

**MASTER**

**Formal modeling of paper path exception handling using CIF**

Kooiman, P.J.P.

*Award date:*  
2011

[Link to publication](#)

**Disclaimer**

This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

**General rights**

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

Formal modeling of paper path  
exception handling using CIF

P.J.P. Kooiman (0514731)

SE 420649

Master's Thesis

Supervisor: Prof.dr.ir. J.E. Rooda

Advisors: Dr.ir. D.A. van Beek

Ir. R. Fabel (Océ )

Dr. J. Markovski

Committee: Prof.dr. J.C.M. Baeten

Dr.ir. P.J.L. Cuijpers

EINDHOVEN UNIVERSITY OF TECHNOLOGY  
DEPARTMENT OF MECHANICAL ENGINEERING  
SYSTEMS ENGINEERING GROUP

Eindhoven, July 11, 2011



## FINAL ASSIGNMENT

EINDHOVEN UNIVERSITY OF TECHNOLOGY  
Department of Mechanical Engineering  
Systems Engineering Group

October 2009

Student P.J.P. Kooiman  
Supervisor Prof.dr.ir. J.E. Rooda  
Advisors Dr.ir. D.A. van Beek  
Dr. J. Markovski  
Ir. R. Fabel (Océ)  
Start October 2009  
Finish September 2010  
Title Control software synthesis for exception handling in the paper-path in Océ printers using supervisory control theory

### Subject

The topic of the project is modeling and supervisory control of Océ printing machines. Modern printing machines are complex multidisciplinary systems which combine electro-mechanic components with embedded control software. The paperpath is the part of a printer where sheets are transported by pinches from an input tray to an output tray. The exception handling takes care that the paperpath comes to a safe rest, when an error has occurred. Prior investigations have shown that supervisory control theory might be suitable for software synthesis for exception handling in Océ printing machines. Conclusions were that automatic generation of correct-by-design software may lead to a decrease in development time with a less number of errors. This project is a follow-up of previous work. The purpose of the project is to have the exception handling software implemented into an Océ printing machine. For this purpose also simulation and testing have to be performed in order to verify proper functioning of the control software.

### Assignment

- Get familiar with the paperpath and the exception handling in an Océ printing machine.
- Use supervisory control theory to synthesize a supervisor for exception handling.
- From the supervisor, generate control software for exception handling and integrate this software into the embedded software environment.
- Illustrate the concept in a simulation environment.

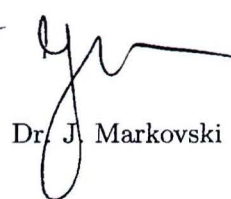
The project will be carried out in cooperation with Océ and the student is expected to spend some time at the research and development headquarters of Océ in Venlo.



Prof.dr.ir. J.E. Rooda



Dr.ir. D.A. van Beek



Dr. J. Markovski



Ir. R. Fabel

Systems  
Engineering



Department of Mechanical Engineering



# Preface

This thesis is presented to the Eindhoven University of Technology to obtain my master's degree in Mechanical Engineering. It marks the end of the period in my life as a student and I feel I am ready for the next step. During this period I have developed academical skills and knowledge. Besides that, I was an active member at W.S.V. Simon Stevin and E.S.V.V. Hajraa. Through these associations I was able to make new friends and develop my soft skills in a pleasant environment.

The reasons for me to join the Systems Engineering Group in 2008 were the emphasis on thorough analysis, the constant drive for improvement and the challenging industrial applications. I thank all members of the System Engineering Group for their interesting and instructive courses. I did my internship at the Arizona State University and I thank Dieter Armbruster for giving me this opportunity and his coaching. Staying in the USA for six months was a wonderful experience.

The result of this project was not possible without the help of many people. I thank Bert van Beek for his help on modeling and his passion to improve and expand these models. I thank Ronald Fabel from Océ for his contagious enthusiasm and for sharing his knowledge on printers and printer development with me. I thank Jasen Markovski for his help with this thesis and his formal abilities. Professor Rooda supervised this project and I am grateful for his interest, advice, and guidance. Through his wisdom and knowledge I widened my self knowledge.

Special thanks goes out to my parents and sister for their unconditional support and their patience. Finally, I thank Mariska for always being there for me.



# Summary

Modern competitive markets of high-tech machines impose ever-increasing requirements on product quality, ease of use, safety, and performance. Other actors are price and time to market. One of the greatest challenges in multidisciplinary environments, such as the one in which high-tech printers are made, is proper communication between engineers of diverse backgrounds. Formal models greatly alleviate the communication barriers between the involved parties.

The paper path of a printer is the part of the printer that transport sheets. Various components are used to detect, guide, and accelerate sheets in the paper path. Sensors detect the edges of the sheets. The paper path might hold a duplex track and a turn track for duplex printing. For more complex paper paths several routings may exist. The software that controls the paper path is sheet based. By this is meant that each sheet is controlled by a piece of software called a sheet procedure. This sheet procedure controls the hardware such that the sheet follows the desired trajectory in time and place in the paper path.

Exception handling is the desired behavior of sheets in the paper path if an error has occurred. An error means that an edge is not detected on time at a sensor. Besides errors of late edges also errors concerning skewness and size exist. The most important goal of exception handling is to make sure all sheets come to a safe stop in the paper path. Requirements specify whether or not to stop sheets at particular locations. For example, stopping a sheet in the hot zone is not allowed since this results in unsafe behavior. The exception handling software is added to the sheet procedures. The most important drawbacks of the current exception handling and its development are: the manual coding of the software, the distributed control concept which does not fit the integral behavior requirements, the lack of reusability for other printers, and the hardware based incremental exception handling design.

A new approach for exception handling and its development are suggested. This approach is based on so called signs. These signs only exist in the software. Variable instructions can be set on these signs. Sheets procedures check these signs and make sure the sheet behaves as instructions prescribe. The signs are used to instruct sheet procedures in case of an error. A lookup table is used to store which instruction to set on which sign in case of which error. An advantage of this approach is that the specific



stop control for exception handling in the sheet procedure is replaced by generic stop control via the signs.

A case is used to test the new approach. The CIF2 formalism is used to model the sheets in the paper path, the signs, the errors and the exception handling via the lookup table. First it was checked by simulation if the sheets in the paper path behave according to the signs instructions. Next, the model was simulated to derive the lookup table, i.e. which instructions on which sign for each error such that the requirements for exception handling are met. Simulation of the model proves to be a useful when trying to understand the effect of a certain instruction on a sign on the behavior of sheets in the paper path. For all fourteen errors in the case, the signs approach is able to instruct sheets such that the requirements are not violated. The modeling approach can be used easily to model other paper paths. The model can be used to derive the lookup table early in the design process of a printer. The derived lookup table can be easily implemented in the software.

# Samenvatting

Moderne concurrerende markten voor hoogtechnologische machines leggen steeds hogere eisen op ten aanzien van kwaliteit, gebruiksvriendelijkheid, veiligheid en prestaties. Andere actoren die een rol spelen zijn de prijs en de marktintroductietijd. Een van de grootste uitdagingen in een multidisciplinaire omgeving, zoals die waarin hoogtechnologische printers worden gemaakt, is heldere communicatie tussen ingenieurs met verschillende achtergronden.

Het papierpad van een printer is het gedeelte van de printer dat vellen transporteert. Verschillende componenten worden gebruikt voor de detectie, geleiding en het versnellen van vellen in het papierpad. Sensoren detecteren randen van vellen. Papierpaden kunnen een dubbelzijdig spoor en/of een keerspoor hebben om dubbelzijdig te kunnen printen. In complexe papierpaden bestaan verschillende papierroutes. De software die het papierpad aanstuurt is velgebaseerd. Hiermee wordt bedoeld dat ieder vel aangestuurd wordt door software dat een velprocedure genoemd wordt. Deze velprocedure stuurt de hardware zodanig aan dat het vel de gewenste trajectorie in tijd en plaats door het papierpad volgt.

Exceptieafhandeling beschrijft het gewenste gedrag van vellen in het papierpad wanneer er fouten optreden. Met een fout wordt bedoeld dat een rand van een vel niet op tijd door een sensor gedetecteerd is. Behalve fouten die betrekking hebben op randen die te laat zijn, bestaan er ook fouten die betrekking hebben op de scheefstand en de afmetingen van een vel. Het belangrijkste doel van exceptieafhandeling is ervoor te zorgen dat alle vellen op een veilige manier stoppen in het papierpad. Gedragseisen specificeren of vellen wel of niet dienen te stoppen op specifieke lokaties. Bijvoorbeeld, een vel mag niet stoppen in het warme gebied omdat dit een onveilige situatie tot gevolg heeft. exceptieafhandelingsoftware wordt toegevoegd aan de velprocedure. De belangrijkste nadelen van de huidige exceptieafhandeling en de ontwikkeling hiervan, zijn: het handmatig coderen van de software, het gedistribueerde besturingsconcept dat niet afgestemd is op de integrale gedragseisen, het gebrek aan herbruikbaarheid voor andere printers en het hardwaregebaseerde incrementele ontwerp van exceptieafhandeling.

Een nieuwe aanpak voor exceptieafhandeling en het ontwerp hiervan wordt voorgesteld. Deze aanpak is gebaseerd op zogenoemde signaalborden. Deze borden bestaan alleen in de software. Verschillende aanwijzingen kunnen op deze borden worden ingesteld. De

velprocedures bekijken deze borden en zorgen ervoor dat vellen zich gedragen zoals de aanwijzingen voorschrijven. De borden worden gebruikt om aanwijzingen aan vellen te geven wanneer er fouten optreden. In een opzoektabel wordt opgeslagen welke aanwijzing op welk bord dient te komen staan voor iedere fout. Een voordeel van deze aanpak is dat de specifieke stopcommando's voor exceptieafhandeling in de velprocedure worden vervangen door generieke stopcommando's via de borden.

Om de nieuwe aanpak te testen, wordt een case gebruikt. Het CIF2-formalisme wordt gebruikt om de vellen in het papierpad, de borden, de fouten en de opzoektabel voor exceptieafhandeling te modelleren. Door gebruik te maken van simulatie is allereerst gecontroleerd of vellen in het papierpad de aanwijzingen op de borden opvolgen. Daarna is simulatie gebruikt om de inhoud van de opzoektabel af te leiden. Oftewel, welke aanwijzingen op welke borden moeten komen te staan voor welke fout zodanig dat er aan de eisen ten aanzien van exceptieafhandeling wordt voldaan. Simulatie blijkt een nuttig hulpmiddel te zijn wanneer men het gedrag van alle vellen in het papierpad bij een bepaalde aanwijzing op een bord probeert te begrijpen. Voor de veertien fouten in deze case is gebleken dat de borden-aanpak in staat is voor iedere fout aanwijzingen op borden te plaatsen zodanig dat er aan de gedragseisen voldaan wordt. Het model kan gemakkelijk worden aangepast voor andere papierpaden. Door gebruik te maken van het model, kan de opzoektabel al vroeg in het ontwerpproces van een printer worden afgeleid. Deze opzoektabel kan eenvoudig in de software worden geïntegreerd.

