ISSN 0280-5316
ISRN LUTFD2/TFRT--5754--SE

Model-based Grade Change Support

Jonas Jägerhök

Department of Automatic Control
Lund Institute of Technology
May 2005

Department of Automatic Control
Lund Institute of Technology
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
May 2005

Document Number
ISRNLUTFD2/TFRT--5754--SE

Author(s)
Jonas Jägerhök

Supervisor
Johan Åkessonat Automatic Control in Lund
Bengt Nilsson at AssiDomän Frövi in Frövi

Sponsoring organization

Title and subtitle
Model-based Grade Change Support (Modellbaserat stödsystem för produktionsomställningar i pappersmaskiner)

Abstrac
Customer demands for board adapted for their specific needs make a flexible production with many different qualities advantageous. The board quality changes can include a change in the composition or only a change in grammage. A grade change is a product quality change in a paper or board machine.
Grade changes may however cause problems in the production. When changing the production from one kind of cartonboard to another, losses of time and raw material are sometimes inevitable. Sometimes the losses may be reduced by optimizing the grade change control. The losses due to control that is not optimized can be separated into two different groups. The first group contains losses due to a non-optimal grade change strategy, where the grade change control is badly planned. The other kind of losses is result of a poor control scheme execution such as important steps being forgotten or not successfully performed.
In this report a Sequential Function Charts (SFC) model developed in JGrafchart is suggested to address both of these problems. The result is a computerized tool meant to help operators by suggesting control strategies and remind them if important steps are not carried out. Control laws and strategies are derived from physical models of the cartonboard making process together with information collected from experienced operators and engineers at AssiDomän Frövi. Timing issues and other optimization is handled with the Matlab, Simulink and GESOP tools.
The goal of the thesis was to present an example which shows how a semi-automatic grade change control with JGrafchart could contribute to the grade change control at AssiDomän Frövi.

Keywords
Grade change, support, optimal control, cartonboard, board, paper, SFC sequential function charts, JGrafchart

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
84

Recipient's notes

Security classification

Contents

Acknowledgments

First of all I want to express my gratitude to everyone who has helped and encouraged me during the past few months of struggle. In particular:

Niklas Jansson for guiding me through the jungle of cartonboard making, answering questions and showing me around at AssiDomän Frövi; Johan Åkesson for all help during the optimization and modeling part of the project, and for all supportive feedback regarding the report; Karl-Erik Årzén for your fast responses with advices, updates and fixes to my JGrafchart problems; Bengt Nilsson and Lars Jonhed for helping me to stay on the right track; Niklas Keijser for valuable advices and friendly support; Jan Sjögren for information and help regarding software issues and for providing me with all the grade change statistics.

I would also like to dedicate some words of appreciation to everybody else at AssiDomän Frövi who participated in interviews, answered questions or in any other way supplied me with valuable material. Without your help this project would not have been possible.

Thanks also to all of you who have been around me during the last months without getting sick of my lengthy expositions of grade changes. Special credits go to my Mother, my Father and my Sister, and of course to all friends in Lund, Örebro and Linköping. Thank you for being patient and understanding.

This research was partially done in the framework of the HYCON Network of Excellence, contract number FP6-IST-511368.

1. Introduction

1.1 Problem Formulation

The aim of this thesis was to investigate the possibilities of using JGrafchart function charts (also referred to as models here) for improving the grade change control at AssiDomän Frövi. The objective was to illustrate how semi-automatic grade change control with JGrafchart could contribute to a faster and safer grade change.

1.2 Background

Grade changes provide a way for the cartonboard manufacturer to offer board with different properties with a single board machine. While making the production more versatile grade changes might also introduce additional problems. Among these are losses of time and pulp, and a higher risk of web breakage. Altogether these effects lead to an average loss of several tonnes of cartonboard in every grade change. If the losses were reduced, the cost of a grade change would be smaller and grade changes could be executed more frequently. This in turn, makes it possible to have less board in stock and to offer many different grades.

1.3 Approach

To fully utilize the benefits of a JGrafchart solution all control steps carried out during a grade change should be included in the JGrafchart model. To find a comprehensive solution is also a general desire at AssiDomän Frövi; if the model only applies to a certain part of the problem the user also needs to know in what situations it can be used and when to resort to other tools. This uncertainty makes the application less user friendly.

Creating a complete JGrafchart model for a grade change requires profound knowledge about both the grade change process and the control adjustments connected to it. This includes information about how, why and in what order the control actions should be performed. A major part of the problems connected to model-based control implementation as described here, is therefore to organize the control actions into a control sequence. When documented control schemes for the grade change control (or parts of it) exist, the implementation of the JGrafchart models is therefore greatly simplified.

In cases where there are no control execution recommendations documented one has to rely on other methods when implementing the model. This can be for example observations from live studies. Another alternative is to use data analysis to identify the used control patterns in previous grade changes.

When no documented guidelines exist the implemented JGrafchart models can also act as a kind of control documentation, which simplifies communication between shifts and further development of the control strategies. Care has therefore been taken to make the models easy to overview.

The proposed JGrafchart guidance model still requires the operators to perform the control. In this sense it is a passive solution, which is completely disconnected from the control systems in the cartonboard making process. The implementation also makes it possible to run several instances of the tool simultaneously without any risk of conflicts. Thus, the JGrafchart models can also be used as observation tools by anyone who wants to see how the grade change control is performed.

1.4 Contribution

A JGrafchart guidance model for the control in the grade change *Carry 425* to *Bright 390* has been developed. The presented model includes advisory features that suggest when specific control actions are to be taken and supervision features for selected steps in the control sequence. The described control actions are organized into four groups corresponding to different parts of the machine. The model structure of the stock preparation and wet end parts is mapped out. For the two other sections, the coating color kitchen and drying section, several control actions still need to be identified.

The JGrafchart guidance model implementation includes interfaces for the communication between JGrafchart and the integrated information systems at AssiDomän Frövi. Data from earlier grade changes together with information from operator interviews have been used to determine the structure and to estimate timing variables of the model. Simulation and off-line testing of the model's supervision features have also been conducted.

The potential of using JGrafchart models for grade change optimization has also been examined. This was done with a new control strategy suitable for a semi-automated JGrafchart model, which was acquired through the use optimal control theory and simulations. The optimization is based on physical models of the cartonboard production process previously developed at AssiDomän Frövi.

1.5 Outline of the Thesis

Chapter 2 and 3 contain information about cartonboard, paper/board making and the grade change problem in general. The two chapters make a good start for people who have little or no experience in how to make cartonboard. They also contain mill specific production details which are referred to in later chapters. The reader is therefore advised to get familiar with this part of the thesis before reading Chapters 7-11.

Chapter 4 lists working methods and sources of information. It is intended as a help for anybody who is interested in continuing the project or needs information about the information sources that have been used. If neither of these things is of interest this chapter can be skipped without the risk of missing any valuable information.

In Chapter 5 a study of the present grade change control strategies at AssiDomän Frövi is presented. It lists the important set point changes and briefly discusses the possibilities to identify control strategies by analyzing stored process data. In addition, the chapter includes some comments on the reliability of the data and the data analysis itself.

Chapters 6 and 7 describe how to generate JGrafchart models. The contents of the first chapter should only be of interest for readers that have no experience of JGrafchart since it only covers very general design and basic features. The second chapter on the other hand presents a JGrafchart model generated for control simulation. This includes both a shorter illustrated description of the model and information on the strategies used when designing it.

Chapter 8 describes the supervision extensions added to the basic model presented in Chapter 7. It is a good idea to read at least the latter part of that chapter before starting to read Chapter 8.

The optimization part of the project is presented in three chapters. Chapter 9 contains information about the simple models used in the optimization. Chapter 10 is an introduction to how the optimal control theory can be applied to the grade change problem. In Chapter 11 is described how GESOP and Simulink have been used to generate a new strategy for the grade change control of the changes in pulp mix composition in the mixing stage.

At last, Chapter 12 contains practical details on the interface programming that has been done during the work on the thesis.

2. Background

This chapter contains background information on paper/cartonboard making, both in general and at the cartonboard mill at AssiDomän Frövi. It briefly explains the characteristics of cartonboard and the different steps of the manufacturing.

2.1 What is Cartonboard?

Paperboard, or board, is thick, stiffish paper that often consists of several plies. Board for making consumer packs is usually called cartonboard. The different layers are the reason of many of the board's special properties. This structure makes it possible to choose different mixes for each of the plies. Sometimes the cartonboard is coated. Just as for the board itself coating can be applied in more than one layer. An example of how a four-layer board can be composed is illustrated in Figure 2.1. The composition of each ply is chosen

Figure 2.1 An example of a four-layer cartonboard composition.

with the desired properties in mind. For example, a good mix for the top layer results in a very flat and white surface that is easy to print on. When choosing the pulp mix for the inner plies, on the other hand, cost and good strength characteristics are more important. Coating layers are used to improve the look and printing properties of the surface. At AssiDomän Frövi the produced board is separated into two different categories, *packaging board* and *liquid board*. The first category contains board which is used for packaging of for example frozen food or chocolate, board of the second category is mainly used for beverage packages. Examples of the two types of cartonboard are shown in Figure 2.2. The *liquid board* produced at AssiDomän Frövi has a different chemical compound, usually a slightly lower grammage and does not always have coating applied to it.

2.2 Cartonboard Production

The raw materials used for the pulp mixes can be chemical pulp from pulp mills and mechanical pulp from chip refiners. The different pulps may be produced at the mill (integrated mills) or may be delivered by an external supplier. The pulp delivered from outside the mill usually arrive dried in bales. The bales have to be dissolved before they can be mixed with the rest of the pulp. This

Figure 2.2 Three products made of *packaging board* and, to the upper right, a beverage package made of *liquid board*.

is referred to as pulping and is done in units that are called pulp slushers or pulpers.

From Wood to Pulp

The wood used for making pulp at AssiDomän Frövi has to pass several steps before becoming pulp that can be used in the cartonboard production. The wood is from both coniferous and deciduous trees and do mainly come from Swedish forests. The fresh wood is first measured and de-barked. The bark removed is transported to the bark boiler where it combusted and converted to energy. After the bark has been removed the wood is sent to the wood chipper where it is turned into chips about 25 mm long and 8 mm thick.

The chips are digested down in the continuous digester and batch digesters. In this process the chips are exposed to high temperature (170°C) and pressure (6-7 atmospheres) and the treatment liberates the fibers from the lignin. After a subsequent washing the pulp is ready to be used in the cartonboard manufacturing. Part of the pulp do however pass through an additional bleaching step in the bleaching plant before it is pumped to the board mill.

Stock Preparation

The stock preparation is a process where the pulp suspension known as the stock, is prepared. In this step the pulp is cleaned and diluted and different additives are added. The outermost layer of the cellulose fibers in the pulp is also treated so that the fibers bond to each other more easily and make a strong board. This is done by beating the stock in mills.

Many times the pulp stock is a mix of different kinds of pulp. This way it is possible to achieve the right properties for the final board. The mixing process is done in mixing tanks in conjunction with the stock preparation. At AssiDomän Frövi there are three mixing tanks, one for the bottom layer, one for the top layer and a common one for the two mid layers. In Figure 2.3 can be seen a Dymola simulation model of the the top layer stock mixing. After

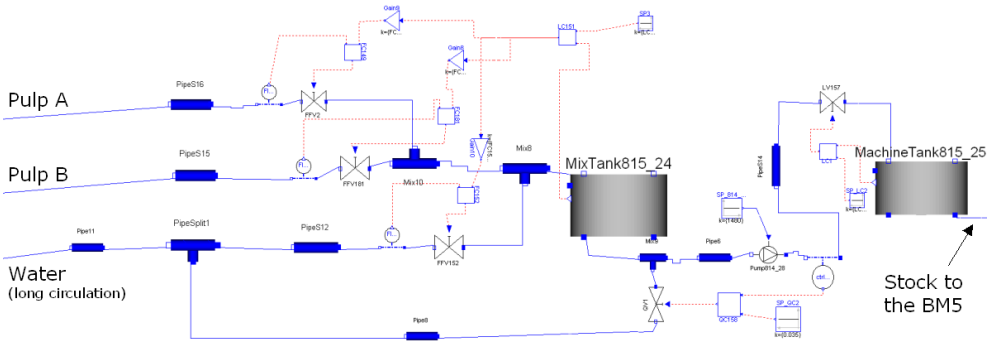

Figure 2.3 A Dymola model of the mixing step in the stock preparation for the top layer. The components used in the model are part of the AssiDomän Frövi component library.

the pulp has been mixed in the mixing tanks it is diluted and pumped into the machine tanks. At this point the consistency of the stock is around 3%. On the way to the boardmachine chemicals for controling pH, strength, color and other printing properties are added to the mix. Clay (kaolin) and chalk are examples of fillers that enhance the opacity and brightness of the board. The mixing pumps pump the stocks from the machine tanks towards the wet end of the boardmachine. Before the stock reaches the boardmachine it is further diluted and cleaned in several steps. When the stock is dispensed onto the wires in the boardmachine more than 99% is water.

Wet End

The first part of the boardmachine is called the wet end. The wet end consists of the wire section and the pressing section. The stock from the stock preparation gets to the wire section through a headbox. The headbox is a chamber that dispenses pulp stock evenly onto a moving wire. The wires are flat belts of metal or plastic mesh on which the board web is dewatered.

The configuration of the headboxes, wires and press rolls of the wet end in the KM5 boardmachine at AssiDomän Frövi can be seen in Figure 2.4. The figure also illustrates the other parts of the machine, all the way to the reel-up. Large quantities of water are drained off the wire by suction boxes and wire foils. The drained water and some of the fibers that pass through the wires are fed back to the mixing pump. This means that some of the water and fibers will circulate many times through the cleaning units and the headbox. This circulation system is referred to as the *short circulation* or *white water system*.

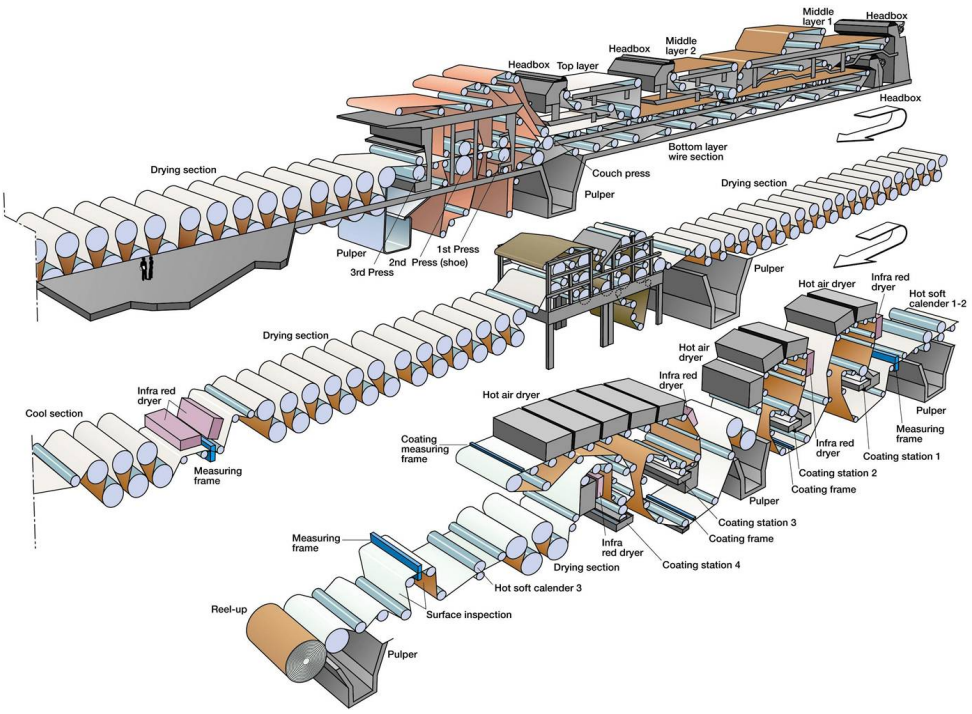

Figure 2.4 The KM5 Board Machine. The machine can be divided into different sections, the wet end with the headboxes, wires and presses, the drying section with the dryers, and the calenders and coating units. When the board is dried and coated it is winded up in the reel-up.

Some of the excess water is also fed back through the longer circulation system known as the *long circulation*. The water in the *long circulation* is used in the stock preparation and pulp slushers.

As mentioned earlier the headboxes are the units through which the pulp stock enters the boardmachine. The angle and speed at which the stock from a headbox meets the wire have a large impact on the quality and characteristics of the produced board. The spraying angle and amount of the stock dispensed by the headbox is affected by the headbox slice and pressure. It is also important to keep the variations in concentration and temperature of the stock as low as possible.

The bottom layer wire at the KM5 board machine is longer than the other three. In the multi-layer web forming the three upper layers are merged with the bottom layer one at a time. The result is a four-layer board by the end of the bottom layer wire. When leaving this wire the web goes into the wet pressing section. At this point the board web has reached a consistency of about 30-40%.

To dry the web by wet pressing is cheaper and more efficient than using the dryers in the drying section. On the other hand it is not possible to achieve a low enough level of humidity by only pressing the board. This is why the drying section is still needed. When the board reaches the drying section it can have a consistency of about 55%.

Drying Section

In the KM5 the board is dried with steam cylinders and infrared dryers. It is important to dry the board evenly. This means that no parts of the web, either

on the outside or inside, ever have a water content that is very different from the rest. If this happens the produced board might curl or show other quality defects. It is also important to prevent the vapor from forming bubbles within the web, this can happen if the power to the dryers is too high. The vapor bubbles might explode and cause web breakage. By the time the board leaves the drying section the water content is less than 10%.

Coating and Surface Treatment

After the board has been dried its surface is treated in calendars. The calendars make the surface of the board flat and give it a glazed finish by passing it between two or more rolls.

The board at AssiDomän Frövi is also coated in an on-machine coating unit. The coating unit in the KM5 boardmachine consist of three separate coating stations, together they can apply three different layers of coating. Today however, only two of them are used at a time. In Figure 2.4 the coating stations can be seen to the bottom-right and the calenders at each side.

When the cartonboard is dried and glazed it is wound to reels in the reel-up. The large reels from the reel-up are later cut into smaller rolls. These rolls are sometimes cut into sheets in the sheeter. Whether or not the cartonboard is cut into sheets depends on customer requests. The off-quality production is usually fed back to the process as broke and is included as one of the components in the mid ply mix.

2.3 More Information

The intention of this chapter is to help readers not familiar with cartonboard/paper manufacturing to follow the discussions in later chapters. The AssiDomän Frövi cartonboard mill includes many processes of which some are not even mentioned here. For more information about the cartonboard production at AssiDomän Frövi [2] is an informative source.

3. The Grade Change Problem

In this chapter the grade change problem connected to cartonboard making is presented. Some mill specific statistics and information from AssiDomän Frövi regarding grade changes is included in Section 3.2. In the latter parts of the chapter is presented examples of what in the grade change control that could be improved and how this could be done. First of all is however some general information on what a grade changes is. This section also contains definitions and explanation of terms connected to the grade change control.

3.1 The Grade Change

A grade change can be defined in different ways. According to [5] the grade change, at some paper mills, does not have to be a change of the product at all. In this case it is only a change in the labels used when labeling the produced board. In this thesis the term grade change is used only for changes that require some kind of change in the production.

It is worth mentioning that even though a grade change, as defined in this thesis, requires a change in the production it is still possible that the produced board fulfills the specifications for two different grades. This happens when the quality acceptance limits of the grades overlap.

The more extensive the grade change is the more and larger are the differences in quality specifications between the two products. Major changes do usually require more control signals to be changed, but also simpler changes need several controller set points to be adjusted. The time it takes to go through with a grade change depends on how the grade change time is defined. In general terms the time of a grade change is the time from the moment when the produced board does not qualify as the grade that was produced before the grade change was started until it fulfills the quality constraints of the new grade. For changes that involve a change in the pulp mix the grade change takes longer than when only the basis weight needs to be changed.

Minor grade changes are usually done without stopping the production, in what is referred to as on-line changes. The most complex ones however require a stop and a restart of the production.

3.2 Grade Changes at AssiDomän Frövi

At AssiDomän Frövi there exist several board grade groups, known as "products". The grades within a group share a common name and have a very similar chemical compound. The main differences between two grades from the same group are differences in thickness and basis weight. Examples of product names are *Bright*, *Duplex* and *Carry*. Many times a distinction is made between a change within the group, change in basis weight ("ytviktsomställning"), and a change from one group to another, change of product ("produktomställning"). In this thesis both of the two are referred to as grade changes.

Grade Change Execution

The grade change at AssiDomän Frövi is controlled and supervised by the operators. There are six operators working at the same time and they are responsible for different parts of the process control. In close communication the operators decide on when to do the change and how to do it. Some of the things that have to be agreed upon are what wire speed to use for the next grade and how to adjust the presses and dryers. To their aid they have software information systems and automated control systems. The most advanced control system is the HMPC controller. During a grade change this controller ramps the speed of the wire, the basis weight and the humidity level of the board to their new set points. The most important information system is the PI System. The PI System gathers information from many parts of the process as well as from other information systems. In Figure 3.1 is a simplified illustration that shows the information flow during the preparation and execution of a grade change.

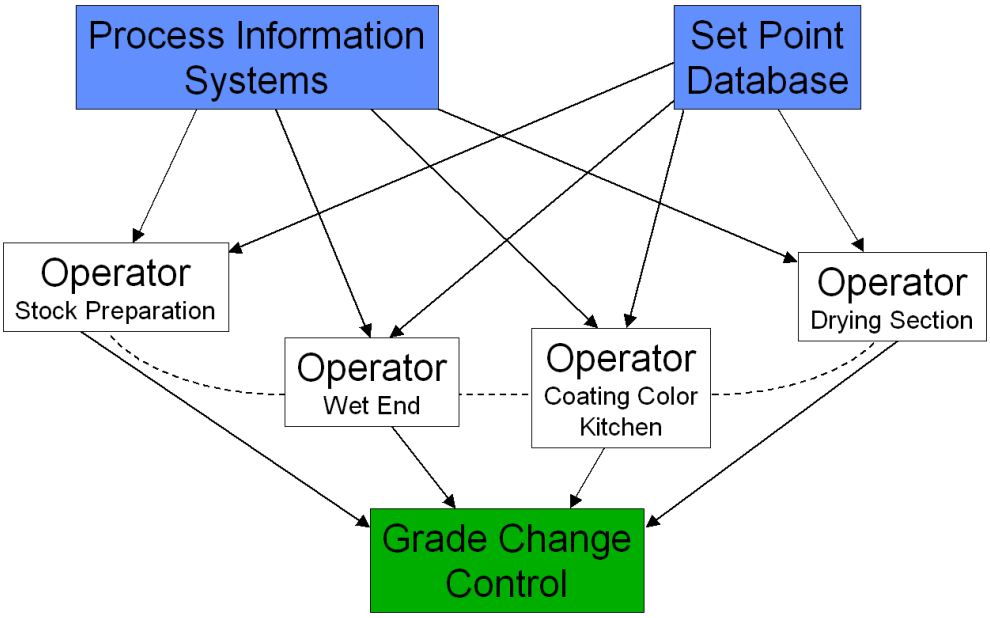

Figure 3.1 A schematic illustration of the information flow in the grade change control. The operators base their control on information from the information systems and communicate with each other to coordinate their actions.

Grade Change Statistics

At AssiDomän Frövi several grades, or board qualities, are produced. During the last year 30 different grades has been produced. The production follows a cyclic scheme with a span of a little more than a month. On average a little more than one grade change is carried out every day. By the rules used for grade change evaluation in [7] the grade change time can vary from no time at all to more than an hour. According to results from a grade change investigation available at AssiDomän Frövi, the losses associated with an online grade change correspond to about 13 tonnes of board.

