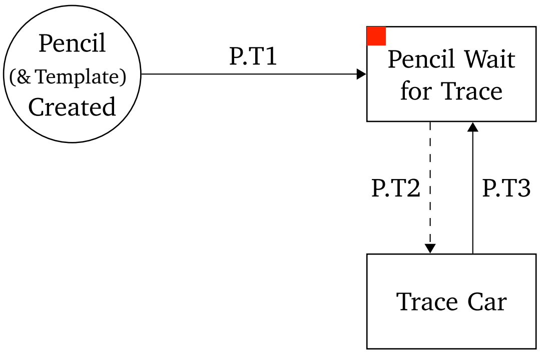

Figure 3.1: Pencil Activity Diagram

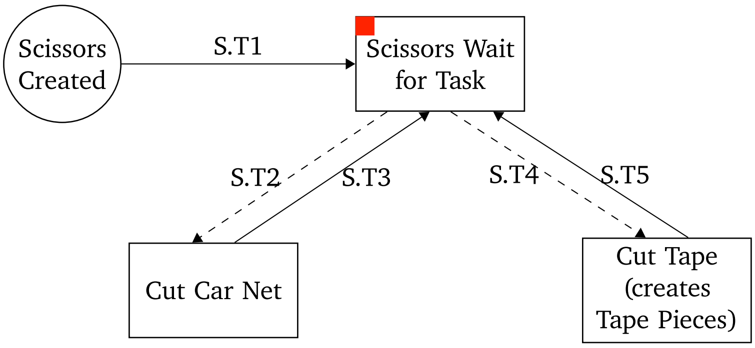

Figure 3.2: Scissors Activity Diagram

In the boxes below draw the activity diagrams for the remaining entities.

4 Data, Structure, and Logic

In this lab you will complete the remainder of the fourth phase of the HCCM framework, with your group.

4.1 Define the Data

Firstly, you need to give detailed definitions of the data modules. You may not have collected data during car making, but complete the following table that describes the kind of data you would need to collect to simulate car making. Also add a comment on how the entry for CutTapeDuration would change if no person-by-person data was available, but an Exponential distribution that estimated the time it takes for a person to cut tape was available.

Table 4.1: List of Data Modules

Name	Source	Model	Type	Input	Output
NumPencils	System info	Constant	Deterministic	-	The number of pencils available
NumScissors	System info	Constant	Deterministic	-	The number of scissors available
NumTape	System info	Constant	Deterministic	-	The number of rolls of tape available
Num	System info	Constant	Deterministic	-	
TraceCar	Experimental data	Lookup	Deterministic	Person	Time to trace car
CutNet	Experimental data	Lookup	Deterministic	Person	Time to cut the net out
CutTape	Experimental data	Lookup	Deterministic	Person	Time to cut a piece of tape

4.2 Define the Structure

The first part of the structure to define is the entities. Table 4.2 lists the entities again, but adds attributes that the entities will need to capture the performance of the system, e.g., waiting time until the cube was cut. It is assumed that all entities have the three attributes: ID, CurrentActivity, and CurrentStart. These are omitted in the table to prevent repetition.

The next part of the structure is the transitions, which describe how entities move between activities and events. Table 4.3 lists the transitions for making paper cars. These transitions are prefixed by entity of the behavioural pathway they come from. Complete the transitions for the Scissors pathway.

Table 4.2: List of Entities

Entity	Attributes
Paper	WaitForTrace[0.0] WaitForCutShape[0.0] WaitForFold[0.0] WaitForTapeCube[0.0]
Pencil	WaitForTrace[0.0]
Scissors	WaitForTask[0.0]
Tape	WaitForCut[0.0]
TapePieces	WaitForTape[0.0] ArrivalTime[0.0] LeavingTime[0.0]
Person	WaitForTask[0.0]