# Contents

Assignment	iii
Preface	v
Summary	vii
Samenvatting	ix
<b>1 Introduction</b>	<b>1</b>
<b>2 Printers</b>	<b>3</b>
2.1 Hardware . . . . .	3
2.2 Paper path . . . . .	5
2.3 Software . . . . .	8
<b>3 Exception handling</b>	<b>13</b>
3.1 Errors . . . . .	13
3.2 Concept and goals . . . . .	14
3.3 Requirements . . . . .	15
3.4 Current solution . . . . .	16
3.5 Development . . . . .	16
<b>4 Signs</b>	<b>19</b>
4.1 Signs approach . . . . .	19
4.2 Software . . . . .	20
4.3 Suggested development . . . . .	23

<b>5 Case</b>	<b>25</b>
5.1 Description . . . . .	25
5.2 Derivation of exception handling . . . . .	26
5.3 Modeling . . . . .	26
5.4 Simulation . . . . .	29
5.5 Results . . . . .	32
5.6 Conclusions . . . . .	35
<b>6 Conclusions</b>	<b>37</b>
6.1 Recommendations . . . . .	37
<b>Bibliography</b>	<b>39</b>
<b>A CIF2 code</b>	<b>41</b>

# Chapter 1

## Introduction

Modern competitive markets of high-tech machines impose ever-increasing requirements on product quality, ease of use, safety, and performance. Other actors are price and time to market. Among else, this puts high pressure on the error-detection, error-handling, and failure-recovery part of the machine. In this study, high-tech Océ printers are considered, in particular the exception handling of errors in the paper path, i.e., the part of the printer responsible for transferring paper sheets from the source, to the printing process, and, finally, to the finisher.

One of the greatest challenges in multidisciplinary environments, such as the one in which high-tech printers are made, is proper communication between engineers of diverse backgrounds. The design-validation loop that involves prototyping high-tech machines requires specification and design by domain engineers, which typically communicate with software and other engineers using informal specification, to be interpreted and implemented as hardware or software. Previous investigations in this area applied supervisory control, a theory that enables automated generation of control software based on the models of the hardware and control requirements, to enable proper exception handling on the paper path [Ber09b, Ber09a]. Supervisory control enables high-level coordination and control of the discrete-event behavior of a system. One of the conclusions drawn in this work, is that formal models greatly alleviate the communication barriers between the involved parties. This holds especially for engineers that easily accommodate their reasoning to (in)formal specifications, relying on precisely defined models.

The major contribution is the definition of a formal framework that intends to capture the detailed, hybrid behavior of the paper path, and enable convenient, yet formal, specification, manipulation, simulation, and verification of the notions and properties involved. Hybrid modeling languages are used as the paper path carries both continuous and discrete behavior, where the former is not easily abstracted from. The CIF modeling framework [Bee07] is chosen and many concepts introduced in the new version of the language, referred to as CIF2 [Bee09] are used. Unfortunately, for simulation had to be

relied on the previous release of the language, referred to as CIF1.

In the proposed framework, the notion of error and exception handling are formally defined, relying on models of standard hardware concepts employed in the paper path like sensors, motors, and pinches. Also a method is proposed for exception handling, referred to as signs, which in essence present models of control signals sent to the paper sheet controllers in order to safely resolve the occurred failures. The main goal is to minimize the number of necessary service calls due to error occurring on the paper, while ensuring safe operation of the printer. For example, the toner is applied to the paper in a so-called hot zone, where high temperatures are necessary, and if paper sheets stop in the hot zone, a fire may break out. This must be prevented, ensured by both hardware devices and in the exception handling control software components, in order to prevent damage to the printer and ensure safety of the user. Some locations in the printer are easily accessible by the user, whereas in certain cases only certified service personnel can resolve the issues. To this end, the framework provides for convenient and detailed modeling and simulation of the paper path, incorporating coordination requirements for exception handling. The framework is illustrated on a study incorporating all relevant aspects of the paper path.

The rest of this thesis is organized as follows. Chapter 2 discusses the hardware and software components of the paper path and the sheet controllers. Chapter 3 overviews possible error and defines them in terms of the paper path components and their relations. Chapter 4 introduces the notions of signs and describes the control signals and the model of coordination. Chapter 5 illustrates the framework on a sample paper path that incorporates all relevant features with an accompanying simulator and visualization. Concluding remarks are given in Chapter 6 as well as directions for future research.

## Chapter 2

# Printers

Printers are machines that apply material on selected parts of a surface. A common example is a printer that applies ink to a sheet of paper. Nowadays also polymers and metals can be printed. Polymers are used in 3D printers and metal printing is used for the production of solar cells. In this report only printing on paper is discussed. The material applied to the paper usually is ink or toner. Ink is a liquid with pigment which is applied directly to the paper in small droplets. After applying the ink some time is needed for drying. Toner is a powder which is transferred to the paper via an imaging belt. Heat is used to melt the powder such that it attaches to the paper. Paper can be supplied to a printer in cut sheets or on a roll. In this report only cut sheet printers are considered.

### 2.1 Hardware

A printer can be divided into three modules as is shown in Figure 2.1. The first module is the input module (1) where sheets of paper are stored in stacks. In this module single sheets are separated from the stack. An input module can consist of multiple trays. Sheets are sent from the input module to the engine module (2). In the engine module the image is printed onto the sheet. A printed sheet is sent to the finishing module (3). This finishing module is able to perform finalizing tasks such as stapling or folding and stores the finished sheets. The modules allow a customer to compose a printer by selecting modules. The paper path is the part of the printer that transports the sheets in and between modules. In the figure, several sheets indicate the paper path. Printing on both sides, in the engine module can be achieved in different ways. The first is to equip the engine module with one location where the image is applied to the paper. To print on both sides, the paper path is extended with a duplex and turn track such that a sheet can be turned and visit the location again to be printed on the other side. The second is to equip the engine module with two locations where the image is applied to the paper. This yields a less complex paper path and a higher rate of printing sheets



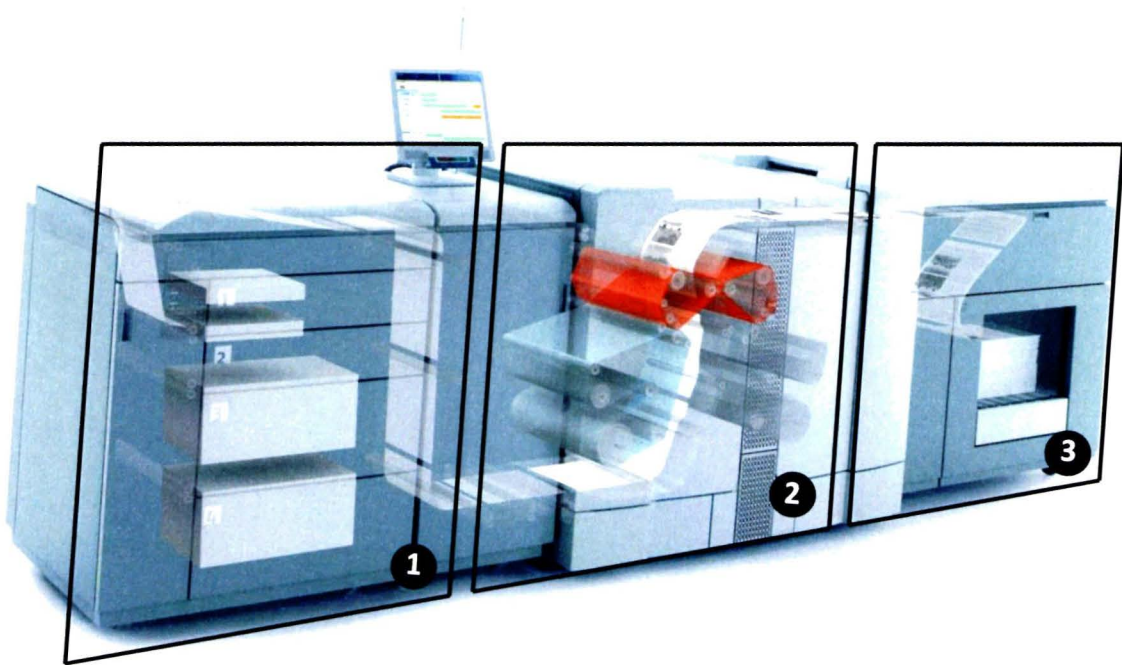


Figure 2.1: Printer with three modules.

but also a more expensive printer. In the next section the transport of paper in the paper path is explained in more detail. This paper path uses the first concept, with a duplex and turn track for printing on both sides.

## 2.2 Paper path

The paper path transports sheets in and between modules. An abstracted side view of the paper path of an example engine module (2) is shown in Figure 2.2 and consists of several *components*. The paper path is presented such that the topology is easy to understand. The corresponding real paper path has the same topology, but distances and bends may vary. As from now, only the paper path of the engine module (2) is considered. The paper path presented here, is used to explain the basics of a paper path. Another paper path is used and explained in Chapter 5 where the case is discussed.

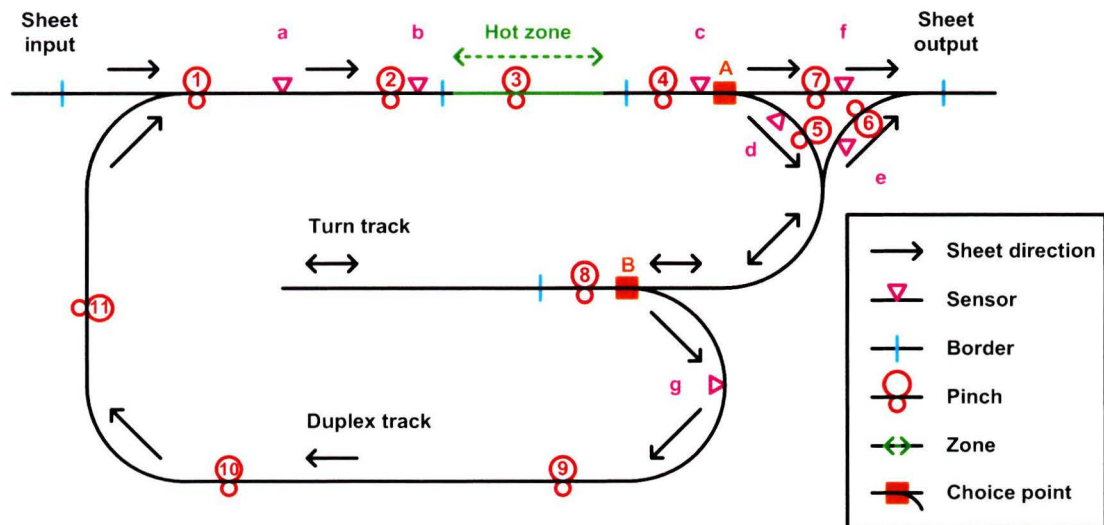


Figure 2.2: Layout of the paper path.

Within the paper path sheets are transported by *pinches*. These pinches are driven by *motors* (not shown). A single motor may drive multiple pinches. Not shown here is the metal tracks that guide sheets in the paper path. At *choice points*, a choice for what part of the paper path the sheet will continue to, is made. The *borders* indicate the location where (sub)modules slide out of the printer. A hot zone is the part of the paper path where the toner is applied to the paper. Heat is needed to attach the toner to the paper. A frot zone (not shown in Figure) is the part of the paperpath where sheets can be squeezed. The frot zone is easily accessible for a user to remove sheets. Two types of edges of a sheet are of special interest for the paper path. The leading edge (LE), which is the front edge of a moving sheet, and the trailing edge (TE) which is the back edge of moving a sheet. If a sheet reverses its direction, the leading edge

becomes trailing edge and vice versa. Within the paper path *sensors* are used to detect these edges. These sensors detect whether or not a sheet is present at the location of the sensor. The sensors are used to monitor the the position of sheets within the paper path. A priori, a timing table specifies the desired behavior for a sheet in the paper path which results in a trajectory (in time, position, velocity and acceleration). An example of a timing table is shown in Table 2.1. For different sheet sizes, different timing tables exist.

Table 2.1: Example of a timing table for a sheet

Time [s]	Position [mm]	Velocity [mm/s]	Acceleration [m/s <sup>2</sup> ]	Comment
0.0	0	500	0	Sheet enters the paper path
0.4	200	500	0	Detect LE at sensor <i>a</i>
0.5	250	500	2	Start accelerating sheet
0.7	390	900	0	Stop accelerating sheet
0.8	480	900	0	Detect LE at sensor <i>b</i>
...	...	...	...	...