Other studies concentrated on the production interruption frequency at AssiDomän Frövi, show the probability of a production stop the first hours after a grade change is significantly larger than during normal operation. This

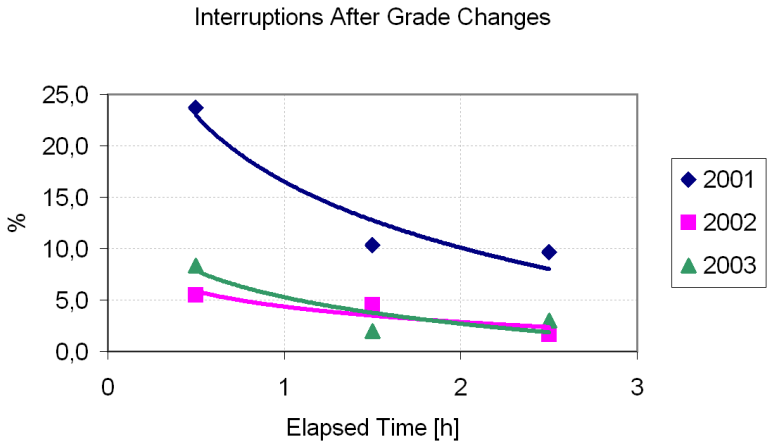

Figure 3.2 The distribution of production interruptions within the first three hours after a grade change. On the y-axis shows how large amount of the total number of interruptions that happened during the first hours after a grade change.

can be seen from Figure 3.2. The probability of interruptions recently after a restart has a distribution similar to the one in the figure. A stop during the first hour is even more likely after a restart than after a grade change. Thus, an interruption after a grade change is likely to cause additional problems.

A study related to the one above also shows that a higher percentage of the board produced during the first hours do not match the accepted quality limits, compared to the normal production. A larger part instead becomes broke, which is fed back into the stock preparation. The effects of the lower quality and increased interruption risk contribute to making the net production much lower during a couple of hours after a grade change. Figure 3.3 shows the ratio between the board that was sold and the board produced from the first reels of the new grade.

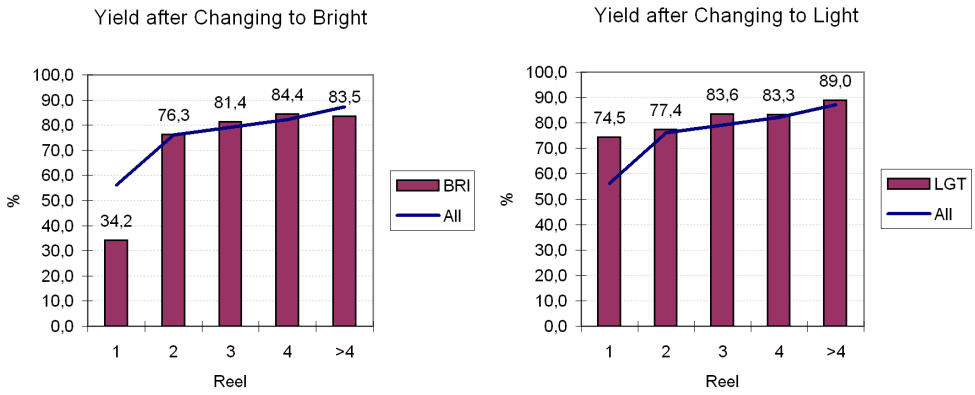

Figure 3.3 The ratio between sold and produced board for the first reels after a grade change. The graph to the left shows the results for *Bright* and to the right is the result for *Light*.

3.3 Improvement Strategies

Besides the obvious goal of keeping the grade change time as short as possible, there are other aspects worth mentioning when considering the outcome of a grade change; the production immediately after a grade change might suffer from quality problems directly related to the grade change. Changes in basis weight, wire speed, and chemical compounds in the different plies might cause for example the humidity and the basis weight to behave erratically until they reach the steady state stability of normal production. Sometimes pronounced oscillations around the desired levels can be noticed. It is obvious that it will then be harder to keep the produced board's quality properties within the specified intervals. It is also likely that oscillations and erratic behavior among the process variables might be the cause of other problems.

Large and sudden changes of the humidity of the web might result in web breakage. A web breakage is a critical and expensive problem. First of all it results in a direct economic loss while the production is halted. Secondly, it requires a restart. Starting the machine can be a problem itself. This process consists of several steps of which some needs to be done manually by the operators. As mentioned in Chapter 3.2 startups have several problems in common with the grade change. On top of this, many stops in the production make planning much harder and may result in problems earlier or later in the production chain.

The humidity is mainly controlled with steam cylinders and infra red dryers in the drying section. The overall control of the humidity is done by a horizon model predictive controller. The HMPC algorithms are based on models which are tuned for the different grades. A somewhat random behavior of controller inputs will make the humidity control perform badly. If all input states on the other hand change in a nice and continuous way it is likely that this will result in a smoother output humidity. The grammage and wire speed are two examples of inputs to the HMPC.

The objective to minimize oscillations and disturbances of the humidity level can be applied to other parts of the process as well. A grade change where the process variables change smoothly is in many ways more similar to the normal production case. Since less problems appear during steady state production it might be reasonable to make this a goal itself. In more general terms a well planned grade change strategy is one that reduces the number of possible problem sources. Making the process's state variables change smoother during the grade change could be one way to achieve this.

Many of the control signals changed in a grade change are adjusted manually according to recommended values from a set point database. An inevitable problem in a case like this is that some of the changes might be forgotten or not successfully performed. The consequences hereof can be hard to spot at an early stage and might not be discovered until the first quality measurements of the new grade are analyzed in the lab, in the worst case it might take even longer. Thus, preventing them from happening in the first place is an attractive solution.

3.4 Introducing Standards

Using the same control scheme for all grade changes of the same kind would also help in communication and evaluation of the grade change control in several ways. It is likely that the grade change control will be executed differently depending on for example which shift is currently working. First of all a scheme following the ideas presented in this thesis could work as a summary illustrating all the important steps that have to be carried out. When new strategy improvements are to be introduced an update of this scheme can simplify the communication between different shifts and help to avoid misunderstandings.

Conforming to a general grade change strategy can also be helpful when trying and evaluating possible improvements; by changing a single parameter (or a few) of the scheme it will probably be easier to identify the direct consequences of this parameter change. If too many parameters are different, which is likely to happen if no scheme is used, the effects of a specific parameter will be harder to see.

4. Preparation

Being the first project of its kind at AssiDomän Frövi, it required a lot of preparation. This chapter contains information about the planning and outcome of the preparatory steps. Sections 4.1-4.3 explain the steps that needed to be finished before the generation of the JGrafchart model and the optimization could be started.

In the last part of the chapter a brief listing of the most important information sources is included.

4.1 Collecting Background Information

In order to generate a good JGrafchart model of a grade change strategy for cartonboard making one has to be familiar with the manufacturing process. The first couple of weeks therefore consisted of studies of books and papers on the paper making subject. The paper/board making is a complex process where many parts in the production is mill specific. To relate the general information from the books to the conditions at AssiDomän Frövi, the literature studies were done in parallel with studies of drawings of the cartonboard machine and diagrams from software tools supplied by AssiDomän Frövi.

Knowledge of the grade change process and control strategies was achieved through literature studies as well. Mill specific information on grade change control at AssiDomän Frövi was collected through interviews with the superintendents. These interviews were scheduled at an early stage in order to quickly get good insight into the problems connected to a grade change.

Since a large amount of information at AssiDomän Frövi is stored in extensive databases with different software interfaces the preparation stage also included investigation of several software tools and environments. Internet documentation on the interfacing options between Java and for example C and SQL was studied to understand what possibilities that existed for JGrafchart to communicate with the AssiDomän information systems. Some examination of the local software applications was also done to get acquainted with the data extracting tools and their interfacing features.

More details on the practical execution of the steps conducted in the grade change process were gathered from live observations from the control rooms. Together with discussions with the operators this provided important information on the currently used strategies.

4.2 Selecting a Grade Change Case

To select a suitable grade change to analyze closer there are several factors that should be considered. Which grade change to examine in this thesis was decided at an early stage in a meeting with the superintendents. An intuitive option is to select the grade change with the best possibilities of improvement but this might be hard in practice. Especially when a completely new approach is in mind, it is very hard to know exactly what can be achieved.

One alternative is to choose the grade change that causes the most problems in the current production. Another solution is to choose the most common

grade change since the accumulated wins will this way be higher. Both these strategies are easy to use since they can be based on studies of collected data, interviewing operators and model analysis, without any knowledge of the new approach. There are however no guarantees that these grade changes will be easy to improve with the method in this thesis.

In an investigating project like this other aspects might instead be of greater importance. To more easily evaluate the results it might be more interesting to test as many things as possible from a single test case. Using this approach it will be easier to identify strengths of the method as well as problem sources. A grade change including many and complex adjustments might therefore be suitable. If the resulting method is promising it is also good if the results can be reused. Thus, it is advantageous if the selected grade change is as general as possible; the more steps that are similar the higher the reusability of the results.

The selected grade change was the one going from *Carry 425* to *Bright 390* (referred to as CRY425-BRI390). This grade change includes extensive adjustments, mainly since the pulp mix of the bottom ply is totally different in the two grades. In addition, it has several steps in common with other "packaging board-to-packaging board" grade changes. CRY425-BRI390 therefore seemed to be a suitable test case for the presented method.

One disadvantage with this selection is that this grade change is currently uncommon. The possibilities of live studies is therefore limited and there is also less stored data to analyze. The CRY425-BRI390 will however get more common if the production of *Bright* increases according to general predictions.

4.3 Adapting JGrafchart

Efforts have been made to make interfacing and communication between JGrafchart and the information systems at AssiDomän Frövi possible. The interfacing has greatly simplified later work and is also a necessary condition for on-line studies and experiments. All interactions between JGrafchart and the local information systems are handled through Java interfaces.

What changes and extensions that were made during the preparatory phase are described in Chapter 12. This chapter also contains more details on how the interface is implemented.

4.4 Sources of Information

This project has included searching among many different information sources such as books, papers, websites and local sources at AssiDomän Frövi. There is a lot of valuable information stored in the information systems and among the employees. To find the desired information can however be a challenging task.

Below is a summary of the most important sources used the project. The list contains brief comments on what the sources have been used for and how they have been utilized. This section might be interesting in case of further development of the method or similar grade change studies at AssiDomän Frövi.

Discussions and Interviews

A very valuable source in the grade change research was discussions and interviews. Especially when it comes to issues specific for the cartonboard mill in Frövi this source of information has been a great help.

The superintendents have a lot of experience from board production in general and they know the control systems and routines at AssiDomän Frövi very well. Each one of them is specialized in his own field of the board making process. The interviews with the superintendents mainly addressed the grade change control topic and provided a lot of information about interesting issues and possible improvements of the grade change process.

Very valuable information has also been collected in discussions with the operators. These discussions are usually more concentrated on the practical execution of the grade change or board making in general. They have also been a good source of feedback in the developing of the grade change control models in JGrafchart.

PI System

The PI system has been one of the most valuable sources of information, and by far the most important data source. The PI system collects and stores all kinds of information in an internal database. Most of the process information, such as process values, actuator signals and set point values, can be found in PI. All measured signals are logged once a minute and the history goes back to early in 2002. Data from lab experiments are also stored in the PI system. The time between the stored lab data values is however a lot longer than a minute.

There are several software applications at AssiDomän Frövi that make use of the PI data for generating trends et cetera. There are also different APIs for interfacing with user implemented applications. In this thesis the C-API, documented in [13], has been used to make the connection between PI and JGrafchart. Details on how this is done can be found in Section 12.1. The C-API has also been used to port the PI data to Matlab. A couple of mex-files and m-files make up an interface which provides a good ground for more advanced and user-configurable data studies than is possible with other software applications at hand. More information about the Matlab programming is found in Appendix A.

Set Point Database

This is an SQL database that contains recommended values for many controller set points and other measurable process values. In general the recommendations are specific for every grade produced at AssiDomän Frövi. The database also includes quality specifications. The values in the SQL database are manually updated once in a while according to new requirements or optimizations.

There are several ways to extract information from this database. For example one can use the local application *Q Info* or Microsoft Excel documents with Visual Basic macros available at AssiDomän Frövi. A more configurable way to extract data from the database is directly through Structured Query Language (SQL) calls from an SQL client. How the interface between JGrafchart and the specification database is implemented is described in Section 12.2.

Software Tools

In addition to the two database systems mentioned above there are many other data sources. To make retrieval of this information easy there are different software tools available. Most of them are developed at AssiDomän Frövi such as *Q Info*, *Prodanareport* and *RawMaterialForecast*. The first two provide features that can be helpful in grade change analysis, the last application is good for keeping track on what grades that are next in the production plan. Although the information from *RawMaterialForecast* is not always perfectly up to date it usually provides a fairly good idea about what grades are coming up.

Among the software applications not developed at AssiDomän Frövi, PI-ProcessBook (OsiSoft) has been the most important. PI-ProcessBook provides illustrative charts of different unit processes included in the board making at AssiDomän Frövi. PI-Processbook also offers easy ways to trend data from the PI system.

Literature and Other Sources

For gaining insight into the paper making process [3] and [4] are two valuable books on the subject. The website [11] is also a good source on paper making. This site contains informative descriptions of the different unit processes in English. For translations during the writing of the report, the glossary at [10] was also helpful.

A large amount of information on grade change strategies and techniques was found in [5] and [6]. These two sources describe the currently used strategies well and also provide valuable information on different approaches to grade change control in paper making.

In the optimization part of the project the theory behind the approach is based on the ideas in [8]. How to use the GESOP environment is described in the GESOP program documentation [14]. Process information such as volumes of tanks and lengths and dimensions of pipes used in the modeling was found in [18] and from the Dymola models of the process, which exist at AssiDomän Frövi.

During the implementation of the JGrafchart-PI System interface [13] was used frequently for the C part of the programming. Most other programming related information was gathered from websites. The sites [15] and [16] contain Java specific information which was of great help for both the Java side implementation of the aforementioned interface and for the communication between JGrafchart and the SQL set point database.

A lot of information connected to the production at AssiDomän Frövi can also be found on the internal website. Among other things this site contains production reports and pictures of the boardmachine.

5. Data Analysis

The grade change analysis in this thesis is concentrated on the grade change from *Carry 425* to *Bright 390*. In the rest of this report this particular grade change is referred to as CRY425-BRI390. This chapter presents results and ideas from the analysis of data from earlier grade changes, which is stored in the PI System. Most of the analysis has been conducted in Matlab. Practical details on the programming part of the Matlab studies can be found in Appendix A.

5.1 Analyzing Current Strategies

A natural question that comes into mind when trying to find a good control scheme is what strategies that are used at the moment. There are several ways to collect information about this. Very important sources in this matter, are the interviews and discussions with the operators and superintendents at AssiDomän Frövi. In addition it is a good idea to examine the historical data.

Set Point Changes

For the steady state production the set point database provides many reference values for controller set points and quality measurements. Since the grade change is a transition between two steady states the reference values are interesting in this case too. In Table 5.1 and Table 5.2 the stored set point recommendations for CRY425-BRI390 and BRI310-BRI290 respectively are gathered. As can be seen by comparing the two tables, some grade changes involve many more controller adjustments than others. The recommended production set points in the database for *Carry 425* and *Bright 390* differ in approximately 60 set point levels.

Order and Timing

There is no information in the set point database on when the changes should be executed. This information has to be found elsewhere. For most of the changes in the two tables there are corresponding PI System tags. When present these are shown in the right-most column. The adjustments to new set points is at the operator level generally done in discrete steps. By analyzing the data from the PI System it is often possible to say when the changes have been conducted. While comparing the data from different grade changes, there are however a couple of things that have to be kept in mind.

Firstly, sometimes some of the expected changes do not occur in the expected way and sometimes not at all. It is usually hard to find specific reasons in single cases, but a likely explanation, especially for older data, is that set point recommendations and control strategies have changed through the years. This evidently makes it harder to spot general patterns.

Secondly, to be able to spot if a certain adjustment was early or late in a specific case one has to define a common time frame for all grade changes. In this report a PI System tag by the name 53PRODBYTE.PV has been used for this purpose. The value of this tag changes at the moment when the last reel of the current grade is changed for a new one. Which reel is the last of the current grade is determined by the planning system COBRA.

Property Name	*CRY425*	*BRI390*	*Change (%)*	*PI Tag Name*
Alun M-kar BS	*	*	*	52FY144.PV
Alun Premix BS	*	*	*	53FY045.PV
Alun Premix MS1	*	*	*	53FY046.PV
Alun Premix MS2	*	*	*	53FY047.PV
Alun Premix TS	*	*	*	53FY048.PV
Andel bl. Barr TS	*	*	*	52FF149_2.PV
Andel bl. Björk TS	*	*	*	52FF181_2.PV
Andel CTMP MS	*	*	*	52FF087_2.PV
Andel mald Högkappa MS	*	*	*	52FF088_2.PV
Andel omald Högkappa MS	*	*	*	-
Andel Utskott MS	*	*	*	52FF086_2.PV
Ånglåda Kalander 1 ytv.	*	*	*	53FX698.PV
Bestr. pålägg Stn 2	*	*	*	CW2F3M01.PV
Bestr. pålägg Stn 3	*	*	*	CW2F2M01.PV
Bestr. recept LAS	*	*	*	-
Bestr. recept Stn 1	*	*	*	-
Bestr. recept Stn 2	*	*	*	-
Bestr. recept Stn 3	*	*	*	-
Bladtyp Stn 2	*	*	*	-
Bladtyp Stn 3	*	*	*	-
Fukt efter Kalander 2	*	*	*	UX1F5M01.PV
Fukt Pope	*	*	*	UX1F1M01.PV
Fyllmedel BS	*	*	*	-
Hartslim Premix BS	*	*	*	53FY067.PV
Hartslim Premix MS1	*	*	*	53FY068.PV
Hartslim Premix MS2	*	*	*	53FY069.PV
Hartslim Premix TS	*	*	*	53FY070.PV
Hastighet Vira	*	*	*	53ST817.PV
Konc Dos-kar Lågkappa	*	*	*	52KCSB70.PV
Konc Inloppslåda BS	*	*	*	53AT524_KONCBS.PV
Konc Inloppslåda MS1	*	*	*	53AT524_KONCMS1.PV
Konc Inloppslåda MS2	*	*	*	53AT524_KONCMS2.PV
Konc Inloppslåda TS	*	*	*	53AT524_KONCTS.PV
Konc LAS	*	*	*	55AT158.PV
L/B-förhållande MS1	*	*	*	53GX830.PV
Läppöppning BS	*	*	*	53GC801.PV
Läppöppning MS2	*	*	*	53GC841.PV
Läppöppning TS	*	*	*	53GC861.PV
Linjelast Kalander 1	*	*	*	53BK1L.CV
Ljushet S2	*	*	*	BR1F6M01.PV
Ljushet TS baskartong	*	*	*	-
Malning bl. Barr TS	*	*	*	52KSRTBA.PV
Malning bl. Björk TS	*	*	*	52KSRTBJ.PV
Malning Högkappa MS	*	*	*	52KSRM75.PV
Malning PBL2 bl.	*	*	*	52KSRPBBL.PV
pH Dos-kar Lågkappa	*	*	*	52KPHB70.PV
Skiktytvikt BS	*	*	*	BCKCTL01.PV
Skiktytvikt MS1	*	*	*	MS1CTL01.PV
Skiktytvikt MS2	*	*	*	MS2CTL01.PV
Styrkestärkelse BS	*	*	*	52FY427.PV
Temp Kartongbana Kal. 3	*	*	*	TM1F1M01.PV
Tryck 2:a press Skopress	*	*	*	53G166_0PV18.PV
Tryck Guskanpress	*	*	*	53G185_3PV2.PV
Utloppskvot MS1	*	*	*	53PF827.PV
Utloppskvot MS2	*	*	*	53PF847.PV
Utloppskvot TS	*	*	*	53PF867_2.PV
Ytvikt baskartong	*	*	*	-

Table 5.1 Set point levels and other process values for which adjustments are recommended in the CRY425-BRI390 (as of May 2, 2005). The numeric values are not shown in this version of the report.

Property Name	*BRI310*	*BRI290*	*Change (%)*	*PI Tag Name*
Andel bl. Barr TS	*	*	*	52FF149_2.PV
Andel bl. Björk TS	*	*	*	52FF181_2.PV
Andel CTMP MS	*	*	*	52FF087_2.PV
Andel mald Högkappa MS	*	*	*	52FF088_2.PV
Ånglåda Kalander 2 ytv.	*	*	*	-
Avvattning Duoformer	*	*	*	53FX3110.PV
Fukt efter Kalander 2	*	*	*	UX1F5M01.PV
Fukt Pope	*	*	*	UX1F1M01.PV
Hastighet Vira	*	*	*	53ST817.PV
Konc Inloppslåda BS	*	*	*	53AT524_KONCBS.PV
Konc Inloppslåda MS1	*	*	*	53AT524_KONCMS1.PV
Konc Inloppslåda MS2	*	*	*	53AT524_KONCMS2.PV
Konc Inloppslåda TS	*	*	*	53AT524_KONCTS.PV
Läppöppning BS	*	*	*	53GC801.PV
Läppöppning MS1	*	*	*	53GC821.PV
Läppöppning MS2	*	*	*	53GC841.PV
Läppöppning TS	*	*	*	53GC861.PV
Linjelast Kalander 1	*	*	*	53BK1L.CV
Linjelast Kalander 2	*	*	*	53BK2L.CV
Ljushet TS baskartong	*	*	*	-
Skiktytvikt BS	*	*	*	BCKCTL01.PV
Skiktytvikt MS1	*	*	*	MS1CTL01.PV
Skiktytvikt MS2	*	*	*	MS2CTL01.PV
Torrhalt Duoformer	*	*	*	53QX459.PV
Tryck 1:a press	*	*	*	53WX898_2.PV
Tryck 2:a press Skopress	*	*	*	53G166_0PV18.PV
Tryck 3:e press	*	*	*	53Y063_0PV1.PV
Tryck Guskanpress	*	*	*	53G185_3PV2.PV
Utloppskvot TS	*	*	*	53PF867_2.PV
Vacuum Stacktork DS	*	*	*	53PC233.PV
Vacuum Stacktork FS	*	*	*	53PC232.PV
Ytvikt baskartong	*	*	*	-

Table 5.2 Set point levels and other process values for which adjustments are recommended in the BRI310-BRI290 (as of May 2, 2005). The numeric values are not shown in this version of the report.

The level change of 53PRODBYTE.PV provides a very easy way to compare the grade change control from different times.

Special Conditions

In many cases it is hard to identify a well-defined sequence with exact timings for the grade change control. If a sequence like this could be found it would be possible to use a single control scheme to describe all grade changes of a certain kind ever performed. In such a scheme all control actions are executed in the exact same way, in the same order and at the same time, every time the grade change is performed. Assuming that the process state is always the same when the grade change control is begun this sounds likely. On the other hand, if the initial state differs it does not sound as probable; if an important process value is lower than usual it probably needs to compensated for in the grade change control, especially if aiming for an optimal solution. If this last assumption holds true a static description of the kind described above can not be found. Instead a better control scheme will depend on the process conditions. The differences in the process state can be of different kinds. Examples are listed below.

- Measurable differences that are controlled by the operators.
- Measurable differences that can be observed but not easily controlled.

- Differences that are unknown or impossible to measure.