Table 4.3: List of Transitions

Participant	Name	From Event	To Event
Paper	P.1	Paper Created	Paper Wait for Trace.Start
	P.2	Paper Wait for Trace.End	Trace Car.Start
	P.3	Trace Car.End	Paper Wait for Cut Net.Start
	P.4	Paper Wait for Cut Net.End	Cut Car Net.Start
	P.5	Cut Car Net.End	Car Wait for Fold.Start
	P.6	Car Wait for Fold.End	Fold Car.Start
	P.7	Fold Car.End	Car Wait for Tape.Start
	P.8	Car Wait for Tape.End	Tape Car.Start
	P.9	Tape Car.End	Car Finished
Pencil	N.1	Pencil/Template Created	Pencil Wait for Trace.Start
	N.2	Pencil Wait for Trace.End	Trace Car.Start
	N.3	Trace Car.End	Pencil Wait for Trace.Start
Scissors	S.1	Scissors Created	
	S.2		
	S.3		
	S.4		
	S.5		
Tape	T.1	Tape Created	Tape Wait for Cut.Start
	T.2	Tape Wait for Cut.End	Cut Tape.Start
	T.3	Cut Tape.End	Tape Wait for Cut.Start
Tape Piece	TP.1	Tape Pieces Created	Tape Pieces Wait for Tape.Start
	TP.2	Tape Pieces Wait for Tape.End	Tape Car.Start
	TP.3	Tape Car.End	Tape Pieces Leave
Person	H.1	Person Created	Person Wait for Task.Start
	H.2	Person Wait for Task.End	Trace Car.Start
	H.3	Trace Car.End	Person Wait for Task.Start
	H.4	Person Wait for Task.End	Cut Car Net.Start
	H.5	Cut Car Net.End	Person Wait for Task.Start
	H.6	Person Wait for Task.End	Fold Car.Start

Table 4.3: List of Transitions

Participant	Name	From Event	To Event
	H.7	Fold Car.End	Person Wait for Task.Start
	H.8	Person Wait for Task.End	Cut Tape.Start
	H.9	Cut Tape.End	Person Wait for Task.Start
	H.10	Person Wait for Task.End	Tape Car.Start
	H.11	Tape Car.End	Person Wait for Task.Start

Table 4.4 lists the activities from the behavioural pathway diagrams along with the state changes for the start and end event of each activity. Complete the activities for:

- Car Wait for Tape Car
- Tape Car
- Person Wait for Task (*Hint* look at Scissors Wait for Task)

Table 4.4: Activities

Activity	Participants	Event	Type	State Change

Paper Wait for Trace	Paper (P)	Start	Scheduled	`(Default, omitted hereafter) P.CurrentActivity = "this activity"`
				`(Default, omitted hereafter) P.CurrentStart = TIME`
				`TRIGGER OnStartPaperWaitForTrace WITH C`

		End	Controlled	`P.WaitForTrace = TIME - P.CurrentStart` `# TRANSITION PAP.2 in logic`
Trace Car	Paper (P), Person (H), Pencil (N)	Start	Controlled	`SCHEDULE END at TIME + TraceCube(H)`

		End	Scheduled	`START Paper Wait for Cut Net WITH P # TRANSITION PAP.3`
				`START Person Wait for Task WITH H # TRANSITION PER.3`
				`START Pencil Wait for Trace WITH N # TRANSITION PEN.3`

Paper Wait for Cut Net		Start		`TRIGGER OnStartPaperWaitForCutNet WITH P`
		End		`P.WaitForCutNet = TIME - P.CurrentStart` `# TRANSITION PAP.4 in logic`
Cut Car Net	Paper (P), Person (H), Scissors (S)	Start	Controlled	`SCHEDULE END at TIME + CutNet(H)`

		End	Scheduled	`START Car Wait for Fold WITH P # TRANSITION PAP.5`
				`START Person Wait for Task WITH H # TRANSITION PER.5`
				`START Scissors Wait for Task WITH S # TRANSITION S.3`

Car Wait for Fold	Paper (P)	Start	Scheduled	`TRIGGER OnStartCarWaitForFold WITH P`
		End	Controlled	`P.WaitForFold = TIME - P.CurrentStart` `# TRANSITION PAP.6 in logic`
Fold Car	Paper (P), Person (H)	Start	Controlled	`SCHEDULE END at TIME + FoldCar(H)`
		End	Scheduled	`START Car Wait for Tape Car WITH P # TRANSITION PAP.7` `START Person Wait for Task WITH H # TRANSITION PER.7`
Car Wait for Tape Car		Start		