As can be seen in Figure 2.2 there are two choice points in the paper path. Choice point *A* is a *switch*, where a sheet arriving from sensor *c* can go either towards sensor *d* or *f*. Choice point *B* is a fixed part of the paper path. Sheets arrive at this choice point from sensor *d* and the direction of all these sheets is reversed in the turn track. The position of the TE before reversing (LE after reversing) at the moment of the reverse determines where the sheet is transported to. A sheet completely in the turn track is always transported to the duplex track to sensor *h*. A sheet of which the TE before reversing still is in the part of the paper path before choice point *B* when it's direction is reversed, will continue to sensor *e* after the reverse. How these choice points are used, becomes clear with the explanation of the 3 possible routes through this paper path. The sensors are used to indicate the route of a sheet.

### Simplex

The sheet enters the paper path and passes sensors *a*, *b*, and *c*. The switch at choice point *A* transports the sheet towards sensor *f* and the sheet output. The sheet leaves and enters this module with the same side up (face up or face down) and is printed on one side.

### Simplex turn

The sheet enters the paper path and passes sensors *a*, *b*, and *c*. The switch at choice point *A* transports the sheet towards sensor *d* and the turn track. Before the entire sheet has passed choice point *B*, the sheet is reversed and starts traveling towards sensor *e* and the sheet output. The sheet leaves this module with the opposite side up that it entered this module with and is printed on one side.

### Duplex

The sheet enters the paper path and passes sensors *a*, *b*, and *c*. The switch at

choice point  $A$  transports the sheet towards sensor  $d$  and the turn track. After the entire sheet has passed choice point  $B$ , the sheet is reversed and starts traveling towards choice point  $B$ . At choice point  $B$  the sheet is directed towards sensor  $h$  and  $i$ . Next it again passes sensors  $a$ ,  $b$ , and  $c$ . The switch at choice point  $A$  transports the sheet towards sensor  $f$  and the sheet output. The sheet leaves and this module with the opposite side up that it entered this module with and is printed on both sides.

Now the routes of individual sheets are explained, also the scheduling of sheets can be addressed. Concerning the scheduling, the sheets enter the paper path in the same sequence as they will exit it. This means for example that a simplex sheet cannot enter the paper path later than a duplex sheet and leave it earlier. When sheets have different routes, sometimes sheets have to wait before entering the paper path, to ensure this. The minimum distance between two succeeding sheets is fixed, for example at 50 mm. The scheduling of  $n$  Simplex sheets is simple: The  $n$  sheets enter the paper path directly after each other are printed and leave the paper path. The scheduling of  $n$  Simplex turn sheets is the same. The  $n$  sheets enter the paper path directly after each other. The sheets are scheduled such that there a sheet has left the turn track before a new one comes in. The scheduling of  $n$  Duplex sheets can be done in two ways: Burst and Interleaved, as is explained below.

### **Burst mode**

Burst mode means that at most  $m$  sheets enter the paper path directly after each other. This is done in such a way that the first sheet comes out of the duplex track directly after the  $m$ -th sheet. A burst of  $m$  sheets is repeated until all  $n$  sheets are printed. The burst size  $m$  depends on the length of the paper path and the length of sheets. For different sheet sizes, a different burst size may be chosen. The timing table constructed such that no collisions occur and a high throughput of sheets is satisfied.

### **Interleaving**

Interleaving means that gaps are created between sheets when they enter the paper path. The size of a gap is such, that an entire sheet can fill this gap, taking into account the minimum distance between sheets. After sheets are printed for the first time, they fill the gaps between the sheets that are not printed yet. In which gap a sheet goes, depends on the length of the paper path and the length of sheets. For different sheet sizes, a different gap may be chosen. The timing table is constructed such that sheets fit the gaps.

Next a small example of five sheets to illustrate burst and interleaving is discussed. In table 2.2 and 2.3 the sequence of the sheets is presented at the sheet input, the hot zone and the sheet output for both duplex modes. The first pass of the hot zone is indicated with  $a$ , the second pass with  $b$ . For different routings through the paper path, separate timing tables exist.

Table 2.2: Scheduling of duplex sheets in burst mode, burst size  $m = 3$ 

Position	Sequence
Entrance	1, 2, 3, gap, gap, gap, 4, 5
Hot zone	1a, 2a, 3a, 1b, 2b, 3b, 4a, 5a, gap, 4b, 5b
Exit	1, 2, 3, gap, gap, gap, 4, 5

Table 2.3: Scheduling of duplex sheets with interleaving

Position	Sequence
Entrance	1, gap, 2, gap, 3, gap, 4, gap, 5
Hot zone	1a, gap, 2a, 1b, 3a, 2b, 4a, 3b, 5a, 4b, gap, 5b
Exit	1, gap, 2, gap, 3, gap, 4, gap, 5

## 2.3 Software

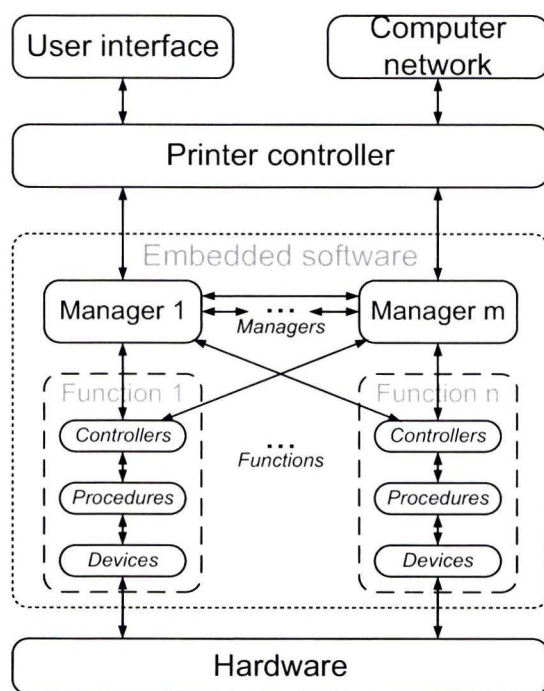


Figure 2.3: Software overview.

In this section, an overview of the software concerning the transport of sheets in the paper path and the exception handling is presented. An overview of the complete software is shown in Figure 2.3. Jobs from the users arrive either via the user interface or via the computer network. The printer controller processes these these jobs and sends a command for each page to the embedded software. The embedded software controls the hardware of the copier. Embedded software consist of multiple managers and functions.

A manager is responsible for managing a specific task of the copier, i.e. planning, energy. For each page, the managers plan a sheet in time, by this is meant that a timing table specification is added to the page command and this is distributed to the functions for execution. A function is responsible for a group of hardware components performing a particular function, i.e. the paper path, the printing process. The function executes commands from the managers. A function is connected to one or more managers. A function consists of the following components:

- **Controllers.** A controller handles request for actions, statuses and selects the appropriate procedures to execute.
- **Procedures.** A procedure defines the behavior over the devices. It controls and directs devices to perform tasks.
- **Devices.** A device controls a hardware device and is connected to the physical world through I/O ports.

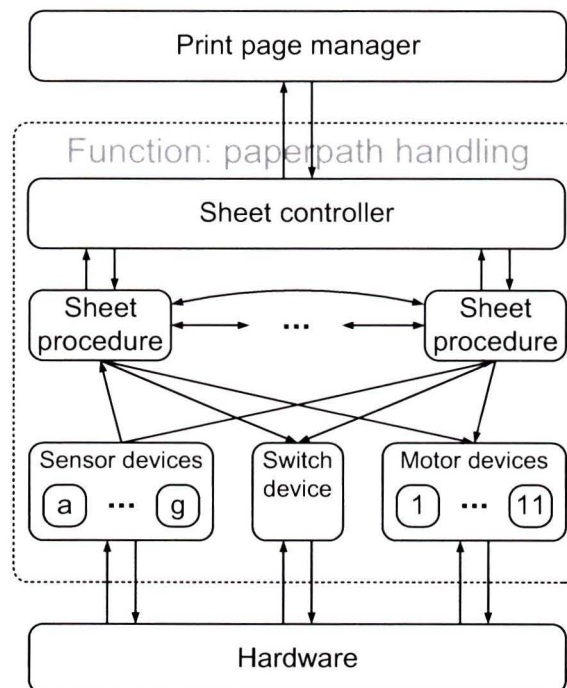


Figure 2.4: Contents and surroundings of paper path handling function.

The function of interest is the paper path handling function and in particular the sheet procedure, since the behavior is defined in a procedure. As can be seen in Figure 2.4 the paper path handling function is connected to the print page manager and to the hardware. Within this function multiple identical instantiations of the sheet procedure are present. The print page manager communicates with other functions and managers

(not shown here) to plan a sheet in time. Once a sheet is scheduled, this is communicated to the functions. The sheet controller performs this communication with the print page manager in the paper path handling function. Once the controller is informed that a sheet is scheduled, the sheet controller starts a sheet procedure. This sheet procedure guides a sheet through the paper path communicating with the devices. These communication consists of receiving sensor information and control of the switches and motors. The devices translate these communications to the hardware via I/O ports. Once a sheet has left the paper path, a sheet procedure can be reused for future sheets. Each sheet is guided through the paper path by its own sheet procedure. The sheet procedure makes sure the sheet follows the predefined timing table.

In Figure 2.5 the contents of a sheet procedure is shown. This sheet procedure is according to the layout of the paper path as presented in Figure 2.2. The sheet procedure waits for a command from sheet control. This command from sheet control contains all necessary information for the sheet procedure to have the sheet move according to the predetermined trajectory, namely the timing table. Also is known what the routing for this sheet is in the paper path. After receiving this command, the sheet procedure starts pinch 1. This is done by communicating with the device for motor 1. Assumed is that each pinch has its own motor, with the same id number. After this, the procedure waits for a LE detection at sensor  $a$ . When sensor  $a$  has detected the sheet, this is communicated via I/O to the sensor device and to the sheet procedure. After checking the skewness and format of the sheet the sheet procedure controls motor 2. The sheet procedure continues transport of its sheet through the paper path as is displayed in Figure 2.5. At the choice point, the sheet procedure selects the appropriate choice to transport the sheet such that it follows its predetermined route through the paper path and operates the switch. The three possible routings through the paper path as described in section 2.2 can be recognized in the sheet procedure. The sheet procedure consists of states. Each of the states in the figure either waits for a sensor or for a delay and then continues with sending control actions to the motors and switch.

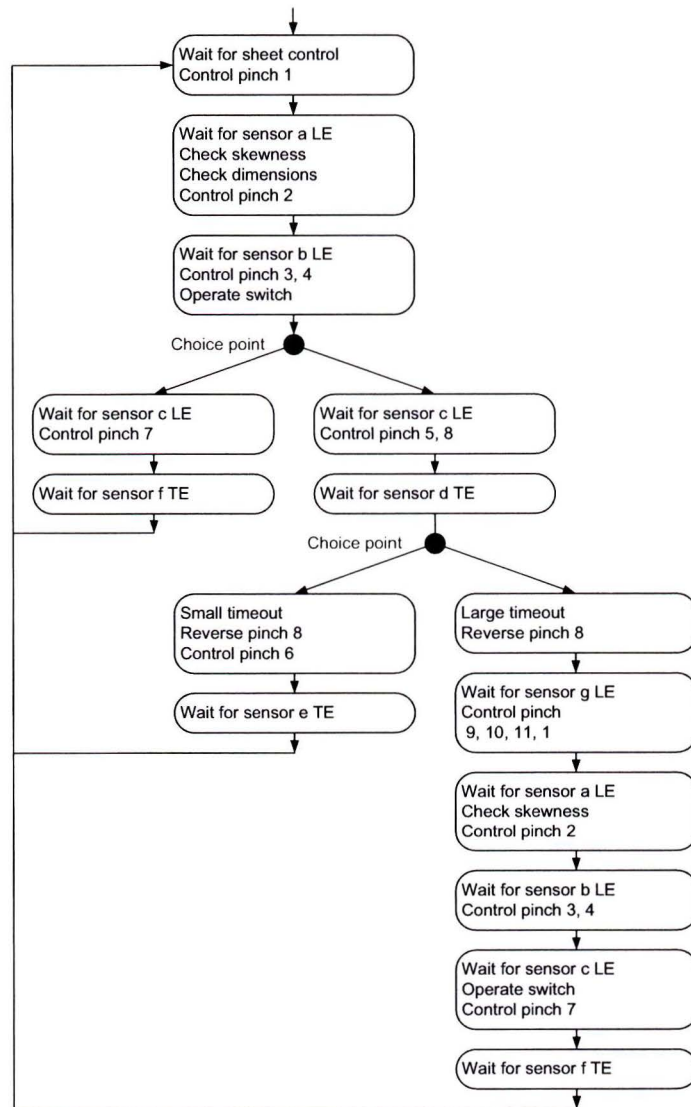


Figure 2.5: Contents of sheet procedure.





