N°d'ordre NNT : xxx

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'Ecole des Mines de Saint-Etienne

Ecole Doctorale N° 488
Sciences, Ingénierie, Santé

Spécialité de doctorat : Génie Industriel
Discipline : (Eventuellement)

Soutenue publiquement/à huis clos le 20/09/2017, par :
Rezvan SADEGHI

Consistency of global and local scheduling decisions in semiconductor manufacturing

Devant le jury composé de :

Nom, prénom	grade/qualité	établissement/entreprise	Président.e (à préciser après la soutenance)
Amodeo, Lionel	Professeur	Univ. de Technologie de Troyes	Rapporteur
Jussien-Guéret, Christelle	Professeur	Univ. d'Angers	Rapporteuse
Moench, Lars	Professeur	Univ. de Hagen	Examinateur
Absi, Nabil	Maître de recherche	Mines de Saint-Etienne	Examinateur
Pinaton, Jacques	Ingénieur	STMicroelectronics	Examinateur
Dauzère-Pérès, Stéphane	Professeur	Mines de Saint-Etienne	Directeur de thèse
Yugma, Claude	Maître de recherche	Mines de Saint-Etienne	Co-directeur de thèse
Vermarien, Leon	Ingénieur	STMicroelectronics	Invité

Spécialités doctorales	Responsables :	Spécialités doctorales	Responsables
SCIENCES ET GENIE DES MATERIAUX	K. Wolski Directeur de recherche	MATHEMATIQUES APPLIQUEES	O. Roustant, Maître-assistant
MECANIQUE ET INGENIERIE	S. Drapier, professeur	INFORMATIQUE	O. Boissier, Professeur
GENIE DES PROCEDES	F. Gruy, Maître de recherche	SCIENCES DES IMAGES ET DES FORMES	JC. Pinoli, Professeur
SCIENCES DE LA TERRE	B. Guy, Directeur de recherche	GENIE INDUSTRIEL	X. Delorme, Maître assistant
SCIENCES ET GENIE DE L'ENVIRONNEMENT	D. Graillot, Directeur de recherche	MICROELECTRONIQUE	Ph. Lalevée, Professeur

EMSE : Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'Etat ou d'une HDR)

ABSI	Nabil	CR	Génie industriel	CMP
AUGUSTO	Vincent	CR	Image, Vision, Signal	CIS
AVRIL	Stéphane	PR2	Mécanique et ingénierie	CIS
BADEL	Pierre	MA(MDC)	Mécanique et ingénierie	CIS
BALBO	Flavien	PR2	Informatique	FAYOL
BASSEREAU	Jean-François	PR	Sciences et génie des matériaux	SMS
BATTON-HUBERT	Mireille	PR2	Sciences et génie de l'environnement	FAYOL
BEIGBEDER	Michel	MA(MDC)	Informatique	FAYOL
BLAYAC	Sylvain	MA(MDC)	Microélectronique	CMP
BOISSIER	Olivier	PR1	Informatique	FAYOL
BONNEFOY	Olivier	MA(MDC)	Génie des Procédés	SPIN
BORBELY	Andras	MR(DR2)	Sciences et génie des matériaux	SMS
BOUCHER	Xavier	PR2	Génie Industriel	FAYOL
BRODHAG	Christian	DR	Sciences et génie de l'environnement	FAYOL
BRUCHON	Julien	MA(MDC)	Mécanique et ingénierie	SMS
CAMEIRAO	Ana	MA(MDC)	Génie des Procédés	SPIN
CHRISTIEN	Frédéric	PR	Science et génie des matériaux	SMS
DAUZERE-PERES	Stéphane	PR1	Génie Industriel	CMP
DEBAYLE	Johan	CR	Sciences des Images et des Formes	SPIN
DEGEORGE	Jean-Michel	MA(MDC)	Génie industriel	Fayol
DELAFOSSE	David	PR0	Sciences et génie des matériaux	SMS
DELORME	Xavier	MA(MDC)	Génie industriel	FAYOL
DESRAYAUD	Christophe	PR1	Mécanique et ingénierie	SMS
DJENIZIAN	Thierry	PR	Science et génie des matériaux	CMP
DOUCE	Sandrine	PR2	Sciences de gestion	FAYOL
DRAPIER	Sylvain	PR1	Mécanique et ingénierie	SMS
FAUCHEU	Jenny	MA(MDC)	Sciences et génie des matériaux	SMS
FAVERGEON	Loïc	CR	Génie des Procédés	SPIN
FEILLET	Dominique	PR1	Génie Industriel	CMP
FOREST	Valérie	MA(MDC)	Génie des Procédés	CIS
FRACZKIEWICZ	Anna	DR	Sciences et génie des matériaux	SMS
GARCIA	Daniel	MR(DR2)	Sciences de la Terre	SPIN
GAVET	Yann	MA(MDC)	Sciences des Images et des Formes	SPIN
GERINGER	Jean	MA(MDC)	Sciences et génie des matériaux	CIS
GOEURIOT	Dominique	DR	Sciences et génie des matériaux	SMS
GONDRAN	Natacha	MA(MDC)	Sciences et génie de l'environnement	FAYOL
GONZALEZ FELIU	Jesus	MA(MDC)	Sciences économiques	FAYOL
GRAILLOT	Didier	DR	Sciences et génie de l'environnement	SPIN
GROSSEAU	Philippe	DR	Génie des Procédés	SPIN
GRUY	Frédéric	PR1	Génie des Procédés	SPIN
GUY	Bernard	DR	Sciences de la Terre	SPIN
HAN	Woo-Suck	MR	Mécanique et ingénierie	SMS
HERRI	Jean Michel	PR1	Génie des Procédés	SPIN
KERMOUCHE	Guillaume	PR2	Mécanique et Ingénierie	SMS
KLOCKER	Helmut	DR	Sciences et génie des matériaux	SMS
LAFOREST	Valérie	MR(DR2)	Sciences et génie de l'environnement	FAYOL
LERICHE	Rodolphe	CR	Mécanique et ingénierie	FAYOL
MALLIARAS	Georges	PR1	Microélectronique	CMP
MOLIMARD	Jérôme	PR2	Mécanique et ingénierie	CIS
MOUTTE	Jacques	CR	Génie des Procédés	SPIN
NEUBERT	Gilles			FAYOL
NIKOLOVSKI	Jean-Pierre	Ingénieur de recherche	Mécanique et ingénierie	CMP
NORTIER	Patrice	PR1	Génie des Procédés	SPIN
O CONNOR	Rodney Philip	MA(MDC)	Microélectronique	CMP
OWENS	Rosin	MA(MDC)	Microélectronique	CMP
PERES	Véronique	MR	Génie des Procédés	SPIN
PICARD	Gauthier	MA(MDC)	Informatique	FAYOL
PIJOLAT	Christophe	PR0	Génie des Procédés	SPIN
PINOLI	Jean Charles	PR0	Sciences des Images et des Formes	SPIN
POURCHEZ	Jérémy	MR	Génie des Procédés	CIS
ROUSSY	Agnès	MA(MDC)	Microélectronique	CMP
ROUSTANT	Olivier	MA(MDC)	Mathématiques appliquées	FAYOL
SANAUR	Sébastien	MA(MDC)	Microélectronique	CMP
STOLARZ	Jacques	CR	Sciences et génie des matériaux	SMS
TRIA	Assia	Ingénieur de recherche	Microélectronique	CMP
VALDIVIESO	François	PR2	Sciences et génie des matériaux	SMS
VIRICELLE	Jean Paul	DR	Génie des Procédés	SPIN
WOLSKI	Krzystof	DR	Sciences et génie des matériaux	SMS
XIE	Xiaolan	PR1	Génie industriel	CIS
YUGMA	Gallian	CR	Génie industriel	CMP

Mise à jour : 03/02/2017

Contents

List of Tables

List of Figures

General Introduction

In the current competitive economy, production management issues are crucial for businesses in particular semiconductor manufacturing companies. Indeed, the complexity of manufacturing new products, the development of production techniques, customer requirements and the competitive context, reinforced by globalization, lead to very challenging issues.

Management of production decisions are traditionally grouped according to the time horizon on which they apply: Long term (strategic), medium term (tactical) and short term (operational decisions). This decomposition allows simplifying the decision-making process. The decisions taken at a level become constraints to meet or targets for the lower levels. However, decisions at the tactical and operational levels are often made independently. And this can lead to inconsistent or not feasible solutions. For scheduling (or dispatching) decisions, a hierarchical approach is generally adopted for semiconductor manufacturing facilities or fabs. This hierarchical approach divides the operational level in global and local levels. It consists of, initially, simulating the start of planned lots at the global level, corresponding to a short-term horizon, in order to determine critical resources and to fix priorities on the lots at the various manufacturing stages. Then, resources or sets of resources are locally managed at the local level, corresponding to real-time horizon, to determine the assignment of lots to resources as well as the sequence of lots on these resources. In this context, ensuring consistency between decision levels means that strategies and global objectives defined at the global level should be followed at the local level, with some degree of flexibility.

The aim of this thesis is to work on the consistency between the global and local scheduling levels using simulation and optimization in a semiconductor manufacturing environment. In addition, this thesis deals with the management of time constraints. In a semiconductor manufacturing facility, time constraints are associated to two process steps to ensure the yield and quality of lots. A time constraint corresponds to a maximum time that a lot can spend between the two steps. If a time constraint is not satisfied by a

lot, this lot will be scrapped or reprocessed.

The manuscript is organized in the following way. The first chapter defines the context of our study, introduces different definitions related to semiconductor manufacturing and the general issues we are interested in.

Chapter 2 presents an approach for production control with time constraints to avoid reprocessing or scrapping of lots.

Chapter 3 describes a multi-method generic simulation model for semiconductor manufacturing. The goal of this chapter is to develop a data-driven generic simulation model which can be automatically generated from external files describing the manufacturing plant.

Chapter 4 discusses a framework for consistency between global and local scheduling decisions. We aim at presenting the interrelation and interaction between global and local scheduling decisions and at ensuring the consistency of the local level with global objective decisions.

Chapter 5 presents an optimization model with various indicators to optimize such as cycle times and the satisfaction of time constraints.

Chapter 6 ends the report with conclusions and different research directions.

Chapter 1

Industrial and Scientific Context

This chapter introduces the industrial context and discusses the objectives of the thesis. Semiconductor manufacturing processes are briefly described, followed by the planning decisions for production management in this context. The first and main objective of ensuring the consistency of global and local scheduling decisions is motivated. The second objective related to managing time constraints is then presented. The main contributions of the thesis are finally summarized

1.1 Semiconductor Manufacturing: An Overview

After a pre-processing that includes silicon polycrystalline growth resulting in a cylindrical ingot, wafers are sliced, polished and an epitaxial deposition is realized. The resulting wafers can be used to start the manufacturing process of chips which usually contains two stages: Front-end and back-end.

■ **Front-end process**

The front-end process includes: Photoresist, Photolithography, Etching, Ion implantation and Chemical Mechanical Planarization (CMP), see Figure 1.1.

- **Photoresist process:** A photoresist or resist is a photo-sensitive material applied to the wafer in a liquid state in small quantities. The wafer is spun typically at 3000 rounds per minute to spread the material into a uniform layer around 2 micrometers thick.
- **Photolithography:** In this process, a scheme of a mask is transferred, normally using an ultra-violet radiation, on the surface of the wafer to specify the different silicon areas which are involved on a specific layer of the wafer.
- **Etching:** The wafer with patterned photoresist is then put into an oxide etch process to remove the oxide where there is no pattern. Etching selectively removes portions of semiconductor layers to leave micro-structures on a device.
- **Ion implantation:** In implantation, the dopant molecules are vertically implanted into the surface of the silicon by exposing it to a high-energy ion beam. These molecules aim to change the electrical characteristics of the target area. This is done in order to finally get the different electronic components.
- **Chemical mechanical planarization (CMP):** This process is used for polishing the surface of the wafer. It can be performed on both oxides and metals. It involves the use of chemical slurries and a circular mechanical action to polish the surface of the wafer.

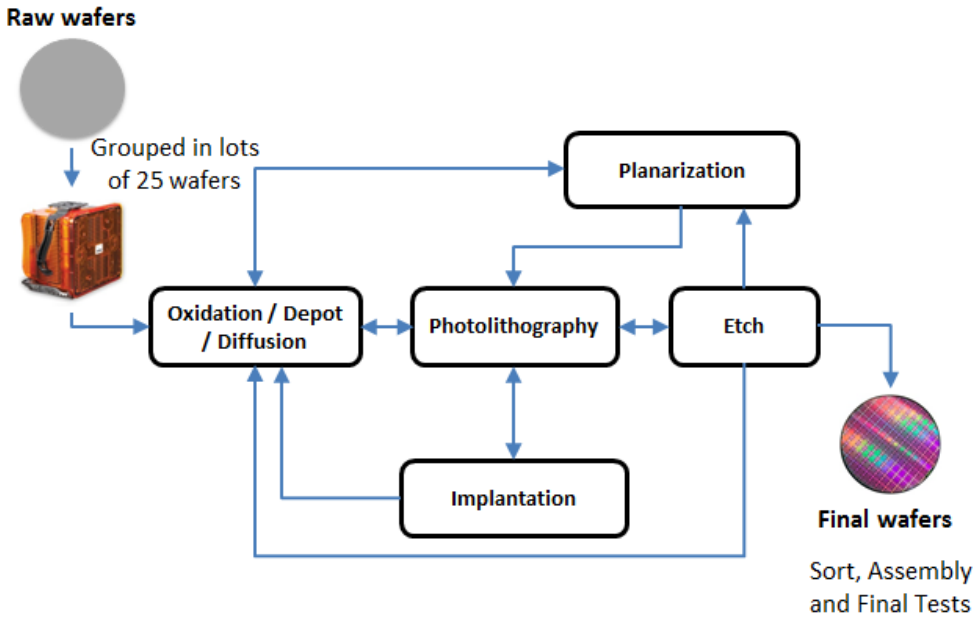

Figure 1.1: Processing steps within wafer fabrication (Mönch et al.(2011)) [52]

■ Back-end process

After undergoing the front-end processes, wafers are then tested, cut and packaged (see Figure 1.2). This is called the back-end stage. The Back-end processes include two major processes:

- **Testing:** Testing generally includes some typical measurements applied to each wafer in order to check its consistency with the predefined specifications. Such measurements typically include wafer flatness, film thickness, electrical properties, critical dimensions. This adds to the complexity of semiconductor manufacturing where sophisticated equipment and qualified workers are necessary.
- **Packaging and assembly:** In order to be connected to other devices or plugged into an electronic card, chips need to be wired. Chips are wired to their appropriate packaging boxes according to their types. A molten plastic material is poured on the whole assembly to form the different well-known existing chips.

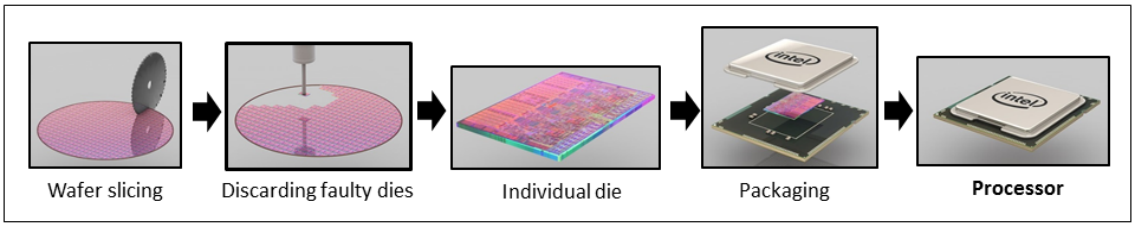

Figure 1.2: Wafer sort test [1]

1.2 Formal Description of Semiconductor Manufacturing processes

In this section, we first describe the production process flow in a semiconductor manufacturing facility (fab) and provide notations. This description will be used throughout this thesis. Then, the notion of time constraints is introduced and discussed.

A wafer fab can manufacture N different types of products on a set of machines $M = \{M_k | k = 1, \ldots, m\}$. Each product type α in the fab is associated with a processing route, denoted r_α. A route involves a sequence of process steps (operations) which must be performed for a lot of the associated product. In a wafer fab, wafers are grouped as a lot to go through the sequences of process steps. Lot l_i of product type α should be performed on r_α. A priority w_i is assigned to l_i according to various criteria such as workload, recipe throughput, technology, due date, customer, etc.

Route r_α has q_α process steps, denoted by the set $\{PS_{\alpha,j} | j = 1, \ldots, q_\alpha\}$. Process step $PS_{\alpha,j}$ can be performed by a set of machines $M_{\alpha,j} \subseteq M$ with tool-dependent processing times. The processing time depends on machine k with $k \in M_{\alpha,j}$ and is denoted $P^k_{\alpha,j}$. Semiconductor fabrication includes reentrant process flows, where lots repeatedly return at different process steps to the same service (Kumar, (1994) [35]) (see production flow of product type β in Figure 1.3).

Although each type of products has a specific route, there are common tool groups between process steps of different routes, e.g. for two product types α and β in Figure 1.3, process steps $PS_{\alpha,i+2}$ and $PS_{\beta,j+2}$ can be per-

formed by machines m_5 and m_6, while the processing times depend on the products type.

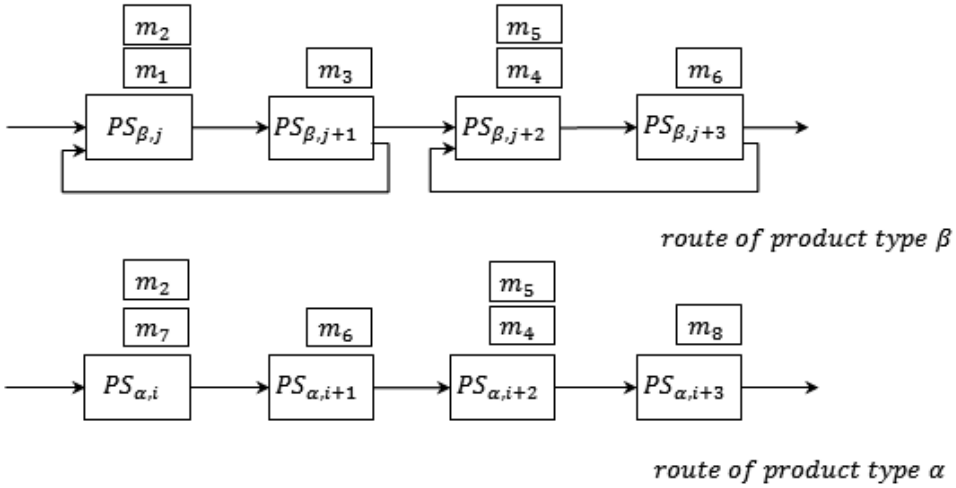

Figure 1.3: Example of a production flow in semiconductor manufacturing

1.3 Planning Decisions for Production Management in Semiconductor Manufacturing

An efficient production process can significantly benefit the competitiveness of semiconductor manufacturing companies. Thus, production management which is the focus of our study, is crucial in the supply chain process. In this section, we first briefly outline the management processes and their interrelation in the supply chain to position the production management process. Then, we focus on the decision levels in production management. Finally, the interrelations and interactions between production management decisions are investigated.

1.3.1 Supply Chain Management Process

Supply chain management refers to all planning and control decisions which have to be made in order to convert raw materials and semi-finished products into final products and deliver the right quantities to customers at the right time at minimum cost. The process consists of four stages: Supply management, production management, distribution management, and demand management. A general overview of these management processes and their interrelation is given in the text book of Stadtler and Kilger (2000) [77]. The supply management process concerns purchasing, material requirement planning and supplier relationship management. The production management process comprises the product development plan and manufacturing flow management. Distribution management includes the planning of transportation of raw materials and distribution of the final product to the warehouses and the customers. The demand management process deals with balancing the demand with supply chain capabilities. The supply chain management process is usually classified into three different decisions levels based on the principles of hierarchical planning proposed by Anthony (1965) [2] and Hax and Meal (1975) [23]. In the following, we provide an overview of the decisions that need to been taken at each of these levels.

Long-term (strategic) planning: The decisions at this level are strategic and are typically taken for several years. The scope of this planning is the whole supply chain and concerns in particular the location, design and structure of new plants. The decisions are based on forecasting the future demand.

Mid-term (tactical) planning: The decisions at this level are tactical and are based on the current status of the facility and pre-determined objectives defined in the long-term planning level. The planning horizon is from several weeks to several months. Decisions include transportation planning between facilities, from facilities to warehouses, and production planning, i.e. how much of each product type should be produced at each period of the planning horizon to meet demand.