Table 4.4: Activities

Activity	Participants	Event	Type	State Change
		End		
Tape Car		Start		
		End		
Pencil Wait for Trace	Pencil (N)	Start	Scheduled	`TRIGGER OnStartPencilWaitForTrace WITH N`
		End	Controlled	`N.WaitForTrace = N.WaitForTrace + TIME - N.CurrentStart` `# TRANSITION N.2 in logic`
Scissors Wait for Task	Scissors (S)	Start	Scheduled	`TRIGGER OnStartScissorsWaitForTask WITH S`
		End	Controlled	`S.WaitForTask = S.WaitForTask + TIME - S.CurrentStart` `# TRANSITION S.2 or S.4 in logic`
Cut Tape	Tape (T), Person (H), Scissors (S)	Start	Controlled	`SCHEDULE END at TIME + CutTape(H)`

		End	Scheduled	`START Person Wait for Task WITH H # TRANSITION PER.9`
				`START Scissors Wait for Task WITH S # TRANSITION S.5`
				`START Tape Wait for Cut WITH T # TRANSITION T.3`
				`CREATE Tape Pieces TP`
				`START Tape Pieces Created WITH TP`

Tape Wait for Cut	Tape (T)	Start	Scheduled	`TRIGGER OnStartTapeWaitForCut WITH T`
		End	Controlled	`T.WaitForCut = T.WaitForCut + TIME - T.CurrentStart` `# TRANSITION T.2 in logic`
Tape Pieces Wait for Tape	Tape Pieces (TP)	Start	Scheduled	`TRIGGER OnStartTapePiecesWaitForTape WITH TP`
		End	Controlled	`TP.WaitForTape = TP.WaitForTape + TIME - TP.CurrentStart` `# TRANSITION TP.2 in logic`
Person Wait for Task		Start		
		End		

Table 4.5 lists the events to start and finish the simulation along with the events from the behavioural pathway diagrams along with the state changes for each event. Complete the activities for:

- Tape Pieces Created
- Person Created

Table 4.5: Events

Event	Participants	Type	State Change

Simulation Start	None	Scheduled	`FOR NumPaper DO`
			`  CREATE Paper P`
			`  START Paper Created WITH P`
			`END FOR`
			`FOR NumPencils DO`
			`  CREATE Pencil N`
			`  START Pencil/Template Created WITH N`
			`END FOR`
			`FOR NumScissors DO`
			`  CREATE Scissors S`
			`  START Scissors Created WITH S`
			`END FOR`
			`FOR NumTape DO`
			`  CREATE Tape T`
			`  START Tape Created WITH T`
			`END FOR`
			`FOR NumPeople DO`
			`  CREATE Person H`
			`  START Person Created WITH H`
			`END FOR`

Paper Created	Paper (P)	Scheduled	`START Paper Wait for Trace WITH P # TRANSITION PAP.1`
Car Finished	Paper (P)	Scheduled	`Calculate statistics for P`
Pencil/ Template Created	Pencil (N)	Scheduled	`START Pencil Wait for Trace WITH N # TRANSITION PEN.1`
Scissors Created	Scissors (S)	Scheduled	`START Scissors Wait for Task WITH S # TRANSITION S.1`
Tape Created	Tape (T)	Scheduled	`START Tape Wait for Cut WITH T # TRANSITION T.1`
Tape Pieces Created			
Tape Pieces Leave	Tape Pieces (TP)	Scheduled	`Calculate statistics for TP`
Person Created			
Simulation Finish	None	Scheduled	`Calculate statistics as required for Pencils, Scissors, Tape , Person entities`

4.3 Define the Logic

The last part of the structure to define is the logic. You need to define the logic for each of the triggers (the red squares in the behavioural paths, and TRIGGER statements in the state changes). Tables XXX show the logic for some of the triggers. Complete the logic for: - On Tape Wait for Cut.Start - On Cube Wait for Fold.Start - the last condition of On Person Wait for Task.Start

Table 4.6: OnStartPencilWaitForTrace

Triggered by: Pencil N

```
IF (any Paper P with P.CurrentActivity = Paper Wait for Trace) AND
   (any Person H with H.CurrentActivity = Person Wait for Task) THEN
  SELECT valid Paper P
  SELECT valid Person H
  START Trace Car WITH P, H, N
END IF
```