The final conditions of the grade change might differ from case to case too, just as their initial counterparts. This fact can be illustrated by examining the speed of the wire during several CRY425-BRI390, shown in Figure 5.1. Variations in

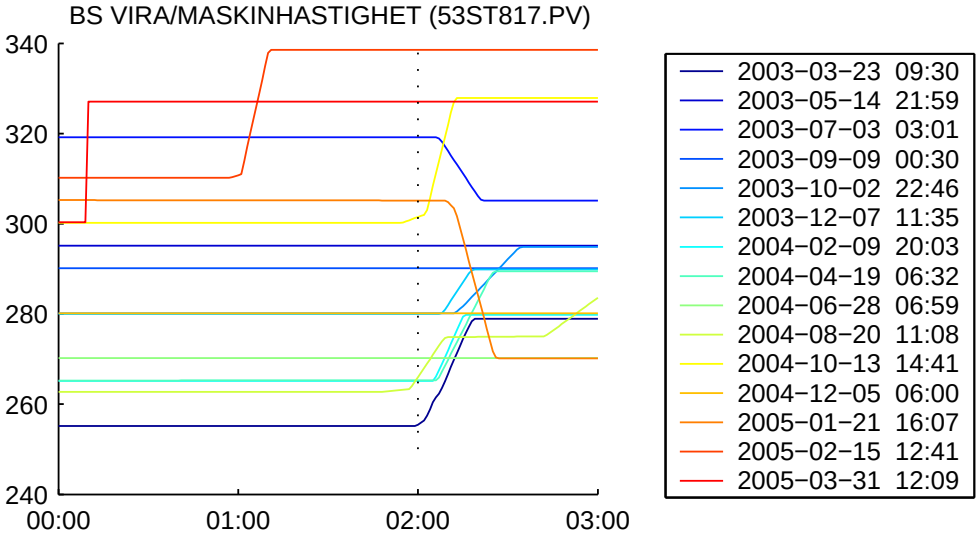

Figure 5.1 The speed of the wire during several CRY425-BRI390. As can be seen in the graph, the sign of the change can be either positive or negative.

the speed of the wire are differences from the first category above, since the speed can both be controlled and measured. As seen from the graph, both the speed when a grade change is started and the goal speed is different from time to time.

Why the speed change in a specific grade change can be different from one case to another can have several explanations. Sometimes a production at lower speed is chosen as a precaution when testing for example a new pulp mix. At other times a higher speed is used to test if the quality still remains as good when the speed is increased.

An example from the category of known sources that are hard to control is the production status of the board machine. In many cases the stops are not planned but the result of unpredicted problems. In Figure 5.2 the production status within a couple of hours before and after a CRY425-BRI390 has been plotted. All the graphs show cases where the production was stopped. The dotted, vertical line corresponds to the moment when the grade change was done, which here means the change in value of 53PRODBYTE.PV. In for example the grade changes at February 15, 2005 and March 31, 2005 the production was restarted with the new grade. In the case when the production is stopped right before the change it is no longer considered to be an on-line grade change. The major differences shown in Figure 5.2 definitely have an impact on how the control has to be executed.

A Suitable Example

Although obvious strategies and control sequences usually are hard to identify there are exceptions. For the CRY425-BRI390 there is a well defined control scheme for the operators to follow when switching the stock used for the bottom ply. This series of actions consists of several steps that are well described in

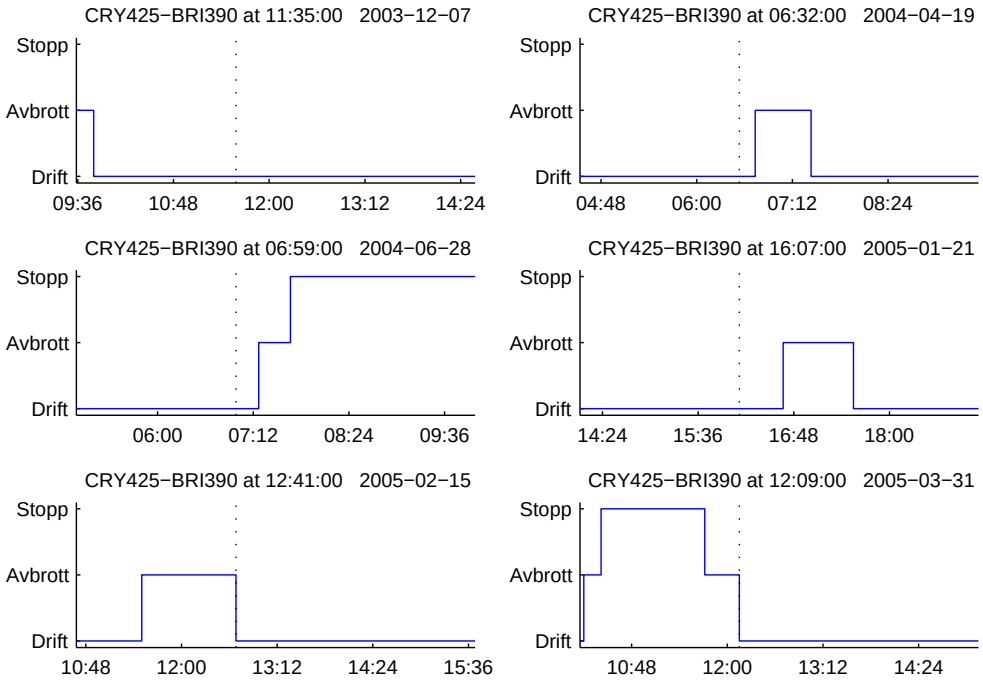

Figure 5.2 The production status during six CRY425-BRI390 where problems occurred.

[19]. When all the steps are carried out the stock from the top layer mixing tank supplies both the bottom and the top layer. Most of the changes conducted in this sequence can be tracked in the PI System data, where each one of the steps corresponds to the change in value for a single tag, or a few. A step can represent for example the opening or closing of a valve or the start of a pump.

Figure 5.3 illustrates the individual changes of the mentioned control scheme in a staircase like configuration. The height of a step shows when in the sequence it is planned to be executed. Thus, the lowest step should be carried out first, then the second lowest and so on. If all the steps are conducted according to the order in the control scheme, the height of the steps will all be in ascending order. This was the case in for example CRY425-BRI390 September 9th 2003, which is seen to the top-right of the figure. It is worth mentioning that the resolution of the data is one sample a minute and the exact order in which the steps was carried out is therefore sometimes impossible to see.

5.2 Results

By using the grade change data together with gathered information from the operators, it is possible to arrange the grade change control in a static control scheme that corresponds well to the data for short well-defined sequences. To make a more general scheme this way is however not possible.

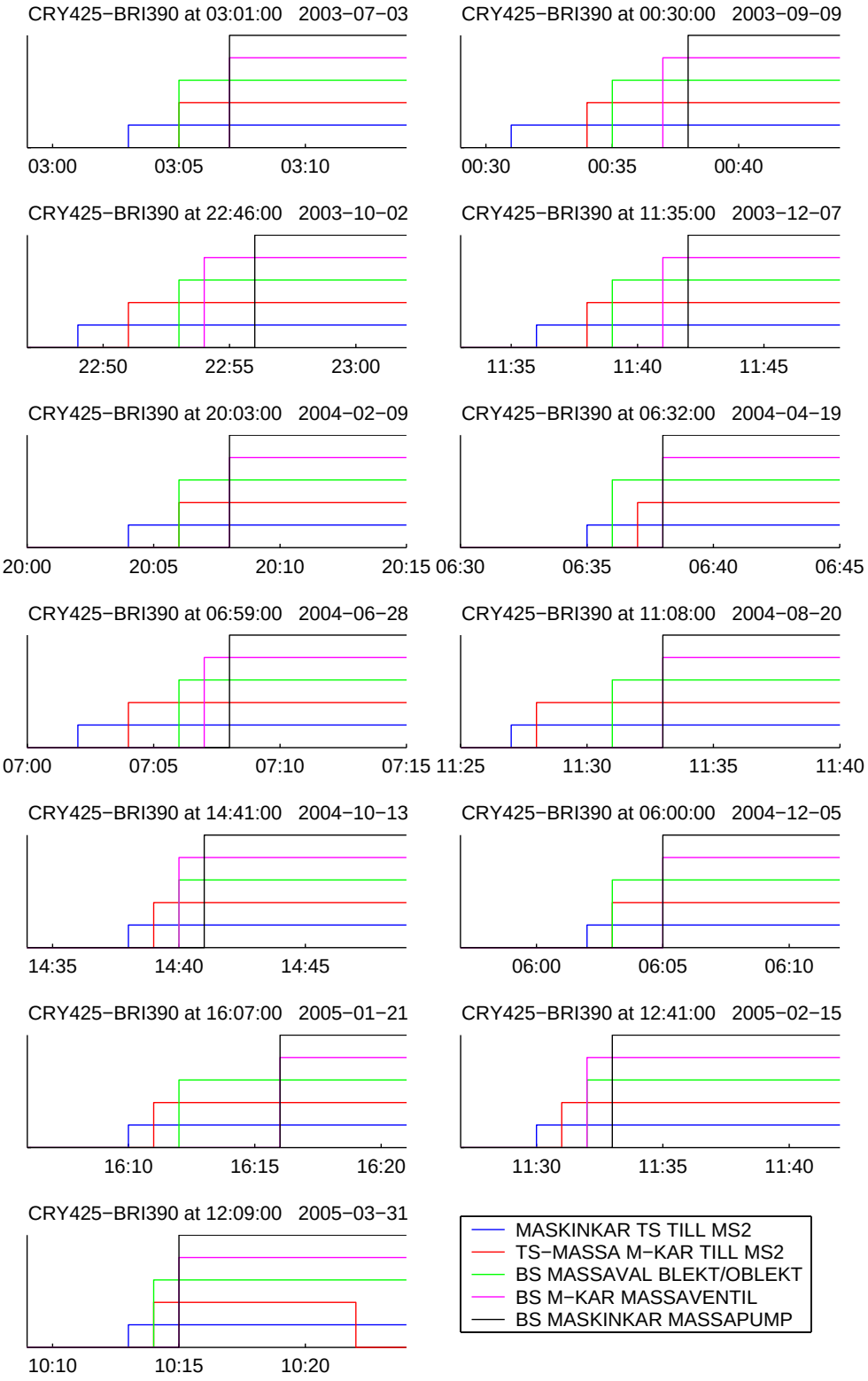

Figure 5.3 An illustration of how the steps in [19] are executed in CRY425-BRI390. Each step corresponds to a control action. A lower step means that the change is scheduled early. Thus, if executed according to the plan the plot will have the look of a staircase with the lowest step to the left and the highest to the right.

5.3 Discussion

No advanced statistical analysis has been conducted on the material presented in this chapter. A deeper study of this kind might be a good way to identify hidden relationships between different variables. If more of the dependencies are revealed it might be possible to outline a dynamic scheme with planning times that are chosen according to the current conditions before a grade change, and thereby uncover more of the current control pattern.

The amount of data for CRY425-BRI390 is however fairly small. At the start of the project only 11 of these grade changes were logged in the PI System and at the writing moment there is data from 15 available. When conducting further analysis it is a good idea to keep this in mind, since it will make it harder to come to general conclusions.

Worth mentioning is also the fact that in 6 out of 15 CRY425-BRI390 cases stored in the PI System the production was stopped within a couple of hours before or after the grade change. It is not guaranteed that all these problems have anything to do with the grade change itself. Some problems in the production, such as forming of blobs in the web that cause web breakage, can happen during the normal production as well. The reason of a stop can also be planned maintenance. At other times the stop is the result of a part of the board machine that is broken and sometimes it is the effect of mistakes in the control. To find the cause of a stop is generally hard by only analyzing the data.

6. JGrafchart

This chapter is an introduction to JGrafchart and contains information about how to use the tool for sequence control. The chapter explains shortly how programs are written and include details on some of the commonly used features. The information is concentrated on the parts of the JGrafchart that are used in the work connected to this thesis. Further details about the general use of JGrafchart and descriptions of the more advanced features can be found in [1] and [9]. In this thesis the term JGrafchart model is used as a synonym to a program implemented with the JGrafchart editor.

6.1 Introduction to JGrafchart

JGrafchart is a programming language for sequential, procedural, and state-transition oriented applications. The development of the JGrafchart toolbox is a project at the control department at LTH, which started in 2001. Together with the included editor, JGrafchart provides a possibility to "draw" sequence control programs instead of typing them in the common way. The source code can be stored in xml-files and make up a structure with different hierarchy levels and other options for abstraction and generalization. The editor also provides features for compiling and running the programs. The JGrafchart editor is therefore, not only an editor but an environment that can handle everything from program editing to the model execution. Figure 6.1 shows a screen shot of the JGrafchart editor. Documented use of JGrafchart can be found in for example [9], where JGrafchart has been tested for batch control.

6.2 Getting Started with JGrafchart

The sequence control in JGrafchart is based on Sequential Function Charts. These charts consist of sets of steps and transitions. Once a program is written and compiled in JGrafchart it can be executed. As soon as the execution of the JGrafchart program, or model as it is called in this thesis, is started a token will start traversing between the steps in the function chart. The token will be traveling downward through the model. All steps are connected through step transitions. An executing model can be seen in Figure 6.2. A step containing a token is referred to as active. An active step can execute actions. The actions can be of different kinds. Examples are assigning values to variables, making function calls and interacting with the user by showing for example a text or an image. The syntax of the actions is similar to the Java syntax. Some of the commands are executed as soon as the token arrive at the step, while others are executed when the token leaves for the next block. An important difference from the Java syntax is the required prefix which precedes the actual command. The prefix is an action qualifier and consists of a single character. The qualifier determines when the action should be performed. Figure 6.3 shows a step with three actions connected to it. In the example three different kinds of actions are shown, all with different action qualifiers. Table 6.1 contains a complete list of the possible action prefixes.

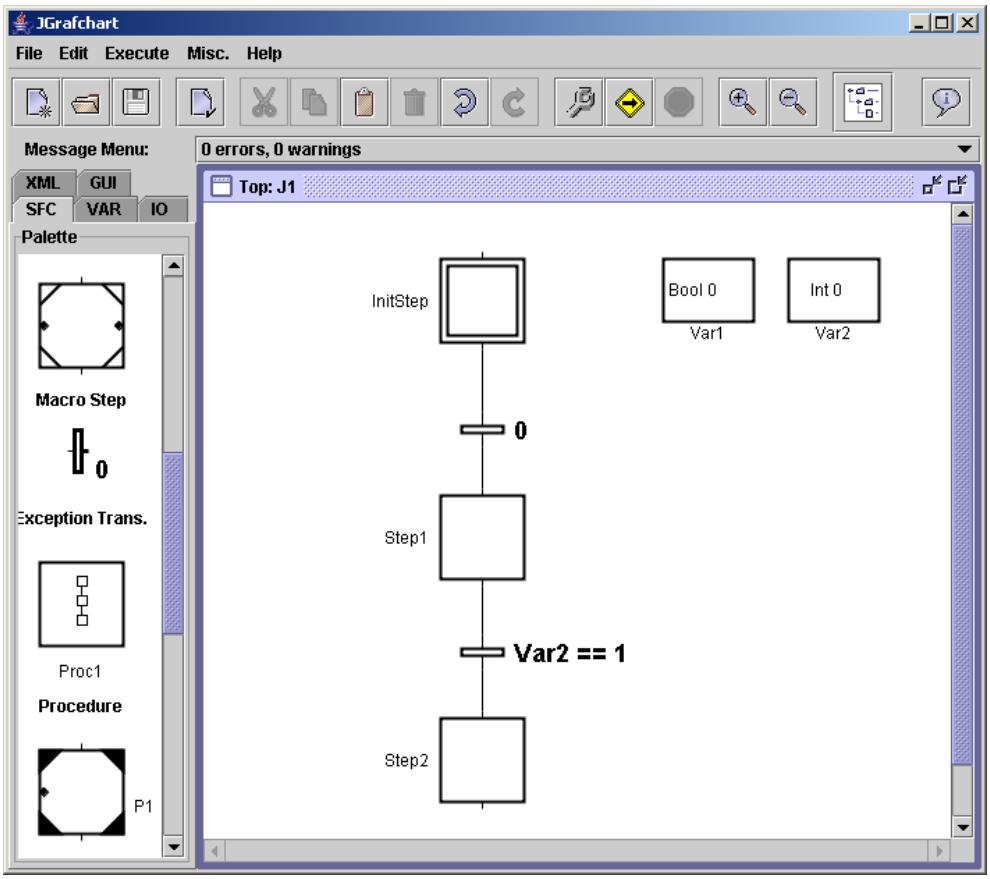

Figure 6.1 A screen shot of the JGrafchart editor.

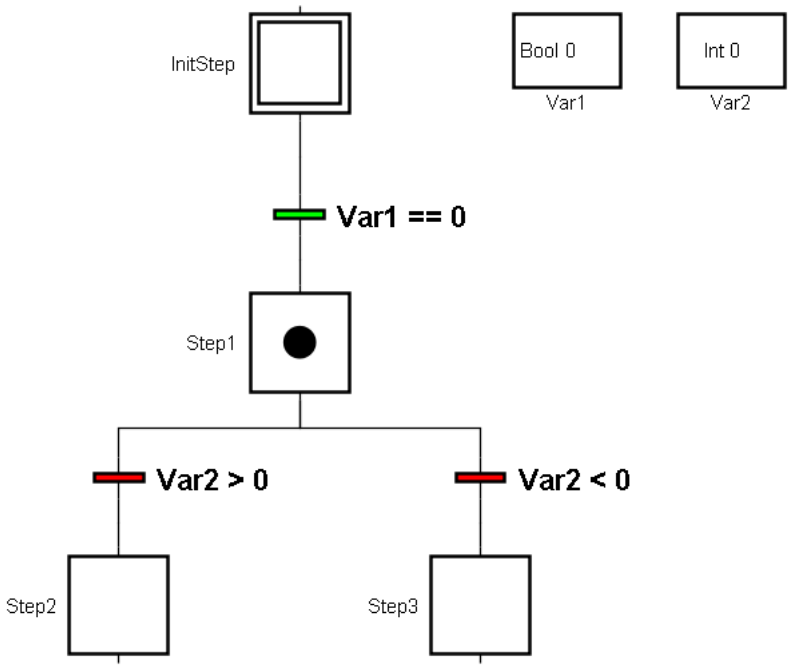

Figure 6.2 A small SFC example from JGrafchart. The model consists of four steps, three step transitions and two variables.

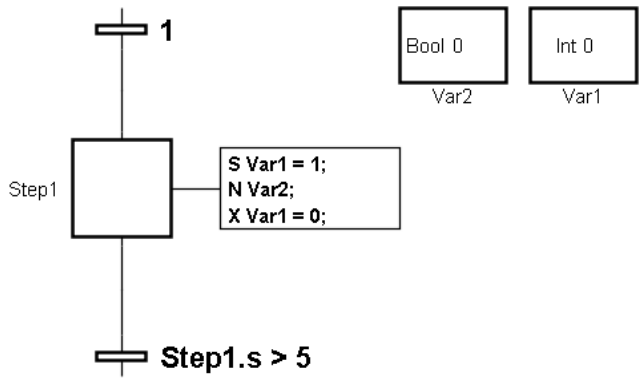

Figure 6.3 The step actions assign values to the two variables `Var1` and `Var2`.

Prefix	*Description*
`S`	Enter Action. With this prefix the action will be executed at the entering moment.
`X`	Exit Action. With this prefix the action will be executed when the token leaves for the next step.
`P`	Periodic Action. This prefix specifies an action that will be repeated as long as the step is active.
`A`	Abort Action. An abort action is executed when a step is aborted due to the firing of an exception transition.
`N`	Normal Action. A line with this prefix is not really an action, instead `N` means that the step will be connected to the value of a boolean variable. By specifying a variable name after the `N` this variable will be set to true when the step is activated and turn false when the token moves on.

Table 6.1 The possible command prefixes used in step actions of the JGrafchart models.

The token will stay in its current position as long as the condition following the active step is false. As soon as the condition turns true the token will continue its journey to the step connected to the other side of the transition. The transition conditions can be used in the same way as conditional logic is used in common textural programming languages. If several step transitions are connected to the same exit of a step the conditions in the step transitions can be used to decide what way the token will take. This is equivalent to "`if... else`" statements in for example Java or C.

The Editor

An important part of the JGrafchart tool is the editor. An overview of the editor window is depicted in Figure 6.1. At the top are the menu and the tool bar. The tool bar contain buttons for easy access of commonly used menu options, for example save, open, compile, run and launching the on-line help. The main part of the editor window consists of the workspace area in which

models are built. To the left of the workspace area is the palette. The palette contains the different components that constitute a JGrafchart model. The components are grouped into several categories, each of them correspond to a separate tab in the palette. By clicking the tabs one can view and access all of the JGrafchart components. The components can be added to the current model by the drag-and-drop method. The blocks and transitions in a model can be connected to each other by "drawing" lines between the connection ports of the blocks and the transitions.

6.3 JGrafchart Components

The different component categories are *SFC*, *IO*, *VAR*, *GUI* and *XML*. Below is a shorter description of each one of the categories. Components from all but the *XML* have been used in this project.

SFC

The SFC group contains the basic elements needed for generating an SFC. Some of these are the steps shown in Figure 6.4. The steps can all execute step

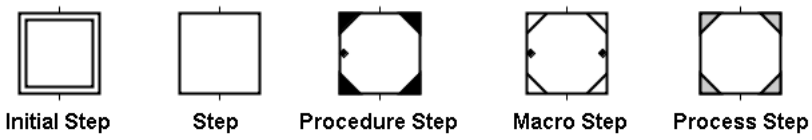

Figure 6.4 The different SFC steps in JGrafchart.

actions when active. The SFC steps in JGrafchart also have three attributes that simplify the modeling. These attributes are a kind of protected variables which can be read but not written to. They are addressed by referring to the block and then appending `.*` to it. The `*` represents any single character of the ones in table 6.2.

Step Attribute	*Description*
`x`	A boolean attribute indicating whether the step is currently active or not.
`t`	A timing attribute that keeps track of how long the step has been active. The unit measured in is sampling periods and can differ between different workspaces in the model.
`s`	A timing attribute that keeps track of how many seconds the step has been executing.

Table 6.2 Step attributes in JGrafchart. Active steps are steps currently visited by a token and is thus equivalent to executing steps. The attributes are addressed in the same manner as object attributes in Java. That is, if the step of interest is `Step1` then `Step1.x` refers to the boolean that indicates if the step is active.

Below is a list of the available steps in JGrafchart.

Step This is one of the very basic components available in JGrafchart. Its only purpose is to execute step actions when activated. Many of the other JGrafchart blocks have the functionality of the step in addition to their own special features. The action step can contain an infinite number of actions of the syntax described above. In other words a single action step can execute many different commands on activation, periodically while active and when it is deactivated. An example is illustrated in Figure 6.3.

Initial Step This is an action step with a special property. An initial defines where the start of the execution will take place. At the start of the model execution all initial steps will have a token inside. Thus, it is possible to have several tokens traversing through the same model simultaneously.

Procedure Step As mentioned before JGrafchart SFCs support function calls. Using a procedure step is one way to do this. The function is defined in a procedure object, which is described further down. When a procedure step makes a procedure call it waits until the procedure is done executing before letting the token move on to the next step.

Process Step The process step offers another way to call a procedure in JGrafchart. In contrary to the procedure step the token in a process step which has just carried out a procedure call is free to continue before the execution of the procedure has finished.

Macro Step This block contains a workspace of its own. Its workspace can store global information of any kind in the same way as the top-level workspace. Inside the macro step there are two special steps, the enter step and exit step. When the macro step is activated the execution of its subworkspace will begin, starting from the enter step. Between the enter and exit steps any combination of transitions and steps can be added. Using macro steps can help organizing control actions into groups that are related to each other. This is a way to improve the overview of a control scheme with many control actions.