Short-term (operational) planning: Decisions at this level are operational and range from minutes to several weeks. They deal with allocating and sequencing jobs on production resources, and routing transportation vehicles. The operational decisions use detailed information and are restricted by tactical decisions, e.g. production scheduling decisions allocate the limited resources to products to be produced and determine the short-term production times. The quantities to produce are determined by decisions taken in the mid-term planning level.

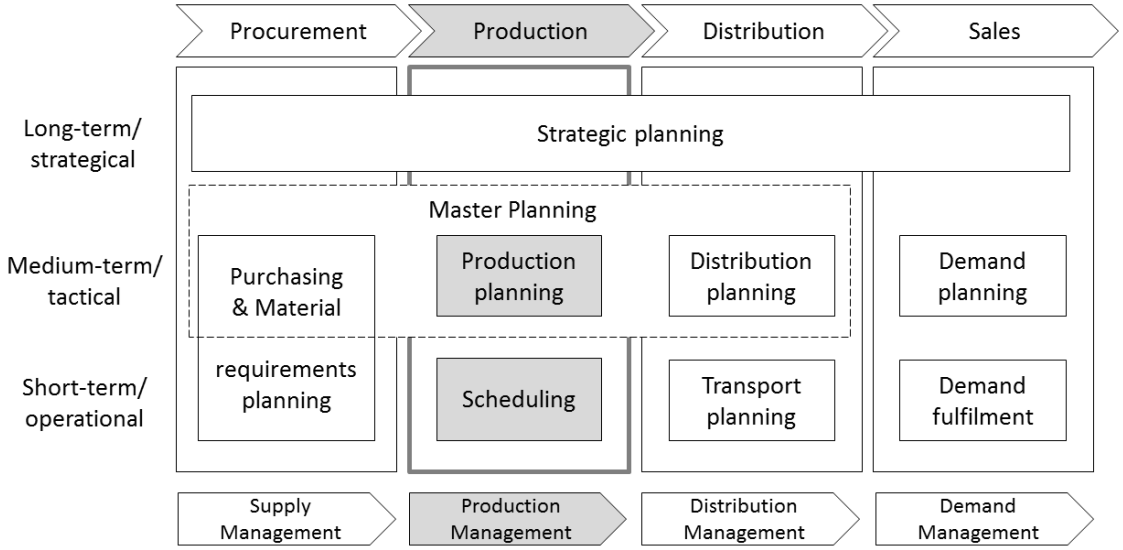

Figure 1.4: Supply chain planning matrix (Rohde et al. (2000) [70])

Rohde et al. (2000) [70] position the planning decisions in a matrix with two dimensions: "Planning horizon" and "Supply chain process" (see Figure 1.4). Due to the interconnections between different management processes of the supply chain, decisions in these processes should be taken simultaneously. Master planning coordinates decisions taken at supply management, production management and distribution management in the medium-term planning level according to the demand forecasts (demand management). Master planning provides targets and plans for each management process. For example, production quantities to be completed are determined within master planning, together with quantities to be released in each site. Furthermore, the decisions taken at different levels in the hierarchy within a management process provide targets and constraints for the next decision levels in the hierarchy and vice versa.

1.3.2 Production Management Process in Semiconductor Manufacturing

We only provide an overview on the last two decision levels in the production management hierarchy which range from planning decisions at fab (site) level to the internal scheduling and dispatching decisions within tools. These two decision levels are called the factory level and the work center level in Mönch et al. (2013) [51].

The *medium-term planning level* or *factory level* comprises the planning decisions which range from several weeks to several months. The scope is the fab and decisions are based on the current status of the fab and predetermined or forecasted demands. The main objective in this level is to determine product quantities at each period of the planning horizon to satisfy demands in a cost efficient manner. An overview of the decisions in the medium-term planning level is depicted in Figure 1.5. Decisions at this level have to interact with other decision making functions in the supply chain process such as demand forecast strategies. The medium-term planning tasks can be decomposed into aggregate (upstream) and detailed (downstream) planning levels. At the upstream level, master planning determines wafer quantities for various types of products and then due dates. The capacity status of the fab, orders and demand forecasts are input to the master planning. The capacity status can be provided by for example Rough Cut Capacity Planning which verifies available capacities to meet the capacity requirements for forecasted demand, i.e. checks the capacity of critical resources. Ponsignon and Mönch (2012) [61] describe and solve master planning problems that determine wafer quantities over products, facilities and time periods for given demand and capacity constraints. At the downstream level, capacity planning allocates production capacity to product quantities calculated by master planning to optimize some performance measures. Release quantities and dates in the fab are defined in this planning level. However, capacity planning decisions have to interact with Material Requirements Planning. Material Requirements Planning ensures the availability of materials and products for the planned production quantities. Delivery schedules and purchasing orders are proposed to minimize costs, in particular inventory

costs.

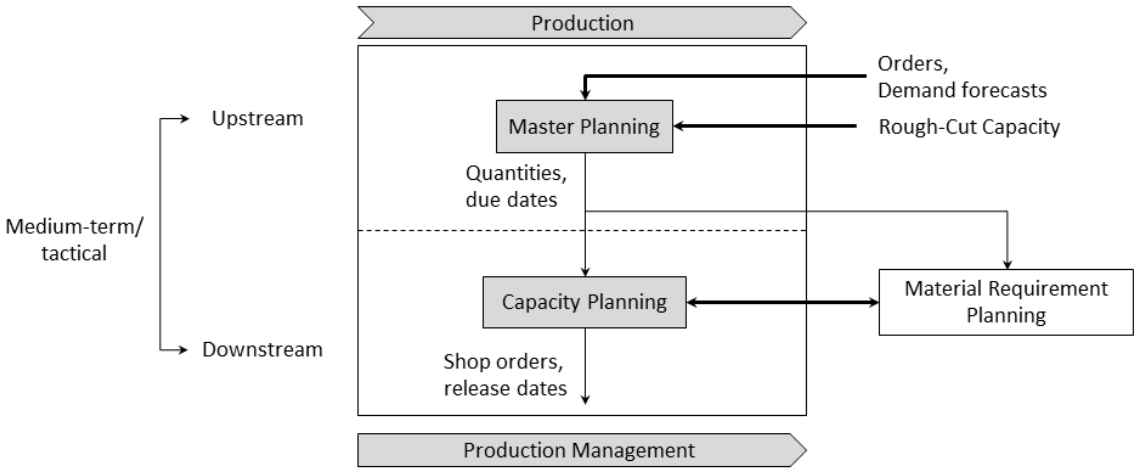

Figure 1.5: Medium-term planning level of production management process

The *short-term planning level* or *work-center level* comprises the decision making functions ranging from seconds to several weeks. The decisions at this level determine lot sizes and the sequence of the lots on the machines. The scope of these decisions ranges from workcenters to dispatching decisions within tools (e.g. cluster tools). Recall that the wafer fabrication process consists of multiple workcenters. Each workcenter is composed of multiple machines and each process step can be performed by a subset of machines in the workccenters. Process steps are restricted by various constraints. It is difficult and time consuming to schedule lots within such a complex and large scale manufacturing system. In addition, each workcenter contains specific types of machines, with their associated constraints, and has its own objectives. Accordingly, the scheduling problem at the short-term planning level is decomposed into sub-problems and the decisions are taken for each of the individual workcenters. For example, the problem of scheduling lots in the diffusion and cleaning workcenter has specific constraints and is addressed in various studies (Yurtsever et al. (2009) [89], Kim et al. (2010) [30], Yugma et al. (2012) [88], Jung et al. (2014) [27], and Knopp et al. (2017) [33]). The photolithography workcenter contains bottleneck machines and requires masks as auxiliary resources for processing operations. Scheduling decisions for such a complex and critical workcenter have been particularly studied

over the past years (Bitar (2015) [5], Pfund et al. (2008) [59]).

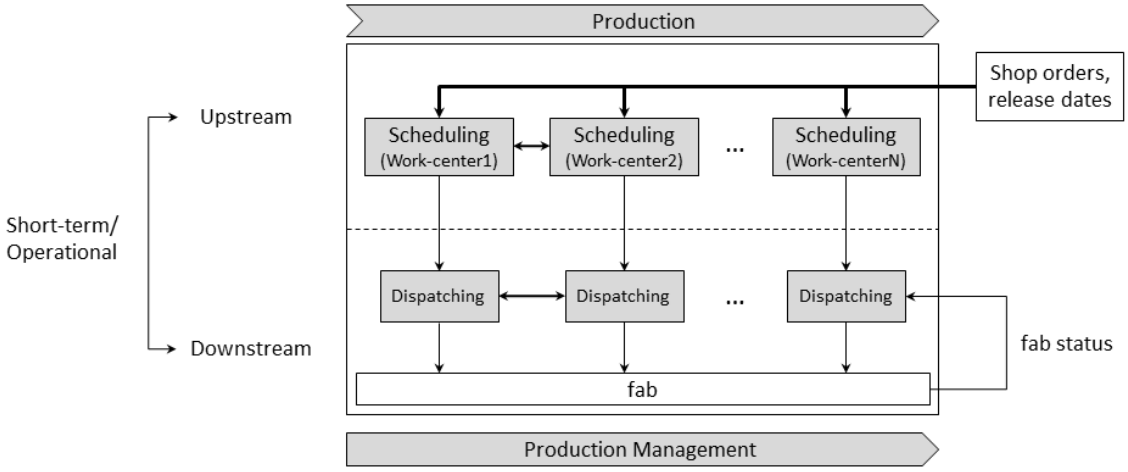

Figure 1.6: Short-term planning level of production management process in semiconductor manufacturing

An overview of the decisions in the short-term planning level is depicted in Figure 1.6. In general, scheduling decisions at the short-term planning level (or in a workcenter) deal with allocating limited resources to the lots which have to be performed. The released quantities are determined at the medium-term planning level. Dispatching rules can be applied to select lots waiting to be processed on machines. Batching strategies decide which lots are grouped in the same batch. Decisions at this level have to take into account the current status of the fab such as the status of machines (up/down). They are influenced by uncertainties in the fab such as machine breakdowns. Other decision making functions in this level support the routing of vehicles to transport lots from and to machines. In fact, scheduling decisions interact with short-term transport planning decisions.

1.4 Thesis Objective I: Ensuring the Consistency of Global and Local Scheduling Decisions

To present the first objective of this thesis, the *global* and *local* scheduling levels are introduced. The global level refers to the planning and scheduling decisions for the whole fab and comprises the decisions at downstream level of the medium-term production management process. The scheduling decisions taken at this level are illustrated as *Global scheduling* in Figure 1.7. The local level deals with scheduling and production control in each work center. This level includes the decisions taken at upstream and downstream levels of the short-term production management process (see Figure 1.7). Local scheduling decisions use local information (e.g. processing times, waiting times and queue lengths of machines) to optimize performance measures, e.g. the Shortest Process Time (SPT) dispatching rule uses the processing times of lots to maximize throughput. Global scheduling decisions use information at the global level such as the arrival of future lots, customer requests, etc., e.g. the Cost Over Time (COVERT) and Apparent Tardiness Cost (ATC) rules use information on future lots to minimize tardiness.

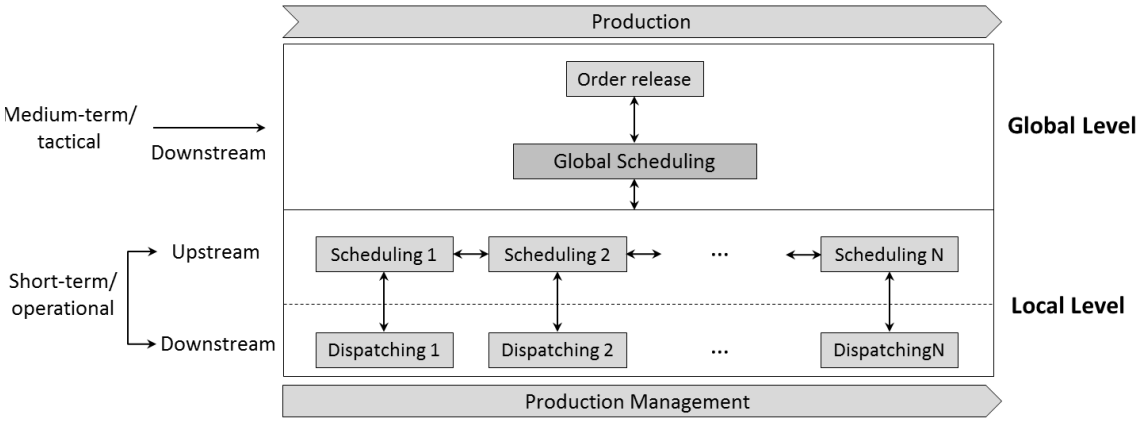

Figure 1.7: Global and and local scheduling levels in semiconductor manufacturing

The overall aim of this thesis is to propose a global scheduling approach

which ensures consistency between scheduling decisions at short-term planning level. The global approach has to take into account the interactions and interrelations between the decisions in each work center.

1.5 Time Constraint in Semiconductor Manufacturing

In wafer fabrication, time constraints are set-up between process steps to ensure the yield and quality of final products. They limit the duration between two given process steps with a maximum time not to be exceeded. For example, after a cleaning process, the chemical condition on the wafer surface deteriorates (contamination or oxidization) over time before another process is completed. To limit the chemical reaction with the environment, technology development engineers set up a time constraint between cleaning processes and furnaces processes. Wafers violating the embedded time constraint have to be scrapped or reprocessed. Also, damaged wafers may continue the production process but with unacceptable yield. Hence, violations of time constraints lead to high scrap and reprocessing costs and significantly increase the production cycle time.

A time constraint $TC_{i,j}$ covers the sequence of process steps from the end of process step PS_i to the end of process step PS_j (see Figure 1.8). The sum of waiting times and processing times in these steps should be lower than or equal to the maximum time $MaxT_{i,j}$ allowed in $TC_{i,j}$. Otherwise, wafers will be scrapped or reprocessed.

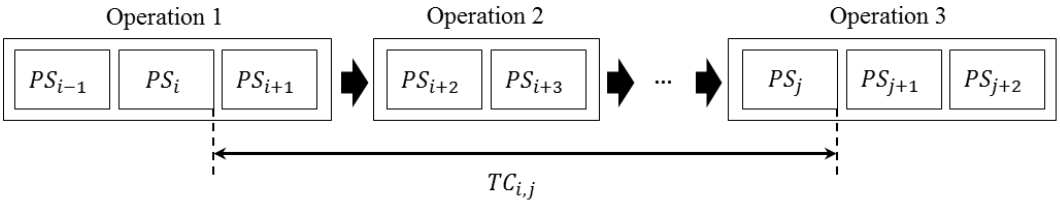

Figure 1.8: An example of a time constraint in semiconductor manufacturing

As in semiconductor manufacturing most machines are extremely expensive and it is important to satisfy customer's demand, managing time constraints to prevent scrap and reprocess costs has become a crucial issue. There are various approaches which tackle time constraints in semiconductor manufacturing. In the following we provide an overview of the related works concerning time constraints.

1.5.1 Literature review on Time Constraints

This literature review starts with an overview of the notion and different types of time constraint in semiconductor manufacturing. Then, the existing literature is presented and discussed. We classify the literature into three groups of papers based on the type of studied problems. The first group corresponds to scheduling problems in which jobs are assigned to resources with various objectives subject to respecting time constraints. The second group of papers in related to production control problems which, in general, try to determine a production rate dominated by time constraints. The third group corresponds to capacity planning problems in which the goal is to determine the maximum number of allowed lots in time constraints. *Time constraint* is a term which is used to denote the limited duration between two process steps in the route of a product. A time constraint can be maximal constraint or a minimal constraint. A maximal time constraint imposes a maximum time between two process steps not to be exceeded and a minimal time constraint imposes a minimum time which must be elapsed between two process steps (Fondrevelle et al. (2008) [17]).

There are differences between time constraints are imposed. This difference is shown in figure 1.8. $TC_{i,i+1}$ impose a maximum time limit between start date of process step PS_i until process step PS_{i+1} is completed (start-to-end). $TC_{i+1,i+2}$ presents a maximum time from the end of process step PS_{i+1} to the completion of process step PS_{i+2} (end-to-end). And $TC_{i+2,i+3}$ starts from the completion of process step PS_{i+2} until the start date of process step PS_{i+3} (end-to-start).

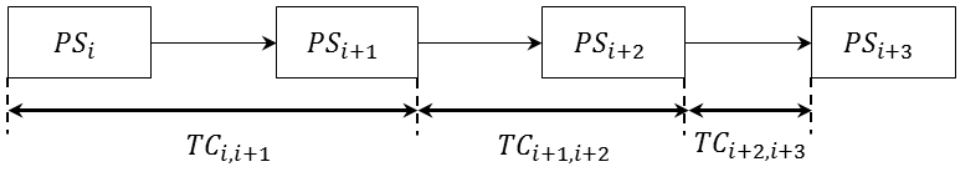

Figure 1.9: Examples of various types of time constraints

In the literature, various terms have been used for time constraint such as "Time lag", "Time bound sequence", "Queue loop", "Queue time constraint", "Delay time", "Time window", "Waiting time constraint", and "Limited waiting time". "Time lag" is a term which is commonly used in scheduling problems. A definition of time lag that comes from the Oxford dictionary is a period of time between one event and another. A time lag covers immediate consecutive process steps with maximum (not to be exceeded) or minimum time (minimal waiting time before further processing can take place).

1.5.2 Types of time constraints

Time constraints can be adjacent or overlapping and, at the same process step, one time constraint can end and another one can start. Klemmt and Monch (2012) [32] classify the different time constraints that can appear in semiconductor manufacturing in five categories.

The first category consists of two consecutive process steps, as shown in Figure 1.10.

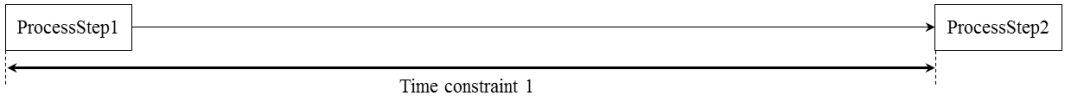

Figure 1.10: An example of the first category of time constraints

The second category allows time constraint between two process steps

that are consecutive but not adjacent in the production flow, as in the example of Figure 1.11.

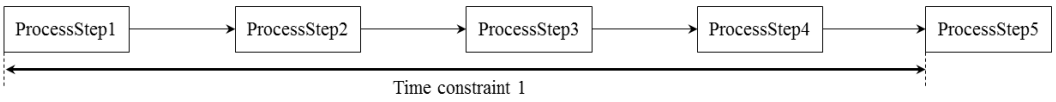

Figure 1.11: An example of the second category of time constraints

The third category belongs to both the first and second categories as shown in Figure 1.12. Note that for the two time constraints are not overlapping.

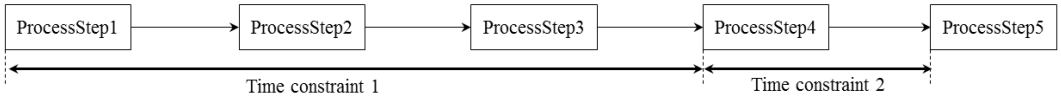

Figure 1.12: An example of the third category of time constraints

In the fourth category, overlapping of time constraints is allowed, as illustrated in Figure 1.13.

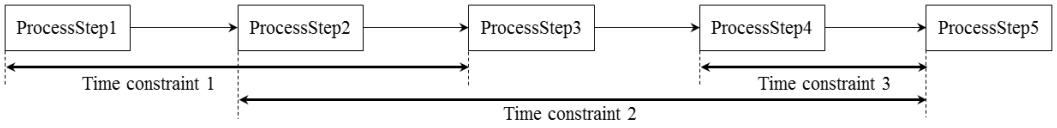

Figure 1.13: An example of the fourth category of time constraints

Finally, the fifth category contains all the time constraints from the third and fourth categories, as shown in Figure 1.14.

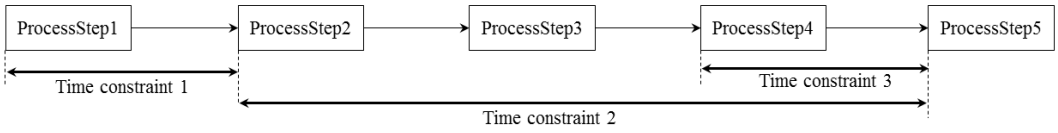

Figure 1.14: An example of the fifth category of time constraints

As in semiconductor manufacturing, productions and most machines are extremely expensive, it is crucial issue to efficiently control and manage lots in time constraints to minimize scrap and reprocess costs. There are various approaches which tackle time constraints in semiconductor manufacturing. In the following, we provide an overview of the related works literature.