Table 4.7: OnStartScissorsWaitForTask

Triggered by: Scissors S

```
IF (any Paper P with P.CurrentActivity = Paper Wait for Cut Net) AND
   (any Person H with H.CurrentActivity = Person Wait for Task) THEN
  SELECT valid Paper P
  SELECT valid Person H
  START Cut Net WITH P, H, S
ELSE IF (any Tape T with T.CurrentActivity = Tape Wait for Cut) AND
        (any Person H with H.CurrentActivity = Person Wait for Task) THEN
  SELECT valid Tape T
  SELECT valid Person H
  START Cut Tape WITH T, H, S
END IF
```

Table 4.8: OnStartTapeWaitForCut

Triggered by: Tape T

```
IF (any Scissors S with S.CurrentActivity = Scissors Wait for Task) AND
   (any Person H with H.CurrentActivity = Person Wait for Task) THEN
  SELECT valid Scissors S
  SELECT valid Person H
  START Trace Car WITH T, H, S
END IF
```

Table 4.9: OnStartTapePiecesWaitForTape

Triggered by: Tape Pieces TP

```
IF (any Paper P with P.CurrentActivity = Paper Wait for Tape) AND
   (any Person H with H.CurrentActivity = Person Wait for Task) THEN
  SELECT valid Paper P
  SELECT valid Person H
  START Trace Car WITH P, H, TP
END IF
```

Table 4.10: OnStartPaperWaitForTrace

Triggered by: Paper P

```
IF (any Pencil N with N.CurrentActivity = Pencil Wait for Trace) AND
  (any Person H with H.CurrentActivity = Person Wait for Task) THEN
 SELECT valid Pencil N
 SELECT valid Person H
 START Trace Car WITH P, H, N
END IF
```

Table 4.11: OnStartPaperWaitForCut

Triggered by: Paper P

```
IF (any Scissors S with S.CurrentActivity = Scissors Wait for Task) AND
  (any Person H with H.CurrentActivity = Person Wait for Task) THEN
 SELECT valid Paper P
 SELECT valid Person H
 START Cut Car WITH P, H, S
END IF
```

Table 4.12: OnStartPaperWaitForFold

Triggered by: Paper P

```
IF (any Person H with H.CurrentActivity = Person Wait for Task) THEN
 SELECT valid Person H
 START Trace Car WITH P, H
END IF
```

Table 4.13: OnStartPaperWaitForTape

Triggered by: Paper P

```
IF (any Tape Pieces TP with TP.CurrentActivity = Tape Pieces Wait for Tape) AND
  (any Person H with H.CurrentActivity = Person Wait for Task) THEN
 SELECT valid Tape Pieces TP
 SELECT valid Person H
 START Trace Car WITH P, H, TP
END IF
```

Table 4.14: OnStartPersonWaitForTask

Triggered by: Person H

```
# Prioritise taping, then folding, then cutting, then tracing
IF (any Paper P with P.CurrentActivity = Paper Wait for Tape Car) AND
   (any Tape Pieces with TP.CurrentActivity = Tape Pieces Wait for Tape) THEN
  SELECT valid Paper P
  SELECT valid Tape Pieces TP
  START Tape Car WITH P, H, TP
ELSE IF (any Paper P with P.CurrentActivity = Paper Wait for Tape Car) AND
        (any Tape T with T.CurrentActivity = Tape Wait for Cut) AND
        (any Scissors S with S.CurrentActivity = Scissors Wait for Task) THEN
  # There is a car waiting to be taped, but no tape pieces
  SELECT valid Tape T
  SELECT valid Scissors S
  START Cut Tape WITH T, H, S
ELSE IF (any Paper P with P.CurrentActivity = Paper Wait for Fold)
  SELECT valid Paper P
  START Fold Car WITH P, H
ELSE IF (any Paper P with P.CurrentActivity = Paper Wait for Cut Net) AND
        (any Scissors with S.CurrentActivity = Scissors Wait for Task) THEN
  SELECT valid Paper P
  START Cut Car Net WITH P, H, S
ELSE IF (any Paper P with P.CurrentActivity = Paper Wait for Trace) AND
        (any Pencil N with N.CurrentActivity = Pencil Wait for Trace) THEN
  SELECT valid Paper P
  SELECT valid Pencil N
  START Fold Car WITH P, H, N
END IF
```