## Chapter 3

# Exception handling

Exception handling is the desired behavior of sheets in the paper path if an error has occurred. In this chapter, first is explained what an error is. After that the exception handling is discussed.

### 3.1 Errors

In the previous chapter, the desired trajectory of a sheet is specified using a timing table. For each sheet a piece of software, a sheet procedure, guides a sheet through the paper path. This sheet procedure always knows the position of its sheet. At sensor positions is checked if the edges of the sheet arrives and leaves at the predefined times. If the sheet is within a short time window around the predefined time, adjustments are made such that the exact trajectory will be satisfied again. If the sheet is outside the time window, an error has occurred. Two different errors exist for the detection of different edges at a sensor:

- **LE error**  
The time window for detecting a leading edge is passed. The LE has not been detected. The sheet has not arrived at the sensor position in time.
- **TE error**  
The time window for detecting a trailing edge is passed. The TE has not been detected. The sheet has not left the sensor position in time.

An error is reported immediately after the time window has ended. The position of the sheet in the sheet procedure might not be equal to the real position of the sheet. The cause of the error remains unknown and is not of interest for exception handling.

Besides errors with respect to the detection of edges also other errors may occur. These errors are related to the dimensions and skewness of the sheet. Groups of sensors are

used to measure the dimensions and skewness. Up to a certain skewness, special pinches are able to correct it.

- **Dimension error - too short**

The detected size of the sheet does not meet the expected size, the sheet is too short.

- **Dimension error - too long**

The detected size of the sheet does not meet the expected size, the sheet is too long.

- **Dimension error - illegal paper size**

The detected size of the sheet is so short, that it is smaller than the largest distance between two succeeding pinches. Such a sheet might get lost in the paper path.

- **Skewness error - too skew**

The skewness of a sheet is such, that it cannot be corrected anymore. The sheet is likely to jam when it is transported further into the paper path.

For the last four errors the position of the sheet in the paper path is still accurately known in the software. For all errors holds that the error sheet is marked as error sheet. This is used for the exception handling.

## 3.2 Concept and goals

In the previous section is explained what an error is and how these errors are detected. This is the **first** step of the concept. After the error is detected, the **second** step is the exception handling: A stopping procedure that stops all sheets and leaves the printer in a safe state. The **third** step is the recovery of the error where the user is helped by the user interface removes stopped sheets from the paper path. For some severe errors, this cannot be done by the user, so a service mechanic is called to fix the problem. In this report, the focus is on the second step of the concept: The exception handling. The third step, the error recovery is not addressed in this report.

If an error has occurred, the exception handling describes the behavior of the sheets in the paper path. The behavior is such, that the paper path is stopped, where the following goals are pursued. In order of priority these goals are:

- **Goal 1: Safety of customer and machine.**

Stop the paper path in such a way that no sheets are stopped in the hot area or cause damage to the paper path.

- **Goal 2: Avoid service calls.**

Stop the paper path in such a way that a user can recover the error and intervention by customer services is not needed.

- **Goal 3: Convenience use in recovery.**  
Stop the paper path in such a way that a user can recover from the error easily.
- **Goal 4: Avoid discarding of sheets.**  
Stop the paper path in such a way that correct sheets are transported to the next module if possible.

### 3.3 Requirements

Because of the unpredictability of errors and all the possible positions of several sheets in the paper path, lots of situations for exception handling exist. To deal with all these situations a generalized concept is used: Protection of particular areas in the paper path, stopping of sheets on positions where they can easily be removed by an operator, and not stopping of sheets on borders or in the hot zone. These concepts combined with the goals presented above result into specific behavior requirements for exception handling, presented below.

- **Requirement 1: Prevent sheets from stopping the hot zone**
- **Requirement 2: Stop sheets to prevent further damage in the area of the error unless they are in the hot zone**
- **Requirement 3: Prevent sheets to stop on borders unless this would result in stopping sheets in the hot zone or unless stopping is prescribed by requirement 2**
- **Requirement 4: Stop sheets with incoming skewness above a certain threshold unless they are in the hot zone**
- **Requirement 5: Stop sheets with incorrect dimensions sheet unless they are in the hot zone**

These requirements cannot be conflicting. Some of the requirements are weakened to satisfy this. It is not straightforward to determine if and where conflicts are. This means that an ideal solution might not be possible. For example a large sheet cannot be stopped while not in the hot area and preferably not on a sliding border. The solution lies in choosing the most important requirement, based on the priorities of the goals presented above. Though the requirements seem simple, due to many sheet formats, routes and scheduling possibilities the resulting behavior control is complex.

To summarize, exception handling concerns the stopping of the paper path after an error has been detected. After stopping the paper path, the exception handling concept is completed by gathering and presenting information on all remaining sheets in the paper path to the user together with instructions how to remove them. Preferably

these sheets are on locations that can be easily accessed by the user. This thesis only concerns the detection of errors and the stopping of the paper path, the manual error recovery is not taken into account.

### 3.4 Current solution

In this section the current solution for exception handling is discussed in more detail. The solution can be different for each error. The general concept of exception handling for one error of the current solution is discussed next.

- After the error has occurred the pinches near the error location are stopped, if this doesn't lead to stopping sheets in the hot zone.
- The error sheet id and which error are communicated to other sheet procedures.
- These other sheets decide for themselves based on position and information, if and what exception handling is necessarily, for example, stopping or rerouting.

Because of stopped pinches also other sheets stop and also errors on these errors are detected. These sheets also stop pinches near their error location and communicate their id to the other sheets. This leads to a cascade of errors and stopping of sheets. If possible, non-error sheets are flushed towards locations where they can easily be removed by an operator.

In Figure 3.1 the contents of the sheet procedure is expanded with the exception handling. In each state where an error is detected also the stopping of the appropriate pinches is controlled. In the Figure this is indicated by 1. The communication of the error sheets is depicted with green arrows. The incoming communication from other sheets is shown with red arrows. Based on which error, which error sheet(s) and its current position, more exception handling is specified in all states, which is indicated by 2. This specified exception handling is conditional, which means that based on available information a choice for particular exception handling is made. The specified exception handling at each number in the figure might be different, since other exception handling and conditions may apply.

### 3.5 Development

In this section the development of a printer with respect to the software of the paper path and the exception handling is discussed. The development of the paper path software is a cyclic process. The first step of this process is the derivation of timing tables, based on the design of the paper path. Happy Flow [Bec07] is a modeling tool used for this step. Using this tool, engineers manually derive the timing tables, which

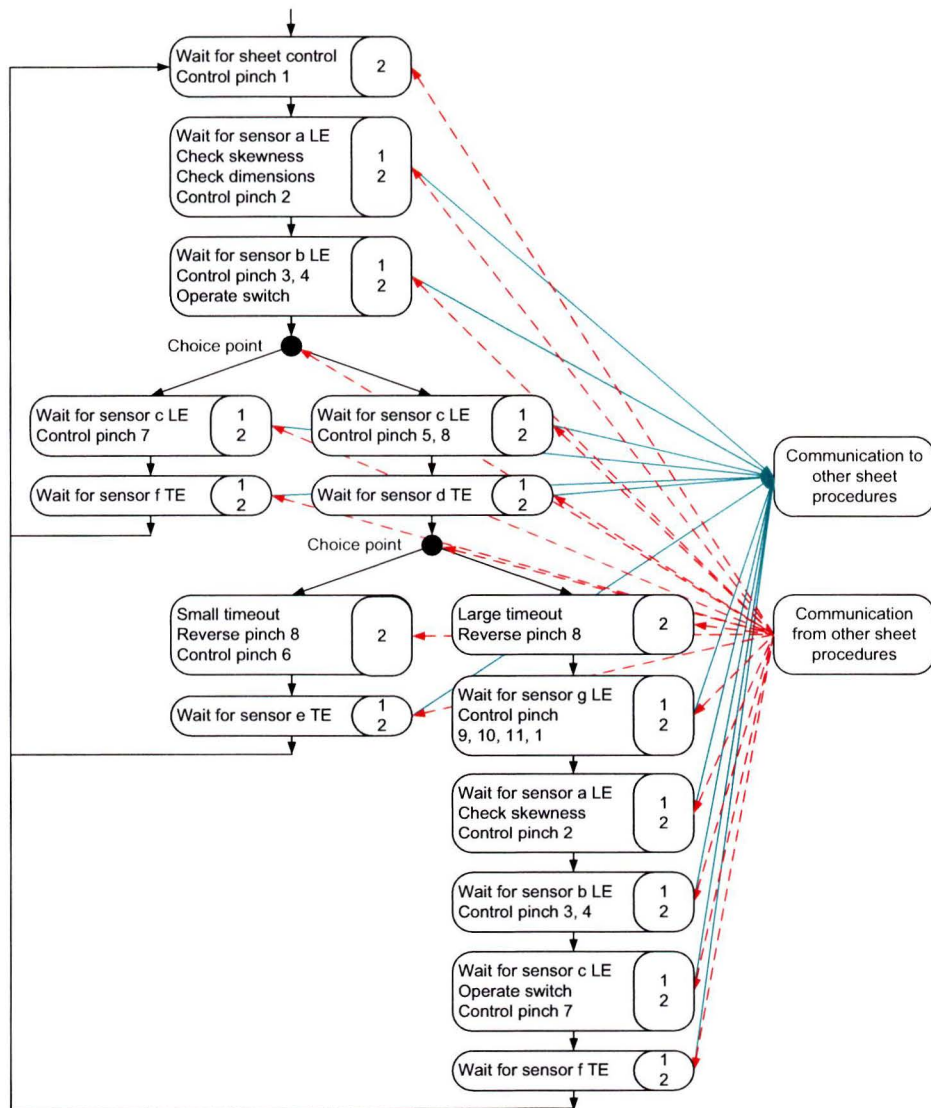


Figure 3.1: Contents of sheet procedure with exception handling.

is time consuming. The second step is the translation of the timing tables into software code (the sheet procedures). This is done by hand and is also a time consuming task. Only after the sheet procedures have been tested on a prototype of the printer, the third step, the exception handling can be derived added and tested. The third step at itself is again a cyclic process. Each of the steps defined above depend heavily on previous steps. The timing tables depend on the design of the paper path, the sheet procedure depends on the timing table and the exception handling depends on the sheet procedure.

The third step, the derivation of the exception handling consists of determining the desired behavior with respect to the requirement for the sheets in the paper path when errors occur. By hand is determined which component instructions yield the correct behavior of sheets in the paper path. These instructions are added to the sheet procedure and next is tested if these instructions yield the expected behavior and if the requirements are violated or not. Relations between these instructions and behavior of sheets are not trivial and are discovered by testing. For each error other exception handling is defined. This exception handling is conditional. This means that the exception handling may depend on the total state of the paper path, for example the position and size of sheets and the position of switches. These conditional instructions make it even harder to understand the relation between the instructions and their resulting behavior.