1.5.3 Scheduling problems with time constraints

In general, a scheduling problem is an optimization problem in which a given set of jobs need to be scheduled on a given set of machines, while trying to minimize a criterion, often the makespan, subject to specified constraints. Here, we provide a survey of scheduling problems with time constraints. Yang and Chern (1995) [86] prove that the two-stage scheduling problem with time constraint is strongly NP-hard problem. Scholl and Domaschke (2000) [73] describe a simulation based approach that includes maximum time lag constraints. Deppner et al. (2006) [13] present a dispatching rule-based construction algorithm coupled with a cluster decomposition method in a job shop. Caumonda et al. (2008) [9] study the scheduling job-shop problem with minimal and maximal time lags. A disjunctive graph is used to model the problem. Then, they propose an approach based on a memetic algorithm. Yurtsever et al. (2009) [89] describe a custom heuristic which has been deployed in an industrial setting and which includes maximum time lag constraints. Yugma et al. (2012) [88] propose a heuristic for scheduling complex job-shops that considers maximum time lags between adjacent operations. The job-shop scheduling problems with time constraints between consecutive process steps is studied by developing decomposition approaches

based on an MIP model in Klemmt and Monch (2012) [32]. Jung et al. (2013) [28] address a multi-objective scheduling problem in the diffusion area with time constraints between multiple process steps. A Mixed Integer Programming model and a decomposition approach to minimize the violation of time constraints are proposed. Kohn et al. (2013) [34] present a parallel batch machine scheduling problem that includes maximum time lag constraints. They propose to use a VNS based approach for a parallel batch machine scheduling problem in combination with simulation. They study the correlation between different key performance indicators and discuss the incorporation of a minimum batch size constraint. Attar et al. (2013) [3] propose a Mixed Integer Programming and three metaheuristics to minimize makespan while respecting time constraints.

Various approaches are developed to deal with the flow-shop scheduling problem (FSP) with time constraints. Su et al. (2003) [78] study time constraints in a two-stage flow-shop scheduling problem with a batch processor at stage 1. They propose a mixed-integer programming model and a heuristic. Fondrevelle et al. (2006) [18] consider minimal and maximal time lags. An exact algorithm based on a branch-and-bound procedure is developed. Joo and Kim (2009) [25] consider time constraints in a two-stage flow-shop scheduling problem. They propose a mixed integer programming model and a heuristic. Fondrevelle et al. (2008) [17] and Dhouib et al. (2013) [14] study the permutation flow-shop scheduling problem with time constraints. Fondrevelle et al. (2008) [17] develop a heuristic based on dispatching rules and a branch-and-bound procedure. Dhouib et al. (2013) [14] propose a mathematical programming formulation and a simulated annealing heuristic with sequence-dependent setup.

For the hybrid flow-shop scheduling problem with time constraints, Botta-Genoulaz (2000) [7] proposes six heuristics with the objective of minimizing maximum lateness. Li and Li (2007) [40] propose a constructive backtracking heuristic composed of a recursive backtracking algorithm. Liu et al. (2008) [45] propose a constructive backtracking heuristic composed of a recursive backtracking algorithm and a Tabu Search.

Bartusch et al. (1988) [4], Dorndorf et al. (2000) [15], Schwindt and

Trautmann (2000) [74], and Nonobe and Ibaraki (2006) [56] consider maximum time lags in the the context of resource constraint project scheduling. Raaymakers and Hoogeveen (2000) [62] study no-wait constraints. Hurink and Keuchel (2001) [24] consider maximum time lags in single machine scheduling. Rossi et al. (2002) [71] consider soft time constraints in a more general setting. Zhang and van de Velde (2010) [92] consider an online open-shop problem with time lags.

1.5.4 Production control with time constraints

There are some studies which consider time constraints in production control. One of the main purposes of these studies is to provide a guideline for production control. They examine production rate under various objectives, in particular respecting time constraints. Lee et al. (2005) [38] try to manage lots within time constraints to minimize scrap rates. They control time constraints by periodically reviewing the remaining workload in the system. Wu et al. (2010) [84] examine the dynamic nature of a two-stage production system with time constraints. They propose a dynamic admission control policy based on a Markov decision model and a value iteration algorithm. They try to minimize the number of job scraps and the sum of the expected inventory holding costs. The proposed approach is extended by considering parallel processing machines in Wu et al. (2012 a) [85]. Wu et al. (2012 b) [83] investigate the same problem where the two-stage system consists of an upstream batch process machine and a downstream single process machine. Cho et al. (2014) [10] use gate-keeping decisions to specify whether the processing of an operation that initiates a maximum time lag can start. A Mixed Integer Programming model with high level abstraction is proposed.

1.5.5 Capacity planning with time constraints

Few studies discuss capacity planning to deal with time constraints. The purpose in these studies is to estimate the maximum number of buffered lots (WIP) without exceeding time constraints. Robinson (1998) [65] and Robinson and Giglio (1999) [64] propose an approximation based on M/M/c

queueing formulas for a two-element system with time constraints. The approximation provides an upper bound on the maximum loading on the machines while respecting time constraints. Kitamura et al. (2006) [31] develop an evaluation method in an M/M/1 queueing system. They estimate the upper bound of the WIP to satisfy the time constraint. Kuo et al. (2008) [36] focus on the bottleneck workstation with time constraint. They determine the WIP level for a bottleneck workstation using backpropagation neural networks. Tu et al. (2009) [80] consider time constraints in the back-end process of wafer fabrication. A GI/G/m queuing model is applied to determine the number of machines needed at each process step for the arriving rate of lots to respect time constraints. The proposed approach is extended by taking batch processing into account in Tu et al. (2011) [79].

1.6 Thesis Objective II: Managing Time Constraints

In recent years, technical changes are frequently affecting semiconductor manufacturing processes. The most recent technologies require more and more complex and constrained processes, i.e. more and more time constraints with shorter time periods and overlapping. Accordingly, controlling and managing lots limited by time constraints becomes a critical challenge. It is necessary to investigate time constraint management more specifically to limit the scraps and reprocessing.

Most studies investigate time constraints within two process steps and in the context of scheduling problems. There are only few papers in capacity planning and production control which directly deal with time constraints. To our knowledge, overlapping time constraints and various types of products have never been investigated.

The focus of this thesis is to consider complete time constraints, i.e. multiple time constraints which cover multiple process steps and overlap. The objective is to provide an admission control in real time before starting the production of a lot in the first step of the time constraint. Figure 1.15 depicts an example of a production process limited by time constraints with an

admission control policy.

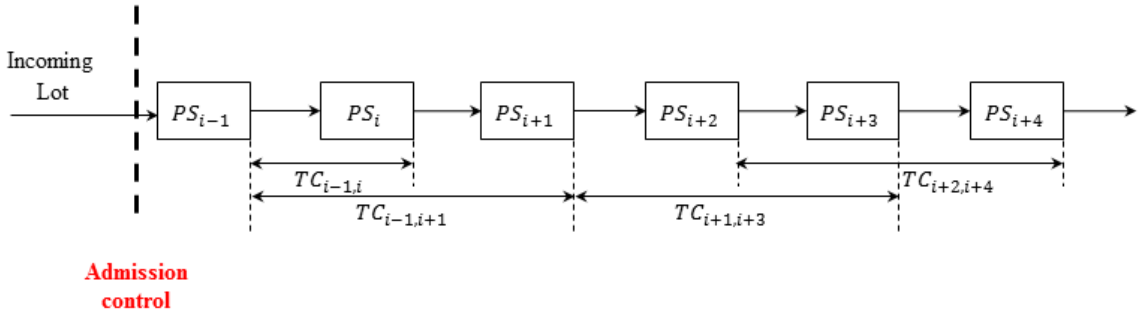

Figure 1.15: Admission control policy for a production process flow in semiconductor fab with time constraints

1.7 Overview and Main Contributions

We present an overview of the structure of this work and highlight the main contributions of each chapter.

Chapter 2, Production Control with Time Constraints

This chapter considers the problem of managing time constraints in wafer fabrication. Our main contribution is to propose multiple approaches which control lots before they enter time constraints to prevent wafer scrap and reprocessing. Multiple types of products and multiple time constraints with overlapping are considered.

Chapter 3, A Multi-Method Generic Simulation Model for Semiconductor Manufacturing

This chapter proposes a simulation model for a semiconductor manufacturing facility to evaluate the approaches proposed in this thesis. The main contribution is the development of a data-driven generic simulation model. A conceptual modelling framework is presented and is then used to develop the model.

Chapter 4, A Framework for Consistency between Global and Local Scheduling Decisions

This chapter considers the consistency problem between global and local scheduling decisions. The main contribution is presenting a general framework which ensures the consistency between global objectives and local scheduling decisions. Two global objectives are considered and, for each objective, an evaluation-based global strategy is presented.

Chapter 5, An Approach for Consistency between Global and Local Scheduling Decisions

This chapter studies the same problem than chapter 4. The main contribution is an optimization model for the global objectives. Three global objectives are considered.

Chapter 2

Production Control with Time Constraints

Time constraints between production steps in wafer fabrication become more and more important as technologies are more and more advanced. Constraints between two steps are necessary due to quality and yield of wafers. Lots of wafers violating these constraints have to be scrapped or reprocessed. Due to the advance in technology, constraints are now shorter than before and tend to be chained and overlap in many cases. Consequently managing time constraints properly is now critical to ensure quality and avoid scrap of wafers and becoming an extremely complex problem. The goal is to develop an approach to control lots before they enter time constraints to prevent wafer scraps or reprocessing.

2.1 Introduction

This chapter specifically deals with time constraints in the wafer fabrication process. A time constraint limits the waiting and processing times between two process steps of lots of a given product. In practice, physical and chemical processes set up time constraints between different steps to ensure the quality of the final product. For example, the time between some steps must be limited to avoid contamination and oxidation. Lots that violate a time constraint have to be scrapped or reprocessed for qualitative recovery. This imposes high reprocess and scrap costs and increases cycle times which may cause to exceed delivery dates. This chapter aims at presenting an approach to control lots before starting the production in time constraints. Regarding the classification of time constraints presented by Klemmt and Monch (2012), the time constraints that we consider in our study belong to the most general class that allows overlapping between time constraints as well as time constraints which cover multiple process steps.

First, we describe a new notion called *time constraint tunnel* to provide a clear problem statement. A Time Constraint Tunnel (TCT) includes a set of time constraints between two safety zones. A safety zone is a sequence of process steps without time constraints. An example of time constraint tunnels can be found in Figure 2.1.

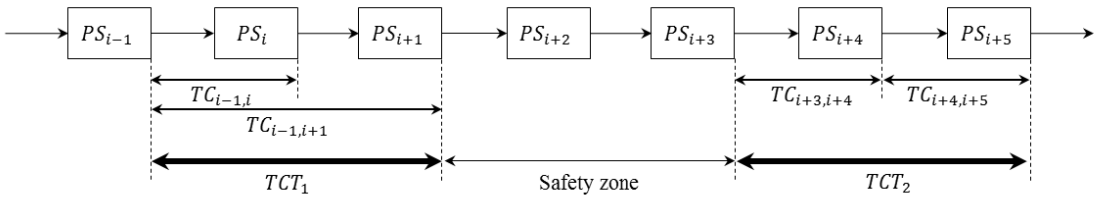

Figure 2.1: An example of Time Constraint Tunnels (TCTs)

The focus of this thesis is to investigate time constraints within a TCT, because respecting a time constraint will depends on respecting adjacent and overlapping time constraints. For example, assume that $TC_{i+3,i+4}$ within TCT_2 is not respected by an incoming lot. It is a waste of cost and time if

the lot with unacceptable yield is allowed to be produced in the remaining process steps, even if $TC_{i+4,i+5}$ is respected by the lot.

Actually, the current lots in the fab and the real time status of machines directly impact the satisfaction of time constraints. These elements induce waiting times within time constraints and increase the risk of violating the time constraints. The objective is to provide an admission control right in front of each TCT in a real fab situation. The admission control will be activated before initiating lots into the first process step restricted by a time constraint tunnel. It estimates whether the incoming lot may violate the time constraints or not. If the time constraints are satisfied, the incoming lot enters the TCT, otherwise it would be put on hold. Figure 2.2 depicts an example of a production process limited by a TCT with an admission control. The problem is formally described in section 2.2.

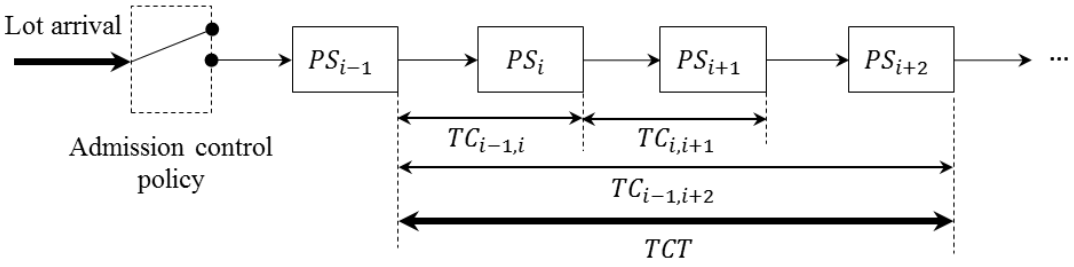

Figure 2.2: An admission control for a time constraint tunnel

A review of the literature on time constraints is presented in section 1.5. Time constraints have been specifically studied in the context of capacity planning and production control problems. Capacity planning problems study time constraints between two process steps and estimate the maximum number of lots which can be buffered in the process steps without violating time constraints. An incoming lot can respect its time constraint if the buffered lots are lower than the estimated number of lots in the buffer. However, because of the dynamic situation in fabs, time constraints with multiple process steps and overlapping between them which affect the number of allowed lots should be considered. Production control problems estimate the

production rate under time constraints. They provide a control policy which hold or allow incoming lots in the production process flow. A two-stage system with only one type of products is considered in these problems. In this chapter, we present three approaches to control incoming lots in time constraint tunnels:

1. *WIPMax Calculation approach*: The admission control is based on the concept of capacity which is commonly used for managing time constraints in real fabs. The parameter *WIPMax* is defined which represents the maximum number of lots which can be buffered in the TCT. An incoming lot can enter the TCT if the number of current lots in the TCT is lower than WIPMax. This approach which is explained in detail in section 2.3 has major drawbacks in supporting time constraints with more than two process steps.

2. *Deterministic approach*: This approach evaluates whether a given lot can respect time constraints within a TCT. This approach which is presented and discussed in section 2.4 is based on computing the completion times of the incoming lot within time constraints.

3. *Probability Estimation approach*: This is an extended version of the deterministic approach which is developed to estimate the probability of satisfying each time constraint within a TCT. This approach is described in section 2.5.

The main purpose of the "Deterministic" and "Probability Estimation" approaches is to provide a guideline for decision maker in the fab. In addition, additional indicators can be extracted using these approaches which enables decision makers to understand the root causes of time constraint violations. Section 2.6 includes discussion on these indicators. Numerical results on various industrial instances are presented and discussed in section 2.7.

2.2 Formal Problem Description

This section provides a formal description of the problem at hand based on the notation presented in chapter 1. We are given a lot l_O of product type α with a route r_α with q_α process steps. The route r_α is limited by a set of time constraint tunnels $TCT^\alpha = \{TCT^\alpha_{m,n} | 1 \leq m < n \leq q_\alpha\}$ where $TCT^\alpha_{m_1,n_1} \bigcap TCT^\alpha_{m_2,n_2} = \phi$ for any m_1 and m_2 such that $m_1 \neq m_2$. A time constraint tunnel $TCT^\alpha_{m,n}$ is a combination of time constraints between process step $PS_{\alpha,m}$ and process step $PS_{\alpha,n}$, which corresponds to the set $TCT^\alpha_{m,n} = \{TC^\alpha_{g,h} | 1 \leq g < h \leq q_\alpha \quad \text{where} \quad m \leq g \quad \text{and} \quad h \leq n\}$. A time constraint $TC^\alpha_{g,h}$ is associated to a maximum time $MaxTime^\alpha_{g,h}$ that a lot l_i of type α can spend between the two process steps $PS_{\alpha,g}$ and $PS_{\alpha,h}$. Overlapping may occur between the time constraints $TC^\alpha_{m,n}$, such that

$$\exists TC^\alpha_{g,h} \in TCT^\alpha_{m,n} \quad and \quad \exists TC^\alpha_{v,w} \in TCT^\alpha_{m,n} \quad \text{where} \quad g \leq v \leq min(h,w) \tag{2.1}$$

Time constraint tunnels should be respected by lot l_O, otherwise reprocessing or scrap will happen. A set of lots $L = \{l_i | i = 1, \ldots, n\}$ of various types of products are already being processed in the fab when l_O is at the entrance of a TCT. Hence, the satisfaction of a TCT by l_O depends on how the other lots are processed on the machines on which l_O can be processed. In semiconductor fabs, machines are shared between different routes. Thus, in addition to the route r_α, lots with routes with common machines with r_α must be considered. The real time status of machines (up or down) are taken into account.

The problem is illustrated in Figure 2.3 on an example with two routes and one time constraint tunnel. Assume l_O arrives in $PS_{\alpha,i}$ where a time constraint tunnel with three time constraints is starting. Lots l_1 and l_2 are already being processed. Process steps $PS_{\alpha,i+1}$ and $PS_{\alpha,i+3}$ of lot l_O can respectively only be processed on M_6 and M_8, and these machines could be used to process steps of lots l_1 and l_2. A waiting time may happen for l_O which may cause a violation of its time constraints. Therefore, before starting l_O in the TCT, an admission control is required to predict whether

the incoming lot l_O satisfy the time constraints in the TCT or why are the time constraints violated.

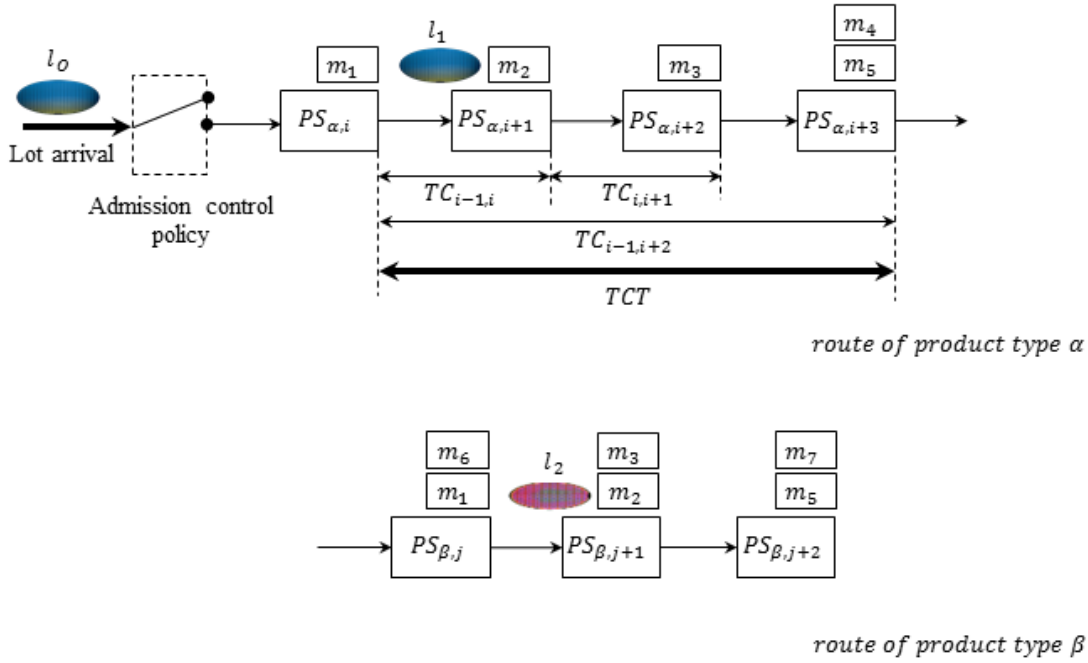

Figure 2.3: Production line in a semiconductor fab with time constraints

2.3 WIPMax calculation approach

Since a time constraint limits the sum of the waiting time and the processing time between two process steps, the satisfaction of time constraints strongly depends on the lots waiting in the process steps that are covered by the time constraints. The WIPMax calculation approach tries to compute the maximum number of lots which can be buffered in a time constraint tunnel to satisfy a time constraint. Lots arriving at the first step can enter the TCT if the following condition is valid:

$$\text{lots currently in } TCT < WIPMax \tag{2.2}$$

Below, the *parameters* and *variables* used throughout this approach are presented.