Part III

Conceptual Models

5 Radiology Clinic

5.1 Data

Table 5.1: List of Global Variables

Name	Description	Initial Value
NextPatIdNum	The Id number that will be assigned to the next patient	1
NextReceptionistIdNum	The Id number that will be assigned to the next receptionist	1
NextCTMachineIdNum	The Id number that will be assigned to the next CT Machine	1
P	The set of all patients	$\emptyset$
R	The set of all receptionists	$\emptyset$
C	The set of all CT Machines	$\emptyset$

Table 5.2: List of Data Modules

Name	Source	Identification	Input	Output
PatientInterarrivalTime	Poisson Process	Parameter	Mean interarrival time	Sample from Distribution
NumReceptionists	Constant	Parameter	N/A	Value
NumCTMachines	Constant	Parameter	N/A	Value
CheckInTime	Uniform Distribution	Parameter	Min and max time	Sample from Distribution
ScanTime	Log-normal Distribution	Parameter	Mean and std. dev.	Sample from Distribution

5.2 Components

Table 5.3: List of Entities

Entity	Attributes
Patient	ID State StateTimes
Receptionist	ID State StateTimes
CT Machine	ID State StateTimes

Table 5.4: List of Transitions

No.	Participant	From Event	To Event
1	Patient	Arrive(P)	Wait for check in.Start
2	Patient	Wait for check in.End	Check in.Start
3	Patient	Check in.End	Wait for scan.Start
4	Patient	Wait for scan.End	Scan.Start
5	Patient	Scan.End	Leave(P)
6	Receptionist	Arrive(R)	Wait for task(R).Start
7	Receptionist	Wait for task(R).End	Check in.Start
8	Receptionist	Check in.End	Wait for task(R).Start
9	Receptionist	Wait for task(R).End	Leave(R)
10	CT Machine	Arrive(CT)	Wait for task(CT).Start
11	CT Machine	Wait for task(CT).End	Scan.Start
12	CT Machine	Scan.End	Wait for task(CT).Start
13	CT Machine	Wait for task(CT).End	Leave(CT)

Table 5.5: Activities

Activity	Participants	Event	Type	State Change
Wait for Check In	Patient (p)	Start	Scheduled	`TRIGGER OnStartWaitForCheckIn WITH p`
		End	Controlled	
Check In	Patient (p), Receptionist (r)	Start	Controlled	`SCHEDULE Check In.End at TIME + CheckInTime()`
		End	Scheduled	`START Wait for Scan WITH p` `START Wait for Task (R) WITH r`
Wait for Scan	Patient (p)	Start	Scheduled	
		End	Controlled	`TRIGGER OnStartWaitForScan WITH p`
Scan	Patient (p), CTMachine (c)	Start	Controlled	`SCHEDULE Scan.End at TIME + ScanTime()`

Table 5.5: Activities

Activity	Participants	Event	Type	State Change
		End	Scheduled	`START Leave (P) WITH p` `START Wait for Task (CT) WITH c`
Wait for Task (R)	Receptionist (r)	Start	Scheduled	`TRIGGER OnStartWaitForTaskR WITH r`
		End	Controlled	
Wait for Task (CT)	CTMachine (c)	Start	Scheduled	`TRIGGER OnStartWaitForTaskCT WITH c`
		End	Controlled	

Table 5.6: Events

Event	Participants	Type	State Change

Simulation Start	-	Scheduled	`SCHEDULE Arrival (R) at TIME`
			`SCHEDULE Arrival (CT) at TIME`
			`SCHEDULE Arrival (P) at TIME + PatientInterArrival()`

Arrival (P)	Patient (p)	Scheduled	`p.ID = NextPatIDNum`
			`p.Priority = PatientPriority()`
			`NextPatIDNum = NextPatIDNum + 1`
			`SCHEDULE Arrival (P) at TIME + PatientInterArrival()`
			`START Wait for Check In WITH p`