Because of the high number of cycles of the sheet procedure derivation and the exception handling derivation, the dependencies of the sub-processes, and the manual execution of steps, printer development is time consuming. Besides that, the development involves several engineers from different disciplines and informal documents, which makes communication difficult. The exception handling instructions are added to the sheet procedure, whereas the desired exception handling behavior transcends the sheets. Unfortunately, the derived instructions and behavior are printer specific, so they cannot be reused for other printers.

### **Problem statement**

The problem statement for which this study addresses solutions:

1. The informal behavior requirements lead to misconception and more communication loops.
2. The distributed control concept doesn't fit the integral behavior requirements.
3. The software is coded by hand.
4. The incremental exception handling design is hardware based.
5. The software is not reusable for other printers.

# Chapter 4

## Signs

A new approach for exception handling is discussed in this chapter. First, the new approach is explained. Second, the software architecture for this approach is discussed. Finally, the development of this approach and its relation to the total printer development is addressed.

### 4.1 Signs approach

The new approach for the exception handling concept is based on signs that provide instructions to sheets. These signs only exist in the software and are positioned along the paper path. Signs are always in exactly one mode and several different modes exist. Depending on the mode of the sign and the position of the edges of the sheet with respect to the position of the sign, instructions for stopping apply to the sheet. Sheets are expected to behave as the instructions prescribe. The goal of this exception handling approach is to stop sheets in the paper path using the signs in case of an error. Several modes of signs are discussed below.

- **Mode free.**

This mode means that there are no stopping instruction for sheets.

- **Mode stop.**

This mode means that there are stopping instructions for sheets. Sheets that cover the location of this sign (the LE has passed the sign location and the TE has not), are expected to stop as soon as possible. Sheets that approach this sign (the LE has not passed the sign location) are expected to stop, such that the LE stops at the position of the sign. Sheets that do not meet the location requirements, are expected not to stop.

- **Mode block.**

This mode means that there are stopping instructions for sheets. Sheets that



approach this sign (the LE has not passed the sign location) are expected to stop, such that the LE stops at the position of the sign. Sheets that do not meet the location requirements, are expected not to stop.

- **Mode frot.**

This mode means that are instructions for sheets. This mode can only be set on a sign at the end of a frot zone. When the sign is in mode frot, the part of the sheet in the frot zone is squeezed. This is done by stopping the pinch at the end of the frot zone and forcing the the pinch in the beginning of the frot zone to continue transporting the sheet.

It is possible that a sheet gets instructions from multiple signs and that these instructions are conflicting because the distance of two succeeding signs is smaller than the length of the sheet. For example, a sign instructs the sheet to continue and another sign instructs the sheet to stop. This is dealt with in the following way: Any stop instruction overrules all continue (free) instructions and also an instruction to stop as soon as possible overrules an instruction to stop on a particular location.

In addition to the signs two additional rules are defined. The first rule is that a sheet always satisfies a minimum distance with respect to the sheet directly in front of it. If the distance between two sheets is below the predefined minimum intersheet distance the backmost sheet is instructed to stop. The second rule is that a sheet always stops immediately if the sheet directly in front of the sheet is an error sheet. Which sheets are error sheets is detected in the first step of the exception handling concept and this information is assumed to be shared between all sheets.

In order to solve the stopping of sheets at locations they are not supposed to stop (for example a hot zone), all stop instructions are neglected when any edge of a sheet is within such a zone. This behavior is included in the sheet procedure and does not influence the genericity of this approach. If this exception is made, is based on the priority of goals and requirements (Chapter 3). In this approach instructions are based on sheets. The sheets are assumed to control pinches such that these instructions are satisfied. The rules presented above form the base for the new approach for exception handling. This base is suitable for extensions, for example additional modes for signs, or conditional instructions based on sheet length.

## 4.2 Software

In this section is discussed how the approach presented above can be transformed into software. First the communication between sheet procedures and the signs is discussed. In Figure 4.1 the paper path handling function is shown. Previously the procedures consisted only of sheet procedures. Now also the signs are present. Sheet procedures communicate which error has occurred to the signs. The signs are put in the appropriate

mode for each error based on a lookup table. A lookup table holds information on what mode to put on which sign for which error. A small example of a lookup table is presented in Table 4.2. The mode of the signs is communicated to the sheet procedures.

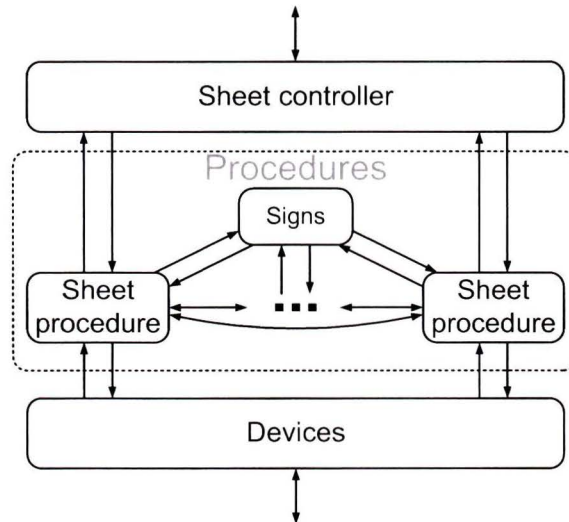


Figure 4.1: Contents of paper path handling function.

Table 4.1: Example of a part of a lookup table

Error \ Sign	1	2	3	4	5	...
1	block	stop				...
2	block		block	stop		...
3	block		block	stop	stop	...
4	block		block	block	stop	...
...	...	...	...	...	...	...

In the example lookup Table, 4 errors and 6 signs are considered. When error 1 occurs, the lookup table sets sign 1 in mode block and sign 2 in mode stop. The other signs are not changed. When looking in columns can be seen which modes occur for each sign. When multiple errors occur, a conflict between what mode to set a sign may appear. This is solved by defining mode stop as the command with priority over mode block. For example if first error 3 and next error 4 occurs, sign 4 first will be set to mode stop and next it will stay in mode stop. If first error 4 and next error 3 occurs, sign 4 first will be set to mode block and next to mode stop.

In Figure 4.2 the sheet procedure adapted to the signs approach is presented. Most states of the sheet procedure hold error detection. Which error and which error sheet is directly communicated to the other sheet procedures and the lookup table. As discussed above, the lookup table sets the appropriate modes on signs. The mode of signs is

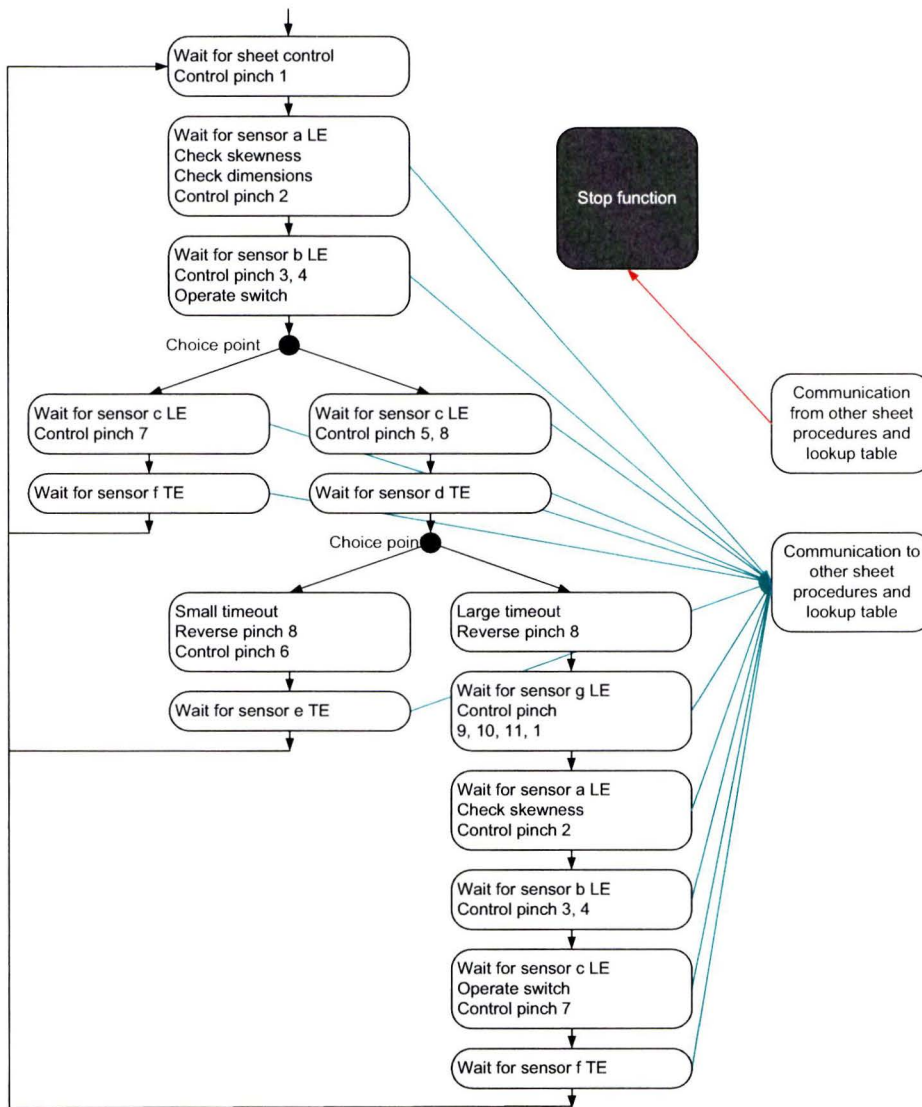


Figure 4.2: Contents of sheet procedure with exception handling using signs approach.

communicated to the sheet procedures. Each sheet procedure now contains a stop function. This function determines based on available information if the sheet should stop or not. The specific conditions for stopping based on this information is discussed in the previous section. If a sheet should stop or not is based on:

- The position of the edges of the sheet
- The length of the sheet
- The position of the signs
- The mode of the signs
- Which sheet is directly in front of this sheet
- The position of the TE of the sheet directly in front of this sheet

The stop function decides if a sheet should stop as soon as possible or at a particular location, and communicates this to all states in the sheet procedure. These states are able to deal with stop requests and control pinches such that the sheet stops appropriately.

### 4.3 Suggested development

Using the new approach and the software implementation as suggested above the development of the exception handling concept is, to determine the position of the signs and in which mode to put signs when which error occurs. A particular solution can be tested on a prototype version of a printer with the sheet procedure adapted as described in the previous section. If the behavior of the sheets in the paper path is not as desired, the positions of the signs and the lookup table have to be changed such that the desired behavior is met. Advantages of this new approach are that the signs approach is easy to understand. Furthermore the development cycle is short. It is easy to make changes to the lookup table and/or the positions of sign in order to satisfy the desired behavior.

The signs approach is less sensitive for changes in timing of sheets or in the design of the printer. This is because of the signs approach is generic, flexible and simple. Once the sheet procedure has been adapted for the signs approach, quick development cycles are used to derive the appropriate exception handling behavior. Also this approach is suitable for reuse in other printers. The sheet procedures now long have specific but generic control for stopping sheets.



# Chapter 5

## Case

In this chapter the signs approach is tested on a case. To do so, a paper path and the new approach for exception handling have been modeled in CIF. In the first section the case is explained. Next, the models are discussed. These models are used for simulation. The results of the simulations are shown in section 5.5. Finally some conclusions are presented.

### 5.1 Description

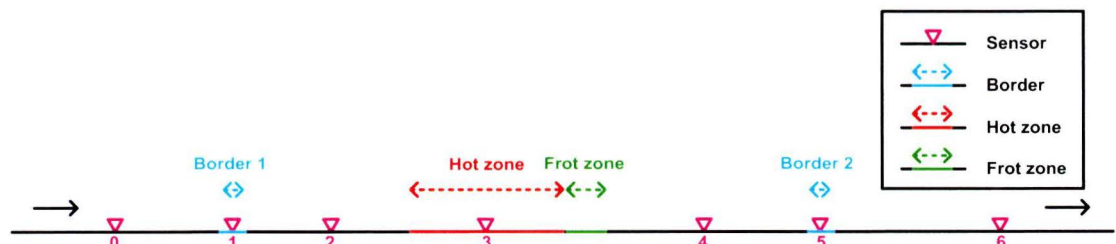


Figure 5.1: Layout of the paper path.