Parameters

$TC_{g,h}$	Time constraint between process steps PS_g and PS_h
$MaxTime_{g,h}$	Maximum time of time constraint $TC_{g,h}$
$WIPMax_{g,h}$	Maximum number of lots waiting in $TC_{g,h}$
N	Number of process steps covered by $TC_{g,h}$
P_i	Processing time in i^{th} process step

Variables

N_i	The number of lots waiting in front of PS_i
$CTime_{g,h}$	The completion time between two process steps PS_g and PS_g

In the literature, respecting time constraints using the WIPMax concept is studied in capacity planning problems in a two-stage system(Kitamura et al.(2006))(see Figure 2.4). In this system, a lot is allowed to start processing in process step PS_{i-1} if it has a chance to be completed on time. The concept of capacity (or WIPMax) is defined for a time constraint to control incoming lots. The parameter $WIPMax_{i-1,i}$ of time constraint $TC_{i-1,i}$ is the maximum number of lots that can be buffered in PS_i without exceeding $maxTime_{i-1,i}$. Thus, an incoming lot can start processing in PS_{i-1} if the number of lots waiting in process step PS_i is lower than $WIPMax_{i-1,i}$.

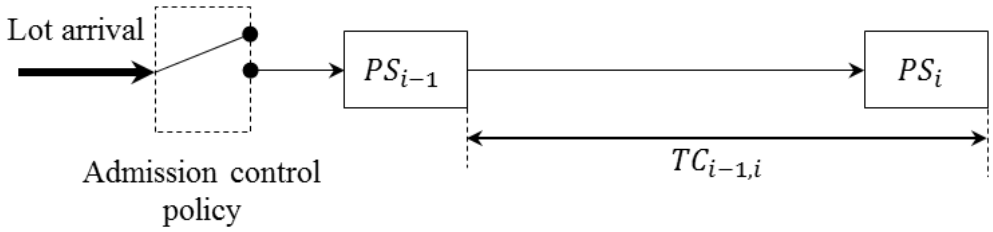

Figure 2.4: An example of a time constraint with two process steps

Here, we extend the concept of WIPMax for a time constraint tunnel. Lot l_O is allowed to enter the processing in $TCT_{p,q}$ if the sum of lots waiting in process step PS_p up to process step PS_q is lower than $WIPMax_{p,q}$.

We first propose an algorithm to estimate the parameter WIPMax of time constraint $TC_{g,h}$ which covers multiple process steps. The proposed algorithm is based on estimating the maximum allowable completion time for lot l_O between the first process step PS_p and the last process step PS_q while satisfying $TC_{g,h}$. The completion time called $CTime_{g,h}$ which must be lower than $MaxTime_{g,h}$. Assume $TC_{g,h}$ covers N process steps, and N_i with $g \leq i \leq h$ is the number of lots waiting in front of process step PS_i. The idea is to estimate the maximum possible value for each N_i while $CTime_{g,h}$ still remains lower than $MaxTime_{g,h}$. The WIPMax calculation algorithm for a time constraint is presented below.

Algorithm 2.1. WIPMax Calculation approach for time constraint $TC_{g,h}$

1: Initialize $N_i = 1$ with $g \leq i \leq h$ ($N_g = 1$ is the incoming lot l_O)
2: **for** $i = g + 1$ to h **do**
3: **while** $CTime_{g,h} \leq MaxTime_{g,h}$ **do**
4: $N_i = N_i + 1$
5: **end while**
6: Fix the value for N_i
7: **end for**
8: $WIPMax_{g,h} = \sum_{i=g+1}^{h} N_i$

First, we set one lot waiting in each process step within $TC_{gh}(N_i = 1)$. We assume that there are no lots waiting in the first process, so $N_g = 1$

represents the incoming lot l_O. The process of calculating N_i starts from process step PS_{g+1} and respectively continues until the last process step. This is because lots waiting in PS_i impose a longer waiting time for l_O compared to the lots waiting in PS_{i+1}. The value of N_j with $g < j < h$ is estimated by computing the completion time of l_O in $TC_{g,h}$ whereas the value of each N_i with $i < j$ has been already fixed and the value of each N_i with $i > j$ has been initialized to 1. With regards to this conditions, the value of N_j increases as long as $CTime_{g,h} \leq MaxTime_{g,h}$. The value obtained for N_j is fixed and the procedure continues with estimating the value of N_{j+1}. The above steps are repeated for each process step in TC_{gh}. Finally, the maximum number of lots which can be buffered in $TC_{g,h}$ is the sum of the value obtained for each N_i.

After computing WIPMax for a time constraint with multiple process steps, we propose the following approach to estimate WIPMax for a time constraint tunnel.

Algorithm 2.2. WIPMax Calculation approach for time constraint tunnel $TCT_{p,q}$

Sort time constraints in $TCT_{p,q}$ by beginning step and shortest length or longest length
$TC_{g,h}$ ranked before $TC_{k,l}$ where $g \leq k$.
Length is defined as the number of existing process steps in a time constraint
for all ranked $TC_{g,h} \in TCT_{p,q}$ **do**
    Apply *Algorithm 2.1* to compute $WIPMax_{g,h}$
    for $i = g$ to h **do**
        if N_i has not already been fixed **then**
            Fix the value for N_i
        else
            if $N_i >$ Obtained value in this stage **then**
                Replace the value of N_i by the obtained value in this stage
            end if
            if $N_i <$ Obtained value in this stage **then**
                Do not change the value of N_i
            end if
        end if
    end for
end for
$WIPMax_{p,q} = \sum_{i=P+1}^{q} N_i$

The impact of overlapping between time constraints on WIPMax of time constraint tunnel $TCT_{p,q}$ is taken into consideration through computing WIPMax of each time constraints in $TCT_{p,q}$. The first step is to determine a sequence for time constraints within $TCT_{p,q}$. Time constraints are sorted according to the beginning step because the incoming lot goes through time constraints which are triggered earlier. $TC_{g,h}$ ranked before $TC_{k,l}$ where $g < k$. In the case where several time constraints are triggered simultaneously, the order should be specified based on the *length* of time constraints. Length is defined as the number of existing process steps in a time constraint. Then, Algorithm 2.1 will be applied on each time constraint according to the specified order. The obtained value for N_i in each stage will be affected by other time constraints in $TCT_{p,q}$ which are overlapping. Assume that process step PS_i is covered by two time constraints $TC_{g,h}$ and $TC_{k,l}$ where

$TC_{g,h}$ ranked before $TC_{k,l}$. The value of N_i is fixed by applying Algorithm 2.1 on $TC_{g,h}$. When we apply Algorithm 2.1 on $TC_{k,l}$, the value of N_i has already been fixed in the previous stage. To take overlapping into account and satisfy both time constraints at the same time, if the obtained value for N_i in this stage is lower than its fixed value, then the fixed value of N_i will be replaced by the obtained value in this stage.

The performance of the proposed approach is investigated on the time constraint tunnel $TCT_{1,3}$ presented in Figure 2.5. It includes two time constraints $TC_{1,2}$ and $TC_{1,3}$ whose maximum times are 3 and 12 respectably. Each process step can be performed by an independent machine with processing times $\{P_1 = 0.019, P_2 = 0.018, P_3 = 0.058\}$. We examine the algorithm within two examples which differ in the sorted list. In Example 2.3.1 time constraint are sorted according to the beginning step and shortest length. The sorted list with longest length is considered in Example 2.3.2.

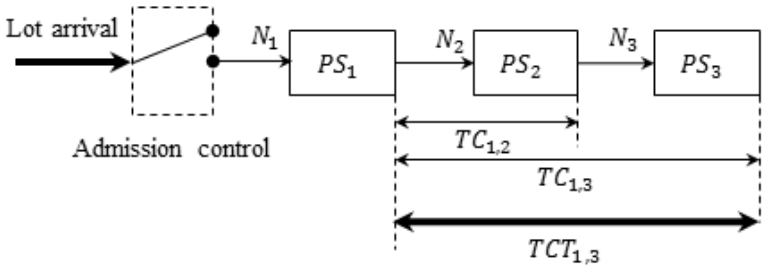

Figure 2.5: Production line in a semiconductor fab with a time constraint tunnel

Example 2.3.1. Sorted time constraints list: $\{TC_{1,2}, TC_{1,3}\}$
Starting with N_1 initialized to 1, which represents the incoming lot, we apply the WIPMax calculation approach on each time constraint according to their order in the sorted list.
WIPMax of $TC_{1,2}$:
For $TC_{1,2}$, the number of lots which can wait in front of PS_2 while $CTime_{1,2} <$ 3 estimates 159 and so N_2 is fixed to 159.
WIPMax of $TC_{1,3}$:
To compute WIPMax of $TC_{1,3}$,N_1 and N_2 are already fixed and only N_3

should be estimated. Note that before incoming lot l_O, N_2 lots at PS_2 and $N_3 + N_2$ lots at PS_3 will be processed which will cause waiting time and probably violation of $TC_{1,3}$. N_3 is fixed to 47 where $CTime_{1,3} < 12$.

WIPMax of $TCT_{1,3}$:

Finally, 206 lots can be buffered in $TCT_{1,3}$ while respecting time constraints $TC_{1,2}$ and $TC_{1,3}$ ($WIPMax_{1,3} = N_2 + N_3 = 206$).

Let us assume, when lot l_O arrives in front of $TCT_{1,3}$, that 200 lots are already in the TCT and all lots are accumulated in $TC_{1,2}$. Based on the approach, 6 lots ($WIPMax_{1,3}-$ *Current number of lots*) can enter the TCT. Although $TC_{1,3}$ within $TCT_{1,3}$ can be respected, according to the proposed approach, the incoming lot will violate $TC_{1,2}$ because the current number of lots is larger than $WIPMax_{1,2}$.

Example 2.3.2. Sorted time constraints list: $\{TC_{1,3}, TC_{1,2}\}$

Starting with N_1 initialized to 1, which represents the incoming lot, we apply the WIPMax calculation approach on each time constraint according to their order in the sorted list.

WIPMax of $TC_{1,3}$:

By applying Algorithm 2.1 on $TC_{1,3}$, N_2 and N_3 are fixed to 205 and 1 where $CTime_{1,3} < 12$.

WIPMax of $TC_{1,2}$:

To compute WIPMax of $TC_{1,2}$, N_2 is already fixed from the previous stage. We should examine whether $TC_{1,2}$ can be satisfied by the fixed value or not. After applying the WIPMax calculation approach on $TC_{1,2}$, N_2 is estimated to be 159. To satisfy $TC_{1,2}$ and $TC_{1,3}$ at the same time, the value of N_2 will be replaced by 159.

WIPMax of $TCT_{1,3}$:

Finally, 160 lots can be buffered in $TCT_{1,3}$ where $WIPMax_{1,3} = N_2 + N_3 = 160$.

Let us assume, when lot l_O arrives in front of $TCT_{1,3}$, that 160 lots are already in the TCT and all lots are accumulated in front of process step PS_3. Based on the approach, no lot ($WIPMax_{1,3}-$ *Current number of lots*) can enter the TCT. Whereas, if the two time constraints are investigated separately, 159 lots can enter $TC_{1,2}$ and 45 lots can enter $TC_{1,3}$. Two time

constraints have the same beginning step, therefore, 45 lots can enter the TCT without a violating of time constraints.

Contrary to what was expected, the result of the proposed approach depends on how time constraints within a TCT are ranked. In addition, the analysis shows that satisfying time constraints is influenced by the position of lots in the TCT. Accordingly, it is not possible to provide a relevant admission control in front of a TCT based on the WIPMax concept. In the following sections, we present approaches which evaluate time constraints that consider the current position of buffered lots in the TCT.

2.4 Deterministic Approach

We first propose a deterministic approach which determines whether a given lot can satisfy a TCT by considering the current status of the fab. Lots which are already waiting within a time constraint may be scheduled before the incoming lot. These lots and those already being processed on machines can cause long waiting times for the incoming lot and accordingly a violation of time constraints. The deterministic approach estimates whether a given lot can be completed within its TCT by considering lots currently being processed in the fab. Hence, our approach is based on computing the completion times of the incoming lot in the first process steps and the last process steps of its time constraints and then comparing with the maximum time allowed by the time constraints. The complexity of the problem is increasing with the diversity of routes and products which is a feature of high-mix/low volume semiconductor fabs. In addition to the route of the incoming lot, production flows of products that do not follow the same route but share the same set of equipment must be taken into account.

To determine the completion times of the incoming lot, we first model the problem through a disjunctive graph representation. Disjunctive graphs, introduced by Roy and Sussmann (1964) are one of the most popular approaches for modeling scheduling problems. A disjunctive graph consists of a set of nodes associated with process steps and a set of conjunctive (oriented) arcs between every two consecutive process steps and a set of disjunc-

tive (non-oriented) arcs between process steps that may be processed on the same machine. The routes of various types of products with hundreds of process steps, a huge number of lots in progress and sets of machines in each step can be modeled using this graph. A conjunctive graph can be obtained by applying a scheduling approach on the disjunctive graph and fixing a direction to each disjunctive arc or eliminating them. The completion time of a lot within a time constraint is the length of the longest path between two nodes in the conjunctive graph linked to the first process step and the last process step of the considered time constraint. We use a list scheduling algorithm to assign and schedule lots on the machines. This algorithm has been developed for acyclic graph which is used to describe the data dependencies and precedence relationships between tasks. The computation is based on topological orderings. In an acyclic directed graph, a topological ordering is a linear ordering of the nodes of the graph. The advantage of a list scheduling algorithm is its low computational complexity. Hence, completion times of lots in each process step are computed. Section 2.4.1 provides the disjunctive graph representation of the problem at hand. Section 2.4.2 describes our implementation of a list scheduling algorithm. Finally, the evaluation of time constraints for a given lot is presented in section 2.4.3.

2.4.1 Disjunctive Graph Representation

Disjunctive graphs model properties and components of a scheduling problem. Nodes correspond to operations (process steps) and arcs represent constraints. For the problem at hand, we apply a disjunctive graph $G = (V, C, E)$, similar to the one described by Dauzere-Peres and Paulli (1997). This model allows the assignment and sequencing of process steps in an integrated way. The following terminology is required for the disjunctive graph representation.

Notation

r_α	Route of product type α
q_α	Total number of process steps of r_α

l_α	Total number of lots of type α to be processed
N_p	Total number of different routes
$PS^i_{\alpha,j}$	j^{th} process step of r_α which have to be performed for lot l_i
$C^i_{\alpha,j}$	Completion time of lot l_i of type α in process step j
$P^k_{\alpha,j}$	Processing time of j^{th} step of route r_α on machine m_k with $m_k \in M_{\alpha,j}$
$S^i_{\alpha,j}$	Starting time of process of lot l_i in j^{th} process step of r_α
V	Set of nodes with one node per process step of a lot
C	Set of conjunctive arcs
E	Set of disjunctive arcs
$V^i_{\alpha,j}$	Node corresponds to the j^{th} process step of r_α which has to be performed for lot l_i
S	Artificial start node
F	Artificial end node

We use the notation $V^i_{\alpha,j}$ for a node that represents the j^{th} process step of lot l_i of type α corresponds to the route r_α with q_α process steps. l_α represents the number of lots of type α which have to be performed in the fab.

The arcs are partitioned into two sets C and E. C is the set of conjunctive arcs that represents precedence relations between process steps of routes. For each route r_α with $1 \leq i \leq l_\alpha$ and $1 \leq j \leq q_\alpha$, all conjunctive arcs $(V^i_{\alpha,j}, V^i_{\alpha,j+1})$ are added to the disjunctive graph. Figure 2.6.a represents the precedence constraints on the problem described in section 2.2. This graph itself is a *Directed Acyclic Graph* (DAG). Finally, we connect the artificial node S to the node corresponds to the starting process steps $PS^i_{\alpha,1}$, and the

artificial node F to the node corresponds to the last process steps PS^i_{α,q_α} with $1 \leq \alpha \leq N_p$ and $1 \leq i \leq l_\alpha$. In a disjunctive graph, the length of a conjunctive arc is equal to the processing time of the process steps from which it starts, e.g. arc $PS^i_{\alpha,j} \rightarrow PS^i_{\alpha,j+1}$ has a length $P^k_{\alpha,j}$ depending on machine m_k which should be selected from set $M_{\alpha,j}$.

E is the set of disjunctive arcs between process steps that may be processed on the same machine. For each $\alpha, \beta \in \{1, \ldots, N_p\}$ and $1 \leq i \leq l_\alpha$, $1 \leq h \leq l_\beta$, a disjunctive arc E_k is added between two process steps $PS^i_{\alpha,j}$ and $PS^h_{\beta,g}$ if $m_k \in M_{\alpha,j} \cap M_{\beta,g}$. Thus, there may be multiple disjunctive arcs between two nodes with different lengths because the processing time of a process step depends on the machines. Figure 2.6 shows a disjunctive graph for the problem in section 2.2. This disjunctive graph models all possible assignments of process steps to machines and sequences of process steps on the machines using undirected arcs. By replacing undirected arcs by directed arcs while satisfying some feasibility constraints, a conjunctive graph is constructed which corresponds to an assignment of process steps to machines and an sequencing of process steps on the machines.

Finally, the completion time of lot l_i of type α can be computed using the resulting conjunctive graph. It is equal to the length of the longest path from node S to the node associated to the last process step in route r_α plus the processing time in this step. Let $L(v, w)$ be the length of the longest path between node v and node w. The completion time of lot l_i in process step j, denoted by $C^i_{\alpha,j}$, is

$$L(S, V^i_{\alpha,j}) + P^k_{\alpha,j} \quad where \quad m_k \in M_{\alpha,j} \tag{2.3}$$

In fact, $L(S, V^i_{\alpha,j})$ is the time that lot l_i starts its jth process step of route r_α, denoted by $S^i_{\alpha,j}$. In the following section, we propose a scheduling algorithm to determine the completion times of lots in their process steps using the disjunctive graph.

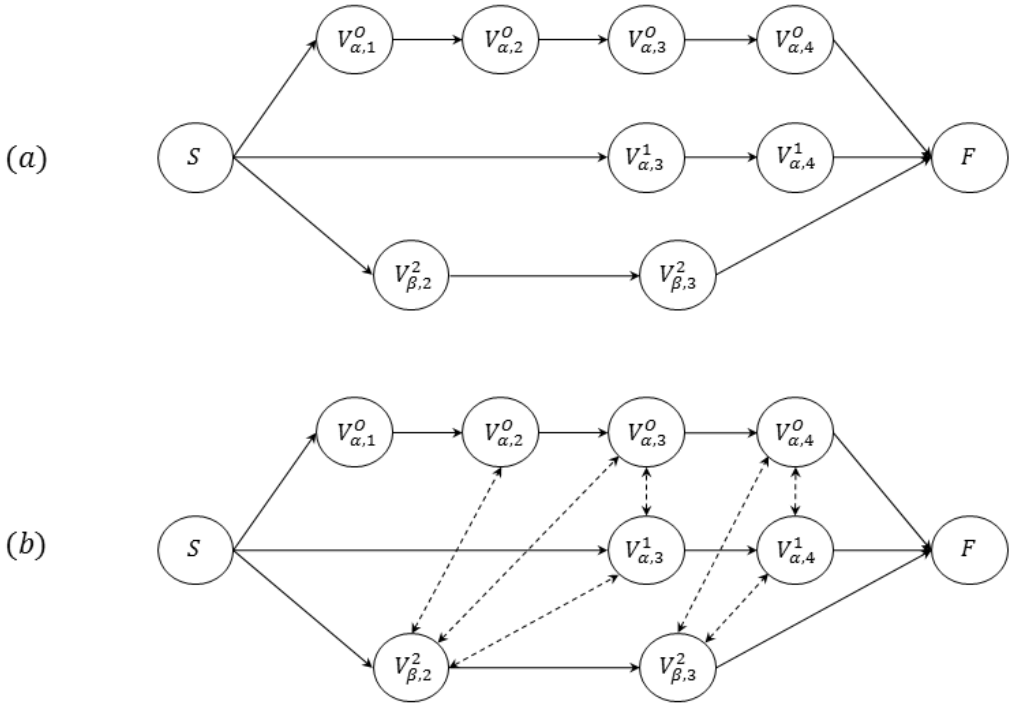

Figure 2.6: Disjunctive graph model

2.4.2 List Scheduling Algorithm

A list scheduling algorithm is a greedy heuristic which is commonly used for resource constrained scheduling problem (Cooper et al. 1998). The basic list scheduling algorithm has originally been developed for acyclic graphs with positive weight arcs. It constructs a schedule with all nodes in the acyclic graph that satisfies precedence constraints in the graph and the resource constraints of the machines. The following notations are used in the algorithm:

Notations

ReadyList	Unscheduled nodes whose predecessors are scheduled
ProcessingList	Set of nodes that are being processed
$Idle_k = \{1, 0\}$	Status of machine M_k, i.e. $Idle_k = 1$ if M_k is idle

$CurrentTime$	Current time in the algorithm

This algorithm contains three main steps: 1) Determining unscheduled nodes whose predecessors are scheduled, 2) Defining a priority to the available nodes and 3) Assigning nodes with the highest priority to an idle machine. This procedure continues until all nodes are scheduled. Algorithm 2.3 provides the pseudo-code for the scheduling algorithm used in this study. The Directed Acyclic Graph (DAG) presented in the previous section is set as input data. Each node in the graph can be performed by a set of machines. The list of nodes whose parents have already been scheduled is called "ready list". Initially, the ready list includes the nodes related to the first process step of each lot. A priority is assigned to each node according to the duration of their availability within the ready list (FIFO strategy is set as the priority rule). Then, the nodes with the highest priority in the ready list are first assigned to machines which are idle and able to execute them. The assigned nodes are first removed from the ready list and then are placed in the processing list. The processing list is defined to distinguish between ready nodes and those that are being processed. Once the processing of a node is completed, it is removed from the processing list and its following node in the route is added to the ready list. The associated machine becomes available and the procedure that assigns the ready nodes on machines will be triggered. The above steps are repeated until all nodes are scheduled. Contrary to classical scheduling problems, our objective is not to find an optimal schedule of lots but to find a feasible and realistic schedule and to compute the completion times of the lots in their process steps.