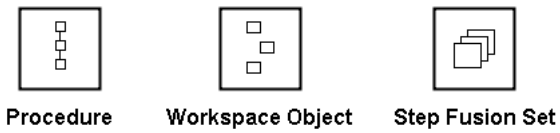

Figure 6.5 Example of JGrafchart components without connection ports.

There are also components which do not have any input or output connections. These are referred to as objects. A list of the available objects is presented below. Figure 6.5 shows some examples.

Workspace Object The Workspace object shares many characteristics with a Java object. This object contains a subworkspace which, with a few exceptions, work just like the top-level workspace of the JGrafchart model. Since the subworkspace have no input or output connection it can not be called in the same way as an ordinary SFC step. However, the workspace object may contain procedures that can be called from other parts of the model. All data and all objects within the workspace object are global and thus accessible from any part of the model.

Procedure The procedure object is something close to a function definition in other programming languages. The procedure object can be called from several steps in the model, even simultaneously if needed. The object can contain action steps and transitions and can be used for doing the same things as a combinations of these two components in the top level workspace. Using procedures provides a higher reusability of the model since desired changes only have to be carried out in one place instead of several. Procedures can also help reducing the file size of the model significantly. A procedure block can not contain any global variables but it can take variables as arguments and it can also return values.

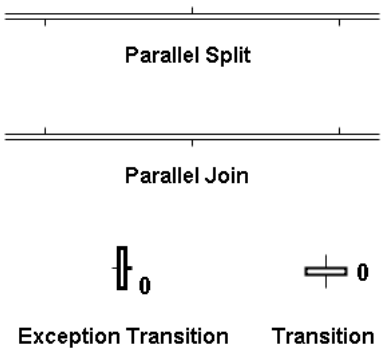

Figure 6.6 Example of JGrafchart components without connection ports.

To successfully build an SFC some other elements are needed as well. In Figure 6.6 transition elements and elements for introducing parallel execution threads are shown.

Parallel Split and Join The parallel split and join are two components that make parallel control sequences possible. Once a token has reached a parallel split it splits in two and continues through both the sequences connected to the split. The first of the tokens that reaches the parallel join will wait until the other token has reached the other input. The tokens then merge and continue as one again.

Step Fusion Set This component provides an alternative to the parallel splits and joins and can simplify complex structures.

Transition The transition is, beside the ordinary step, the most basic element of the JGrafchart SFC components. Every transition has a condition connected to it. If the condition is true, it is possible for a token pass from one step through the transition and on to another step.

Exception Transition This is the equivalent to the step transition that has to be used when connecting to the exception output of a macro step or a procedure step. The exception transitions have higher priority than the normal step transitions, which means that if one each of these are true, the execution of the model will continue through the exception transition.

VAR

Variables are JGrafchart components representing simple data types. The variables in JGrafchart can be used in ways very similar to most other programming languages. The data types in JGrafchart are `int`, `real`, `boolean` and `String`.

Variables can be accessed from anywhere within in the model by using an absolute reference. When addressing a variable or another object within a JGrafchart model the general Java object notation is used, for example, say that a variable by the name `var1` is located in the the workspace object `myObj` in the top-level workspace `Top`. The variable can then be addressed through `Top.myObj.var1`. If the variable is referenced from within the same workspace as `myObj` (in this case `Top`) it can also be addressed through `myObj.var1`. The first alternative on the other hand can be used from anywhere within the model. The different ways to address an object are called respectively, absolute and relative referencing.

GUI

This group of workspace objects provides ways for easy user interaction. Among the objects in this group are the browser, different buttons and the text object. The browser is a convenient way to log messages and provides a good way to back trace and keep track on what has happened during an execution. The text object can be used to make comments in the model, but its message can also be changed automatically during the execution if desired. Buttons can be used to design user friendly models which requires user interaction. The buttons can have actions connected to them in a similar way to steps.

IO

JGrafchart offers features for interfacing with other software or hardware. Using input and output blocks is one way to do this. Input and output blocks are in many ways similar to the JGrafchart variables; they can be read from and assigned values, although the second property is only true for output blocks.

In this thesis all communication with external programs has been carried out through input and output blocks and their provided Java I/O interfaces. These interfaces are designed as a user-configurable way to send and receive information directly from within the JGrafchart model. The four available I/O interfaces are listed in table 6.3. Provided that the interfaces are implemented

Interface Name	*Description*
`AnalogInput`	Provides a way to get external `float`/`double` data into the JGrafchart runtime environment.
`AnalogOutput`	Provides a way to send `float`/`double` data from JGrafchart to external applications.
`DigitalInput`	Provides a way to get external `boolean` data into the JGrafchart runtime environment.
`DigitalOutput`	Provides a way to send `boolean` data from JGrafchart to other applications.

Table 6.3 Java Interfaces that make communication with other software applications through the JGrafchart I/O blocks possible.

in a correct way the corresponding JGrafchart I/O blocks can be connected

to values in for example external databases. This means that as soon as the interfaces are implemented, interaction with external software is very similar to accessing internal variables in the JGrafchart model. Through the interface implementations in this project the I/O blocks manage to handle both the PI System communication and the retrieval of set point recommendations. What system and value the input or output is connected to is determined by the channel string of the object. This string is passed to the interface classes at the start of the model execution and tells the interface what value to connect the I/O object to. Further information on the programming details is presented in Chapter 12.

XML

This group of objects provides other ways for JGrafchart to communicate with external software. No XML objects have not been used in this project.

7. The JGrafchart Model

To make full use of the benefits with a well planned grade change strategy the control has to be performed according to the plan. In this thesis JGrafchart is suggested as the tool ensuring that this criterion is fulfilled. This chapter describes how a defined control scheme can be translated into a JGrafchart model. The idea behind the model in this chapter was to simulate the control actions of a skilled operator with a consistent and well planned execution. The model have options and settings that should be tuned to optimize the performance. It is thus not a complete solution to the grade change control problem, but constitute a framework for optimization and tuning. Details on the strategies used in the design and general discussions about implementation issues are presented herein. The supervision features mentioned earlier was added by extending the original JGrafchart model. This extra functionality is addressed separately in Chapter 8.

7.1 Designing the Model

As described above JGrafchart has several ways to manage the complexity of extensive control problems. As good programming practice it is desirable to make use of these possibilities when generating the function charts. In general the following strategies have been used in the model implementation:

- External data sources have been modeled as Workspace Objects which are placed in the top level of the model.
- The process control has been divided into several parts by using macro steps. The separation of parts is done in accordance with how the operators work.
- Important controller actions such as set point changes are represented as enter actions connected to ordinary steps one by one.
- Process, procedure and macro steps do not have controller actions directly connected to them.

To make use of the advantages of the graphical interface of the JGrafchart models an attempt has been made to keep the actions connected to an action step as simple as possible; even though many controller actions might be scheduled to take place at the same time the model will be much harder to overview if all of them are put in the same block. If a step only contains a single important action it will be easier to give it a short name which explains exactly what it does. This is the meaning of the third item in the list above. More complex control actions can be implemented through a structure of action blocks with simpler actions in either parallel or serial configuration. These sets of blocks might, if desired, be bundled and put into macro steps. To put any commands related to the control among the step actions of a macro, process or procedure step has also been avoided since it makes the model harder to overview.

7.2 Parallel or Serial?

When organizing a grade change control strategy with SFCs one have to know the order in which the different steps should be executed. In theory there are two extremes describing how a function chart for this purpose can be designed. The first alternative is to put all the action steps in a serial configuration, this means that there are no concurrent actions and the execution of one step has to finish before anything else happens. The second alternative is to make all actions parallel which means that there will be no limits for how many changes and adjustments that can be made simultaneously. There are obvious advantages and disadvantages with both the serial and the parallel configuration. This section will describe why none of the extremes are suitable for the grade change control in paper/cartonboard making. It will also explain what designing approach that was used in this thesis.

Serial Implementation

The serial implementation of a control scheme provides a very good overview of the control. It is obvious what has been carried out already and what is left to be done. It is thus easy to see exactly what state the grade change control is in at any moment. The serial approach also guaranties that all changes will happen in the same order every time. This reduces the risk of problems connected to control actions happening in an unpredicted order. The one-change-at-a-time method might also make the outcome of the grade change control easier to analyze; since differences in the control execution between two runs will be easier to spot it should be easier to find the cause of a certain effect.

There are however disadvantages with this approach too. If all blocks are to be outlined in a sequential way (no parallel actions) the control strategy has to be an exact serial sequence. In many cases it is hard to determine the exact order of the required actions and in some cases the perfect order might depend on the state of the process when the grade change execution is begun. The serial configuration also precludes solutions where several actions happens simultaneously and may lead to inefficient control.

Parallel Implementation

The parallel solution is much more flexible than the serial alternative. With the right conditions for the step transitions it can exactly mimic the serial behavior. In addition the parallel approach provides an option to have several control steps executing at the same time. This means that all changes can be carried out independently and that problems with a single action will not prevent the rest from being executed. When separating a large model into several parts, which are known to be independent, this property is very useful.

The downside of the parallel implementation is instead the increased complexity. An easy overview of the control scheme is one of the reasons for using SFCs for grade change support. If one step is currently executing in a parallel function chart this provides no information about what actions that have been conducted or what is left to be done. This uncertainty makes it harder to find errors in the generated JGrafchart model since many more cases need to be studied in a testing phase.

Final Approach

In the presented JGrafchart model the serial and the parallel techniques have been combined. The approach has been to use the serial configuration for control actions that have to take place in a certain order. The rest of the steps have been organized in parallel. The control sequence when switching stock mix to the bottom layer during the change to *Bright* production is an example where the serial design is applied. This particular sequence was addressed in the last part of Section 5.1.

An example of how the parallel approach can improve the overview and functionality is the separation of the four main parts of the control scheme in the JGrafchart model described later in this chapter. The changes conducted in each of them are connected to different parts of, either the pulp preparation section or the boardmachine. The set of actions connected to each part also belongs to the responsibilities of different operators. The parallel configuration of macro steps results in four individual workspaces with a set of closely related control actions in each of them. This way it is easy for the operators to view the changes related to their part of the process. The parallel approach also prevents timing problems in one of main parts from spreading to the other parts. This could be the case if their actions were mixed and nested in a serial configuration.

7.3 Model Description

The JGrafchart model structure is based on the information collected in the preparatory stages of this project described in Chapter 4. This section presents how this information has been translated into a JGrafchart model.

To achieve a user friendly and well organized design the model is implemented in a modular way with several layers of abstraction. The top level of the model is depicted in Figure 7.1.

Top Level

The top level of the model provides an overview of the JGrafchart model. At this level the most important information from the control scheme execution is summarized and presented to the user.

To the upper right of the workspace there are two workspace objects which contain data from outside the model. These are the `Pi` and `Recipes` objects. Their workspace contents are shown in Figure 7.2 and Figure 7.3.

The PI Object The `Pi` object handles the PI communication and stores data retrieved from this system. The `Settings&Such` workspace object inside the PI object contains information about the sampling frequency used for the PI calls. It also contains the SFC that handles the periodic PI calls. Each of the other workspace objects inside the PI object contain PI data collected from PI tags. The data is retrieved with `AnalogInput` blocks.

By using a `AnalogOutput` the value of a specific tag in the PI System can be changed just like changing the value of an internal `real` variable. This feature might be a convenient way to analyze the outcome of the SFC strategy and compare the JGrafchart control to the control performed by the operators. The PI interface currently available supports simple communication through `AnalogOutput` blocks. Simulations with output objects have been conducted,

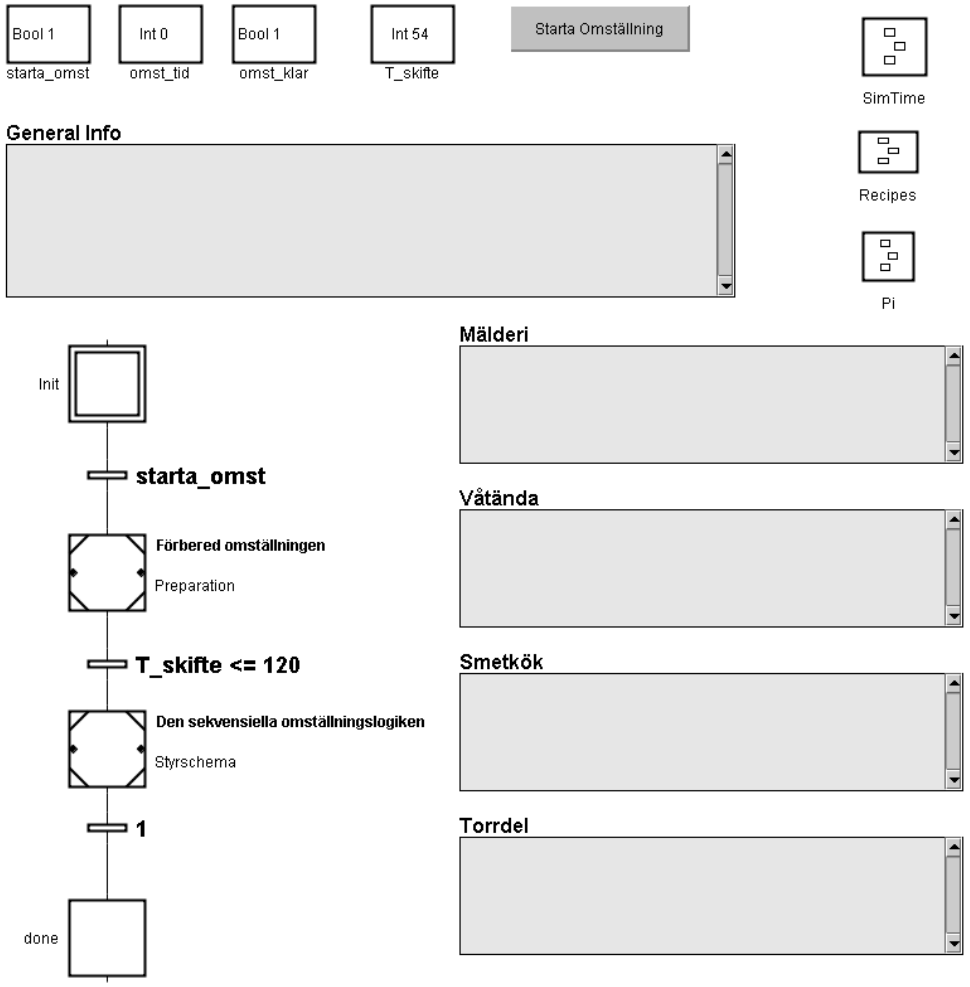

Figure 7.1 A view of the top level workspace of the CRY425-BRI390 SFC.

but the output feature of the PI interface is not as efficient or stable as the input part. There are currently 10 PI tags available for JGrafchart to write to. To log more values this way requires administrative changes in the PI System configuration. No `AnalogOutput` blocks are included in the JGrafchart model described in this thesis.

When addressing a PI value with an input or output object, the channel string of the object has to start with "PI" followed by the tag name of interest. The channel strings of PI I/O blocks are case insensitive. The JGrafchart object references on the other hand are not. This means that when referring to a PI value through an I/O object in the model the name of the object has to be matched exactly. The tag names in the PI System follow a certain naming convention described in [12]. The same strategy has been used when naming the PI tag objects in the JGrafchart model implementation. According to [12] a tag with extension `.PV` is a tag referring to a process value, `.SP` is a set point value and tag names ending with `.OP` are so-called, operating points. If the name extension is the only difference between two tags that should be included in the model, the `AnalogInput` blocks are to be put inside the same workspace object. An example of how an input is represented can be seen in Figure 7.2. For the illustrated object `52E320\_1` only the process value (`.PV`)

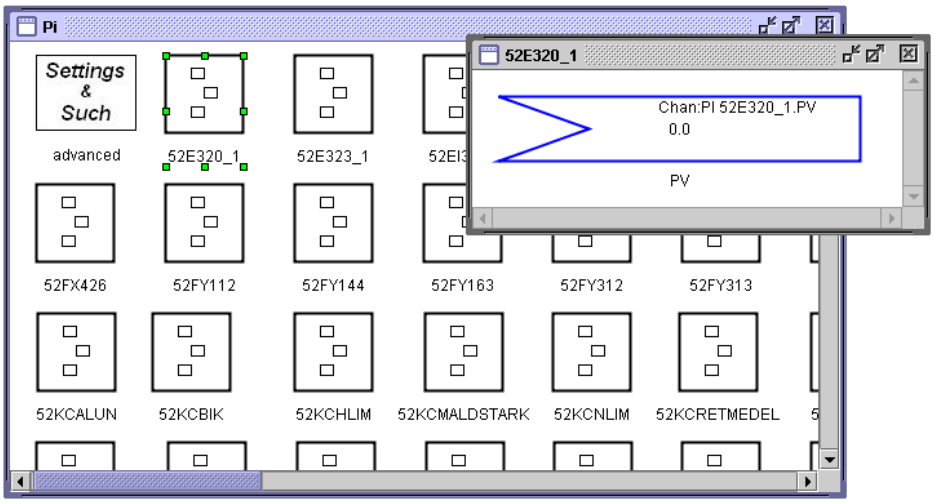

Figure 7.2 A view of the PI object workspace. This object handles the PI communication and stores data retrieved from the PI System.

is used in the model.

The Recipes Object The `Recipes` workspace object and its two contained workspace objects, `BRI390` and `CRY425`, contain set point recommendations that are retrieved from the aforementioned SQL database. Practically this is done through `AnalogInput` blocks with "RV" as the initial word of the channel string. The second token represents for which grade the set point value should be fetched. The terminating number determines for what property the set point value should be fetched. A close up of an input example can be seen in Figure 7.3.

A reference list for variable names and numbers can be found in the database too. Many times the properties have corresponding PI tags which contain the current values for these properties. Two lists of properties can be seen in the Tables 5.1 and 5.2. When present, the corresponding PI tag is included in the rightmost column of the tables.

In practice the values from the SQL database is only retrieved once during the execution. This is done during the preparatory step of the SFC in the top level of the model. To prevent the inputs from updating periodically the *cyclic update* property should be deselected for all the the input objects connected to the SQL data.

The SimTime Object The object `SimTime` in the model makes it possible to use a different time than the current time for the PI calls. This provides a way to test the control strategies by trying them on earlier grade change cases. Experiments have also been done with the use of a "time multiplier". This means that a real-time second correspond to several seconds in the simulated time. Hereby the time of a simulation can be reduced significantly. Setting the "time multiplier" to 2 means that after one real second the two seconds have passed in the simulation time. Using a "time multiplier" not equal to one will not give the expected results for `AnalogOutputs` as the interface is currently implemented. This is one of the reasons why no `AnalogOutput` blocks are included in the

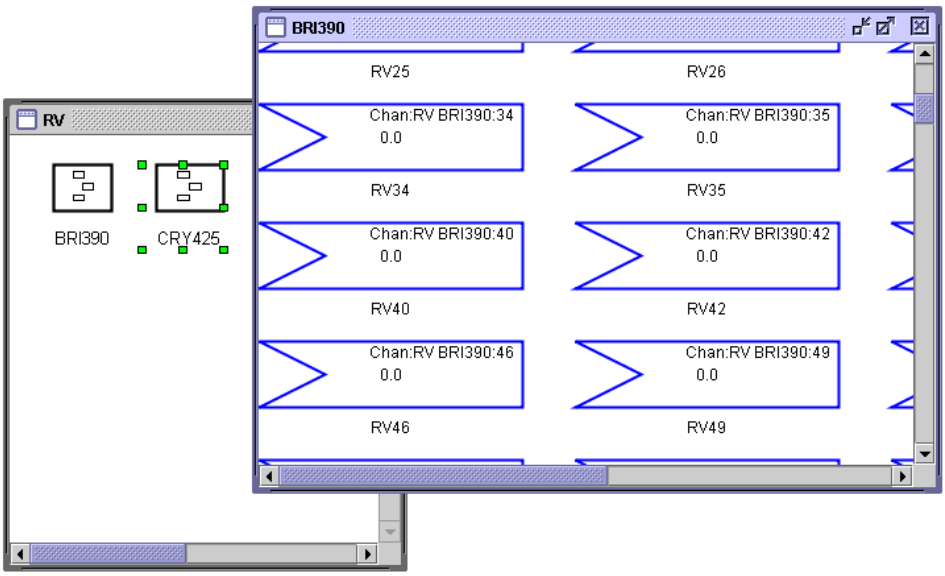

Figure 7.3 The contents of the `Recipes` workspace object. This workspace contain one workspace object with the recommended *BRI390* values and one with the corresponding ones for the *CRY425*.

JGrafchart model described here. Obviously a faster simulation time can only be used when the model runs with old data.

Browsers As can be seen there are also several browsers in the top-level workspace. The browsers are GUI components that provide a way for the different parts of the model to output information to the user. During execution the JGrafchart control actions taken will be printed in these browsers.

Variables There are four variables in the top level workspace of model. The boolean `starta_omst` turns true when the user clicks the start button. In `omst_tid` the elapsed time of the grade change control is stored. The value in `omst_klar` turns true when the grade change control has finished. The fourth variable, `T_skifte`, contain the time left until the change of reels. When producing the last reel of a grade this corresponds to the value of the PI tag 53SPTID2.PV. The value of this tag should be the number of minutes until the value of the PI tag 53PRODBYTE.PV changes. The meaning of the 53PRODBYTE.PV value was discussed in Chapter 5.

The SFC The function chart to the left contains four different steps. The initial step at the top clears the browsers and resets the values of the grade change variables. The second step is executed as soon as the start button is clicked. This step loads the recipe values from the SQL database.

The step `Styrschema` contains the actual grade change control. The contents of this step are presented in detail further down.

When the grade change is finished the last step of the sequence is reached. During the execution of this, results and general information about the outcome will be printed to the main browser.

Navigation Level

The logics and functionality connected to the grade change control is concentrated to the `Styrschema` block. The grade change control step is a macro step that represents another level of the model hierarchy. The contents of this step can be seen in Figure 7.4. As shown in the diagram, this step contains four parallel sequences that correspond to different parts of the boardmachine and the stock preparation processes. This is an intuitive way to organize the required changes. The parts respectively, correspond to tasks that belong to the responsibilities of different operators.