Leave (P)	Patient (p)	Scheduled	`Calculate statistics for p`

Arrival (R)	Receptionist (r)	Scheduled	`r.ID = NextReceptionistIDNum`
			`NextReceptionistIDNum = NextReceptionistIDNum + 1`
			`IF NextReceptionistIDNum <= NumReceptionists THEN`
			`    SCHEDULE Arrival (R) at TIME`
			`END IF`
			`START Wait for Task (R) WITH r`

Leave (R)	Receptionist (r)	Scheduled	`Calculate statistics for r`

Arrival (CT)	CTMachine (c)	Scheduled	`c.ID = NextCTMachineIDNum`
			`NextCTMachineIDNum = NextCTMachineIDNum + 1`
			`IF NextCTMachineIDNum <= NumCTMachines THEN`
			`    SCHEDULE Arrival (CT) at TIME`
			`END IF`
			`START Wait for Task (CT) WITH c`

Leave (P)	CTMachine (c)	Scheduled	`Calculate statistics for c`

5.3 Activity Diagrams

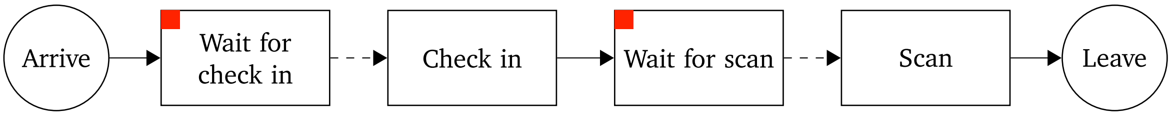

Figure 5.1: Patient Activity Diagram

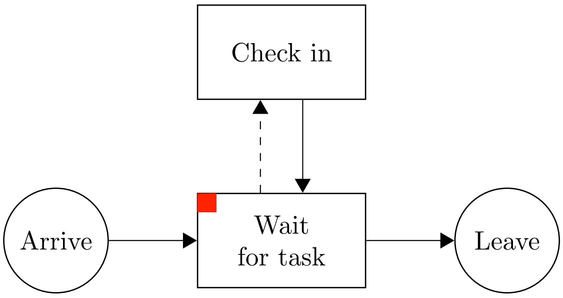

Figure 5.2: Receptionist Activity Diagram

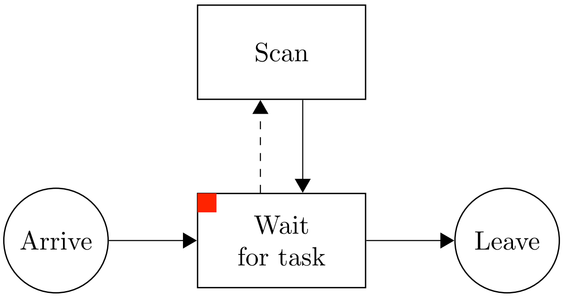

Figure 5.3: CT Activity Diagram

5.4 Logic

Table 5.7: OnStartWaitForCheckIn

Triggered by: Patient p

```
receps = {r FOR r IN R IF r.State = "Wait for task (R)"}
IF receps IS NOT empty THEN
    r_hat = argmin{r.CurrentStart FOR r IN receps}
    START Check In WITH p, r_hat
END IF
```

Table 5.8: OnStartWaitForScan

Triggered by: Patient p

```
cts = {c FOR c IN C IF c.State = "Wait for task (C)"}
IF cts IS NOT empty THEN
    c_hat = argmin{c.CurrentStart FOR c IN cts}
    START Scan WITH p, r_hat
END IF
```

Table 5.9: OnStartWaitForTaskR

Triggered by: Receptionist r

```
patients = {p FOR p IN P IF p.State = "Wait for Check In"}
IF patients IS NOT empty THEN
    p_hat = argmin{p.CurrentStart FOR p IN patients}
    START Check In WITH p, r_hat
END IF
```

Table 5.10: OnStartWaitForTaskCT

Triggered by: CTMachine c

```
patients = {p FOR p IN P IF p.State = "Wait for Scan"}
IF patients IS NOT empty THEN
    p_hat = argmin{p.CurrentStart FOR p IN patients}
    START Scan WITH p, r_hat
END IF
```