Algorithm 2.3. List Scheduling algorithm

Initialize *ReadyList* with the nodes without predecessors in the graph,
$ProcessingList = \phi$,
$Idle_k = 1$ for all machines M_k,
$CurrentTime = 0$.

1: **while** $ReadyList \neq \phi$ **do**
2: **for** each $V^i_{\alpha,j} \in ReadyList$ **do**
3: **if** machine $M_k \in M_{\alpha,j}$ with $Idle_k = 1$ exists **then**
4: $Idle_k = 0$
5: $S^i_{\alpha,j} = CurrentTime$
6: $C^i_{\alpha,j} = S^i_{\alpha,j} + P^k_{\alpha,j}$
7: Add $V_{i,j}$ to $ProcessingList$
8: Remove $V^i_{\alpha,j}$ from $ReadyList$
9: **end if**
10: **end for**
11: Select node $V^p_{\beta,q}$ with smallest completion time from $ProcessingList$
12: $CurrentTime = C^p_{\beta,q}$
13: **for** all $V^m_{\gamma,n} \in ProcessingList$ **do**
14: **if** $C^m_{\gamma,n} \leq CurrentTime$ **then**
15: $Idle_k = 1$ (M_k is occupied by $V^m_{\gamma,n}$)
16: Add all following nodes of $V^m_{\gamma,n}$ to $ReadyList$
17: Remove $V^m_{\gamma,n}$ from $ProcessingList$
18: **end if**
19: **end for**
20: **end while**

2.4.3 Evaluating Incoming lot

Consider lot l_O which is available at a process step where one time constraint tunnel $TCT^{\alpha}_{m,n}$ starts. A time constraint $TC^{\alpha}_{g,h} \in TCT^{\alpha}_{m,n}$ is satisfied by l_O if

$$C^O_{\alpha,h} - C^O_{\alpha,g} \leq MaxTime^{\alpha}_{g,h} \tag{2.4}$$

$C_{O,h}$ and $C_{O,g}$ are estimated with a list scheduling algorithm based on the priority rule FIFO. Hence the inequality is only checked with one schedule and a Yes/No answer is provided. To consider the impact of different sequences of lots on whether time constraints are satisfied or not, we present

in section 2.5 an alternative approach which enables us to take into account different sequences of lots within the ready list.

2.5 Probability Estimation Approach

In the deterministic approach, one schedule is computed and the given answer is either Yes (the lot has a completion time lower than the TCT duration) or No (the completion time is strictly larger than the TCT duration). However, a time constraint may be respected or violated with another schedule because it has impact on the completion time of a lot within time constraints. Accordingly, to better evaluate time constraints, different schedules should be considered. Our goal is to provide a probability for the lot to satisfy the time constraints. The notion of service level is adapted to express the probability in a time constraint satisfaction problem. The empirical probability based on the list scheduling algorithm estimates the probability of satisfying time constraints.

2.5.1 Service level

The notion of service level, as expressed for instance by (Dauzere-Peres et al. 2008), is used in this approach. The service level corresponds to the probability that a criterion is smaller (or larger) than or equal to a fixed value:

$$P(\text{Criterion} \leqslant \text{Fixed value}) \tag{2.5}$$

Providing the service level for a time constraint satisfaction problem is very relevant from a practical point of view. For example, if we identify a time constraint that can be satisfied by a given lot or not, it may be very valuable to know that, using different schedules, the probability of satisfying the time constraint is 80%. Thus, the service level for a given time constraint $TC^{\alpha}_{g,h}$ is

$$P(C^{O}_{\alpha,h} - C^{O}_{\alpha,g} \leq MaxTime^{\alpha}_{g,h}) \tag{2.6}$$

2.5.2 Empirical probability

Various schedules have to be considered to evaluate the probability for a given lot to be completed within its time constraints. The empirical probability determines the probability of respecting time constraints from experience and observation. The sample space is a set of feasible and realistic schedules. The event or criterion is completing a given lot within its time constraints. The empirical probability is the ratio of the number of times in which the specified event occurs to the total number of schedules. To create the sample space, various schedules have to be constructed. The list scheduling algorithm presented in section 2.4.2 is modified to construct different schedules. Determining a priority function which assigns a priority to the lots in the ready list is a critical step in a list scheduling algorithm. This function has an extreme impact on the success of the algorithm and so on the completion time of each lot (Wang et al. 2008). In the modified version of the list scheduling algorithm, we use a random function to determine the priority of ready lots. Lots within the ready list are randomly assigned to the available machines. Each time the list scheduling algorithm is run, a new schedule is computed. The sample size determines the number of runs of the list scheduling algorithm. The pseudo-code for the probability estimation approach is given in Algorithm 2.4

Notations

l_O	Incoming lot to the time constraint tunnel $TCT^{\alpha}_{m,n}$
N	Sample size
$Count^{\alpha}_{g,h}$	Number of times time constraint $TC^{\alpha}_{g,h} \in TCT^{\alpha}_{m,n}$ is satisfied

Algorithm 2.4. Probability Estimation approach

$n = N$
while n>0 **do**
    Apply list scheduling algorithm in Algorithm 2.3 with random priority function
    for all $TC^{\alpha}_{g,h} \in TCT^{\alpha}_{m,n}$ **do**
        if $C^{O}_{\alpha,h} - C^{O}_{\alpha,g} \leqslant MaxTime^{\alpha}_{g,h}$ **then**
            $Count^{\alpha}_{g,h} = Count^{\alpha}_{g,h} + 1$
        end if
    end for
    $n = n - 1$
end while

for all $TC^{\alpha}_{g,h} \in TCT^{\alpha}_{m,n}$ **do**
    $P(C^{O}_{\alpha,h} - C^{O}_{\alpha,g} \leqslant MaxTime^{\alpha}_{g,h}) = \frac{Count^{\alpha}_{g,h}}{N}$
end for

2.6 Additional Indicators

The proposed algorithm can not only determine the probability of satisfying time constraints, but it can also be adapted to extract relevant information to support decisions. The following indicators can be provided to help decision makers to understand the root causes of violations.

1. Deviation from time constraints

The probability estimation approach determines the probability of satisfying time constraints by evaluating various schedules (sample space). In each schedule, the completion times of an incoming lot within time constraints are estimated. A time constraint is satisfied if the incoming lot has a completion time lower than the time constraint duration. Once a time constraint is violated, it would be valuable to know how far the completion time is from the time constraint duration. As the completion times within a time constraint are computed by a set of schedules, a measure is required to quantify whether the completions times are close to the time constraint duration and whether the completion times are spread out over the sample. To this aim,

two popular measures as the *Average deviation* and the *Standard deviation* are presented. The formulas are as follows, where N equals the number of schedules (sample size), l_O is the incoming lot of type α to time constraint $TC_{g,h}$ with maximum duration $MaxTime^{\alpha}_{g,h}$ and the completion time of l_α within $TC_{g,h}$ is $C^O_{\alpha,h} - C^O_{\alpha,g}$:

- **Average deviation from time constraints**
  The average deviation is the average deviation from the duration of the time constraint.

$$AverageDeviation = \frac{\sum \left((C^O_{\alpha,h} - C^O_{\alpha,g}) - MaxTime^{\alpha}_{gh}\right)}{N} \quad (2.7)$$

- **Standard deviation from time constraints**
  As the term implies, in this context, the standard deviation is a standard amount of deviation from the duration of the time constraint. To calculate the standard deviation, first the variance is determined. This is done by subtracting each completion time estimated by a schedule from the time constraint duration and then squaring, summing and averaging the differences.

$$StandardDeviation(SD) = \sqrt{\frac{\sum \left((C^O_{\alpha,h} - C^O_{\alpha,g}) - MaxTime^{\alpha}_{gh}\right)^2}{N-1}} \quad (2.8)$$

  Without the standard deviation, we cannot compare two sets of schedules effectively. Suppose the probability of satisfying a time constraint by evaluating two sets of schedules is the same, but that dose not means that two sets are exactly the same. For example, assume that the maximum time of a time constraint is 10 hours. Two set of schedules S_1 and S_2 with 3 schedules in each set are conducted to compute the completion time within the time constraint. The obtained completion times with each set are respectively 9,11,12 and 10,40,80. The probability of satisfying the time constraint by considering both sets is the same ($\frac{1}{3}$) whereas

the first set has a very small standard deviation ($SD_1 = 1.3$) compared to the second set ($SD_2 = 44$).

2. Critical process steps

Process steps that could possibly cause the violation of time constraints. The waiting time of the incoming lot l_O in front of these steps is too long which causes a large completion time within a time constraint. Two reasons can explain this fact: First, there are many lots of the same type in the fab which must go through these steps and, second, these process steps share some machines with the process steps of other existing routes in the fab. The critical steps can be identified among the process steps located on the longest path within the disjunctive graph representation. Since our approach is able to compute the completion time of lot l_i of type α in each process step (see Algorithm 2.3); the waiting time of l_i in front of process step $PS^i_{\alpha,j}$, called $WaitingTime^i_{\alpha,j}$, is computed by:

$$WaitingTime^i_{\alpha,j} = C^i_{\alpha,j} - C^i_{\alpha,j-1} - P^k_{\alpha,j} \quad with \quad k \in M_{\alpha,j} \tag{2.9}$$

By adding this formula just after line number 6 in Algorithm 2.3, the waiting time of l_i in each process step can be calculated. Finally, the critical steps can be obtained by sorting the process step by their waiting times.

3. Bottleneck machines

Machines which cause long waiting times and so late completion times of process steps within a time constraint. In the production process, each process step can be performed by a set of machines. The bottleneck machines are those who perform critical process steps. Accordingly, the bottleneck machines can be identified through the process of identifying critical steps which has been described previously. This information allows us to improve time constraint satisfaction in two ways:

- **Minimizing the deviation of time constraints:** Machines can be decomposed into bottleneck machines and non-bottleneck machines. According to the significant impact of bottleneck machines on satisfying a time constraint, a bottleneck-based schedul-

ing approach can be proposed. In this algorithm, the bottleneck machines are first scheduled optimally with the objective of minimizing the deviation of time constraints. Note that a bottleneck machine may perform limited and non-limited lots of various type of products. By assigning a high priority for the limited lots in the bottleneck machines, waiting times and accordingly violation of time constraints will be reduced. The non-bottleneck machines are then scheduled under consideration of the solutions of the bottleneck machine schedule.

- **Satisfying time constraints:** By indicating critical process steps, bottleneck machines can be identified. A time constraint can be satisfied if the waiting times in front of bottleneck machines are reduced. Accordingly, setting up some new machines with the same characteristics of the bottleneck machines in order to perform critical process steps is a possible solution. Consider the incoming lot l_O of type α that is not able to satisfy time constraint $TC^{\alpha}_{g,h}$. Assume that $PS_{\alpha,j}$ with $g \leq j \leq h$ is the critical process step which cause the longest waiting time for l_O. $PS_{\alpha,j}$ can be performed on a set of machines $M_{\alpha,j}$ with size N. The following algorithm estimates the required number of machines to respect time constraint $TC^{\alpha}_{g,h}$ by the incoming lot l_O.

  Although this approach is theoretically promising to ensure that time constraints can be satisfied, it is difficult to apply in practice, because bottleneck machines are often the most expensive machines in the fab.

4. Evaluating multiple lots

The proposed approach evaluates one single lot l_O arriving in front of time constraint tunnel $TCT^{\alpha}_{m,n}$. Assume that a set of lots $L = \{l_1, ..., l_n\}$ arrive simultaneously in $TCT^{\alpha}_{m,n}$. In this case, the probability of satisfying each $TC_{g,h} \in TCT^{\alpha}_{m,n}$ by each $l_i \in L$ must be estimated. A nice feature of our algorithm, is that it can be easily extended by adding new nodes to the directed acyclic graph. Accordingly, to evaluate multiple lots in $TCT^{\alpha}_{m,n}$, the corresponding nodes must be added to the Directed Acyclic Graph. Since

our approach is able to compute the completion times in each node, the satisfaction of the time constraint can be investigated on a set of lots. The pseudo-code for the probability estimation approach with multiple incoming lots is given in Algorithm 2.5. This is an extended version of Algorithm 2.4.

Notations

$\mathcal{L}$	Set of lots $\{l_1, ..., l_n\}$ arriving to the time constraint tunnel $TCT^{\alpha}_{m,n}$
N	Sample size
$Count^{\alpha,i}_{g,h}$	Number of times time constraint $TC^{\alpha}_{g,h} \in TCT^{\alpha}_{m,n}$ is satisfied by lot l_i

Algorithm 2.5. Probability Estimation approach with multiple lots

for all $l_i \in \mathcal{L}$ **do**
    add corresponding nodes and arcs to the directed acyclic graph
end for
$n = N$
while n>0 **do**
    Apply list scheduling algorithm in Algorithm 2.3 with random priority function
    for all $l_i \in L$ **do**
        for all $TC^{\alpha}_{g,h} \in TCT^{\alpha}_{m,n}$ **do**
            if $C^{i}_{\alpha,h} - C^{i}_{\alpha,g} \leqslant MaxTime^{\alpha}_{g,h}$ **then**
                $Count^{\alpha,i}_{g,h} = Count^{\alpha,i}_{g,h} + 1$
            end if
        end for
    end for
    $n = n - 1$
end while

for all $l_i \in L$ **do**
    for all $TC^{\alpha}_{g,h} \in TCT^{\alpha}_{m,n}$ **do**
        $P(C^{i}_{\alpha,h} - C^{i}_{\alpha,g} \leqslant MaxTime^{\alpha}_{g,h}) = \frac{Count^{\alpha,i}_{g,h}}{N}$
    end for
end for

This algorithm can be applied to approximate the maximum number of lots which can enter a time constraint.

2.7 Experimental Evaluation

The objective of this section is to evaluate the performance of the deterministic approach and probability estimation approach on industrial instances. These approaches were implemented in Java and compiled using Eclipse SDK in version 4.2.1. A prototype of these approaches has been implemented and tested on real fab data. Before a lot enters its time constraints, the proposed approaches are applied. For each instance, various files providing a snapshot of the fab are created. The basic data files are “Lots”, “Steps”,“TCTs”,and “Objective”. The file “Lots” includes the information of the lots that are already being processed in the fab such as ID and current process step. The processing information (route) of each type of product is provided in the file “Steps”. It contains the sequence of process steps which has to be performed for each lot, the set of machines in each process step and the corresponding process times. The actual state (up/down) of each machine is captured in this file. The file “Objective” has the information of the lot which is going to start the production. The route of the incoming lot is covered by multiple time constraints. The component of time constraints such as ID of first step and last step and maximum allowed waiting time are in the file “TCTs”. The graphical interface of the approaches is shown in Figure 2.7. It allows quick and simple access to the data sets by pressing the button “Insert input data files” and choice of the instance to evaluate. Once the program has been executed, a dialog box appears and an output file is created.

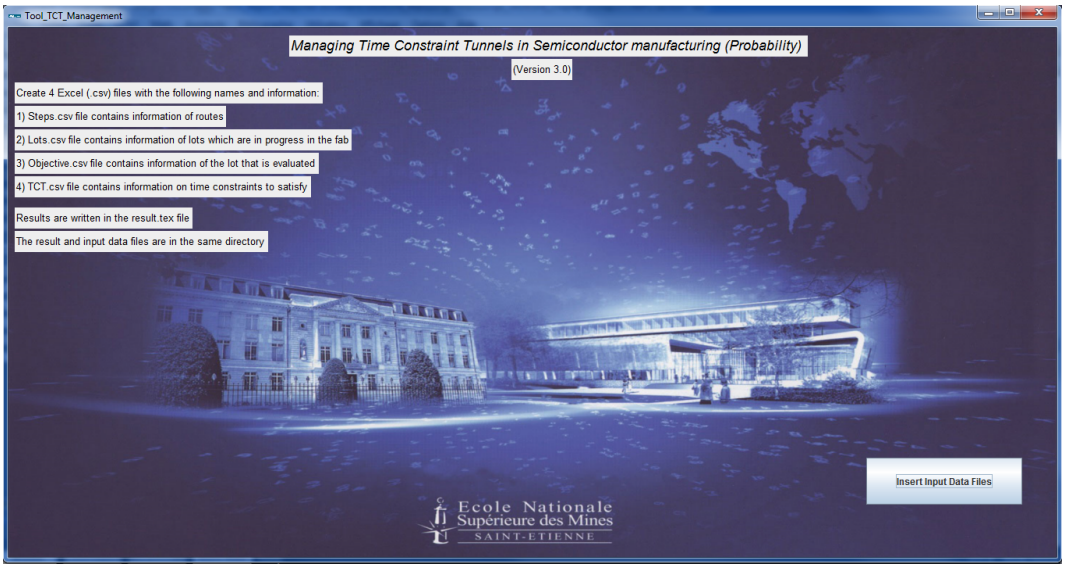

Figure 2.7: Graphical interface of tool

The first studied industrial instance includes 1,638 lots in process on 82,234 different process steps. There are 370 tools available to perform these steps. The route of the incoming lot includes 8 time constraints, each covering a different number of process steps. The principle is to evaluate these time constraints and to compare the results obtained with the deterministic approach and the probability estimation approach. In the deterministic approach, one schedule is computed and the given answer is either True or False. In the probabilistic approach, n schedules are computed and the result is a probability of satisfying time constraints. Table 2.1 summarizes the results obtained with these two approaches. The probability estimation approach is executed with different number of schedules (sample size) varying from 10 to 1000. The results revealed that, although some time constraints are not satisfied with only one schedule (Deterministic approach), increasing the number of generated schedules may lead to time constraints being sometimes satisfied, e.g. TC6 in Table 2.1 with a probability of 50% in 10 runs. The proposed approach can provide some extra information about the root cause of time constraint violations in the special cases when there is not even one machine to perform a process step covered by the time constraint, e.g. TC8 in Table 2.1. Assume that TC8 covers PS_i till PS_{i+14}, the result shows

there is at least one process step PS_j with $i \leq j \leq i+14$ where $M_j = \phi$. The processes needed in our approach to achieve these results are as follows: The actual state (up/down) of each machine is considered in our approach. If the actual state of machine m_k corresponds to PS_j is down, this machine will be removed from M_j. The procedure of computing the completion times of the process steps within time constraint $TC_{g,h}$ continues as long as the process step PS_j with $M_j = \phi$ can be seen. Since PS_j cannot be performed, evaluating the remaining process steps (PS_t with $j < t \leq h$) is not relevant. The procedure is interrupted and the result "No^*" is returned as the performance of $TC_{g,h}$.

Table 2.1: Performance of probability estimation approach on first industrial instance

TC	Nb.steps	1	10	50	100	150	200	250	300	500	750	1000
TC1	5	*No*	20%	34%	24%	17%	20%	20%	18%	19%	18%	21%
TC2	2	*Yes*	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
TC3	3	*No*	10%	18%	17%	10%	10%	16%	17%	18%	18%	16%
TC4	3	*Yes*	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
TC5	3	*Yes*	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
TC6	6	*No*	50%	30%	24%	34%	37%	36%	31%	31%	32%	32
TC7	3	*Yes*	100%	100%	100%	100%	100%	100	100%	100%	100%	100%
TC8	14	*No**	0	0	0	0	0	0	0	0	0	0
CPU(Min.)		0.50	0.58	1.18	1.92	2.63	3.35	4.14	4.96	7.74	11.39	15.27

* No tool up to production in this TC

Note that, after about 300 runs, the results of the probabilistic approach are stabilizing.