The workspace in Figure 7.4, provides information about the execution status of the different steps. In other words, in what parts of the process that the grade change control sequence have been started, where adjustments are still being made and where the changes are already carried out. This view also provides an easy way to navigate between the grade change control schemes for the different parts of the cartonboard making process. Each of the macro

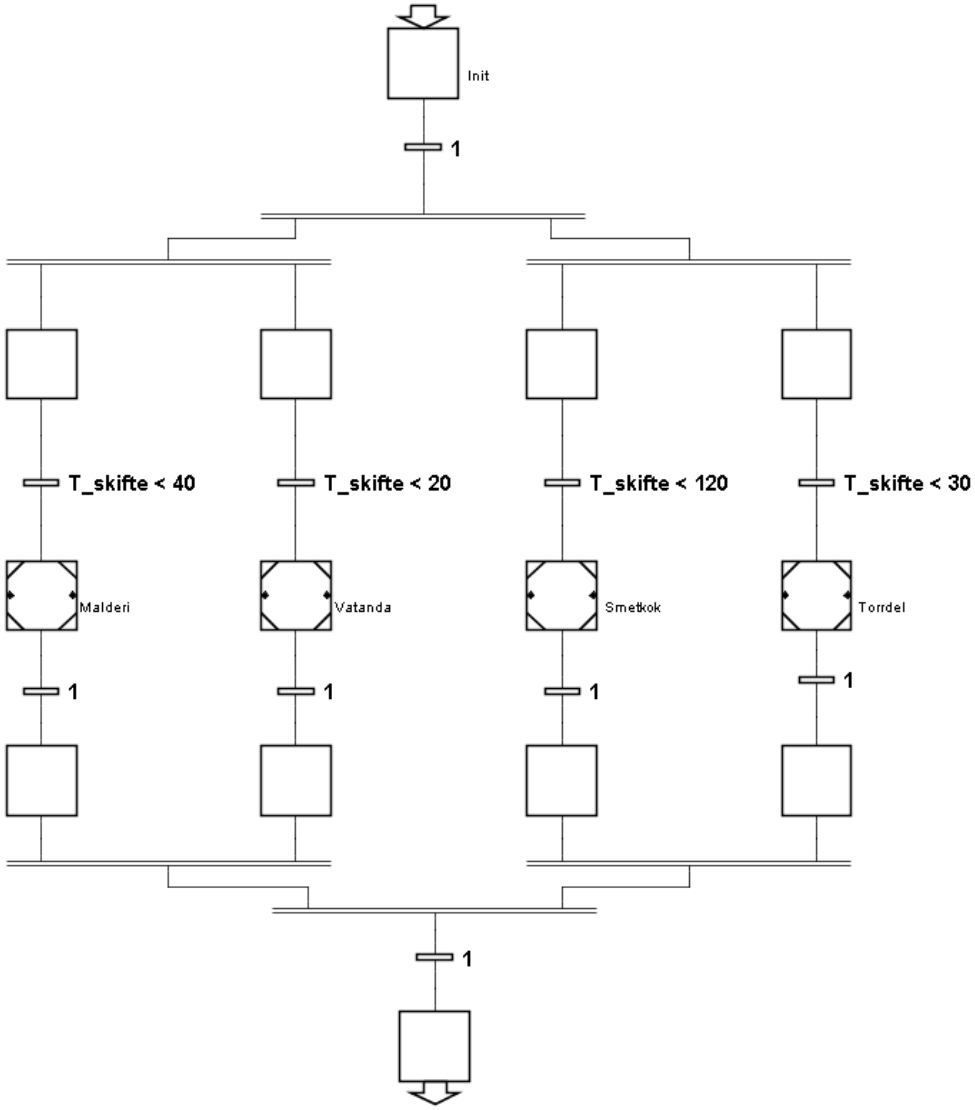

Figure 7.4 Subworkspace of the block `Styrschema`, which contains an overview of the grade change control.

steps contain a subworkspace with other steps. These workspaces are located at the level which we may call the operator level.

Operator Level

The workspaces at the operator level contain steps that correspond to the actions the operators take during a grade change. All parts but the coating color kitchen (`Smetkok`) sequence have been analyzed in this thesis. The remaining three parts are depicted in Figures 7.5, 7.7 and 7.8. These are all operator level workspaces.

The workspaces at the operator level share many characteristic features in their design. At the top they all have variables. These are provided to help the planning and execution of the control. The idea behind the design of the control scheme at the operator level is to have the planning done in a separate preparatory step and then the execution at a later stage of the control. This is a commonly used grade change strategy that is mentioned in for example [5].

What could be done in a planning step is, for example, to have the stored set point values for the next grade adjusted with a small offset relative to the offset for the same variable in the current production. Why set points different from the set point recommendations are used can have different reasons. Sometimes it can be due to effects of calibration problems which are assumed not to change during the grade change.

Also the timing of the steps could be adjusted during the planning phase. As seen in Figure 7.5 some of the transition conditions contain times, which are workspace variables instead of constants defined before the start of the execution. By checking for example the speed of the wire just before the grade change is about to be performed it is possible to determine the time it will take for the changes just before the wet end of the machine to reach the reel-up. Similarly, levels in the mixing tanks can be taken into account when planning the pulp mix adjustments.

The macro steps at the operator level contain components corresponding to operator actions that should be executed during the grade change. They contain steps with actions that execute actual set point change. An example of this is depicted in Figure 7.6.

Stock Preparation Figure 7.5 shows the workspace of the `Malderi` macro step. This macro step contains more steps than the other three in `Styrschema` workspace. This indicates that more adjustments are done in the stock preparation than in the other sections analyzed. The order and placement of the steps has been determined from interviews with the operators. The serial step sequence to the left contains five macro steps. The first two, which are specific for CRY425-BRI390, represent actions stopping the flow of bottom-layer pulp (`BottenStop`) and enabling the bleached coniferous wood pulp flow (`BlektBarr`). The other three describe set point adjustments. Many of these can be taken directly from the recommendations in Table 5.1. The step `MassaMix` corresponds to the change in proportions of the different pulps used in the mixing stages for the separate layers of the board. The last two, `LimKem` and `Malning`, represent adjustments of chemical additives and milling power respectively. The two macro steps parallel to the serial sequence describe a preparatory control of the level in the top-layer mixing tank (`NivaBlKarTS`) and enabling the flow of clay to the top-layer (`Lera`).

Figure 7.6 shows the contents of the `MassaMix` step. The picture illustrates how control actions can be implemented in the JGrafchart model. To the left

is the adjustment of the mix to the top layer mixing tank (`TS`) and to the right is a corresponding step for the mid layer mix (`MS`). In both steps the actions are step changes of the ratios in the flows to the mixing tanks. In the CRY425-BRI390 the top layer pulp mix is changed from 100% birch pulp to X% birch and Y% coniferous wood pulp.

Assume that appropriate values for the ratios are stored in planning the variables `BjorkTS` and `BarrTS`. Suppose further that we have an output block connected to the controller set points for the top layer mix proportions. Under these circumstances the step actions for `TS` can be expressed the following way:

```
S Path.To.TopLayer.BirchPropSetPoint = BjorkTS;
S Path.To.TopLayer.ConiferousPropSetPoint = BarrTS;
```

Wet End The contents of `Vatanda` in the `Styrschema` workspace is shown in Figure 7.7. The steps correspond to the things the operators in the wet end do during the grade change. Step `BottenMassa` represents the switch from ordinary bottom-layer stock for the bottom-layer to the same pulp as is supplied to the top-layer. `Rampning` describes how the controller set points for the wire speed and basis weight should be adjusted. Goal values for these parameters can be found in Table 5.1.

Drying Section The drying section in Figure 7.8 is not as well surveyed as the other two and needs completion. The two steps that are included represent drying changes to meet the different humidity set points in Table 5.1 (`Fukt`), and switching to the right kind of coating color (`Bestrykning`), which is also specified in the same table.

Coating Color Kitchen Control Most of the coating color kitchen production is done in batches. At AssiDomän Frövi also other mixes of chemicals besides the coating color are prepared by the operators in the coating color kitchen. A grade change in this part of the process has to be planned at an early stage, usually at least 24 hours ahead. Some actions in the coating color kitchen are therefore required long before the grade change is started in the other parts. In addition to the early changes there are a couple of steps in the coating color kitchen control that should be executed around the same time as the grade change adjustments are made in for example the drying section. An example is the switch between what tank to take the coating color from. Changes of this kind should be possible to include in the JGrafchart model but this has not been done yet.

7.4 Discussion

The idea behind the model presented in this chapter is to mimic the behavior of a group of experienced and consistent operators. For well-defined control sequences this is a very attractive solution. The planning ideas provide options for optimization. With good and reliable models it would be possible to extend the optimization features further by implementing interfaces with possibilities to communicate with for example the Mathworks Simulink or Dynasim Dymola Modeling environments in real-time. This would open a door to more advanced, dynamic, optimization of the grade change control.

If more advanced planning should be carried out however, it might be a good idea to revise the model design guidelines for storing planning data. This is because the current method, where planning set points are stored as variables at the operator level and below, will make the model bloated if a lot of data has to be stored. Perhaps the model can be made more efficient if all planning is done at a stage before the actual control is started.

Another benefit with the model is the good overview of the control. The main reasons behind this are the separation of the grade change control into four main parts and the habit of connecting the different changes to separate steps. The possibility to follow the grade change execution makes it easier to supervise and find problems.

The most important problem with the presented technique is the difficulty in defining control sequences that are simple to translate into SFCs. In a grade change process decisions are based upon a lot of different information. To summarize the experiences which the operators base their control on into simple rules that can be translated to JGrafchart SFCs is a challenging task.

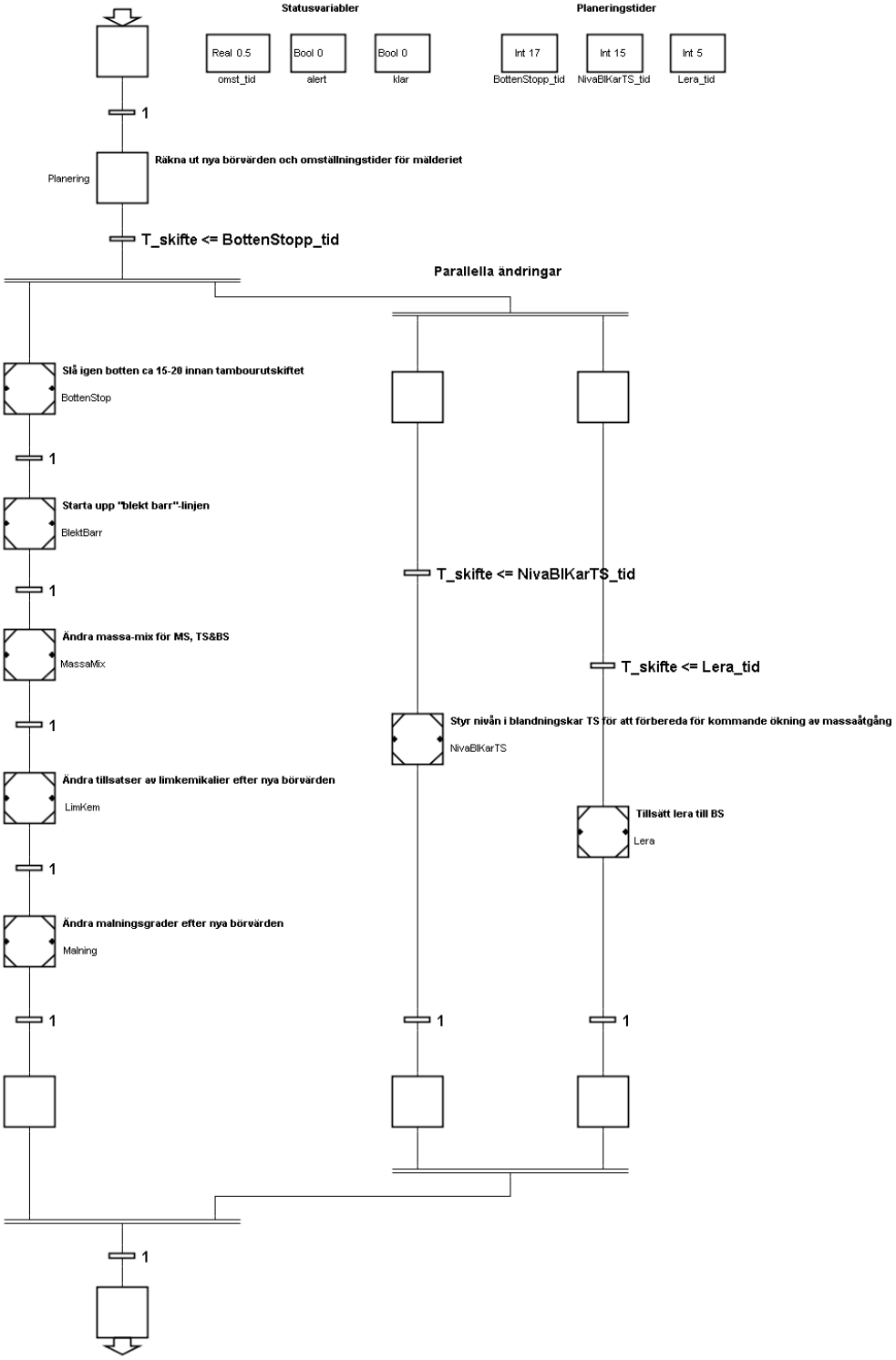

Figure 7.5 The contents of the stock preparation macro step.

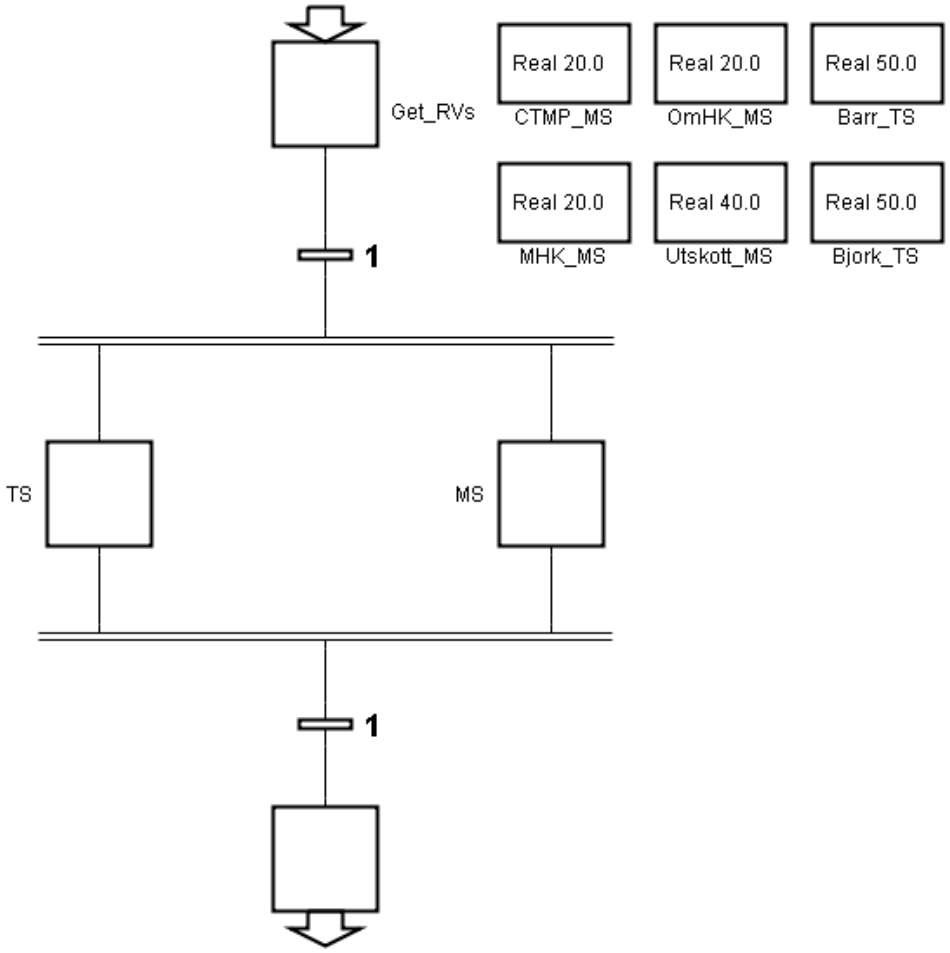

Figure 7.6 The contens of the pulp stock proportion adjustments. The mix change times can be scheduled differently due to the parallel design. (The values of the mix proportions are assumed.)

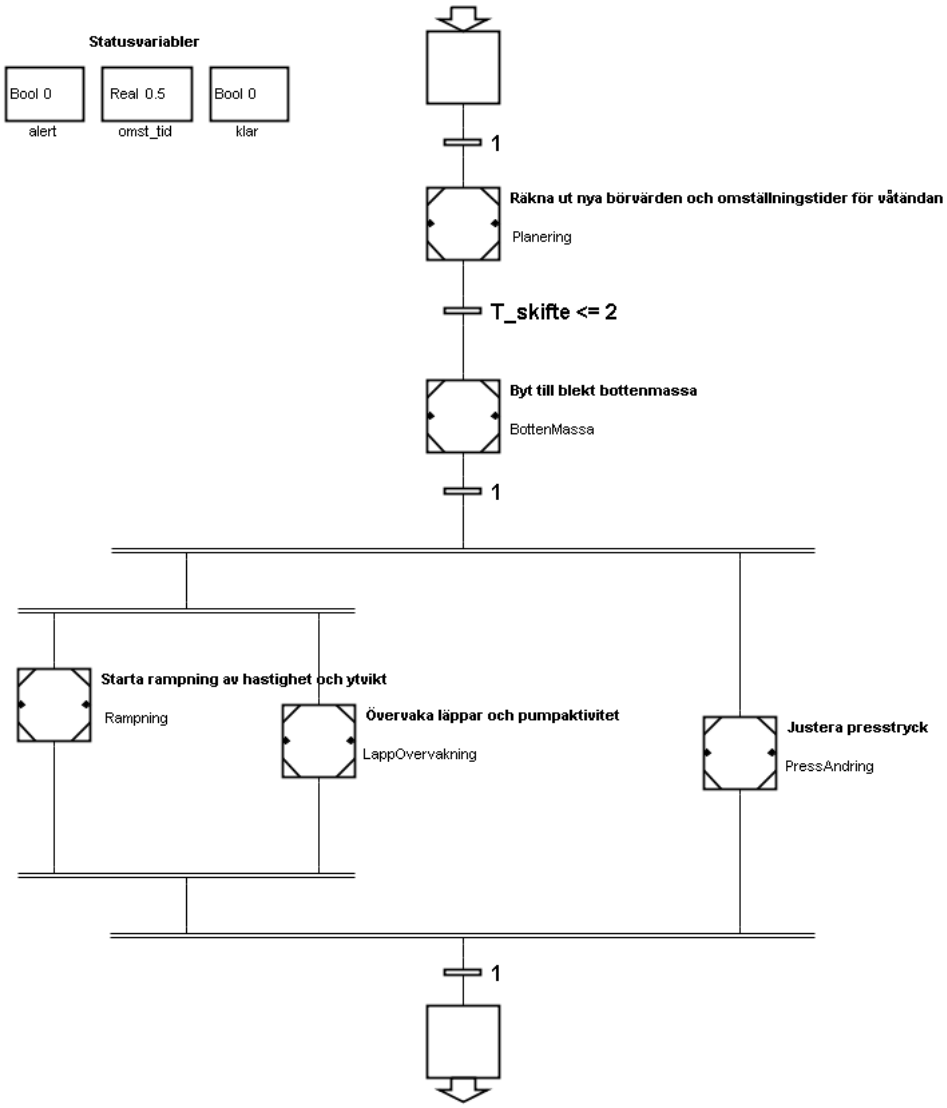

Figure 7.7 The contents of the wet end macro step.

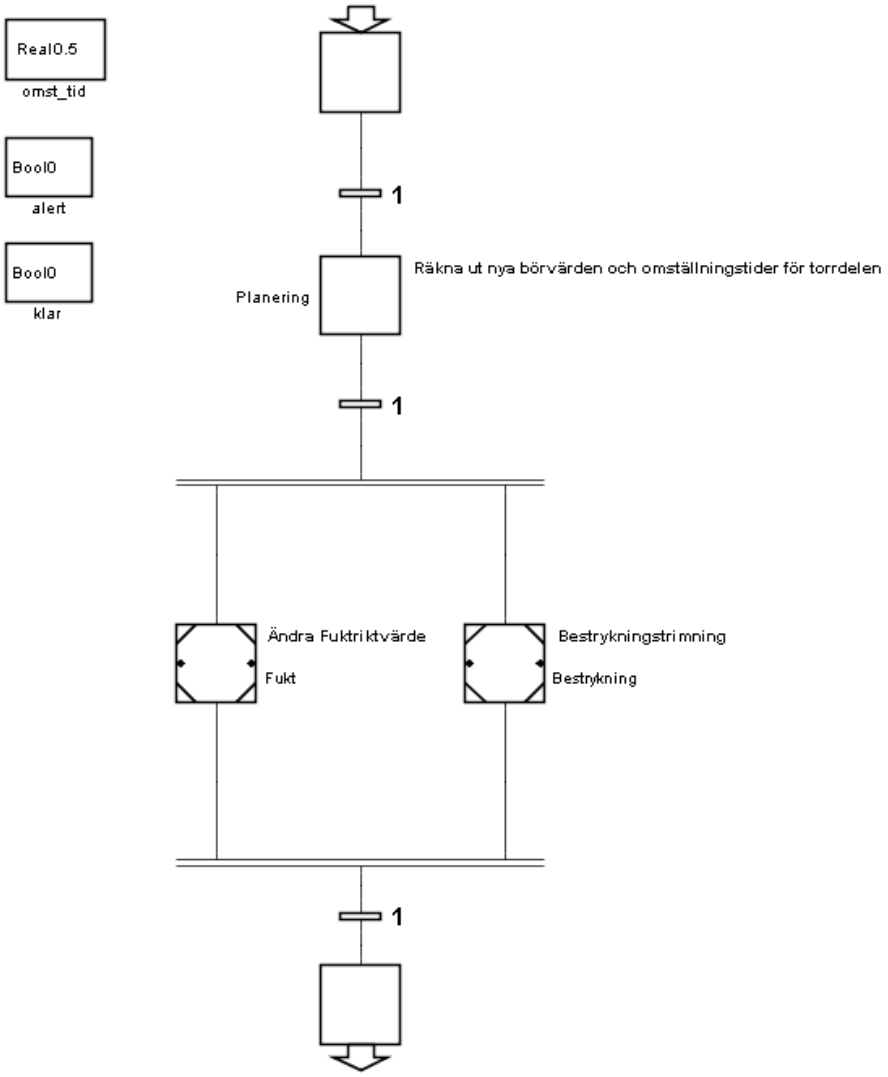

Figure 7.8 The contents of the drying section macro step.

8. Grade Change Supervision

As mentioned before, the use of JGrafchart in this thesis differs from previous cases. Instead of only being a tool for sequential control the JGrafchart model is also supposed to supervise the set point changes and timings to reduce the risk of problems due to operator errors or regulator problems. These new functions add some complexity to the overall problem and require the development of an extended modeling strategy. In this chapter problems, solutions and thoughts connected to the supervision functionality are discussed.

Some slight modifications and changes in the configuration of the JGrafchart source code have also been necessary. These changes include features for showing and hiding pop-up windows and support for check box buttons.

8.1 Why Supervise?

During the time spent on this project the reason why to implement a supervising feature in the JGrafchart model has become more clear. Having an automated sequential control tool in mind a feature like this might seem somewhat redundant. In this case the goal is however, not a fully automated grade change control, but something closer to a semi-automated solution. Instead of having JGrafchart controlling all of the adjustments and tunings connected to a grade change it will act more like an advisor, it is up to the operators in the control rooms to make sure that the steps of the control strategy is performed accordingly. Keeping this in mind it is easier to understand that also supervision of the control sequence would be helpful.

As mentioned in the first chapter forgotten and badly performed adjustments might not be spotted until hours have passed. This is an expensive worst-case scenario that could be prevented if there were supervising functions in the JGrafchart model that showed warning messages about steps that are not performed, already during the grade change.

8.2 How to Supervise

A good monitoring model should include functions to warn for as many errors as possible. This does not mean that all warnings from a software system necessarily are good. When looking at the problem from the operator's point of view one can easily imagine that too many and/or unnecessary warnings from a software system result in a lack of confidence for the tool. For example, it is possible that one problem that occur at an early stage will be the cause of many others. If many warnings show at the same time it will be very hard for operator to identify exactly what is the cause. It is therefore very important that the supervision and warning system is both reliable and accurate. For one thing, it should not show confusing or unnecessary warning messages.

Most of the general strategies presented in Chapter 7 can be used in the supervision model. These include all the items listed in Section 7.1. The general design approach of mixing the parallel and serial technique from that chapter has also been used here.

In addition to these ideas there are also new design issues to consider. The general ideas when extending the simpler model with extra supervision features can be summarized in the following way.

- Add supervision to every control action that is proposed.
- Avoid unnecessary warnings by having old warnings disappear automatically when the required actions are carried out.
- Use configurable warning levels and timings.
- Make it easy to find the problem with only the information supplied in the warning message.