In Figure 5.1 the paper path used in this case is shown. This paper path has seven sensors, two borders, a hot zone, and a frot zone. There are no branches in this paper path, so only one route through the paper path is possible. Pinches are not shown in the Figure. Assumed is that sheet procedures always can control the position and speed of sheets. In this case fourteen errors are considered. Two errors for the two edges for all seven sensors. The odd error numbers correspond with the LE errors, the even error numbers correspond with TE errors. Errors concerning wrong dimensions and skewness are not taken into account. The goal of this case is to determine where signs should be located and in which mode these signs should be set in case of which error, such that the requirements for exception handling (see section 3.3) are satisfied.

## 5.2 Derivation of exception handling

In this section is explained how the exception handling is derived. For each of the fourteen errors the location of signs and in what mode to set them if that error occurs is selected. Next this exception handling is implemented in the model (discussed in the next section) and tested is if the behavior of the sheets does not violate the requirements. Most likely not all errors result in the desired behavior, so changing the signs and using the model to test the behavior is repeated until the behavior is satisfactory.

A total of ten signs is used, six at the locations of the sensors (except sensor 3), three at the entrances of the borders and the hot zone, and one sign at the end of the frot zone. The paper path with the signs is shown in Figure 5.2.

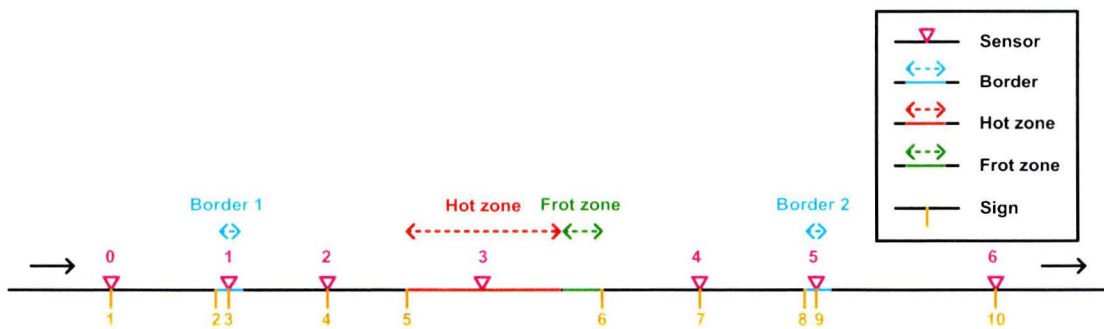


Figure 5.2: Layout of the paper path with signs.

The setting of modes on signs in case of errors is presented in the lookup table. In this table the fourteen errors and ten signs are shown. This table is the result of several iterations. For signs 2, 5, and 8 at the beginning of the zones hold that these signs are set in mode block when the location of the error is after the location of these signs. This also holds for sign 6 at the end of the frot zone. Signs 1, 3, 4, 7, 9, 10 are placed at sensor locations. The sign at the sensor that detected the error is put in mode stop. Besides the signs also the stopping of sheets in case its preceding sheet is stopped or is an error sheets contribute to the correct stopping behavior.

## 5.3 Modeling

The CIF2 language is used to model the case described above. The complete syntax of the CIF2 modeling language is specified in railroad diagrams [Hen10]. The most important elements are discussed automata, locations and edges. A CIF model consists of parallel automata. Each automaton consists of at least one location. Edges connect locations within an automaton. A CIF model in general also holds actions and variables. Which automata control variables is modeled explicitly in CIF.

Table 5.1: Lookup table

Error \ Sign	1	2	3	4	5	6	7	8	9	10
1	stop									
2	stop									
3		block	stop							
4		block	stop							
5		block		stop						
6		block		stop						
7		block			block	frot				
8		block			block	frot				
9		block			block	frot	stop			
10		block			block	frot	stop			
11		block			block	frot		block	stop	
12		block			block	frot		block	stop	
13		block			block	frot		block		stop
14		block			block	frot		block		stop



Figure 5.3: Toolchain.



In Figure 5.3 the toolchain used in this thesis is shown. Models of the paper path are written in CIF2 ASCII. CIF2 ASCII has a user friendly and easy to understand syntax. Though a simulator for CIF2 is under construction, it is not ready yet at the time this thesis was written, so the CIF1 simulator is used. In order to use this simulator, first the CIF2 ASCII model is transformed to a CIF1 ASCII model. This model transformation is one of the translations between two languages, in this case CIF2 and CIF1. The `cifascii2.cifascii1` command performs this translation and also linearizes the model. In this context linearizing means that the model is reduced to a single automaton with a single location. How this is done is described in a.o. the theses of Engels [Eng10] and Fonteijn [Fon11]. The next step of the toolchain is to compile the CIF1 ASCII model to a CIF1 core model using the `cifc` command. A CIF1 core model can be used for simulation.

In Appendix A the full CIF model for this case including the derived exception handling is shown. In this section this model is explained step by step. Values for length or size are in centimeters.

## Model: Paper path

The model consists of seven parallel automata. Four automata **PaperX** each represent one sheet in the paper path. The automaton **Setsigns** holds the lookup table and sets the signs in the appropriate mode when an error has occurred. Automaton **Sequence** makes sure the sheets start in the right sequence and also obey the intersheet starting distance. Automaton **Usecase** is used to change the settings at the beginning of a simulation and determines based on these settings when errors occur and are detected. These seven automata share several actions and variables. In the Appendix the model declaration is at lines 14 to 39.

## Automaton: PaperX

The **PaperX** automaton models one sheet in the paper path. Within the automaton is checked if the sheet should be frotted, stopped or started and also if it is in the hot zone. Within these checks also the modes of the signs are incorporated. Based on the checks the automaton sets the speed of the sheet. As time progresses, the position of the sheet increases, if it has a positive speed. The derivative of the position of the LE of the sheet is the speed. The position of the trailing edge of a sheet is equal to the position of the LE minus the length of the sheet. The sheets are assumed to be able to start and stop immediately. In the Appendix the **PaperX** automaton is at lines 41 to 99.

### **Automaton: Setsigns**

The **Setsigns** automaton sets the signs when an error has occurred. Based on which error has occurred the automaton sets the signs according to the lookup table. The lookup table is incorporated in this automaton. The variables representing the signs are controlled by this automaton. In the Appendix the **Setsigns** automaton is at lines 101 to 130.

### **Automaton: Sequence**

The **Sequence** automaton determines the sequence in which the four sheets are started. Furthermore this automaton makes sure that the intersheet starting distance between sheets is realized. The intersheet starting distance is the distance between the TE of the preceding sheet and the LE of the current sheet. In the Appendix the **Sequence** automaton is at lines 132 to 144.

### **Automaton: Usecase**

The **Usecase** automaton is used to set all kinds of parameters for simulation convenience. These parameters are: The size of the sheets, The intersheet starting distance, The sensor at which an error is going to occur, The edge at which the error is going to occur, and The jamming distance. The jamming distance is the distance between the jamming position of the edge that is supposed to jam and the position of the sensor that is going to detect the error. Three predefined sets of simulation parameters can be selected using the UC actions. The **Usecase** automaton also controls the occurrence and detection of the error. In the Appendix the **Usecase** automaton is at lines 146 to 204.

Within the model, the position of the sheet according to the software is modeled. After an error has occurred it takes some time before the error will be detected. At the moment the error occurs the position of the sheet is remembered in the model. Since the error is not detected the software and the model assume the sheet is still moving. The moment of detection of the error is when the edge of the error sheet should have been detected at the sensor. Assumed is that sheets can start and stop instantaneously.

## **5.4 Simulation**

The model as explained in the previous section is now ready for simulation. The CIF1 simulator is called using the `simcif` command. This simulator has an automatic and manual mode. In manual mode the simulator writes to the screen which transitions in the model are allowed. The user can chose one of these transitions by selecting the corresponding number on the keyboard. This transition is executed by the simulator.

The process of displaying all possible transitions is repeated. The user can the manual mode to step through the model. In automatic mode choices are automatically made by the simulator. The simulator has built-in functionality for plotting graphs and visualizing automata. This simulator is extended with a button panel and a visualization as is shown in Figure 5.4.

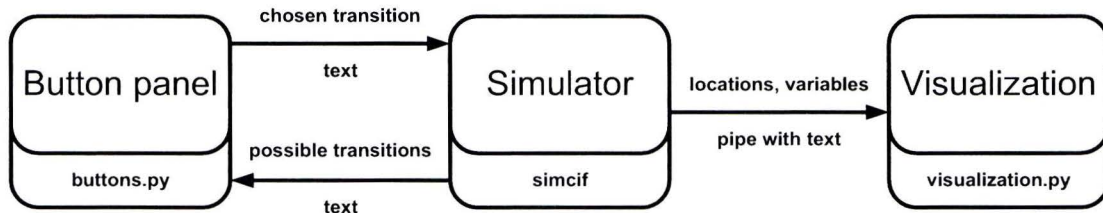


Figure 5.4: Simulation.

In the middle the CIF1 simulator is shown. Via a pipe, the simulator sends the states of the automata and the name and value of all variables to the visualization. The visualization is generated by a python script that parses the textual input. Based on the information from the pipe a figure is drawn. For different values of parameters, items in the figure are drawn on other positions or in other colors. The visualization helps to understand the current state of the system, especially if the user doesn't have to know the meaning of all states and variables. Another addition to the simulator is the button panel, generated by another python script. This python script receives the output from the simulator which usually is written to the screen, see Figure 5.5. For each transition in the model a button is present on the button panel. The buttons of all transitions that are allowed are enabled, the other buttons are disabled. The button panel with enabled and disabled button is presented on screen to the user. The user can select one of the enabled buttons. The appropriate transition is selected and this is communicated from the button panel to the simulator. The button panel provides a clear overview of all transitions and which of these are enabled. The button panel also has functionality for the automatic selection of transitions. The automatic mode can be switched on and off with a checkbox. A set of transitions is selected for which the automatic mode applies. These selected transitions are executed automatically by the button panel when the automatic mode is turned on. The button panel provides the user a convenient way of stepping through the model.

The first column holds all transitions to configure the simulation. The first two are used to increase and decrease the size of all four sheets. The third and fourth button are used to increase and decrease the intersheet starting distance. The fifth and sixth button are used to select at which sensor the error occurs. The seventh button is used to select at which edge of the second sheet the error occurs. The eighth and ninth button are used to increase and decrease the distance between the occurrence of the error and the detection of the error. The three use case buttons are used to select preconfigured sets of initial settings. The 'Start' button confirms all initial settings. The 'Again'

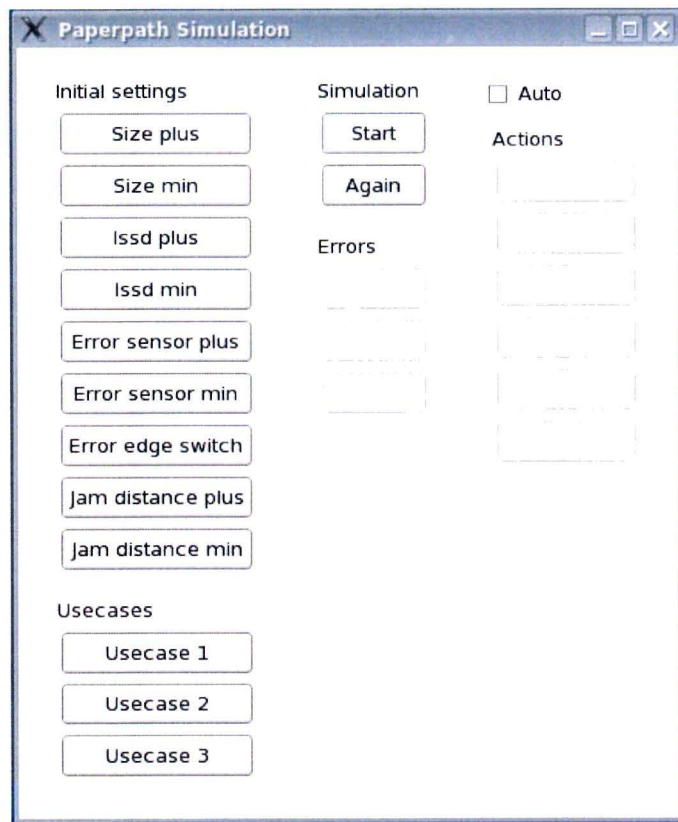


Figure 5.5: Screenshot of button panel for the case.