Table 2.2: Performance of probability estimation approach on second industrial instance

TC	Nb.steps	1	10	100	300	500	1000
TC1	5	*Yes*	50	51	55	56	52
Time (Min.)		0.4	0.4	1.5	3.8	6.2	12.0

Let us consider the second instance in which the route of the evaluated lot includes only one time constraint. In this route, there are 1,567 lots being processed in the fab and their routes include 78,786 process steps. The results of the probability estimation approach can be found in Table 2.2. In this case, TC1 is satisfied with one run but the probability of satisfying TC1 with more than 10 runs is about 50%.

These instances clearly show the relevance of the probability estimation approach compared to the deterministic approach in which only one run is performed. The numerical results also show that, even though instances of large sizes are considered, computational times are relatively small, even with a large number of runs. However, the convergence of the probability to satisfy time constraints with the number of runs is not clear. We still need to study what is the appropriate number of runs to balance between the quality of the results and acceptable computational times.

2.8 Conclusion and Perspectives

In this chapter, we proposed three approaches to manage time constraints. The first approach computes values of WIPMax which represents the maximum allowed number of lots within a TCT. A given lot can enter the TCT if the number of current lots waiting in TCT is lower than WIPMax. This approach does not take into account the position of lots inside the TCT, i.e. situations where all the lots are in the first step of the constraint and situations where all lots are in the last step are seen as identical. Given

the limitation of this approach, we proposed a deterministic approach which estimates whether a given lot can be completed within its TCT by considering the current status of the fab. This approach uses a disjunctive graph and a list scheduling algorithm. The limitation of this approach is that only one schedule is computed and therefore the result is a Yes/No answer. To consider the impact of different schedules on satisfying time constraints, we proposed an approach which estimates the probability of satisfying time constraints. This can be seen as an extended version of the deterministic approach. The probability of satisfying each time constraint is computed by running multiple times the list scheduling algorithm where lots are randomly selected. A prototype of this probabilistic approach has been implemented and tested on real fab data. The approach can help to support decisions in reasonable time on whether lots should enter time constraints or not, and thus to avoid reprocessing or scrapping lots. In addition, the approach can be extended to detect the root cause of time constraint violations.

To our knowledge, problems including overlapping and various types of products have never been investigated in previous works related to time constraints. Furthermore, because of large computational times, the approaches in the literature can only consider a limited number of process steps and machines. Our approach can handle problems of industrial sizes.

Estimating the right number of runs to balance between computational times and the quality of the results is a perspective of our research. Moreover, only 0 and 1 answers are used to evaluate time constraints satisfaction. Another potential and perhaps more relevant model is to consider boundary values in the evaluation (see Figure 2.8).

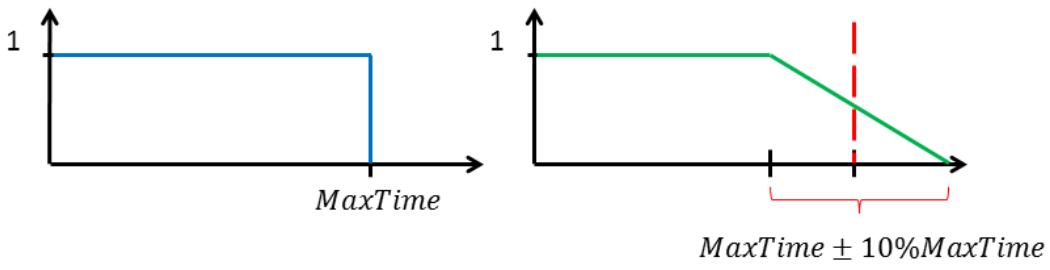

Figure 2.8: Time constraint evaluation models

In addition, analyzing and managing time constraints to support decisions to allow lots to enter time constraints based on the computed probabilities is one the main goals of our future research.

Chapter 3

A Multi-Method Generic Simulation Model for Semiconductor Manufacturing

Simulation is a powerful tool that provides a clear understanding of a system, predicts the future system behaviour, evaluates the impact of changes of parameters, analyses the operational performance and in general supports decision making in the context of manufacturing. However, developing an accurate, reliable and realistic simulation model is very costly and time consuming. In addition, most simulation models are designed for single-use applications which are not reusable. Our goal is to develop a data-driven generic simulation model for complex semiconductor facilities which is automatically generated from an external data source.

3.1 Introduction and Motivation

Semiconductor manufacturing is probably the most complex, advanced, and competitive manufacturing process. The underlying causes include multiple product types, constrained and large production process, re-entrant process flows, large number of expensive and sophisticated machines, and batch processing. In recent years, technical changes are frequently affecting semiconductor manufacturing processes. In order to stay competitive, semiconductor manufacturers have to quickly incorporate advanced technologies in products. Many different planning and scheduling problems have been raised to manage such dynamic, large and complex systems. The available solution approaches can be classified into two categories:

- *Mathematical or analytical approaches:* Represent the system with known mathematical concepts and language. The behavior of the system, relations and interactions between its components are described using mathematical formulas and constraints. Mixed Integer Linear Programming, Constraint Based Programming, or heuristics are applied to generate a solution that satisfies the specified constraints and also optimize target performance measures.

- *Simulation approaches:* Represent the system by a simulation model. The temporal behavior of the system is modeled by applying alternative methods for making short-term decisions. A number of scenarios are evaluated and compared to determine the suitable configuration of decisions depending on the situation and target performance measures.

Although mathematical approaches can provide optimal solutions, a high degree of abstraction and simplification from the real world system is often required. The stochastic elements and dynamic behaviour of a wafer fabrication facility (fab) cannot be described with formulas and strict mathematical relationships. In addition, in order to reduce the computational complexity of mathematical approaches and develop solvable models, considered problems are often of small size. On the contrary, simulation approaches are

capable to build a detailed model of the real system by incorporating realistic features and dramatically reduce computational times. Nevertheless, these approaches do not guarantee the optimality of solutions.

In recent years, the combination of both categories of approaches has attracted growing attention to achieve optimal solutions with high accuracy in less time (Nguyen et al. (2014) [55]). A comprehensive classification and survey of simulation optimization approaches is recently provided by Juan et al. (2015) [26]. The procedure of the most common simulation-based optimization approach is illustrated in Figure 3.1. It is based on an iterative process between simulation and optimization. The most important stage is formulating the optimization problem and designing the simulation model with regard to the objectives that should be achieved and the communication that has to be established between them. The optimization problem contains variables that influence the behaviour of the simulation model and the output of the simulation experiments determine some parameters of the optimization problem. The procedure begins by initializing the simulation model and performing a simulation run in order to get the parameters which are then transferred to the optimization. After solving the optimization problem, the values of decision variables are fed into the simulation model and then a new simulation experiment begins. This cycle is repeated until convergence of the solution.

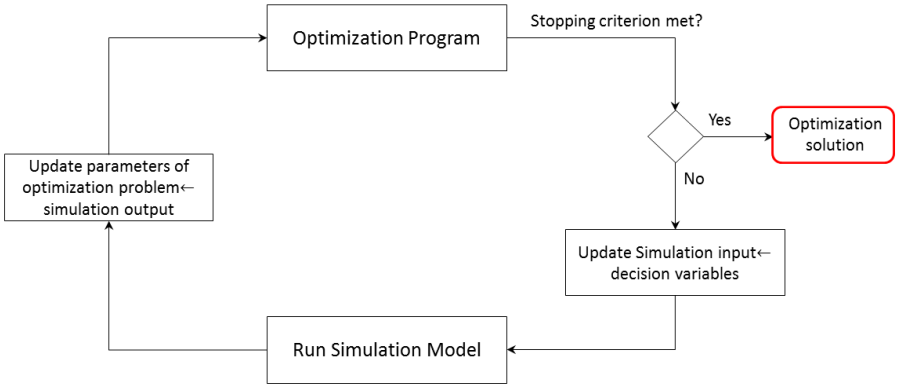

Figure 3.1: A simulation-based optimization approach procedure

The above-mentioned discussion of different approaches reveals the practical importance of simulation modelling in the management and optimization of production processes. In addition, simulation modelling enables the validation and verification of mathematical approaches before employing them in real systems since conducting experiments on a real system are costly and time consuming. Along with those mentioned above, simulation is an essential tool for designing a new facility or planning the expansion of an existing one to reduce risks and avoid unnecessary costs. It allows designers to evaluate different design alternatives even before the physical prototype is built. Simulation is a powerful tool that provides a clear understanding of the system, predicts the future system behaviour, evaluates the impact of changes on parameters, analyses the operational performance and in general supports decision making in the context of manufacturing. A comprehensive review of simulation with a particular focus on applications in manufacturing is provided in Negahban and Smith (2014) [54].

The quality of the decisions based on simulation depends on the quality of the model. The development process of an accurate, reliable and realistic simulation model is very costly and time consuming and usually should be done by an expert in simulation. Most simulation models are designed for single-use applications and investigate a restricted set of problems that are not reusable afterwards. In most cases, the model components such as machines are manually drawn by developers. Extending such models to large size problems requires internal changes of the source code that is usually impossible for the user. The manual procedure for building a simulation model of a large-scale system consisting of hundreds of equipment is time consuming and error prone. Moreover, rapid technological changes in advanced systems lead to more constraints and detailed elements in the production process which must quickly be incorporated into the existing simulation models. Because designing a simulation model from scratch requires considerable effort and time, a classification and detailed description of these challenges is provided by Fowler and rose (2004) [19] with the purpose of reducing time and effort for building simulation models as well as integrating simulations in the real world.

One way to cope with the challenges above is to use a data-driven generic simulation model. Pidd (1992) [60] defines a data-driven generic simulation model as "one which is designed to apply to a range of systems which have structural similarities." Such a model is generic to a target domain and is able to simulate different instances of systems in the domain without any change of the code. More precisely, a simulation model is automatically generated from an external data source using algorithms for creating the model and interfaces which allow users to easily interact with a simulation environment without being aware of the code. Data-driven generic simulation modeling is a promising approach for speeding up the simulation model building time, reducing the validation and verification time of the model and in addition decreasing the need for simulation experts.

Simulation has been applied for various applications in semiconductor manufacturing. Low et al. (2005) [46] and Lin and Chen (2015) [43] describe a simulation system for semiconductor assembly and test operations. Lin and Hung (2014) [44] propose a simulation based optimization approach for an automated material handling system in the photolithography zone. Simulation is extensively applied in scheduling and optimization problems for semiconductor manufacturing (Gupta and Sivakumar (2002) [22], Zhang et al.(2009) [90], Werner et al. (2006) [82]). There are only few papers which particularly address the development of simulation models for semiconductor manufacturing systems. For instance Lin and Long (2011) [42] develop a multi-agent-based distributed simulation platform for semiconductor manufacturing. The model is build using the Java Agent Development (JADE) framework.

The focus of this chapter is on the modeling and development of a data-driven generic simulation model for complex semiconductor manufacturing facilities. The main purpose of this simulation model is to evaluate the performance of the proposed approaches in the remainder of this thesis since conducting experiments on a real semiconductor facility is impossible. This model enables us to investigate and analyze manufacturing processes on a wide variety of instances for predicting the effect of changes, extracting relevant information and detecting critical problems and applying new objectives and strategies. In contrast to most simulation studies, we will specifically fo-

cus on the conceptual modelling process in the simulation study which is independent of the computer programming and the simulation software with which the simulation model will be developed. The conceptual modelling is concerned with how to structure the real system, how to decide which characteristics should be included, and how to specify the appropriate simulation paradigm (Discrete event, Agent based, System dynamic) to model the problem. Indeed, it is known as a bridge between stakeholders (customer, domain experts, and simulation developer) which facilitates communication and information exchange.

This chapter is structured according to the typical simulation modelling process which consists of conceptual modelling, model coding, experimentation, and implementation (see Figure 3.2).

1. *Conceptual modelling* is about abstracting a simulation model from the system that is being studied, identifying the components to be included in the model, describing the component behavior and relationships and determining the modeling approach. The outcome of this stage is a description of the model that can be developed in a computer. This stage in simulating a semiconductor manufacturing facility is explained in section 3.2.

2. *Model coding* is the process of developing a computer model for the conceptual model. It refers to the computer programming of transferring information from a conceptual model into a computer model using a programming language or simulation software. In this study, AnyLogic is chosen as the tool to implement a conceptual model of a semiconductor manufacturing facility. The process of implementation is described in section 3.3.

3. Once the development is completed, it is possible to conduct experiments on the simulation model according to the specified goals of the simulation study. The result of the experimentation process provides a solution for the real world problem and enhances the understanding of the system. This stage in our study is discussed in section 3.4.

4. Eventually, the obtained results will be established in the real world in various aspects such as implementing the solutions in practice or giving some guides to the users. This stage is considered as a future perspective of this study which is discussed in section 3.5.

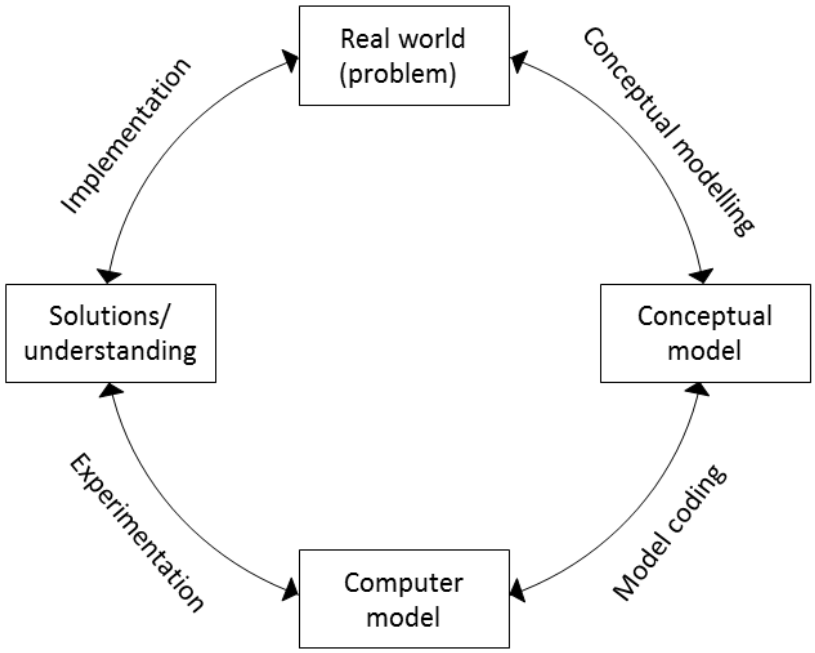

Figure 3.2: Key stages and processes in simulation studies (Robinson (2004) [66])

3.2 Conceptual Modelling

Conceptual modelling is the primary and most important stage in a simulation study. It refers to the process of abstracting a simulation model from the real world (Robinson (2006) [67]). The outcome of this stage is a conceptual model that represents the model components, structure and behavior. As illustrated in Figure 3.2, the conceptual model is required for the subsequent stages in the simulation study and affects their performances and quality. Constructing an applicable and reliable conceptual model is thus important.

The conceptual modelling framework will be used as a guideline for developing a simulation model for a semiconductor manufacturing system in this research. The framework that will be used in this study is adapted from the framework presented by Robinson [69] and the Hierarchical Control Conceptual Modelling (HCCM) framework presented by Furian [21]. In fact, the HCCM is an extended version of the framework of Robinson [69] with a focus on control policies such as dispatching as one of the main steps of the conceptual modelling framework. Both frameworks are presented for discrete event simulation where discrete event (DE) method is used to develop the model. However, agents based (AB) and system dynamic (SD) are two other methods which can be used to model a system. The process of selecting an appropriate simulation method through the conceptual modelling process is neglected in the existing frameworks. In addition, single type simulation methods may not be adequate to represent and design all features and behaviors of complex systems. Thus, the process of selecting a combination of multiple simulation methods must be placed in the framework. Note that the process of choosing a simulation modelling method or a combination of simulation modelling methods can be one of the main steps in the conceptual modelling framework, because the type of simulation method affects the design of the model structure and behavior. Figure 3.3 represents the framework used in this study. The steps of the framework will be briefly explained below.

In order to demonstrate the utility of the framework in practice, we also illustrate the application of each step in the modelling of a real semiconductor manufacturing system.

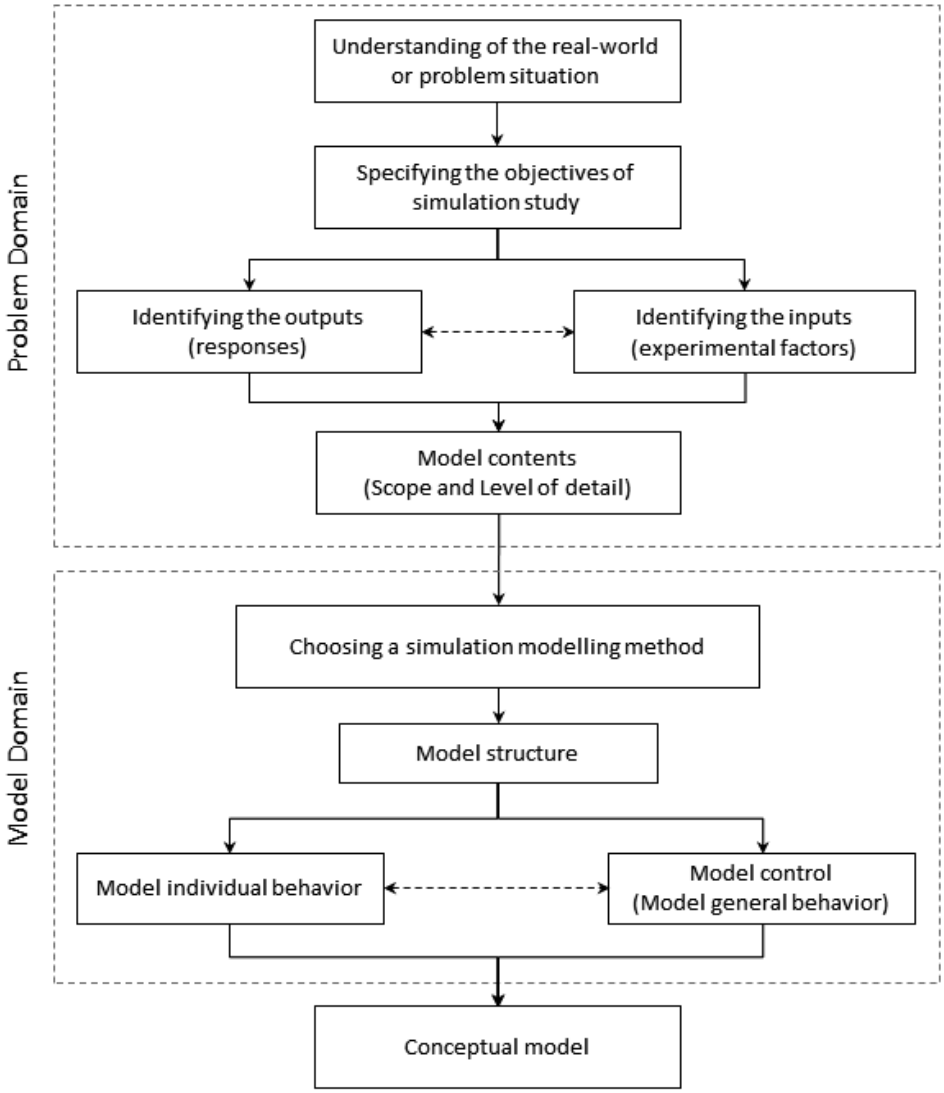

Figure 3.3: A framework for developing conceptual model (revised from Robinson et al. (2011) [69] and Furian et al. (2015) [21])

■ Understanding the real-world or problem situation

The first step in conceptual modeling is understanding the real-world or problem situation. In order to understand the real system that is the subject of the simulation study, understanding the problem context is essential. Problems are often defined as situations in which something has gone wrong or situations that can be improved based on the objective. The requirement for a simulation model should always be driven by the need to improve a

problem situation (Robinson et al. (2011) [69]).

Because of the complexity of the real-world or problem situation, investigation on understanding it is not an easy task. Most of the time problem situation is either not understood or expressed clearly; or we can find different points of view on the behavior of the system. Speaking with the right people and asking the right questions help this understanding (Robinson 2008 [68]).