8.3 The Supervision Model

To achieve the goals mentioned above the ordinary steps that simulated the execution of the changes from the previous chapter have been exchanged for a combination of a procedure step and a set of workspace objects. Figure 8.1 illustrates how a change step is represented in the supervision model. Every

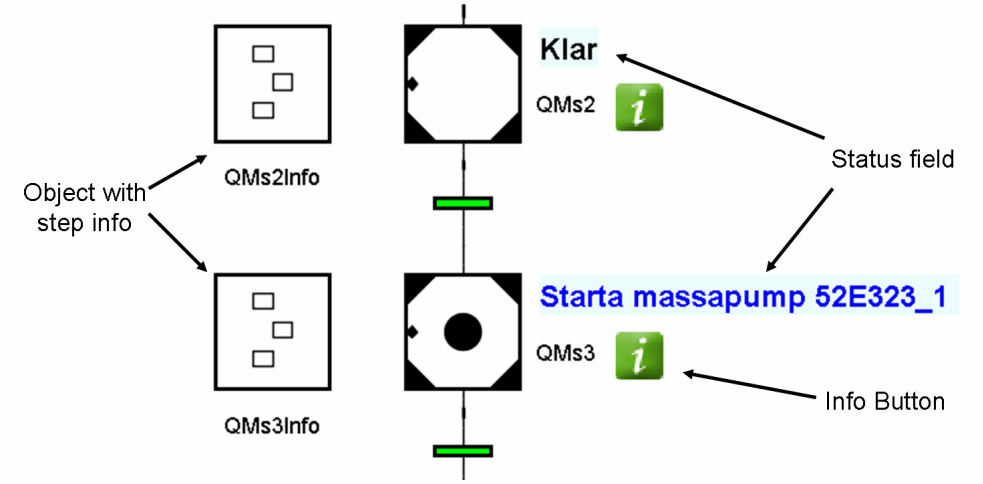

Figure 8.1 Detailed view of a step in the supervision model. The picture also shows the objects connected to the step.

procedure step is closely connected to a workspace object referred to as an information object. This object contains step specific information, including what variables (PI tags) to supervise, the target interval for each of them and how long to wait before alarming. Among the information stored is also a shorter description of the action that should be executed in the change step.

The information about the status of a step change can reach the user in two ways. Firstly, it can be seen directly by looking at the status field to the right of the process step of interest. When a change is suggested the field shows a short description about what should be carried out. The text is in blue. If the action is not executed fast enough the status field text changes color to red, indicating that the action still has not been carried out. When a suggested action is executed the status text changes to “Klar” (done). Figure 8.2 shows a shorter sequence from the implemented supervision model that illustrates the

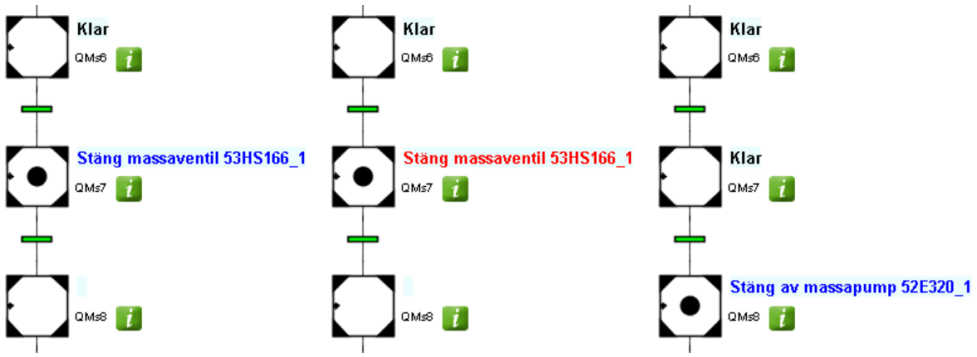

Figure 8.2 Three images from the execution of the supervision model. Furthest to the left the JGrafchart model tells the user that an action is waiting to be carried out (text in blue). In the mid sequence the planned change `QMs7` has not been carried out within the acceptable time interval and JGrafchart therefore changes the step status text color to red. To the right the `QMs7` action has been carried out, and the model suggests another action (blue text).

status changes. The steps corresponding information objects have been left out of the figure of space reasons.

The other way in which JGrafchart can inform the operator of the change status is through pop-up windows with status information. These are also referred to as warnings and appear at the same time as the status text color changes to red. An example of what one of these can look like can be seen in Figure 8.3.

Figure 8.3 A pop-up advice, which shows that a step has not been executed according to the plan.

The supervision model also has some changes in the top-level workspace. The new top-level workspace is illustrated in Figure 8.4. Some of the objects are new while others are inherited from the model in the previous chapter. Objects of the latter kind are `Pi`, `SimTime` and `RV` (called `Recipes` in the previous chapter), which are the same as in the model in Chapter 7.

8.4 The Supervision Procedure

When the procedure steps in the model are activated, that is get visited by the token, they call a procedure named `OperatorAdvice` in the top-level workspace. This procedure handles the supervision of the specified process values and also

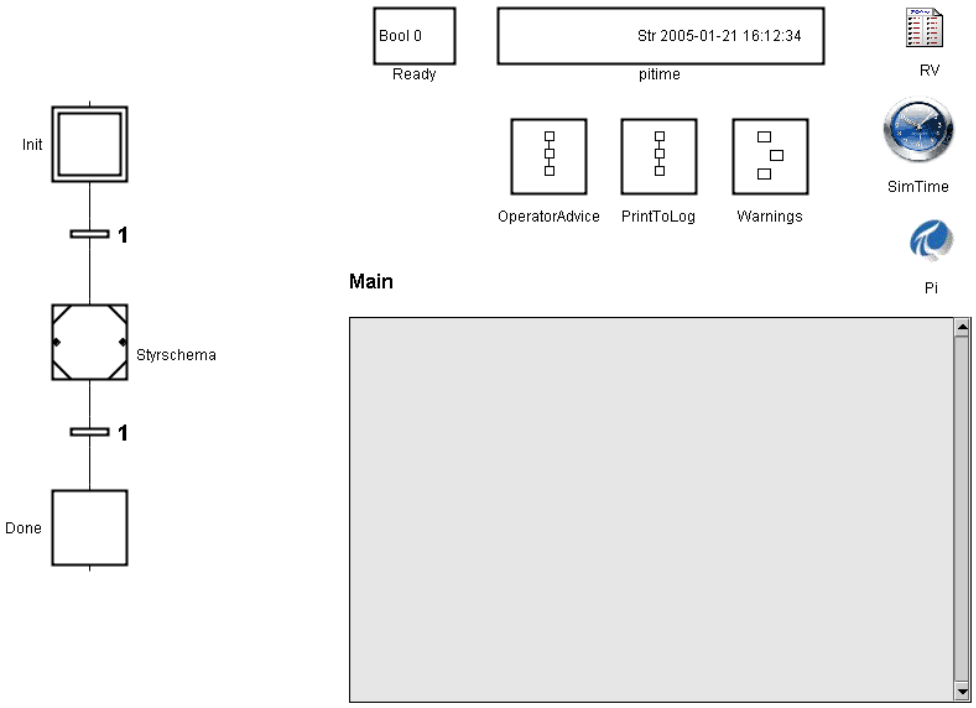

Figure 8.4 Overview of top level workspace in the supervision model.

communicates with the user to let him or her know about the grade change status. The mentioned features are provided through calls to other procedures placed, either directly in the top-level of the JGrafchart model, or in a workspace object in this workspace.

The supervision procedure has the ability to supervise four tags for every call. This means that each procedure step can be used for supervision of four variables. As mentioned in Chapter 7 it is generally a good idea to avoid having many changes among the step actions of a single step. The same way of reasoning can be applied to the supervision case too and the four tag limitation is therefore not serious.

Though it might be tempting to have only one change in every single step this, on the other hand, makes the physical size of the model very large and impossible to overview. Therefore it is sometimes preferable to group changes that are closely related into a common procedure step.

When the supervision procedure is called it will start monitoring the set point variables passed from the procedure step. This is done in a loop which will execute until all variables are within their target zones.

If the levels of acceptance are not met after a predefined number of seconds the procedure will make a hidden workspace visible to the user. This workspace is called a pop-up window or a warning. The `OperatorAdvice` procedure also changes the color of the status text next to the process step.

During its execution the procedure makes use of other procedures, the `PrintToLog` procedure and two procedures in the `Warnings` workspace object. `PrintToLog` handles logging of events and warnings. It includes options for both global and localized logging in lower levels of the model. The `Warnings` object and its contained procedures keep track of warnings make sure that a pop-up window will not hide other warnings when it is shown.

8.5 Supervision of the Wet End Control

To further illustrate what a more complete supervision model looks like this section includes a translation of the wet end control (`Vatanda`) SFC from Chapter 7. Figure 8.5 shows the supervision version contents of this step. The most obvious difference between the two workspaces is the single step `BottenMassa` (Bottom-layer pulp) from the first model, which has been expanded to an eight-step step sequence. This sequence describes how the change of the bottom-layer pulp should be executed and is unique for CRY425-BRI390. Exactly how this should be done is covered in the handbook [19] and it is therefore straightforward to include it in the model.

A second difference is the extra step `Drag`, which is meant to supervise the force that keeps the web stretched from the point at which the web leaves the wire until it reaches the pressing section.

The short names of the steps `QMs1`...`QMs8` refer to the steps from the handbook's pulp change control scheme. For example `QMs1` corresponds to first step of the scheme and represents the disabling of the HMPC controller. The short names do not reveal any details about the actions they refer to. More detailed information can however easily be accessed, by opening the workspace of the information object to the left of the step. Opening the object's workspace can be done through a single click of the green info button to the right of the step. An example of the contents of the information objects can be seen in Figure 8.6. This figure shows the workspace of the `QMs8` object.

As can be seen from Figure 8.5 one of the steps are missing from the JGrafchart model, namely `QMs5`. The reason to this is that it is impossible to get the required information about the execution status of this step from the PI System. At first this was also the case for `QMs2`, which is a preparatory change done by operators to unlock some of the control settings. Recently, however, a PI tag representing the `QMs2` change has been added; this means that the step is now included. There is however still no old data stored for this tag, which makes it uninteresting to include when executing the model with old data for testing purposes.

8.6 Component Library

As can be seen in for example Figure 8.5 some components appear repeatedly in organized groups. The groups consist of either a procedure or process step, a status text field, a button and an information object. To apply the presented supervision concept to new cases would be simplified if the groups existed as single components in a component library. A simple library could therefore be a read-only JGrafchart model in which the components of interest are included. To implement a new supervision model could then be done by using the copy and paste functions in the JGrafchart editor to import components from the library to the model. JGrafchart offers ways for grouping objects. This function make moving and selecting multiple objects easier.

There is however a problem with a library solution like this. The components of a group reference to each other by using the component names. The references are relative to the workspace which the group reside in.

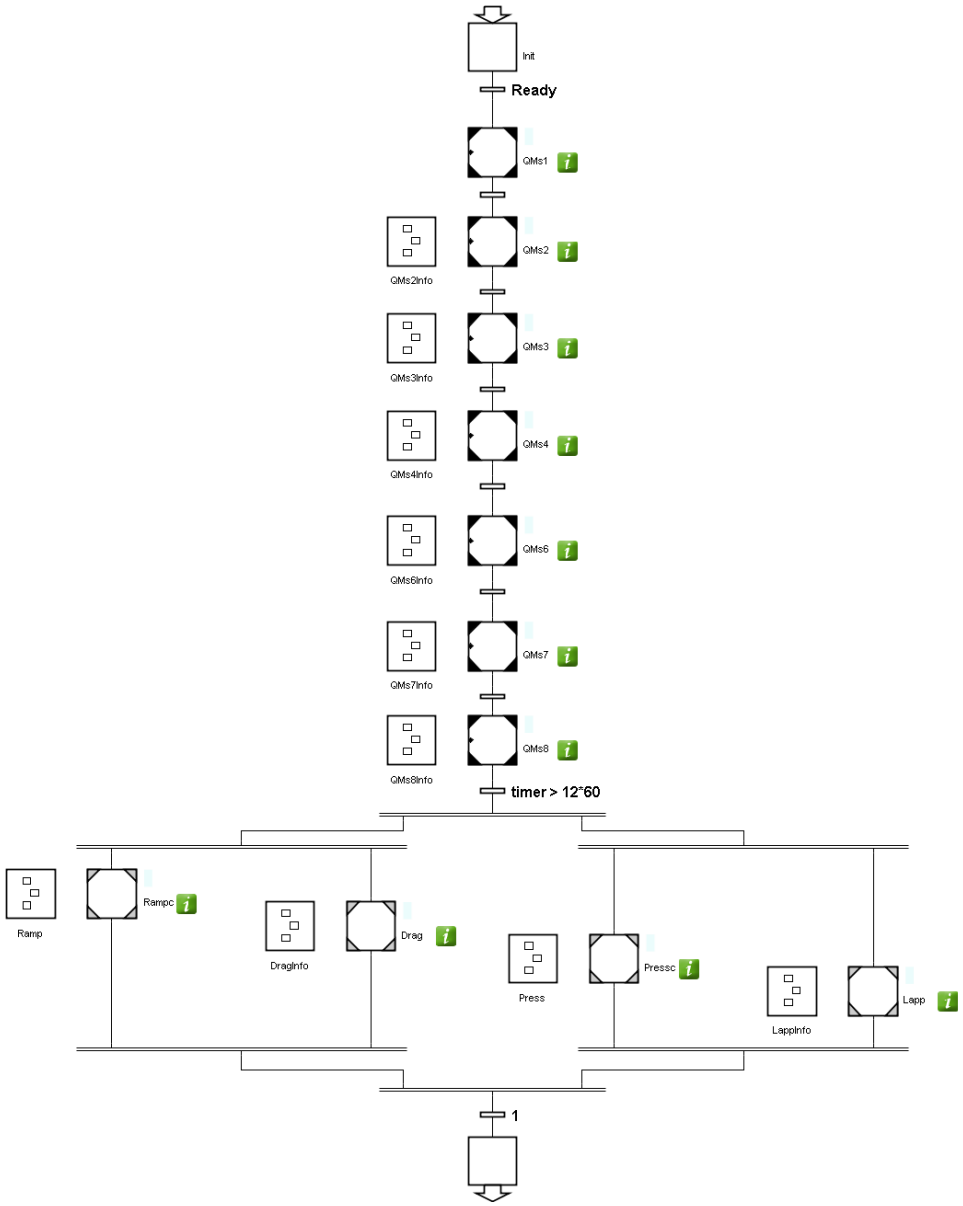

Figure 8.5 The wet end grade change control supervision. Part of this function chart is an implementation of the detailed control plan described in [19].

Example

Say that we have one of these groups in the workspace named `MyWorkSpace`. When clicked, the button of this group shows the workspace of the information object within the same group. If the information object is named `MyInfoObj` the action command of the button would be `showWorkspace(MyInfoObj)`.

This means that when a new group is created from an existing copy all object names and references of the copy have to be changed. This is required since all object names have to be unique and the components must refer to the objects within their own group. For example we could call the new information object `MyInfoObj2` and change the reference in the button command of the

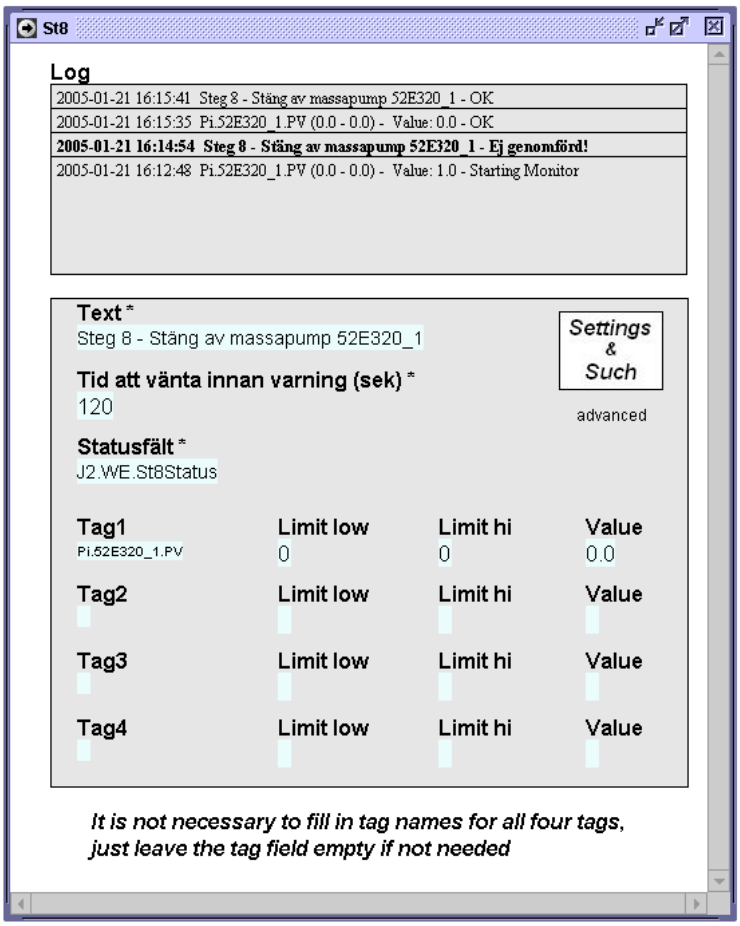

Figure 8.6 The contents of the `QMs8Info` object. In the browser at the top information about the execution of the changes connected to `QMs8` can be found. The text fields below describe what changes are needed for the execution of step to be considered done.

new button to `showWorkspace(MyInfoObj2)`. This configuration step is time consuming and may also introduce errors in the model if any of the references is wrongly configured.

If instead the grouping feature provided its own namespace and references inside a group were relative to the group itself the problem would be overcome. This is explained in the following example:

Example

Say that this time we have two component groups by the name `MyGroup` and `MyGroup2`. The information object in both groups are called `MyInfoObj`. By using the button command `showWorkspace(MyInfoObj)` for the buttons in both groups would then make the button in `MyGroup` point to `MyGroup.MyInfoObj` and button in `MyGroup2` point to `MyGroup2.MyInfoObj`, which is exactly as desired.

There is however a limitation in JGrafchart that makes this implementation impossible; no objects in the same workspace can have the same name. This means that even though it would be possible to distinguish between all objects by using the group names in the references the JGrafchart compiler does not

allow it.

Because of the described limitation no component library has yet been implemented. If the issue connected to relative references in a group is resolved it should be possible to make new models with libraries of grouped components. The only thing that then would need to be changed for the imported copies would be the group names.

A solution might be to implement the grouping function as a "phantom workspace"; an object that share some characteristics with the workspace object, but also lack some. A "phantom workspace" would provide the extra namespace required for distinct references to objects with the same name but residing in different groups. The extra level of abstraction in which the objects of an ordinary workspace are contained inside the workspace object and hidden to the user's view is however not wanted.

9. Process Models

A complete model of the board making process including the stock preparation process and the board machine is under development at AssiDomän Frövi. The main tool used in this modeling project is the software application Dymola which is developed by Dynasim. Dymola is a graphical modeling tool in which mathematical models can be implemented by using components from component libraries. Development, validation and simulation of the board making model have been conducted in several smaller projects. Some of these are the master thesis projects [20], [21], [22] and [23].

9.1 The Dymola Model

The Dymola model structure constitutes of several parts. Figure 9.1 illustrates the different subprocesses. In the pulping process the pulp is prepared. After

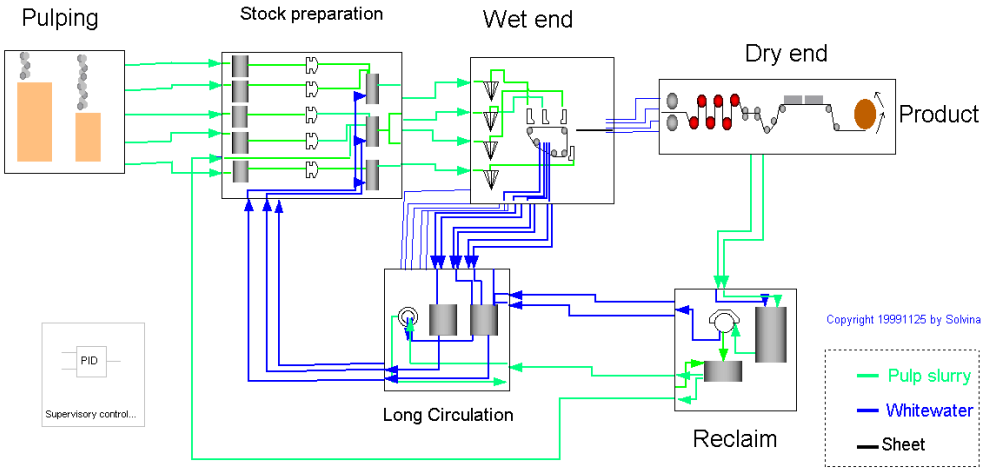

Figure 9.1 Overview of the Dymola/Modelica of the cartonboard making process.

this step, the pulp enters into the stock preparation. Figure 9.2 is a close-up of the stock preparation process. As can be seen from the figure this step contains three subprocesses. Figure 9.3 shows the equivalent for the wet end subprocess. In the optimization part of this project only the mixing part of the model has been analyzed. The contents of the mixing stage for the top layer is illustrated in Figure 2.3. A more detailed view of the first mixing tank in this flow is shown in Figure 9.4. The libraries at AssiDomän Frövi contain for example tanks, pipes, pumps and regulators. Examples of these components can be seen in the figure.

9.2 Optimization Models

The simple models used in the optimization part of this project are based on data and characteristics from the available Dymola model but is less detailed and complex than the original. Most of the modeling has been done in the

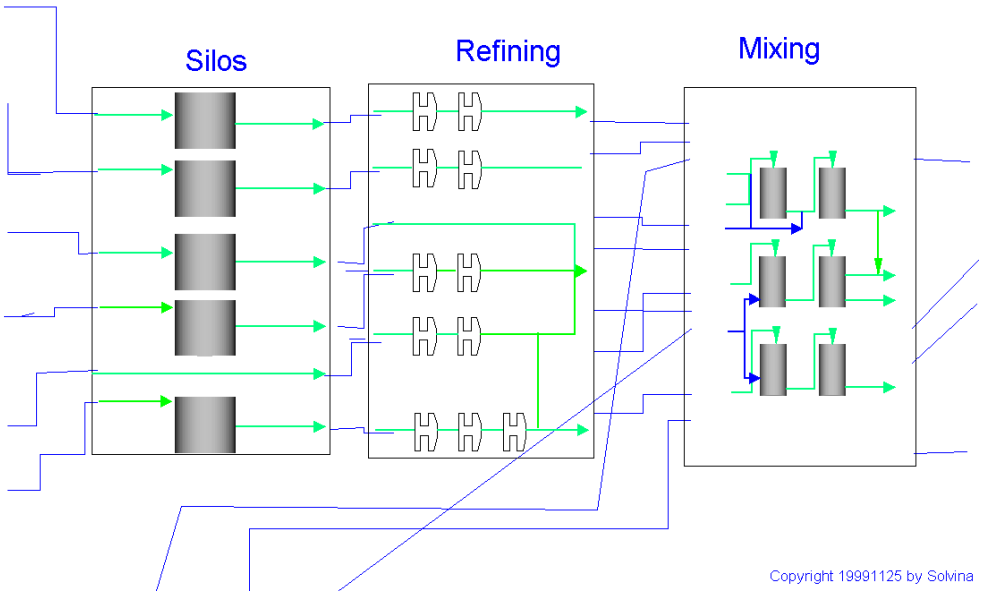

Figure 9.2 Overview of the stock preparation stage.