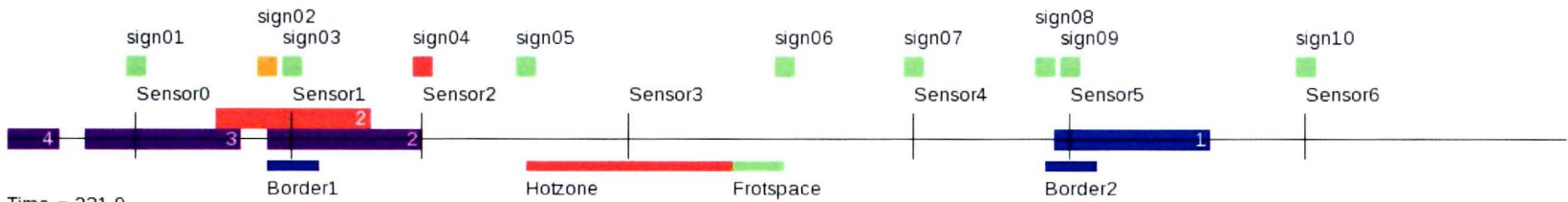
button resets the simulation. The 'Jam' button is used to simulate the occurrence of the error. The 'Detect' button means the detection of the error and the 'Set signs' button sets the appropriate modes on the signs after an error has been detected. The third column holds all transitions that occur automatically when the auto checkbox is checked. A 'tau' transition is a transition with no specific meaning. This is the result of transforming from CIF2 to CIF1. The 'start sheet' buttons are used to start the corresponding sheets. The 'Time step' button makes the simulation do a time step.

In Figure 5.6 a screenshot of the visualization is shown. In the figure, the ten signs, seven sensor, borders, hot zone, and frot zone are shown. The color of the signs corresponds to the mode it is in. Green = free, Red = stop, Orange = block, and Blue = frot. Moving sheets are displayed in blue, stopped sheets are displayed in purple. The location at which sheet 2 jammed is displayed in red, just above the paper path. In the text below the paper path the settings are displayed. From these setting can be concluded that for the case showed in figure the LE of sheet 2 jammed 10 cm before sensor 2 (shown in red). The detection of the error occurred when the LE of sheet 2 reached sensor 2. At that moment in time the signs are set. Sheet 2 stopped because sign 4 tells it to do so. Sheet 3 is stopped because error 2 is the error sheet. Sheet 4 is stopped because it maintains a minimum intersheet distance of 5 cm to sheet 3. Sheet 1 received no instructions from signs or other sheets so transport is continued.

## 5.5 Results

In this section, the results of one use case are discussed. The screenshot of the visualization in the previous section is taken at time 231.9. The visualization is a movie in which papers run trough the paper path. This movie gives a good insight in how the sheets behave. In this report this is represented by a graph where the position of the LE of the sheets is plotted against simulated time.

In Figure 5.7 the position of the LE of four sheets is shown as time progresses in the model. At time 0 sheet 1 is started. At time 35 sheet 2 is started. This is because the length of the sheets is 30 cm and the intersheet starting distance is 5 cm. Sheet 3 and 4 are started on times 70 and 105 respectively. The LE of sheet 2 jams 10 cm before sensor 2. The position of sensor 2 is at 80 cm and sheet 2 is started at time 35 so this is at time  $35 + 80 - 10 = 105$ . The jam is detected by sensor 2 at time 115. Also at this time the signs and error sheet are set. Sheet 2 is stopped by sign 4 which was set to mode stop. Sheet 3 is stopped because sheet 2 is the error sheet. Sheet 4 stops because it maintains the minimum intersheet distance. In this use case the intersheet starting distance is equal to the intersheet distance so the succeeding sheet stops immediately if its predecessor stops. Sheet 1 is not stopped by any signs or predecessors so it continues and leaves the paper path at time 330. There are no sheets stopped in the hot zone, but sheets are stopped on border 1. This is because stopping error sheets is more important then clearing the border.



Time = 231.9  
 Sheet size = 30.0  
 Intersheet starting distance = 5.0  
 Selected sensor for error = 2  
 Selected edge for error = "LE"  
 Jamming distance = 10.0

Figure 5.6: Screenshot of the visualization of the paper path of the case.

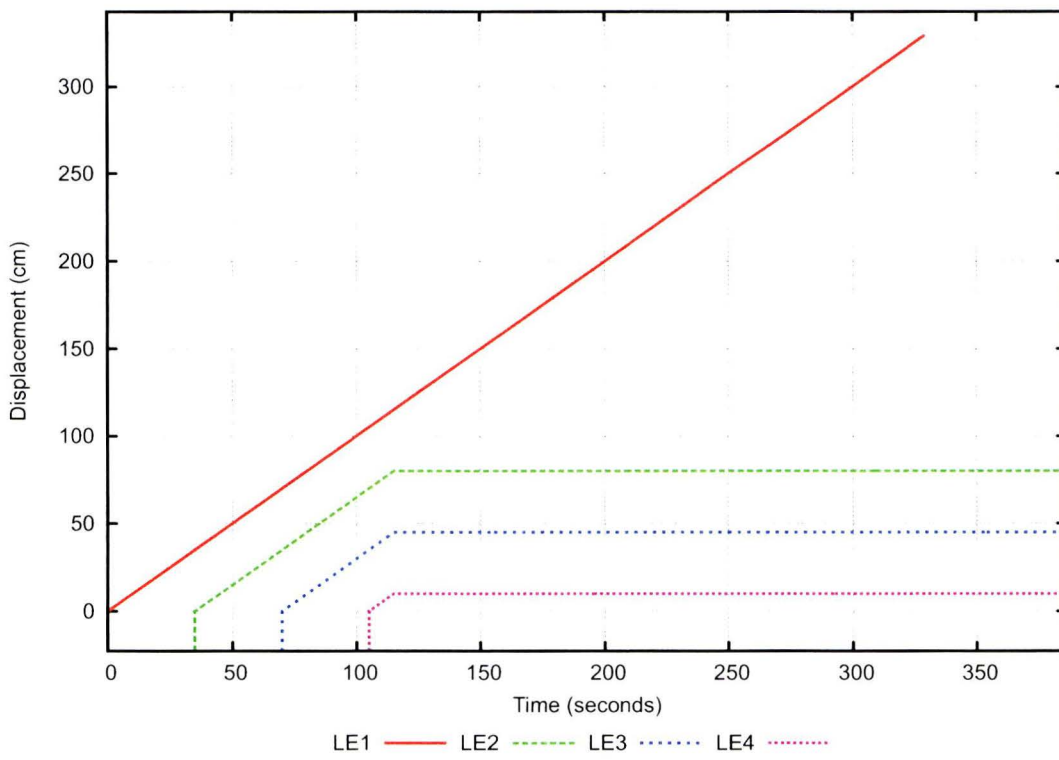


Figure 5.7: Positions of the LE edges of 4 sheets.

Simulations were performed for all errors with different settings of the parameters. If the behavior of the sheets violated the requirements, the exception handling was adapted. For the current exception handling settings no violation of requirements was observed for lots of different parameter settings. There is no proof that there does not exist a set of parameter settings for which the requirements are violated.

## **5.6 Conclusions**

The model of the paper path is suitable for determining exception handling and to check using the visualization if the requirements are violated. The visualization is a very useful way of communicating the behavior of sheets. The modeling approach is suitable for modeling different paper paths. The model can be expanded for paper paths with branches and/or loops, also the starting and stopping of the sheets can be added in more detail.

The most important advantage of the model is that now a tool is available for exception handling. This decouples the dependence of the development of the exception handling from all other development process of the printer. The exception handling derived using this model can easily be implemented in the software.

The model and simulation presented above were used to determine the lookup table presented in Section 5.2.





# Chapter 6

## Conclusions

A formal framework is proposed for modeling of errors and exception handling in high-tech machines. The framework is based on the hybrid modeling language CIF and the accompanying toolset. The approach is illustrated by modeling error handling of a paper path that incorporates all relevant aspects of a high-tech printer. A simulation and visualization environment has been developed on which several possible scenarios are tested. Advantages of the new approach are proper communication, the possibility to derive exception handling earlier in the development process of a printer, the reusability for other printers, and that the implementation of the approach is generic instead of the current specific approach.

### 6.1 Recommendations

The framework proved versatile and up to the challenge. Still, there are features that require further attention:

- The model of the paper path comprises a fixed number of paper sheets. A more general approach is needed. The approach was partly dictated by the older version of the simulator that has been employed.
- The paper path that is considered is linear. To be able to conveniently model branching and looping paper path, some advanced modeling features are necessary. One option is to employ supervisory control to coordination the normal flow of paper sheets.
- More detailed modeling of the motors and pinches can be employed that consider also acceleration/deceleration of sheets.
- The model can be translated to other languages in order to perform automatic verification.

- The dynamics of the paper sheets can be directly deduced from the timing tables, instead of having to model it explicitly using differential equations.
- Errors are currently introduced by nondeterministic labeled transitions. One can also model uncertainty in the timing and detect the errors on the fly.
- In the current model is abstracted from the behavior of frotting of sheets.
- The toolchain can be upgraded to work with the CIF2 simulator.

# Bibliography

- [Bec07] J.M.J. Beckers, W.P.M.H. Heemels, B.H.M. Bukkems, and G.J. Muller. Effective industrial modeling for high-tech systems: The example of happyflow. In *Proceedings of the 17th Annual International Symposium of the International Council On Systems Engineering, INCOSE*, 2007.
- [Bee07] D.A. van Beek, M.A. Reniers, R.R.H. Schiffelers, and J.E. Rooda. Foundations of an interchange format for hybrid systems. In Alberto Bemporad, Antonio Bicchi, and Giorgio Butazzo, editors, *Hybrid Systems: Computation and Control, 10th International Workshop*, volume 4416 of *LNCS*, pages 587–600, Pisa, 2007. springer.
- [Bee09] D.A. van Beek, P. Collins, D.E. Nadales, J.E. Rooda, and R.R.H. Schiffelers. New concepts in the abstract format of the compositional interchange format. In A. Giua, C. Mahuela, M. Silva, and J. Zaytoon, editors, *3rd IFAC Conference on Analysis and Design of Hybrid Systems*, pages 250–255, Zaragoza, Spain, 2009.
- [Ber09a] E.L.G. Bertens. Supervisory control theory applied to exception handling in Océ printers. Master’s thesis, “Eindhoven University of Technology, Department of Mechanical Engineering”, 2009. SE 420580.
- [Ber09b] E.L.G. Bertens, R. Fabel, M. Petreczky, D.A. van Beek, and J.E. Rooda. Supervisory control synthesis for exception handling in printers. In *In Proceedings Philips Conference on Applications of Control Technology*, 2009.
- [Eng10] T.P. Engels. CIF to CIF model transformations. Master’s thesis, “Eindhoven University of Technology, Department of Mechanical Engineering”, 2010. SE 420621.
- [Fon11] J. Fonteijn. And/or superstate refinement in hierarchical compositional interchange format. Master’s thesis, “Eindhoven University of Technology, Department of Mechanical Engineering”, 2011. SE 420640.
- [Hen10] D. Hendriks, A.T. Hofkamp, and P.A.H. Thijs. Cif2.1.1 railroad diagram, 2010.