The result of this step can be an informal, textual description of the problem situation. It also necessary to well document the assumptions that have been made in the gathering of the problem and included in the problem description (Furian et al. (2015) [21]).

■ **Specifying the objectives of the simulation study**

The objective of a simulation study could be different such as estimating the value of one or more characteristics of the system or identifying the best or at least acceptable system configuration (Maria et al. (1997) [49]). Robinson et al. (2011) [69] specified three component which could describe the objectives:

- **Achievement:** The ultimate goal and desire achievement, e.g. increased throughput, reduced cost, improved customer service, improved understanding of the system
- **Performance:** Performance measurement where applicable, e.g. increased production by 20%, reduced cost by $50,000$
- **Constraints:** Constraints exist in real world and conditions in which the clients (modeler) must work, e.g. budget, available space

Determining the general project objectives is also necessary. The nature of the model and its use will impact the conceptual model design. In order to clarify the nature of the model, Robinson et al. (2012) [**?**] proposed to consider some general project objectives: Flexibility, run speed, visual display, ease-of-use and model/component reuse.

■ **Identifying the outputs and inputs**

Identifying the outputs have two purposes (Robinson (2011) [69]):

- To verify whether the modelling objectives have been achieved,
- To identify the reasons why the objectives have not been achieved.

■ **Model contents**

This phase is about developing the model contents which includes identifying the components that must be considered in the model and specifying their interaction and relationships. The model content does not suggest how the components behave and the system is implemented. In other words, it only represents the static structure and composition of a particular system. However, it is designed in a way to be able to accept identified inputs and to provide required outputs.

■ **Choosing a simulation modelling method**

In general, a manufacturing system has a discrete event model. However, once the model grows in complexity, it becomes too complex to understand the model. In this case, developing the simulation model with a single type method may faces various challenges. This phase is about choosing an appropriate simulation method (e.g. DE, SD, AB) or a combination of methods according to the system and specified objectives of the simulation study.

■ **Model structure**

The entity structure defines the model structure. Similar to classes and instances within the object-oriented software paradigm, entity structures define the specification of entity instances. The modeler can determine their preferred way to illustrate entities and their flows (Furian et al. 2015 [21]).

■ **Model individual behavior and control**

There are several possibilities to capture sequences of behavior. Activity diagrams, state charts or sequence diagarams in UML and SysML, business process diagrams, and flow charts are just some examples that can help to model the sequence of behavior. For modeling this part of the system, it is necessary to identify what behavioral artifacts are included and in what detail they are represented in the model. This helps to define the model's scope and the level of detail included in the model (Furian et al. 2015 [21]).

3.3 Conceptual Modelling Representation Methodology

The conceptual model is known as a bridge between stakeholders such as customers, domain experts, and simulation developers to facilitate communication and information exchange. Thus, a standard representation of a conceptual model that can be understood by different stakeholders is essential. Onggo (2010) [58] classifies the methods for conceptual model representation into three categories:

- **Textual representation**: A written document (texts) to communicate information.
- **Pictorial representation**: A pictorial representation to communicate information through pictures. Diagrams (such as activity diagrams, process flow diagrams and etc.) are the most popular pictorial representation for conceptual modelling.
- **Multi-faceted representation**: As explained in section 3.2, a conceptual model consists of several steps. Hence, it is not possible to represent all the steps using a single diagram. The multi-faceted representation consists of a set of diagrams and textual representation, which are components of a conceptual model.

Richter and März (2000) [63] proposed the use of Unified Modelling Language (UML) for representing conceptual models. In this study, we use UML diagrams in order to represent different components of the conceptual model of a semiconductor system (readers not familiar with UML diagrams may refer to e.g. [20]). UML is one of the most popular modelling languages used in software engineering that is intended to provide a standard way to visualize the design of a system. UML diagrams can be divided into two main categories:

- **Structural diagrams**: Focus on the static aspect of the system. The four structural diagrams are Class diagram, Object diagram, Component diagram, and Deployment diagram.

- **Behavioral diagrams**: Focus on the dynamic aspect of the system. The five behavioral diagrams are Use case diagram, Sequence diagram, Collaboration diagram, Statechart diagram, Activity diagram.

In this study, an UML Class diagram has been selected for describing and visualizing the model content which refers to the static aspect of the conceptual model. To this end, an overview of essential notations of a class diagram is provided below. The model behavior which refers to the dynamic aspect of the conceptual model is described using behavioral diagrams. Because of the complexity of the problem, a single diagram is not sufficient to describe the dynamic behavior of the entire system so a layered approach which is a combination of behavioral diagrams is used in this study. Sequence diagram and Activity diagram are used in the layered approach. A brief overview on essential notations of these two diagrams is provided below.

In addition to UML diagrams, an objective diagram (Keeney (1992) [29]) is used to structure objectives of the simulation conceptual model. The essential components and structure of this diagram is presented through specifying objectives of our simulation study for a semiconductor manufacturing system in section 3.4 (see Figure 3.13).

Essential notations of class diagram

- **Class** A class represents a set of objects (or entities) that share the same attributes and operations. Classes in class diagram are represented by rectangles which contain the name of the class and optionally the name of attributes and possible operations. Figure 3.4 is an example of a simple class diagram. The rectangle is partitioned into two: The top partition contains the class name and the bottom partition shows the class attributes. In this example, a lot is considered as a component of the model and so the corresponding class is created. It represents that every lot has an id, type, priority, and release date which will only differ in the values.

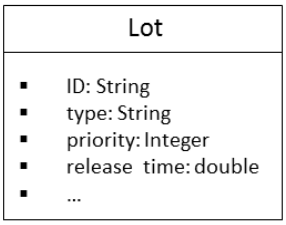

Figure 3.4: Simple class diagram with attributes

- **Relationships** A class diagram may contains multiple classes. The relationships between classes are shown with various lines and arrows. Figure 3.5 illustrates some of the common UML relationships which are used in our study. An *association* relation is depicted by a solid line on a class diagram. It is established between two classes, where their objects need to know about each other in order to perform their task. *Aggregation* is a relationship between two classes C and D, where C is part of D. It is shown by a line with empty diamond. A *composition* relation is a stronger form of aggregation which is indicated by a line with black diamond. A composition typically means objects of class E cannot exist independently of class F. And finally, an association that connects a class to itself is called *self association*.

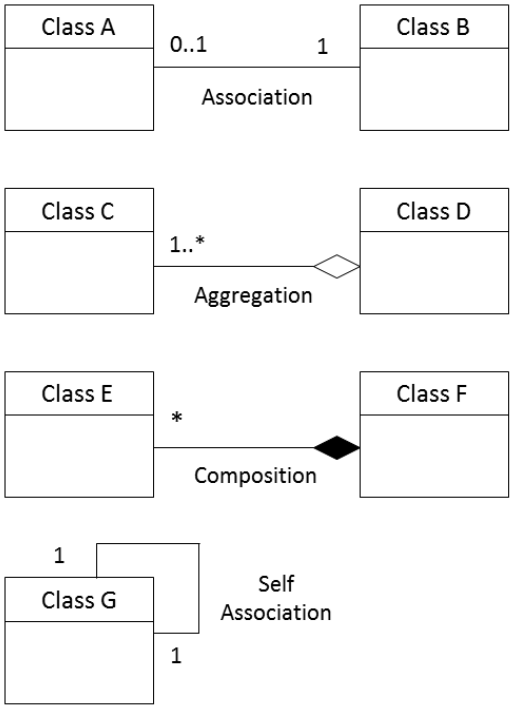

Figure 3.5: Relationships in UML class diagrams

- **Cardinality** Multiplicity limits the number of objects of a class that can be involved in a relationship. Figure 3.6 presents some typical examples of multiplicity that are used in this study. For example, in Figure 3.5, above, an object of class B is associated with no object or only one object of class A. An object of class D contains at least one object of class C. An object of class F may contain zero or more objects of class E. And an object of class G is associated with exactly one object of its class.

Cardinality and modality	UML symbol
One-to-one and mandatory	1
One-to-one and optional	0..1
One-to-many and mandatory	1..*
One-to-many and optional	*

Figure 3.6: Multiplicity syntax

Essential notations of sequence diagram

A Sequence Diagram is one of the behavior diagram in UML. It describes the agent's interaction by representing the sequence of messages exchanged between agents. The essential elements of the sequence diagram are presented in Figure 3.7.

- **Lifeline** Agents that are involved in interactions are shown as parallel vertical lines (life lines). A lifeline represents the sequence of events that occur in an agent during an interaction, while time flows down in the line. Rectangles are drawn on top of lifelines to represent the name of the agents.

- **Message** Communications between agents represented as messages drawn as horizontal arrows from the source agent to the target agent. The time sequence of message flows is shown by moving down on the lifeline. The message flow (or communication act (CA)) is a method call of an agent which triggers a communication. Note that the sequence diagram does not show how agents handle the message flows, meaning that it does not show which methods or activities are conducted.

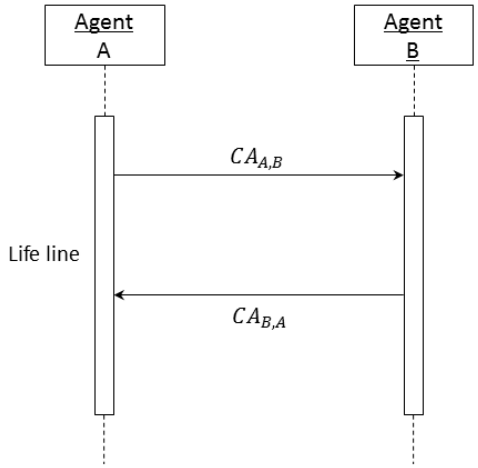

Figure 3.7: Syntax of Sequence diagram definitions

Extended UML diagram to express agent interactions

Odell et al. (2001) [57] discuss some possible extensions to UML that can be used to express interactions between agents. A recommended extension that supports concurrent threads of interaction is presented in Figure 3.8. This figure indicates that all threads $CA_1, CA_2, \ldots, CA_n$ can be sent concurrently (Figure 3.8 (a)). A decision box is used to decide which CAs must be sent. Figure 3.8 (b) indicates that zero or more CAs will be sent. Figure 3.8 (c) includes a decision box containing notation "x" which indicates that exactly one CA will be sent. One way of using these concurrent threads in the sequence diagram is illustrated in Figure 3.9.

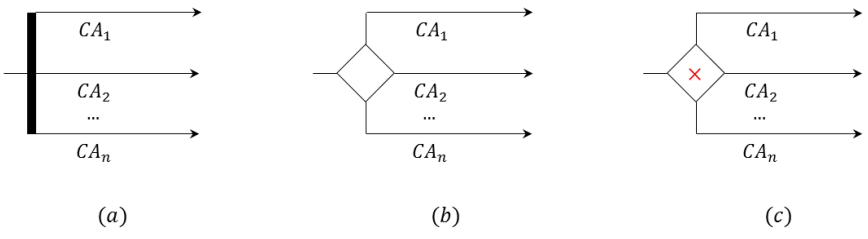

Figure 3.8: A recommended extension that supports concurrent threads of interactions ([57])

Figure 3.9 (a) indicates that all CAs (1) are sent concurrently from one

agent to another. To show that the receiver agent is capable of handling various CAs concurrently, multiple lifelines are drawn for it, one lifeline for each CA (2). Consider a situation in which each CA_i must be sent to a different agent. In this case, each lifeline represents an individual agent. Figure 3.9 (b) indicates that zero or more CAs will be sent to an agent (or various agents). A decision box (2) is set up on top of the lifelines of the receiver agent to specify which methods (or processes) should be activated to handle the received CAs. Once CAs have been sent to various agents, the decision box indicates which agents are receiving a CA. Finally, Figure 3.9 (C) indicates that exactly one CA will be sent to an agent (1) and exactly one agent is receiving this message (2).

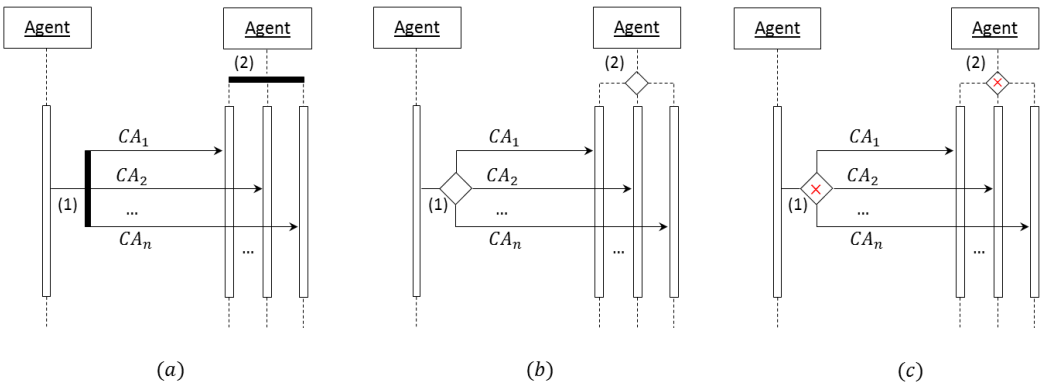

Figure 3.9: A technique that expresses concurrent threads of interactions between agents (Odell et al.(2001))

Essential notations of activity diagram

An Activity Diagram is one of the behavior diagram in UML. It captures the process flows in the system and describes the sequence and condition for workflow behavior. An activity diagram is basically a flowchart diagram while supporting parallel, concurrent, and branched flows. Fundamental elements of the activity diagram can be categorized into actions and control elements (initiation, end, decision, division, merge, etc.) (see Figure 3.10). In the following, we provide an overview on these elements.

- **Activity** Represents the sequence and conditions for coordinating workflow behaviors. It is denoted by a rectangle which encapsulates all the actions and control elements.

- **Action** An action is an individual step that must be executed in the overall activity. Its notation is a round cornered rectangle.

- **Transition** A transition shows the flow from one action to another. It is denoted by a line with an arrowhead.

- **Initial Node** An initial node is shown as a black spot that represents the start point of the flow when an activity is invoked. An activity may have more than one initial node such that each node triggers an individual action within the activity.

- **Final Node** A final node is shown as a circle with a dot inside. An activity may have more than one final node. Once a final node is reached, all flows in the entire activity are terminated.

- **Send Signal Action** An action that creates a message (or a signal) from its inputs, and sends it to another activity. The message can be a "request message" which sends the request for the required data or information to perform the reaming actions or can be a "provider message" which contains the requested data or information in another activity.

- **Receive Event Action** An action that waits for a message from another activity to occur.

- **Fork** A fork node splits current flow (a transition) into a set of concurrent or parallel flows of activities. It has one incoming flow and multiple outgoing flows.

- **Join** A join node combines a set of concurrent or parallel flows of actions into a single flow. It has multiple incoming flows and one outgoing flow.

- **Decision Node** A decision node is used where a flow may branch in alternative paths depending on a condition. A decision node has one input flow and two or more output flows with guard conditions. The input flow is processed along a path depending on the evaluation of the guard conditions.

- **Merge Node** A merge node combines multiple alternate flows. It has multiple incoming flows and a single outgoing flow. Contrary to join nodes, a merge node does not synchronize the incoming flows. Once an incoming flow is received by a merge node, the outgoing flow is executed without waiting for the arrival of other flows. Note that decision nodes and merge nodes have the same notation.

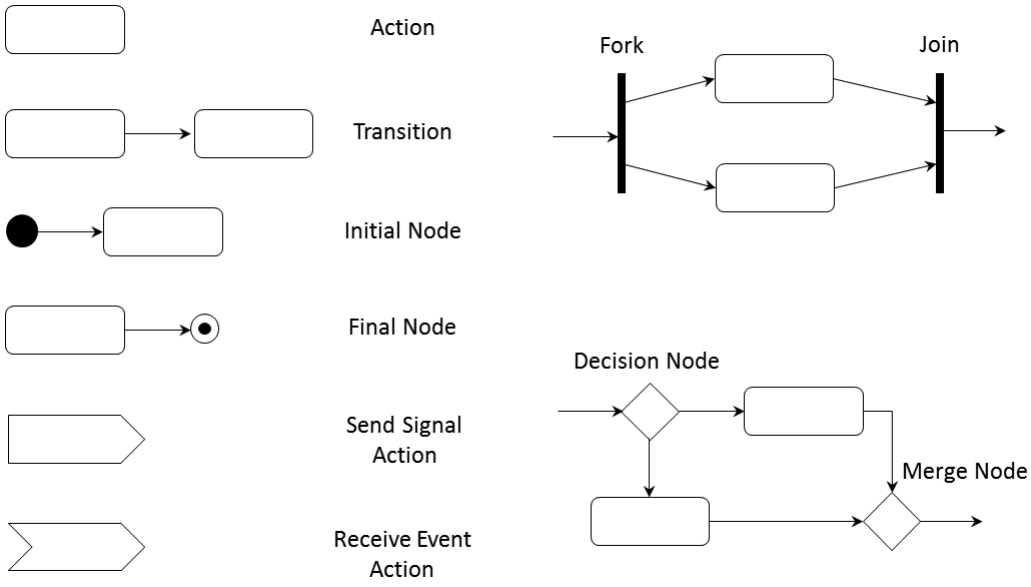

Figure 3.10: Syntax of Activity diagram definitions

A Layered approach

A single diagram is not sufficient to describe the dynamic aspect of an entire system. Odell et al. (2001) [57] introduce a layered approach to

capture the system as a whole in UML. We explain this approach through an example depicted in Figure 3.11.

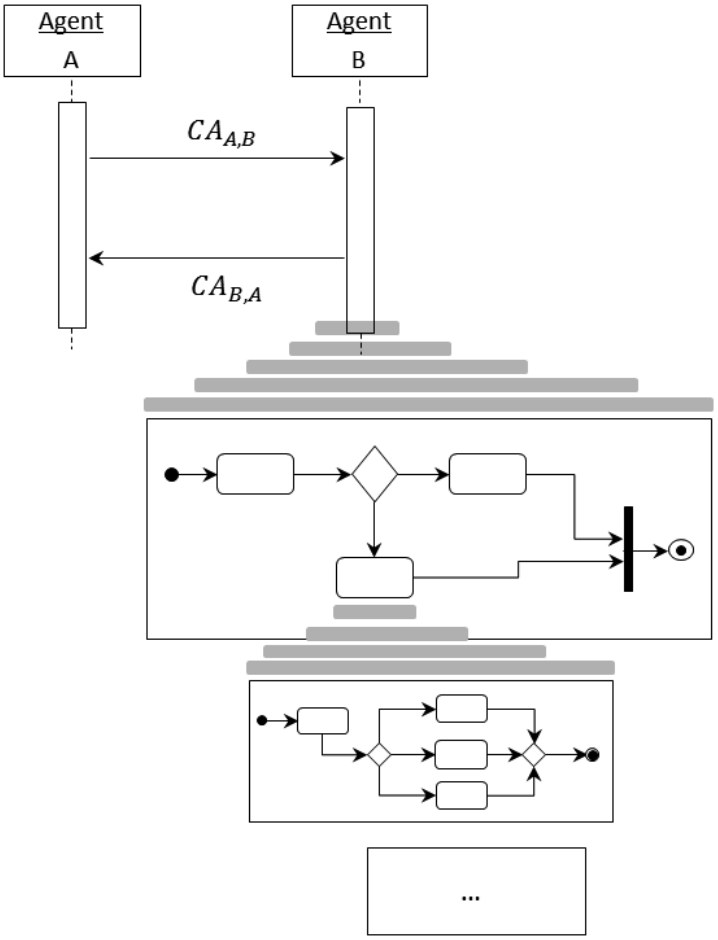

Figure 3.11: Capturing system behavior using a combination of behavioral diagrams (a layered approach)

The UML sequence diagram provides a high-level summary of system behavior (the top layer of the approach). Once the sequence diagram is invoked, Agent A generates the communication act (message) $CA_{A,B}$ and then sends it to Agent B. $CA_{A,B}$ can be a process request message. When $CA_{A,B}$ is delivered, Agent B prepares the communication act $CA_{B,A}$ as response and sends it to Agent A. The second layer describes the internal behavior of individual agents in generating a request or a response message. The internal

behavior of Agent B in generating the response message is shown using an activity diagram. Note that the execution of an individual step (action) of the activity diagram can be complex enough to express its process using a behavioral diagram. The layering process can continue until a proper model behavior has been designed to be used in the model coding stage.

3.4 Conceptual Model of a Semiconductor Manufacturing System

In order to demonstrate the utility of the framework in practice, we illustrate the application of each phase in the modelling of a real semiconductor manufacturing system. Due to the significance of the m odel behavior stage in the conceptual modelling process, it is discussed separately in the section 3.5.