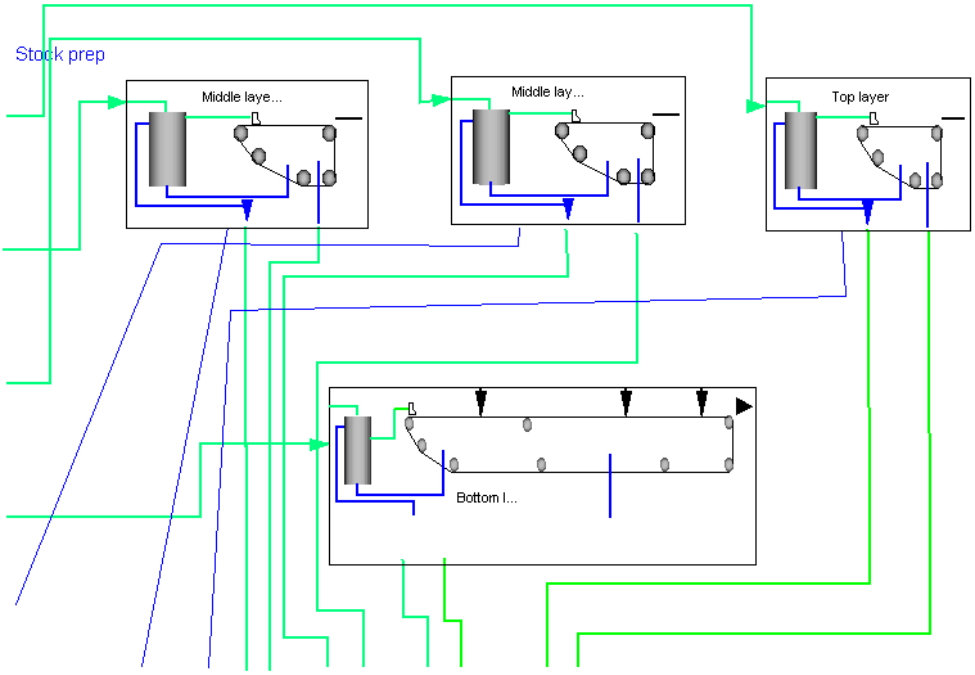

Figure 9.3 Overview of the wet end process.

Matlab Simulink environment. The effects of the dewatering dynamics in the wet end and the short and long circulations have been ignored. Most of these effects are ignored in the current Dymola model version too. Below are brief descriptions on the simple models used for the different parts of the process.

Pipes

In the set of tanks, mill, pipes et cetera that constitutes to the stock preparation and board machine an important part consists of analyzing flows in pipes. The effects of pipes in general, are time delays that depend on the flow rate. To

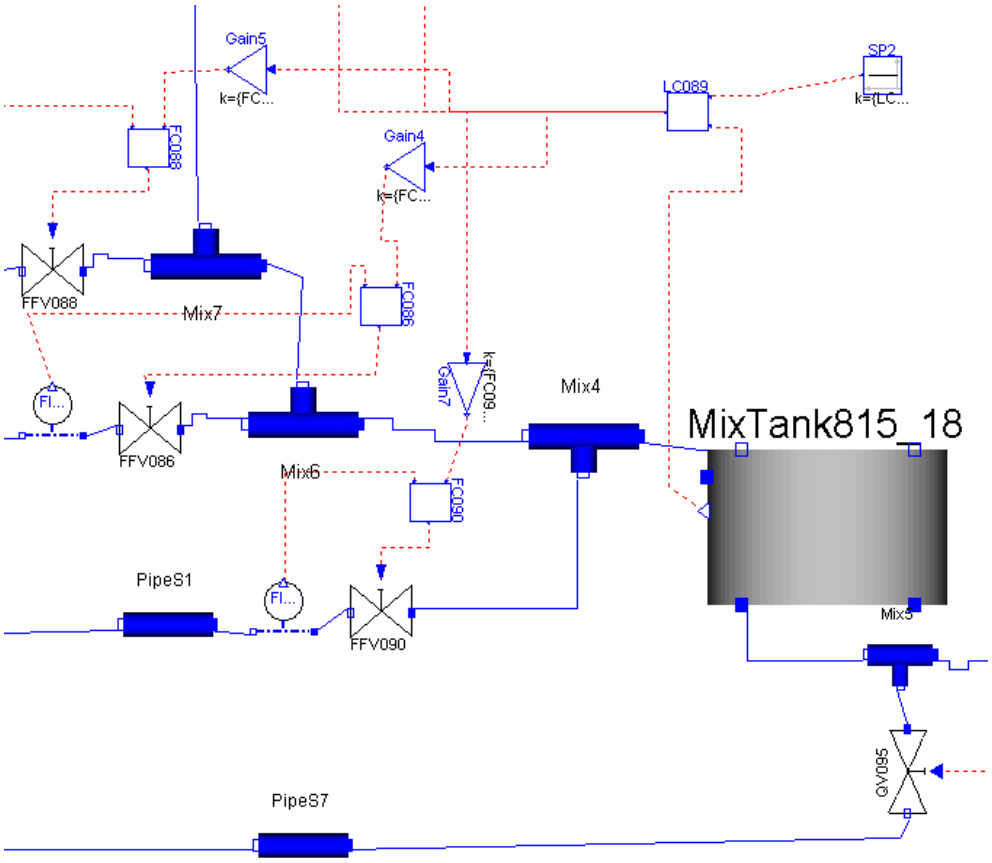

Figure 9.4 Close-up of the first mixing tank for the top layer.

model this, different techniques can be used. In this thesis the method of choice has been the compartment model. This means that every pipe is is modeled as several states where every state corresponds to a section of the original pipe. In a more practical way of speaking the pipe is chopped up in pieces resulting in a discretization of the pipe in position:

$$V_k \frac{dc_{A_k}(t)}{dt} = q(C_{A_{k-1}}(t) - C_{A_k}(t)) \tag{9.1}$$

An alternative way of handling pipe delays is presented in [17].

Tanks

The dynamics of the mixing and machine tanks are assumed to be perfect mixing. Thus, the following equations have been used

$$A \cdot \rho \frac{dL}{dt} = \sum_{i=1}^{j} Q_i^{in} - \sum_{i=1}^{k} Q_i^{ut}. \tag{9.2}$$

Here A is the cross-section area of the tank, ρ is the density of the fluid in the tank, L is the level of the surface in the tank and Q represents the mass flows into and out from the tank.

Boardmachine Delay

The grade change delay of the boardmachine has been modeled in a very simple way. The only effect taken into account is the time it takes for a fiber to get from the headbox to the reel-up, assuming that no fibers pass through the wire in the wet end and is fed back through the short circulation system. This assumption is apparently not correct but is a commonly used strategy for simple analysis.

Under this assumption only two parameters need to be known in order to calculate the time it takes for the stock to get from the beginning to the end of the board machine. These are the speed of the wire (approximately the same as the angular velocity of the pope) together with the length of the web. With this at hand the following equations can be used to find the delay time of the board machine.

$$L = \int_0^{T_{bm}} v \, dx \tag{9.3}$$

Here T_{bm} is the delay time, L is the length of the web and v is the speed of the wire. Assuming the wire speed to be constant or slowly varying this can the simplified as

$$T_{bm} = \frac{L}{v} \tag{9.4}$$

9.3 Discussion

No validation of the models have been carried out in this project and it is not clear how well the used models describe the process behavior in connection with a grade change. For example, is perfect mixing in the tanks a suitable model when concentrations are changing quickly? The results of the optimization suggestion presented in this thesis therefore requires further investigation.

10. Optimal Control

This chapter contains a general discussion on optimal control which could be applied to the grade change control problem.

10.1 Problem Definition

In order to develop an optimal grade change control scheme one firstly has to determine what specifies the optimal solution. To use theoretical results and methods from for example [8] the optimality conditions and constraints have to be specified as a mathematical criterion with boundary constraints. A general description of an optimal control problem can be stated in the following way

$$\text{minimize} \int_0^{t_f} L(x(t), u(t))\, dt + \phi(x(t_f)) \tag{10.1}$$

$$\dot{x}(t) = f(x(t), u(t)) \tag{10.2}$$

$$u(t) \in U, \quad 0 \leq t \leq t_f \tag{10.3}$$

$$x(0) = x_0, \quad \psi(x(t_f)) = 0 \tag{10.4}$$

In Equation 10.1 the functions L and ϕ define the cost function which is to be minimized. The two vectors x and u contain the state variables and the control signals respectively. The set U defines our boundary constraints and includes all the input states allowed during the transition. Lastly, the terminal constraint Equation 10.4 specifies the conditions that have to be fulfilled for the problem to be solved. To summarize a physical problem this way can sometimes be a fairly complex problem.

10.2 Optimal Grade Change Control

In a mathematical way a grade change can be defined as taking the process state from a point within the currently produced grade to a point within the grade to be produced after the change.

These "grade spaces" consist of the accepted intervals for the quality properties of the two grades. The time outside both these sets of states defines the grade change time. In an optimal case the two spaces overlap, which means that the grade change will be instant. It is possible to simplify a grade change by choosing the next grade "close" the current one. How to plan the production cycle is therefore very important. Planning is however outside the scope of this thesis, which instead concentrates on how to efficiently execute a specific grade change in an already defined production cycle.

As mentioned in Chapter 3, there are more than one aspect to consider when analyzing the outcome of a grade change in paper/board making. Having this in mind it is not always obvious how to choose a single mathematical condition as the main objective of the optimization. This matter will therefore be discussed more in detail in a separate section further down.

Besides the criterion used for optimizing the grade change there will be additional things to keep in mind; for some process values it is very important

that certain limits are not exceeded. In our example these limits can be tanks getting empty or pumps running with a higher load than they are designed for. There are also other important requirements that have to be met. These can be to follow a certain execution order in which for example a certain valve needs to be opened before a pump is started.

Translated to the mathematical model the extra requirements can be seen as boundary constraints. These have to be incorporated into either the cost function of Equation 10.1 or, for input states, the set U in Equation 10.3. Denoting the allowed states for x as X, the sets X and U will limit the number of allowed states and preclude some solution trajectories. In a large process with many state variables, like the board making process, the number of boundary constraints will evidently restrict the number of control options significantly.

Simplification

As described in the earlier section the grade change can be illustrated as going from one grade to another in the quality property space. Clearly, there will exist several possibilities of how to do this, one optimal solution (maybe more than one) and an infinite number of less good alternatives. If the optimization criterion is for example a minimum grade change time the optimal solution will be the trajectory that takes the least time.

Because of the large number of states and the system complexity it is hard to determine exactly how the variables are depending on the control of the controller set points. Without this knowledge it is impossible to calculate the optimal solution. Under certain assumptions it might however be possible to get close.

Assuming that the set point values used in earlier production of the desired grade will produce board of acceptable quality once the production has stabilized is reasonable. Process variations such as incorrect calibration of sensors, and other long-term fluctuations make the grade change harder. The effects of these problems usually result in a need of manual adjustments based on professional experience to tune the controller set points. Even if slight adjustments of the recommended values are needed these control settings will however, at least bring the production close to what is desired.

The Optimality Criterion

A reasonable objective in this case is to achieve a short grade change time. Following the ideas in the section above this means to get all the controlled set points close to the recommended values as quickly as possible. By mathematical terms this is a so-called minimum time problem. Using the same notation as before, our criterion in Equation 10.1 then becomes

$$\text{minimize} \int_0^{t_f} 1 \, dt = t_f. \tag{10.5}$$

The approaches in [8] can result in fairly complex control signal patterns. These are not a problem for automated systems with a high level of control of the regulator coordination and synchronization. In a semi-automated case however, this can be a great difficulty. If the signals are not well synchronized and do not follow the optimal scheme perfectly it might not be a good control at all. It is even likely that the results will be worse than the fully manual control, with fewer and easier steps. How good the results will be is a matter of the system's robustness. The problem of following a complex control pattern

manually might also make the resulting control differ from time to time, which in turn, makes evaluation and further optimization harder.

Solutions to minimum time problems usually have the characteristic bang-bang shape. This means that the optimal control signal switches between its maximum and minimum levels a finite number of times to solve the problem. The bang-bang control can be described with step functions which can easily be incorporated in a control scheme for manual execution. The minimum time approach is therefore a tempting option when a simple control pattern is desired. In complex systems the number of steps needed in the optimal control scheme do however increase fairly quickly. See [8] for details on bang-bang control for optimal control problems.

11. Optimizing the Mixing Stage

The first attempt of using a dynamic model for optimizing the grade change control addresses only a small but well-defined part of the process. To use the result in a more comprehensive optimization is however also possible. This is because the JGrafchart models offers good means for synchronizing parameter adjustments, in addition to control or guidance for subprocesses. Synchronization can make the effects of several control changes (in for example pulp composition and amounts of additives) affect the produced board at the same time. Combining cases of subprocess optimization can therefore reduce the amount of off-quality cartonboard produced during the grade change. This chapter presents an improved control strategy for optimizing the control of the pulp mix composition in the mixing stage. The used models are presented in Chapter 9.

11.1 Optimization with GESOP

Graphical Environment for Simulation and Optimization, GESOP, is the name of the optimization tool that has been used in the project. The application is developed by the University of Stuttgart. The development is partly funded by the German Aerospace Research Establishment (DLR) and Deutsche Forschungsgemeinschaft (DFG).

Modeling in GESOP

There are different ways to solve an optimization problem in GESOP. In this thesis the C language interfacing option has been used. The interface consists of four different parts, model interface, model description interface, control law interface and auxiliary interface. There are guidelines on how to implement each one of these parts in the supplied program documentation [14].

The information included in the interface source code includes the dynamics of the models to use. Other things that need to be specified in the interface are path constraints, rules for the control law and boundary conditions for the initial and final states.

11.2 A Pulp Composition Adjustment

As mentioned before the control in the change from *Carry 425* to *Bright 390* involves several parameter adjustments. A well-defined and straight forward problem is the control of the changes in the composition of the pulp mix for the top and bottom layers. As can be seen in Table 5.1 the proportions of coniferous wood pulp and birch pulp for these layers have to be adjusted. This section illustrates what the results of a simple optimization could look like. Details about system dynamics, constraints and the optimality criterion is presented in a separate section further down.

The objective of the optimization is to change the proportions of the different pulps in the flow to the top layer mixing tank so that the desired ratio is reached as fast as possible in the machine tank. In the example the ratio

between coniferous and birch wood pulp in the input flow to the mixing tank has been used as the only control input. The optimization relies on the pulp concentrations of the different pulps in the flows into the mixing tank to be constant. It is important to keep these concentration levels at a constant level for the mills to function properly and this is assured by separate control systems. Thus, the assumption is reasonable. The flows are also assumed to be constant. Figures 11.1-11.3 show the results of the optimization.

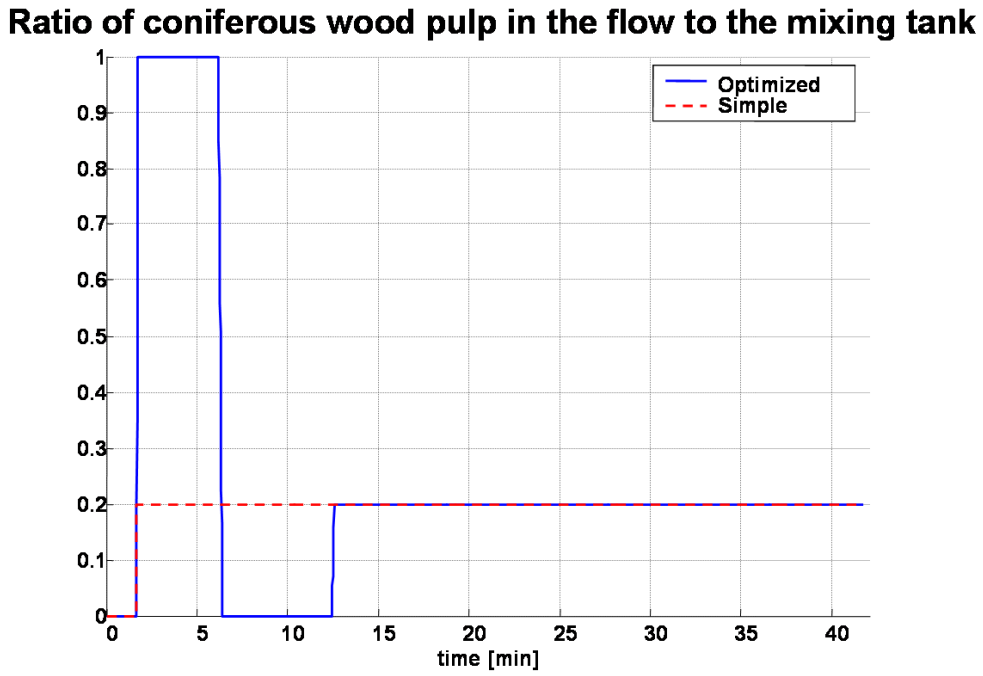

Figure 11.1 The ratio of coniferous wood pulp in the input flow to the first mixing tank. In blue is the suggested way to control the input ratio during the grade change. The dashed, red line shows the currently used control strategy.

11.3 Optimization Problem

Figure 2.3 and Figure 9.4 provide a good overview of the problem at hand. From the figures can be seen that the amounts of coniferous and birch wood flowing into the top-layer mixing tank are controlled by two separate controllers. The third line, at the bottom to the left, represents the long circulation and is used for diluting the pulp to the desired concentration. The mixing tank is connected to the machine tank through a pipe. There is a controller adjusting the concentration in this pipe as well. In the modeling three assumptions have been made:

- The total flow into the mixing tank is kept constant throughout the grade change.
- The levels in the tanks are assumed to be constant, meaning that the amount of stock leaving each tank is equivalent to the amount entering.
- The 2-step dilution of the stock is kept constant, meaning that each controller add the same amount of water all the time.

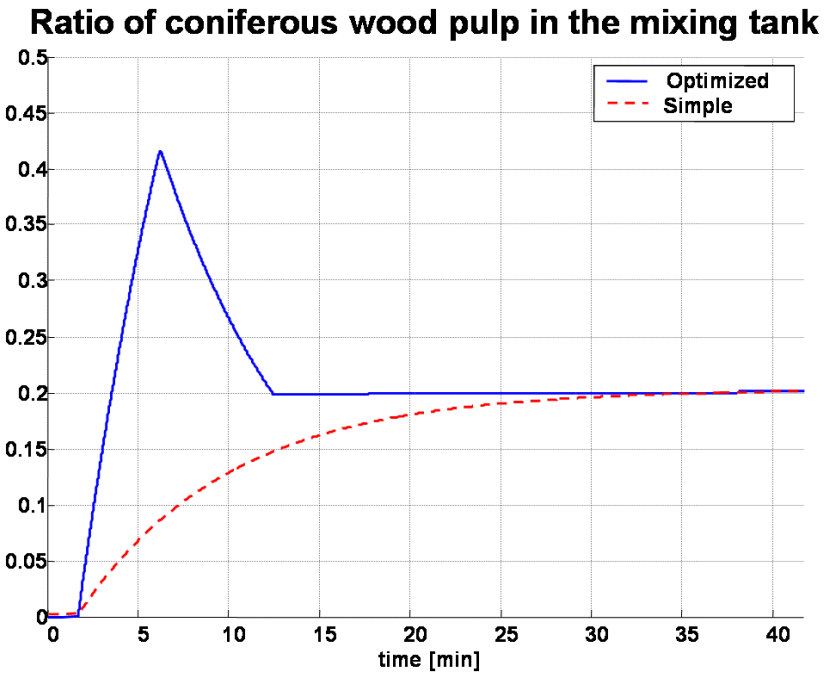

Figure 11.2 The solid, blue line shows how the ratio in the first mixing tank is affected by the optimized control signal. The dashed, red line corresponds to the case when the current control strategy is used.

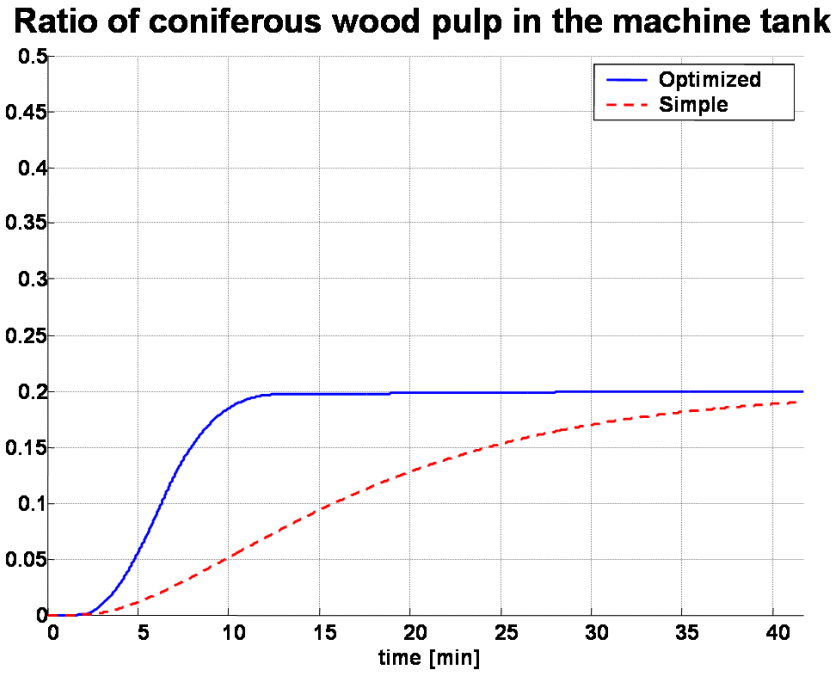

Figure 11.3 The ratio of coniferous wood pulp in the machine tank. The solid, blue line corresponds to the suggested control strategy. The dashed, red line corresponds to the currently used control.

The third assumption makes it possible to simplify the problem as a system with a single input. If the two dilution steps are constant they can be ignored, since we are only interested in the ratio coniferous/birch pulp. The ratio at the inlet of the mixing tank considered the only input signal. To simplify the rest of the discussion the ratio is referred to as the concentration of coniferous pulp even though the actual coniferous pulp concentration is several times lower and different in the separate parts of the system.

System Dynamics

Keeping the levels in the tanks constant leads us to the following equation for the tank dynamics

$$V_{tank}\frac{dC_{tank}(t)}{dt} = q(C_p^{out}(t) - C_{tank}(t)) \tag{11.1}$$

where V_{tank} is the volume in the tank, C_{tank} is the concentration in the tank, q is the flow in and out from the tank and C_p^{out} is the concentration of coniferous wood pulp in the end of the pipe closest to the tank.

The states introduced by the compartment model can be described by the following equations

$$V_k\frac{dC_k(t)}{dt} = q(C_{k-1}(t) - C_k(t)),\ k = 1, \ldots, m \tag{11.2}$$

where k is the compartment number from the inlet of the pipe and m is the total number of compartments in the approximation. Setting for example $m = 10$ makes C_p^{out} from Equation 11.1 equivalent to C_{10} from Equation 11.2 if the tank is connected to the end of the pipe.

Putting the tanks and the pipe together, our system can be described as a linear system of equations:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{11.3}$$

$$\mathbf{A} = \begin{bmatrix} -\beta & 0 & \ldots & \ldots & 0 \\ \alpha & -\alpha & & & \vdots \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & & \alpha & -\alpha & 0 \\ 0 & \ldots & 0 & \gamma & -\gamma \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} \beta & 0 & \ldots & 0 & 0 \end{bmatrix}^T$$

$$\mathbf{x} = \begin{bmatrix} C_{tank1}(t) & C_1(t) & \ldots & C_m(t) & C_{tank2}(t) \end{bmatrix}^T$$

$$\mathbf{u} = u(t) = C_{in}(t)$$

The variables in $\mathbf{A}$ are defined as

$$\alpha = \frac{q_{pipe}}{V_{comp}},\ \beta = \frac{q_{tank1}}{V_{tank1}},\ \gamma = \frac{q_{tank2}}{V_{tank2}}. \tag{11.4}$$

Lengths and dimensions of pipes and tanks are taken from the aforementioned Dymola model.