# Appendix A

## CIF2 code

```
1  const array(7,real) sensorpos = array ( 25.0, 55.0, 80.0, 120.0
2      , 175.0, 205.0, 250.0 )
3      ; real sign01pos = 25.0      , border1pos = 50.0      , pppsize = 300.0
4      , sign02pos = 50.0      , border1size = 10.0      , md = 0.1
5      , sign03pos = 55.0      , hzpos = 100.0      , misd = 5.0
6      , sign04pos = 80.0      , hzsize = 40.0
7      , sign05pos = 100.0      , frotpos = 150.0
8      , sign06pos = 150.0      , frotsize = 10.0
9      , sign07pos = 175.0      , border2pos = 200.0
10     , sign08pos = 200.0      , border2size = 10.0
11     , sign09pos = 205.0
12     , sign10pos = 250.0
13
14  model Paperpath () =
15  |[ act      P1, P2 ,P3, P4, SizePlus , SizeMin, IssdPlus , IssdMin, EsPlus
16      ,      EsMin, EeSwitch , JamdPlus, JamdMin, UC1, UC2, UC3, Start , Again
17      ,      Jam, Detect , SetSign
18  ; disc real  L = 30.0, issd = 5.0
19  ; disc nat   error = 0, fv = 5
20  ; var real   TE1, TE2, TE3, TE4
21  ; disc string sign01 = "free", sign02 = "free", sign03 = "free"
22      , sign04 = "free", sign05 = "free", sign06 = "free"
23      , sign07 = "free", sign08 = "free", sign09 = "free"
24      , sign10 = "free"
25  :: PaperX   ( P1, Again, L, issd, TE1, TE4, 1, fv, sign01, sign02, sign03
26      , sign04, sign05, sign06, sign07, sign08, sign09, sign10 )
27  || PaperX   ( P2, Again, L, issd, TE2, TE1, 2, fv, sign01, sign02, sign03
28      , sign04, sign05, sign06, sign07, sign08, sign09, sign10 )
29  || PaperX   ( P3, Again, L, issd, TE3, TE2, 3, fv, sign01, sign02, sign03
30      , sign04, sign05, sign06, sign07, sign08, sign09, sign10 )
31  || PaperX   ( P4, Again, L, issd, TE4, TE3, 4, fv, sign01, sign02, sign03
32      , sign04, sign05, sign06, sign07, sign08, sign09, sign10 )
33  || Setsigns ( SetSign, Again, error, sign01, sign02, sign03, sign04, sign05
34      , sign06, sign07, sign08, sign09, sign10 )
35  || Sequence ( P1, P2, P3, P4, Again, issd, TE1, TE2, TE3, TE4 )
36  || Usecase   ( L, issd, error, fv, TE2, SizePlus, SizeMin, IssdPlus, IssdMin
```

```

37         , EsPlus , EsMin , EeSwitch , JamdPlus , JamdMin , UC1 , UC2 , UC3 , Start
38         , P1 , Again , Jam , Detect )
39 ]|
40
41 automaton PaperX ( inout act sync    Px , Again
42                   ; inout disc real  L , issd
43                   ; inout var real   TE , TEx
44                   ; val  nat          id
45                   ; inout disc nat    fv
46                   ; inout disc string sign01 , sign02 , sign03 , sign04 , sign05
47                   , sign06 , sign07 , sign08 , sign09 , sign10
48                   ) =
49 |( cont control real LE = -100.0
50   ; disc control real v = 0.0
51   ; disc control bool a = true , b = true , c = true , d = true
52   ; var bool          frot , stop , inhz
53   ; mode X =
54     initial
55     inv LE' = v
56     , TE    = LE - L
57     , frot = // Frot if LE is at or in frot pinch the sign is in mode frot
58             ( frotpos - md < LE and TE < frotpos and sign06 = "frot" )
59
60     , inhz = // Check if sheet is in hz
61             ( hzpos < LE and TE < hzpos + hzsize )
62
63     , stop = // Stop at the sign position if the sign is in mode block
64             ( sign02pos - md < LE and LE < sign02pos and sign02 = "block"
65             or sign05pos - md < LE and LE < sign05pos and sign05 = "block"
66             or sign08pos - md < LE and LE < sign08pos and sign08 = "block"
67
68             // Stop if the sheet covers the sign and the sign is in mode stop
69             or TE < sign01pos + md and sign01pos - md < LE and sign01 = "stop"
70             or TE < sign03pos + md and sign03pos - md < LE and sign03 = "stop"
71             or TE < sign04pos + md and sign04pos - md < LE and sign04 = "stop"
72             or TE < sign07pos + md and sign07pos - md < LE and sign07 = "stop"
73             or TE < sign09pos + md and sign09pos - md < LE and sign09 = "stop"
74             or TE < sign10pos + md and sign10pos - md < LE and sign10 = "stop"
75
76             // Stop if the sheet comes to close to the preceding sheet
77             or TEx - misd < LE and LE < TEx
78
79             // Stop if the preceding sheet is an error sheet
80             or fv = id - 1
81             )
82     // Starting and stopping of sheet
83     ( when LE < 0.0 and v = 0.0 act Px do LE , v := 0.0 , 1.0 )
84     ( when v > 0.0 and stop and not inhz now do v := 0.0 )
85     ( when v = 0.0 and LE >= 0.0 and not stop now do v := 1.0 )
86     ( when TE >= pppsize now do LE , v := -100.0 , 0.0 )
87     ( act Again do LE , v := -100.0 , 0.0 )
88     ( when frot now do LE , v := -100.0 , 0.0 )
89
90     // Rules for proper behavior in simulator

```

```

91     ( when      a and issd          < TE now do a := not a )
92     ( when not a and issd          > TE now do a := not a )
93     ( when      b and sign02pos - md < LE now do b := not b )
94     ( when not b and sign02pos - md > LE now do b := not b )
95     ( when      c and sign05pos - md < LE now do c := not c )
96     ( when not c and sign05pos - md > LE now do c := not c )
97     ( when      d and sign08pos - md < LE now do d := not d )
98     ( when not d and sign08pos - md > LE now do d := not d ) goto X
99 )|
100
101 automaton Setsigns ( inout act sync          SetSign , Again
102                    ; inout disc nat         error
103                    ; inout disc control string sign01 , sign02 , sign03 , sign04
104                    , sign05 , sign06 , sign07 , sign08
105                    , sign09 , sign10
106
107                    ) =
108 |( disc control bool done = false
109 ; mode X =
110   initial
111   ( act Again do done , sign01 , sign02 , sign03 , sign04 , sign05 , sign06 , sign07
112     , sign08 , sign09 , sign10 := false , "free" , "free" , "free"
113     , "free" , "free" , "free" , "free" , "free" , "free" , "free" )
114   ( when not done and ( error = 1 or error = 2 ) now act SetSign do done
115     , sign01 := true , "stop" )
116   ( when not done and ( error = 3 or error = 4 ) now act SetSign do done
117     , sign02 , sign03 := true , "block" , "stop" )
118   ( when not done and ( error = 5 or error = 6 ) now act SetSign do done
119     , sign02 , sign04 := true , "block" , "stop" )
120   ( when not done and ( error = 7 or error = 8 ) now act SetSign do done
121     , sign02 , sign05 , sign06 := true , "block" , "block" , "frot" )
122   ( when not done and ( error = 9 or error = 10 ) now act SetSign do done
123     , sign02 , sign05 , sign06 , sign07 := true , "block" , "block" , "frot" , "stop" )
124   ( when not done and ( error = 11 or error = 12 ) now act SetSign do done
125     , sign02 , sign05 , sign06 , sign08 , sign09 := true , "block" , "block" , "frot"
126     , "block" , "stop" )
127   ( when not done and ( error = 13 or error = 14 ) now act SetSign do done
128     , sign02 , sign05 , sign06 , sign08 , sign10 := true , "block" , "block" , "frot"
129     , "block" , "stop" ) goto X
130 )|
131
132 automaton Sequence ( inout act sync P1 , P2 , P3 , P4 , Again
133                    ; inout disc real issd
134                    ; inout var real TE1 , TE2 , TE3 , TE4
135                    ) =
136 |( disc control nat start = 1
137 ; mode X =
138   initial
139   ( when start = 1 and ( issd < TE4 or TE4 <= -100.0 ) act P1 do start := 2 )
140   ( when start = 2 and ( issd < TE1 or TE1 <= -100.0 ) act P2 do start := 3 )
141   ( when start = 3 and ( issd < TE2 or TE2 <= -100.0 ) act P3 do start := 4 )
142   ( when start = 4 and ( issd < TE3 or TE3 <= -100.0 ) act P4 do start := 1 )
143   ( act Again do start := 1 ) goto X
144 )|

```



```

145
146 automaton Uscase ( inout disc control real L, issd
147                   ; inout disc control nat error, fv
148                   ; inout var real TE2
149                   ; inout act sync SizePlus, SizeMin, IssdPlus, IssdMin
150                   , EsPlus, EsMin, EeSwitch, JamdPlus
151                   , JamdMin, UC1, UC2, UC3, Start, P1
152                   , Again, Jam, Detect
153                   ) =
154 |( disc control nat es = 3
155 ; disc control string ee = "LE"
156 ; disc control bool start = false
157 ; disc control real jamposLE = -100.0, jamd = 10.0
158 ; mode X =
159   initial
160   // Set size of sheets
161   ( when not start now act SizePlus do L := L + 5.0 )
162   ( when not start now act SizeMin do L := L - 5.0 )
163
164   // Set inter sheet starting distance
165   ( when not start now act IssdPlus do issd := issd + 1 )
166   ( when not start now act IssdMin do issd := issd - 1 )
167
168   // Select Error sensor
169   ( when not start and es < 6 now act EsPlus do es := es + 1 )
170   ( when not start and es > 0 now act EsMin do es := es - 1 )
171
172   // Select Error edge
173   ( when not start and ee = "LE" now act EeSwitch do ee := "TE" )
174   ( when not start and ee = "TE" now act EeSwitch do ee := "LE" )
175
176   // Select jamming distance
177   ( when not start now act JamdPlus do jamd := jamd + 3.0 )
178   ( when not start now act JamdMin do jamd := jamd + 3.0 )
179
180   // Select predefined case
181   ( when not start now act UC1 do L, issd, es, ee, jamd := 30.0, 5.0, 2
182   , "LE", 10.0 )
183   ( when not start now act UC2 do L, issd, es, ee, jamd := 30.0, 5.0, 5
184   , "TE", 10.0 )
185   ( when not start now act UC3 do L, issd, es, ee, jamd := 40.0, 20.0, 4
186   , "LE", 20.0 )
187
188   // Confirm above settings and start simulation
189   ( when not start now act Start do start := true )
190   ( when start act P1 )
191   ( act Again do start, jamposLE, error, fv := false, -100.0, 0, 5 )
192
193   // Occurence of the error
194   ( when ee = "LE" and TE2 + L > sensorpos[es] - jamd and jamposLE < 0.0
195   now act Jam do jamposLE := TE2 + L )
196   ( when ee = "TE" and TE2 > sensorpos[es] - jamd and jamposLE < 0.0
197   now act Jam do jamposLE := TE2 + L )
198

```

```
199 // Detection of error
200 ( when fv = 5 and ee = "LE" and TE2 + L > sensorpos[es] now act Detect
201 do error , fv := 2*es+1, 2 )
202 ( when fv = 5 and ee = "TE" and TE2 > sensorpos[es] now act Detect
203 do error , fv := 2*es+2, 2 ) goto X
204 )|
```