Criterion and Constraints

As mentioned before, the goal of the optimization is to make the pulp mix in the machine tank reach the right proportions as quickly as possible. In mathematical terms this can be translated into an optimization criterion together with terminal constraints. The optimization criterion is

$$\text{minimize} \int_0^{t_f} 1 \, dt = t_f.$$

Since the grade change is a transition between two steady state cases we set the initial and terminal constraints in the following way

$$\mathbf{x}_0 = C_{init} \begin{bmatrix} 1 & \dots & 1 \end{bmatrix}^T \tag{11.5}$$

$$\psi(\mathbf{x}(t_f)) = \mathbf{x}_{t_f} = C_{final} \begin{bmatrix} 1 & \dots & 1 \end{bmatrix}^T \tag{11.6}$$

where C_{init} and C_{final} are constant scalars describing the initial pulp mix proportions and the desired goal proportions, respectively.

A third kind of constraints are the so-called path constraints. These provide an option to specify conditions that have to be fulfilled throughout the optimization. This could be applicable if for example the levels in the tanks where allowed to vary. In such a case a path constraint could be a limit for the highest level allowed in a tank.

In the optimization carried out in this thesis path constraints where used on the level of the control signal. In addition to the limits for the input concentration, another constraint has been used for the control signal. This is described in Section 11.4.

Bang-bang Control

Referring to the discussion in the end Chapter 10 our optimization criterion should render a solution suitable for a JGrafchart guidance model. The solution has a very characteristic input signal; according to Theorem 18.5 in [8] the control signal will be bang-bang along extremals of the minimum time problem. To apply the theorem we need to show that the system in Equation 11.3 is controllable and that the terminal constraint $\psi(x(t_f))$ has full rank. The second condition is true since $\psi(x(t_f))$ is a 1-dimensional array and not zero. As for the first condition, we use the rank of

$$\begin{bmatrix} \mathbf{A} - \lambda \mathbf{I} & \mathbf{B} \end{bmatrix} \tag{11.7}$$

to show that the system is controllable. The method is taken from Theorem 6.1 in [26] and states that the pair ($\mathbf{A}$,$\mathbf{B}$) is controllable if and only if the matrix in Equation 11.7 has full row rank at every eigenvalue λ of $\mathbf{A}$. With the values from Equation 11.3 the matrix in Equation 11.7 can be written as

$$\begin{bmatrix} -\beta + \lambda & 0 & \dots & \dots & 0 & 1 \\ \alpha & -\alpha + \lambda & & & \vdots & 0 \\ 0 & \ddots & \ddots & & \vdots & \vdots \\ \vdots & & \alpha & -\alpha + \lambda & 0 & \vdots \\ 0 & \dots & 0 & \gamma & -\gamma + \lambda & 0 \end{bmatrix}$$

Since $\mathbf{A}$ is triangular its eigenvalues are the elements in the diagonal ($\lambda_1{=}{-}\alpha$, $\lambda_2{=}{-}\beta$, $\lambda_3{=}{-}\gamma$). It can easily be seen that the matrix will have full row rank for all λ if α, β and γ are non-zero. From Equation 11.4 follows that if α, β or γ is zero there is no flow through the pipe or the tanks, and thus we can ignore this case.

11.4 Optimization Issues

This section presents the effects and solution of a problem encountered at an early stage of the optimization. At this point the system considered constituted of a single pipe with a tank connected to its end. The problem was to find how the concentration of the incoming flow to the pipe should be adjusted to reach the right concentration in the tank as fast as possible.

The problem arouse during the first attempts of using compartment models for the pipes. The GESOP solution then suggested a control signal according to Figure 11.4.

As can be seen from the figure the concentration in the end of the pipe (compartment 10) and in the tank settle nicely, but the control signal and the concentration in the beginning of the pipe does not look as good. The concentration in the first compartment of the pipe oscillates and the number of switches for the control signal is large. It is obvious that this control strategy is hard to follow with a JGrafchart model.

The background to the problem is the extra states and terminal constraints introduced by the compartment model. For systems with real eigenvalues there is generally an upper bound on the number of switches in the bang-bang control. Theorem 18.6 in [8] states that this limit is connected to the number of states for a system such as the one presented here. The maximum number of switches is one less than the number of states in the system. Thus, if $\mathbf{A}$ is $n{\times}n$ the control signal switches at the most $n-1$ times.

To avoid this problem an additional path constraint was added to the optimization problem definition. This equality path constraint was used to fix the control variable (at the value C_{final}) at the end of the optimization horizon, for a time corresponding to the delay of the pipe:

$$u(t) = C_{final},\ t_f - L_p \leq t \leq t_f$$

where t_f is the final time and L_p is the pipeline delay. In addition, the terminal equality constraints for the pipe states were dropped. The strategy ensures that the terminal values of the pipe states are approximately equal to the desired final concentration. This way GESOP was forced to choose an alternative solution with less switches. Figure 11.5 shows the corresponding plots from the final version of the optimization. In this case the pipe is modeled with 20 compartments.

11.5 Discussion

As can be seen in Figure 11.1 the proposed control is very aggressive. Since the optimization is based on fairly simple models of the process, which were not designed for grade change control, there is a large risk that the real process will not behave accordingly if the control is too extreme. It is therefore not

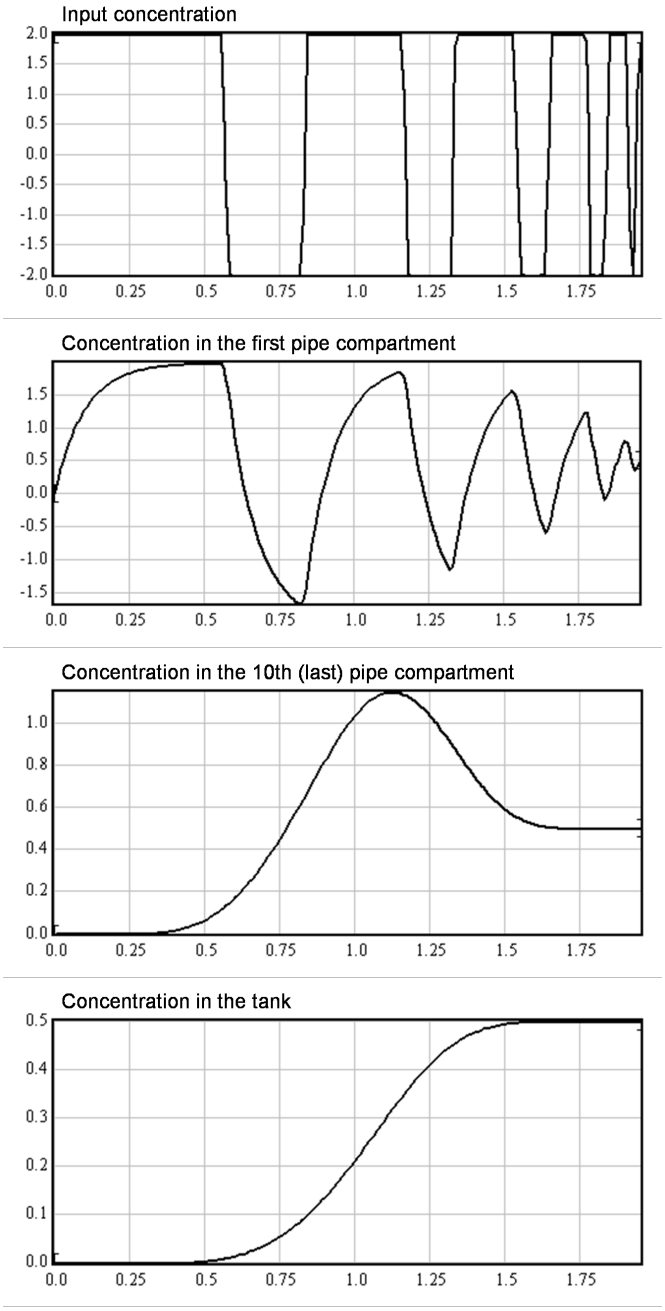

Figure 11.4 The results from the first optimization attempts with a compartment model of a pipe. The control signal has the characteristic bang-bang shape with many switches.

possible to rely on the presented optimization results without validating the models for the right conditions.

Even if the proposed strategy can not be applied as is, the optimization rules can be adjusted to provide a control signal that still performs well but is less aggressive. It is for example easy to add maximum and minimum levels for the control signal. The effects will still be a reduced adjustment time and the moment at which the desired ratios are reached can be calculated in advance.

The proposed control strategy for the mixing stage is not specifically designed for the CRY425-BRI390 and the technique should therefore be applica-

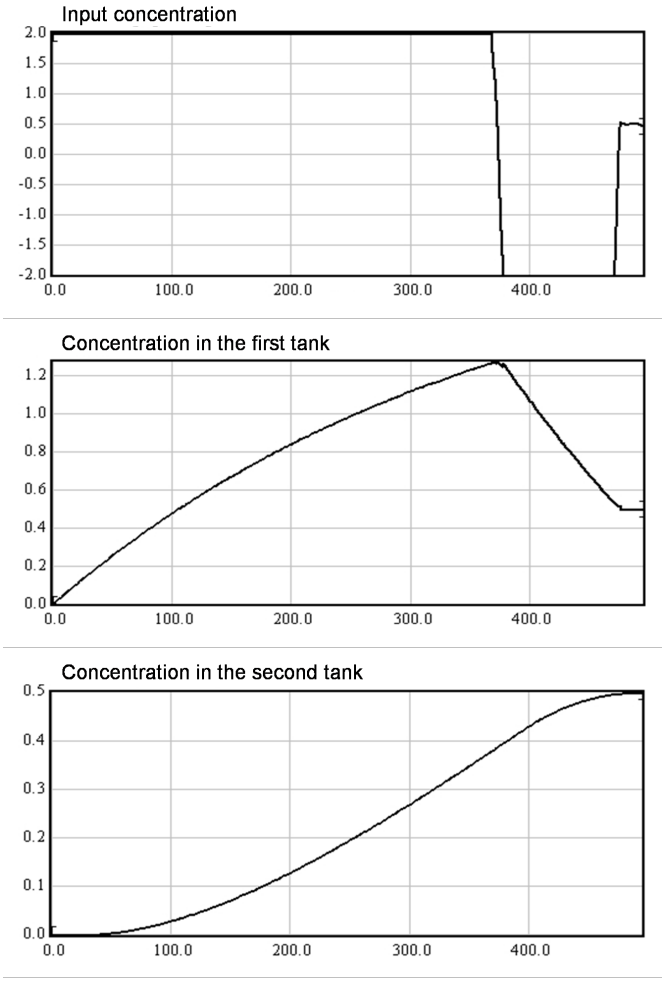

Figure 11.5 The final results from the optimization with modifications to reduce the number of switches. The model considered consists of two tanks with a pipe in between.

ble in other grade changes too. In fact, the results might be even more accurate in other cases. This is because the pulp flow through the top-layer mixing stage do changes considerably in the CRY425-BRI390, which is an effect that adds to the uncertainty of the optimization. In the proposed the strategy the flow through the mixing stage has been assumed constant during the grade change. The value of the flow has been approximated with the average of the flow before and after the change is carried out.

11.6 General Optimization

In the paper making process some adjustments have a fast response time while others take much longer. It is therefore convenient to separate the optimization problem in several subsystems, which can be treated independently. This also makes it easier to reuse the strategies in different grade changes. Since the grade changes have several control steps in common these can be included (with modifications). The steps not needed are simply ignored.

12. Interface Details

12.1 JGrafchart-PI Interface

The interface between JGrafchart and the PI System has been done with Java Native Interfacing (JNI). More information about JNI in general can be found in [15].

From the JGrafchart side the interface is an implementation of the input and output interfaces described in Chapter 6. The interface implementation also makes use of the Osisoft C API which is documented in [13]. The methods in the JGrafchart I/O classes call dynamic link library (dll) functions implemented in C.

The Native Side Implementation

The C files, which are the source files of the dll, call functions from libraries in the Software Development Kit for the PI System. The native side of the interface include four C files briefly described in the table 12.1. Besides the C

C File	*Description*
`pi_info.c`	Contains functions for retrieving point specific information, such as a descriptive text, applicable engineering unit, data type of the stored data.
`pi_get.c`	Contains functions for retrieving on-line and archive data from the PI Points.
`pi_put.c`	Contains functions for storing data in the PI System.
`pi_private.c`	Contains functions that are used from the above listed C files. No functions from `pi_private.c` are directly called from the Java side.

Table 12.1 The native side of the JGrafchart-PI interface consists of four C files.

files, the native side implementation also includes a header file `pi.h` with some data type definitions.

The compilation and linking of the dynamic link library has been done with the Microsoft compiler and linker from VCTool. This tool can be downloaded from the Microsoft website.

The Java Side Implementation

In this project all inputs to the the JGrafchart environment has been retrieved with the `AnalogInput` blocks and all outputs have been sent through the `AnalogOutput` blocks. Inside the JGrafchart environment the data sent and retrieved from PI is therefore handled as `doubles`.

The data logged in the PI System can be of different types. Each “node”, for which data is logged, is usually referred to as a PI point. A PI point can for example refer to a measured value from a sensor somewhere in the process.

It is possible to choose the data type of the stored data so that it fits the source in a good way. For example, a temperature can be stored as a floating point centigrade value and data for the state of a valve might be stored as a two-state variable (open/closed). The PI System supports real values, integer values, digital states, character strings and a couple of other data types. The most common type used with in the PI System at AssiDomän Frövi is the `real`. The only other data type used in this thesis is the digital type.

Internally within the PI System references to PI points are done with a unique integer id. For easier handling from applications and human interaction every PI point also has a unique tag name. At AssiDomän Frövi the tag names follow the standard specified in [12]. The JGrafchart interface handles the extra level of a control provided by the integer ids and hides these values from the JGrafchart user. From the JGrafchart environment references to the PI Points can therefore be done simply by using the PI point tag names in the channel strings of the I/O blocks.

The PI System C API has several functions for retrieving information. The returned parameters from these functions hold information about both the measured value and the validity of the data. Depending on what kind of data type a point stores the return data has to be interpreted differently. The rules for how to do this is specified in [13]. It is possible to identify the status of the returned PI values by using these rules but there is no easy way to pass this on to the JGrafchart environment. At this point, invalid data from a PI value call cause the value in the calling I/O block to be set to 0.

When simulating a control scheme with historical data there are no constraining real time demands for the data retrieval. The interfacing methods for the archive data retrieval is implemented so that data for a specified time for example two hours are fetched at the start of the simulation. If the simulation goes on for longer than this new data will be fetched automatically when the buffer runs out. By using buffer lengths of slightly different sizes at initialization of the input blocks the preceding calls are evenly distributed over time.

More class structure and implementation details can be found in the Javadoc documentation for the `frovi.pi` and `frovi.io` packages.

Practical issues

From the beginning all PI communication was carried out through the interface class `AnalogInput`. This solution means that every `PiAnalogInput` made its own call to the C API when it needed a new value. In a larger model this means that a couple of hundred calls has to be made every time new values should be retrieved. This waste of CPU power and network bandwidth meant that the models had to run at very low sampling rates.

An advantage of the C API is the possibility to make a common call to a large number of points. This means that a single call to the C API can return data for several points and thereby offers a fast and efficient way of communicating with the PI System from other applications.

With a couple of small modifications to the JGrafchart source code methods were added to make it possible to collect data for all points of interest with only one call to the C API. This resulted in increased efficiency and made it possible to run the on-line simulations at about 10 times faster sampling rate. The improvement also reduces the load on the PI System.

12.2 JGrafchart-SQL Interface

In this thesis JGrafchart has been acting as an SQL client. The implementation makes the data in the production recommendation database accessible from the JGrafchart models. Having the control actions in the models directly connected to this data ensures that the suggestions from JGrafchart will always be based on the most recently tuned recommendations.

The communication at the JGrafchart side is handled by `AnalogInputs`, where every sought variable corresponds to an input block. The connection string for the `AnalogInput` should follow the pattern "RV BRI390:35". Where the first token is an abbreviation of Recommended Value, the second token specifies the grade and the number after the colon refers to the property of interest.

The SQL communication is handled with the JDBC Library from Microsoft. For efficiency reasons the calls to the the SQL-server is only made at the initialization of the input block classes. This means that no new data will ever be fetched as long as the model is running. In order to update the values in the model the simulation has to be stopped and started again. To fully optimize the execution time of the model all input blocks referring to an SQL-value should have the *cyclic update* property box unchecked.

13. References

[1] Karl-Erik Årzén. *JGrafchart User Manual ver. 1.5*, Department of Automatic Control at Lund Institute of Technology, 2004

[2] *AssiDomän Frövi: The natural choice*, AssiDomän Frövi

[3] Mats Kassberg, Magdalena Erlandsson, Gunnar Gavelin. *Massa och papper - en grundbok*, Skogsindustrins utbildning i Markaryd AB, 1998, ISBN 91-7322-233-X

[4] Douglas Wahren. *Papersteknik - Del 1 Fundamenta*, STFI institutionen för pappersteknik KTH, 1973

[5] Paavo Viitamäki. *Hybrid Modeling of Paper Machine Grade Changes*, Control Engineering Laboratory at Helsinki University of Technology, 2004, ISBN 951-22-7135-4

[6] Kunko Leiviskä. *Process Control*, Fapet Oy, 1999, ISBN 952-5216-14-4

[7] Jan Sjögren. *Q Info (ver. 3.6.99)*, AssiDomän Frövi, 2005

[8] Torkel Glad, Lennart Ljung. *Control Theory - Multivariable and Nonlinear Methods (chapter 18)*, Taylor & Francis, 2000, ISBN 0-7484-0878-9

[9] Rasmus Olsson. *Exception Handling in Recipe-Based Batch Control*, Department of Automatic Control at Lund Institute of Technology, 2002, ISBN 0280-5316

[10] Finnish Forrest Industries Federation. *Finnish Forrest Industries Federation*, http://english.forestindustries.fi/

[11] Sveriges lantbruksuniversitet. *SkogsSverige*, http://www.skogssverige.se/

[12] Skogsindustrierna Teknik AB. *SSG 5276 - Koder för mät- och styrfunktioner i processflödesscheman*, 2000

[13] OSIsoft, Inc. The API Documentation for *PI-API-NTI 32 Bit PI-API for Microsoft Windows NT and Windows 9x Version 1.3.9.4*, http://www.osisoft.com/

[14] GESOP. *GESOP Software User Manual 4.6.0*, Institute of Flight Mechanics and Control at University of Stuttgart, February 2004

[15] Sun Microsystems, Inc. *Java Technology*, http://java.sun.com/

[16] Sun Microsystems, Inc. *jGuru: JDBC 2.0 Fundamentals*, http://-java.sun.com/developer/onlineTraining/Database/JDBC20Intro/-JDBC20.html

[17] Carl D. Laird et. al. *Contamination Source Determination for Water Networks*, "Journal of Water Resources Planning and Management", March/April 2005

[18] AssiDomän Frövi. Drawings of the KM5 Board Machine.

[19] AssiDomän Frövi. QMsys.

[20] Lars Jonhed. *Modelling, simulation and validation of a wet end simulation model in Modelica*, Chalmers Tekniska Högskola & AssiDomän Frövi, 2000

[21] Stefan Ericsson. *Modelling and validation of a wet end simulation model*, KTH & AssiDomän Frövi

[22] Niklas Jansson. *Dynamic simulation of pressure, flow and pulp consistency in a bleaching plant*, Chalmers Tekniska Högskola & AssiDomän Frövi, 2004

[23] Björn Carlsson. *Dynamic simulation of the wire and press section of a board machine*, Chalmers Tekniska Högskola & AssiDomän Frövi, 2004

[24] *The Gnumex Project*, http://gnumex.sourceforge.net/

[25] *Minimalist GNU for Windows*, http://www.mingw.org/

[26] Chi-Tsong Chen. *Linear System - Theory and Design*, Oxford University Press, 1999, ISBN 0-19-511777-8

A. Documentation of m- and mex-files

To simplify the data analysis in the project some interfacing between the PI System and the Matlab environment was carried out. This was done to be able to compare and analyze historical data from the PI System with the powerful tools in Matlab, without the need of tedious and time consuming steps of exporting and importing data. The interface with Matlab consists of a set of mex-files developed, compiled and linked with the gnumex [24] and the Minimalist GNU (MinGW) [25] softwares. No source code is included in the thesis but user documentation for the most important mex-files is listed below.

```
PIPOINTINFO - Returns an info structure with info about
    the Pi Point of interest.

SYNTAX
info = pipointinfo( tagname )

INPUT
tagname: The name of the Pi Point, case insensitive.

OUTPUT
info: An info structure with information about name,
    description, engineering unit, type and, if point type is
    digital, names of the possible states and the state offset.
```

Figure A.1 The help documentation for `pipointinfo.m`.

```
DATESTR2PITIME - Using Java object SimpleDateFormat to
    parse and convert a date string to a pitime integer
    timestamp.

INPUTS
strdate: The date string to parse. Possible date formats
    are listed below and tried in listed order.
        'yyyy-MM-dd HH:mm:ss'  ex. 2005-04-29 14:16:19
        'yyyy-MM-dd HH:mm'
        'yyyy-MM-dd HH'
        'yyyy-MM-dd'
        'dd-MMM-yyyy HH:mm:ss' ex. 12-Feb-2004 13:00:00
        'dd-MMM-yyyy'

OUTPUT
pitime: The pitime integer time stamp, which is close
    to utc but has an offset.

NOTES
The function can easily be extended to handle additional
formats.
See the Java documentation for the SimpleDateFormat
class for more info on how the date formats should be
interpreted.
```

Figure A.2 The help documentation for `datestr2pitime.m`.

```
PICALL - Retrieve data from PI.

SYNTAX
[times, values, quality] =
    picall(tagname [, from [, to [, res [, timeunit] ] ] ] )

OUTPUTS
times: an array with 'datenums' corresponding to the retrieved
    values. See Matlab documentation for further details on the
    datenum format.
values: array with numeric values returned from PI. For float
    Pi Points these are the retrieved values. For digital
    Points they represent the state number i.e. a value from 0
    to the number of states minus 1. The array may contain NaNs.

INPUTS
tagname: the tag name of the PI Point
from: Either a date string (ex. '2002-02-02 12:03:10') or a
    positive or negative numeric offset. The offset will
    represent the time of 'to' plus the offset in 'from' if
    the 'to' argument is a date string. If 'to' is an offset as
    well they will both be treated as offsets to the current
    time. The time unit of the offset is specified in the last
    input argument.
to: See 'from'.
res: The resolution of the returned values. This value contains
    the spacing in seconds between the collected data. If 0 or
    left out the returned values will be only the values stored
    in PI. If specified the returned array will be interpolated
    values from the PI System. The time unit of 'res' is always
    seconds (independent of 'timeunit').
timeunit: string that determine the time unit of the offsets
    used in the call. The different options are:
        second: 's', 'sec', 'second', 'seconds'
        minute: 'm', 'min', 'minute', 'minutes'
        hour:   'h', 'hour', 'hours'
        day:    'd', 'day', 'days'

NOTES
The order of 'from' and 'to' in the function call is not
important.
See the documentation for DATESTR2PITIME for information on
the possible date string formats.

SEE ALSO
PIPOINTINFO, PIPLOT, DATESR2PITIME
```

Figure A.3 The help documentation for `picall.m`.

B. Javadocs

The documentation of the JGrafchart add-ons is available at: http://www.control.lth.se/publications/extra/5754