3.4.1 Understanding the problem content

The typical components for semiconductor wafer fabrication systems can be categorized into physical structure, production planning, and production control. From a physical structure point of view, a semiconductor wafer fab contains a wide variety of products and a set of unrelated parallel tool groups. Each product type is associated with one processing flow (route) and each processing flow is defined as a sequence of hundreds of process steps (operations). In addition, the processing routes in a semiconductor fab are characterized as a re-entrant process flow, in which the lots repeatedly pass through a similar sequence of process steps. Each process step can be performed by a set of tool groups, the processing time depends on the tool group. Each tool group includes a list of tools which can perform the process step. A tool group has a storage buffer where lots of different types and processing times are waiting for service. In a semiconductor fab, a tool group can be a batch-processing tool, which can process several lots simultaneously as a batch. Figure 3.12 represents a re-entrant processing flow of a product which contains 7 process steps. Each step can be performed respectively by the following set of

tool groups: $\{(TG1), (TG2 and TG3), (TG4), (TG2), (TG1), (TG2), (TG4)\}$.

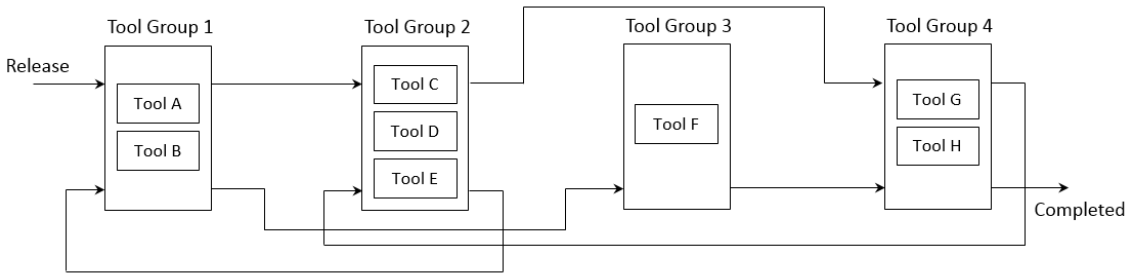

Figure 3.12: Reentrant production line in a semiconductor fab

Because of the limited production resources, batch-processing tools, reentrant process flows, process variability, etc., production planning and production control are required in order to meet the production goals and maximize the profit or minimize costs. Production planning includes capacity planning and order release strategies. Capacity planning strategies determine the quantities and the product mix which have to be performed and release strategies specify their release times in the fab. Production control which is usually based on scheduling and dispatching is then required. Scheduling algorithms focus on arranging limited resources over time, and dispatching rules determine which lot will be processed on each tool from a set of waiting lots. For batch-processing tools, batching rules decide which lots from the storage buffer should form a batch.

3.4.2 Specifying the objectives of the simulation study

The objective diagram is used to present the objectives of the simulation conceptual model in this study. An objective diagram classifies objectives into two categories: fundamental objectives and means objectives. The fundamental objectives are the final result of the simulation study that has to be achieved. They are organized within a tree showing hierarchical relationships and interconnections. To draw an objective diagram, a list of objectives has to be prepared and then ordered into sets of higher level and lower level objectives. The higher level objectives are general objectives of the simulation

study whose measurement can be obtained from the lower level objectives. Figure 3.13 shows an example of fundamental objectives from our case study. The higher level fundamental objectives can be classified into three categories:

- **Generic model**: A motivation of this study is to develop a simulation tool applicable and reusable for a wide range of problems in semiconductor manufacturing. The lower level objectives for constructing such a generic model are developing a simulation model which is flexible in number of resources (tool groups) and product types.

- **Graphical interface**: Developing a graphical and animated simulation model including 2-D and 3-D graphics is one of the higher level fundamental objective in this study. The goal is constructing the layout of the semiconductor fab in question containing hundreds of tool groups and showing lot movements among them.

- **Performance**: Evaluating the performance of the proposed approaches in the rest of this thesis is the main higher level fundamental objective. The performance can be categorized into resources (tool groups) performance and lot performance. The tool group performance refers to the measurement of the utilization of tool groups and so tool utilization and identification of bottleneck tool groups. The lot performance refers to identifying bottleneck process step, respecting time constraints and linearity constraints by lots, achieving their cycle time target.

Means objectives help to achieve fundamental objectives and are placed into the tree. Two examples of means objectives are shown in Figure 3.13. Managing waiting times in process steps helps to respect time constraints and linearity constraints. Selecting appropriate tool groups (a dispatching rule) helps to achieve lower level fundamental objectives corresponding to the lot performance.

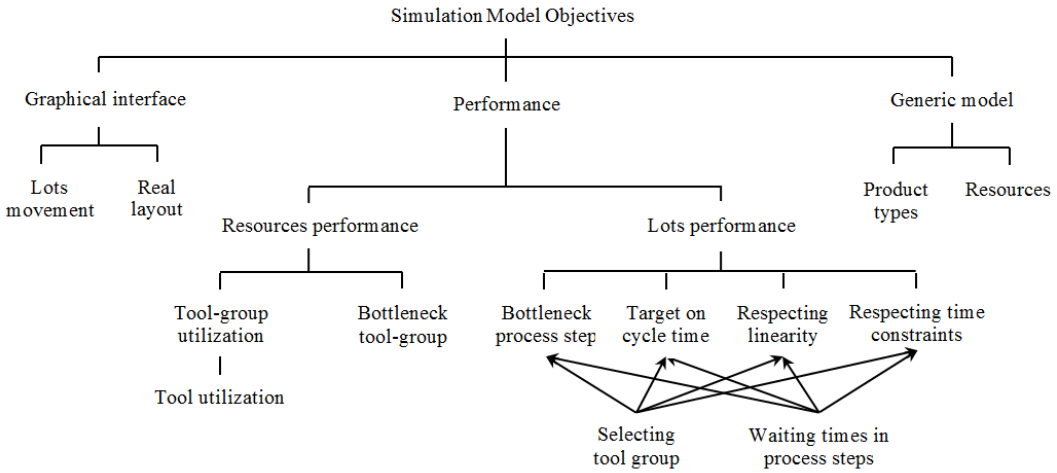

Figure 3.13: Objective diagram

3.4.3 Identifying the outputs and inputs

Here, we present the required data for modelling the global semiconductor manufacturing process. The basic data files are "Tool Groups", "Routes" and "Lots". The file "Lots" includes the information of the lots planned to be released and processed in the fab. Various types of products should be considered. The processing information of each type is provided in the file "Route". Finally, the file "Tool Groups" has the information related to the tools in the fab. The contents of each file are summarized in Figure 3.14.

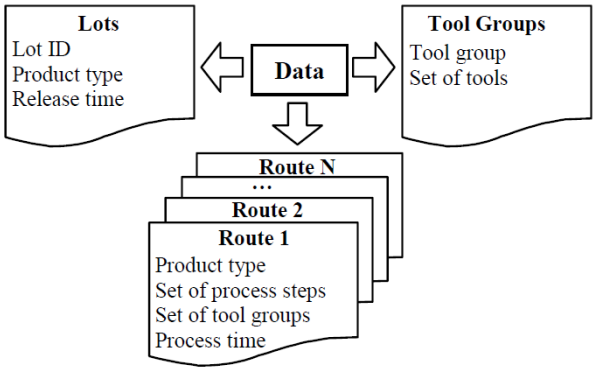

Figure 3.14: General data for simulation modelling

3.4.4 Model contents

The model content diagram in Figure 3.15 abstracts the real word objects used in the semiconductor manufacturing system and explains the relationships between them. A class is created for each physical component such as Lot, Tool and Fab, and for virtual components such as Route, Process step, Tool Group, etc.

On the diagram, a Fab is formed as a collection of one or more Tool Groups which can perform a set of Lots. A Tool Group contains at least one Tool. A Lot can be performed on various Tools. Thus, a Tool can have two different states: "Busy" while it is performing a Lot and otherwise "Idle". An enumeration is used to show these possible values for *Tool State* in the diagram.

A Lot can only be associated to one Product. However, multiple Lots are usually associated to the same Product. The Lot class inherits attribute *Product type* from the Product class. A Product is linked to a Route in order to pursue the right production flow in the Fab. A Route contains the production flow corresponding to a Product. It is composed of a set of Process steps which must be performed for a particular Lot. To specify the sequence of Process steps within a Route, a self-association relationship is defined for Process step. This relationship specifies which Process step should be performed at the later point in the process. A Process step is associated with one or more Tool Groups. However, a Tool Group can exist in the fab without performing a Lot and so its related Process steps.

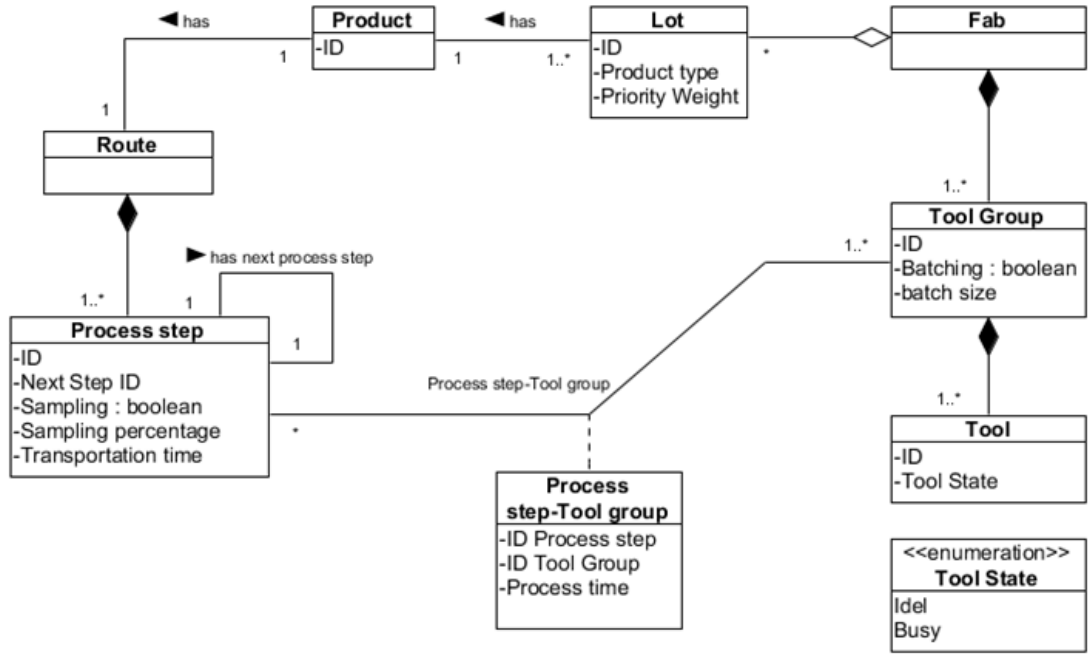

Figure 3.15: Model content diagram: Essential elements and their relations (UML class diagram)

3.4.5 Choosing a simulation modelling method

Here, we review three major simulation modelling methods: Discrete event (DE), System Dynamic (SD), and Agent Based (AB), to select the most suitable methods for modelling semiconductor manufacturing processes (see Figure 3.16).

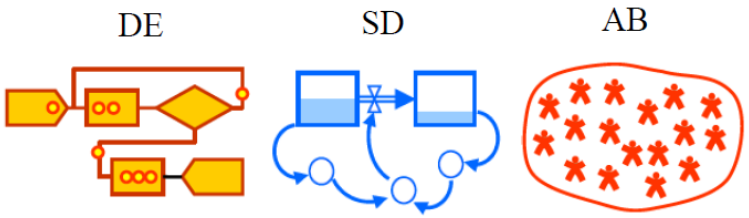

Figure 3.16: Simulation modelling methods (Borshchev (2013) [6])

1. DE models the system as a sequence of operations (or events)

which must be performed over entities. Operations include delay, service by various resources, choosing the process branch, etc. And an entity corresponds to an object or component which moves through the system. Hence, DE is an appropriate method for modelling manufacturing and queueing systems (Fishman (2013) [16]).

2. SD models the system as a network of stocks (stores of objects), flows and delays. In contrast to DE, SD focuses on flows of objects in the system rather than queuing systems (Maidstone (2012) [48]).

3. AB models the system as a population of individual objects (agents or active entities) which communicate with each other and follow a series of rules to achieve their specific objectives. The communication is ensured through a set of communication messages which include the agent sending message, the agent receiving message and the message contents. AB is suitable to model the system in a flexible and realistic way (Siegfried (2014) [75]).

At first glance, DE may seem an appropriate method for modelling semiconductor manufacturing processes as a sequence of operations. But since DE is based on predetermined static information, it is expensive and time consuming to develop large scale systems. In addition, DE can only build static structures of the system which will not be re-usable for the same problem in other facilities.

To overcome the limitations of DE, AB being more flexible, looks to be a suitable method for modelling large scale manufacturing systems. But the concept of queue which is the most important part of manufacturing systems is not defined in an AB method. Accordingly, a simulation method which is flexible such as AB and covers queueing concepts such as a DE method seems to be the most appropriate to model large scale complex semiconductor manufacturing processes.

Combining AB and DE methods can be done in two ways which are used in our study (see Figure 3.17):

(a) Process inside agents can be designed using a discrete event method, e.g. each element of a large scale system can be modelled as an agent such as raw material suppliers, manufacturer, and distributors in supply chain management. Since the service level in each part includes a sequence of queuing processes, the process inside each part is modelled using DE.

(b) In some systems modelled by DE, entities have different plans and strategies in using the system, thus they can be defined as agents which are active entities, e.g. some entities required to repeated processes in the system such as patients with chronic diseases which return to the hospital.

(a) Process inside agents (b) Agents become entities

Figure 3.17: Combination of discrete event simulation and agent based simulation (Borshchev (2013) [6])

3.4.6 Model structure

The model structure is about identifying entities, the entity structure, and their attributes. The entities and their attributes are already presented through abstracting the semiconductor manufacturing system in the model content diagram in Figure 3.15. In order to develop a generic and flexible simulation model for semiconductor manufacturing, a multi-method modelling approach using the combination of AB and DE methods for simulating semiconductor manufacturing is presented.

As already discussed, for each lot and depending on its type, a sequence of process steps must be performed. Each process step consists of a set of

tool groups in which one tool should be selected to perform the lot. Thus, tool groups can be defined as the main physical components in our simulation method. And a lot must be performed on a sequence of tool groups. This sequence involves thousands of tool groups which are not predetermined and depend on the behaviour of other lots in the system. Since in a DE simulation method, an entity must move through a predetermined path (sequence of tools), it would be impossible to use DE to create every possible path for a lot. Accordingly, in our simulation method, lots (entities) and tool groups (resources) are main agents (virtual components such as routes and process steps are also defined as agents). Thousands of tool group agents are created with different information in terms of type, number of tools, process time, etc. Lot agents have different types, routes, release times, etc., and are able to move through the system according to their predefined characteristics and real-time manufacturing information.

As already explained, lots with different types can be performed on some common tool groups. Also, a lot at various process steps can be performed on a specific tool group, thus the queuing concept must be defined for each tool group agent (see Figure 3.18). In this study, we use a DE method to describe the process inside the tool group agent where the processing times in the tool group agent differ according to the lot type.

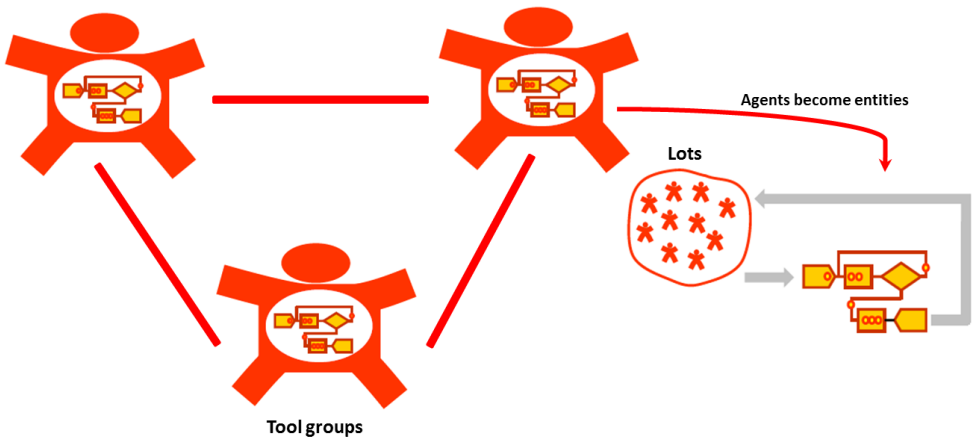

Figure 3.18: Physical structure of simulation model for semiconductor manufacturing

3.5 Model behavior

The proposed model for a semiconductor manufacturing system consists of multiple agents (Lot, Route, ProcessStep, ToolGroup). Every agent is given a set of rules according to which it interacts with other agents to complete the production process of lots in the fab; this interaction generates the overall system behavior. It becomes too complex to understand the behavior of the model using one diagram. Thus, a layered approach is used to represent the model behavior. A sequence diagram is used to display the sequence of communication between agents from when a lot is released into the fab until its production process is completed (first layer). Once an agent receives a message, a set of actions must be executed to provide the response message. The activity diagram is made to understand the flow of actions within each agent (second layer). The overall system behavior is presented in Figure 3.19.

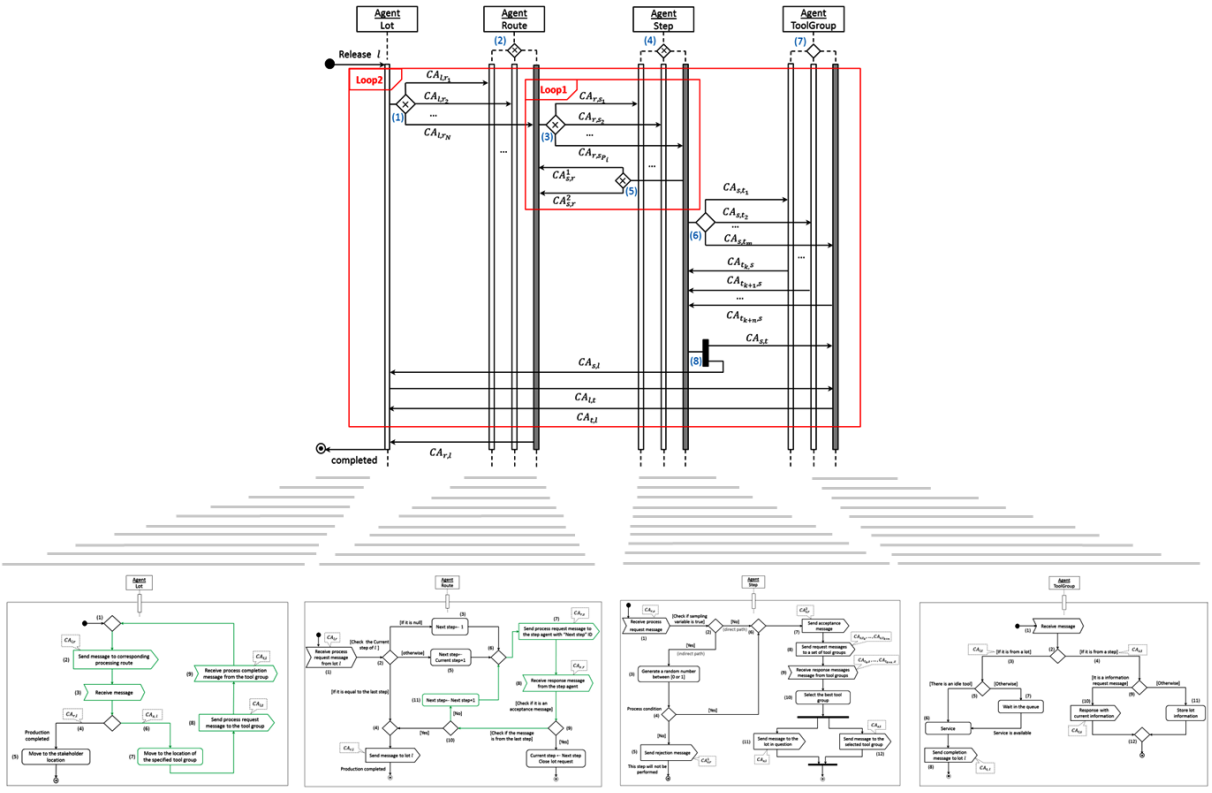

Figure 3.19: The overall system behavior

■ Communication between agents

The communication between agents in the simulation model is specified using a sequence diagram presented in Figure 3.20. The purpose of the sequence diagram is to visualize the interactions between agents in the system.

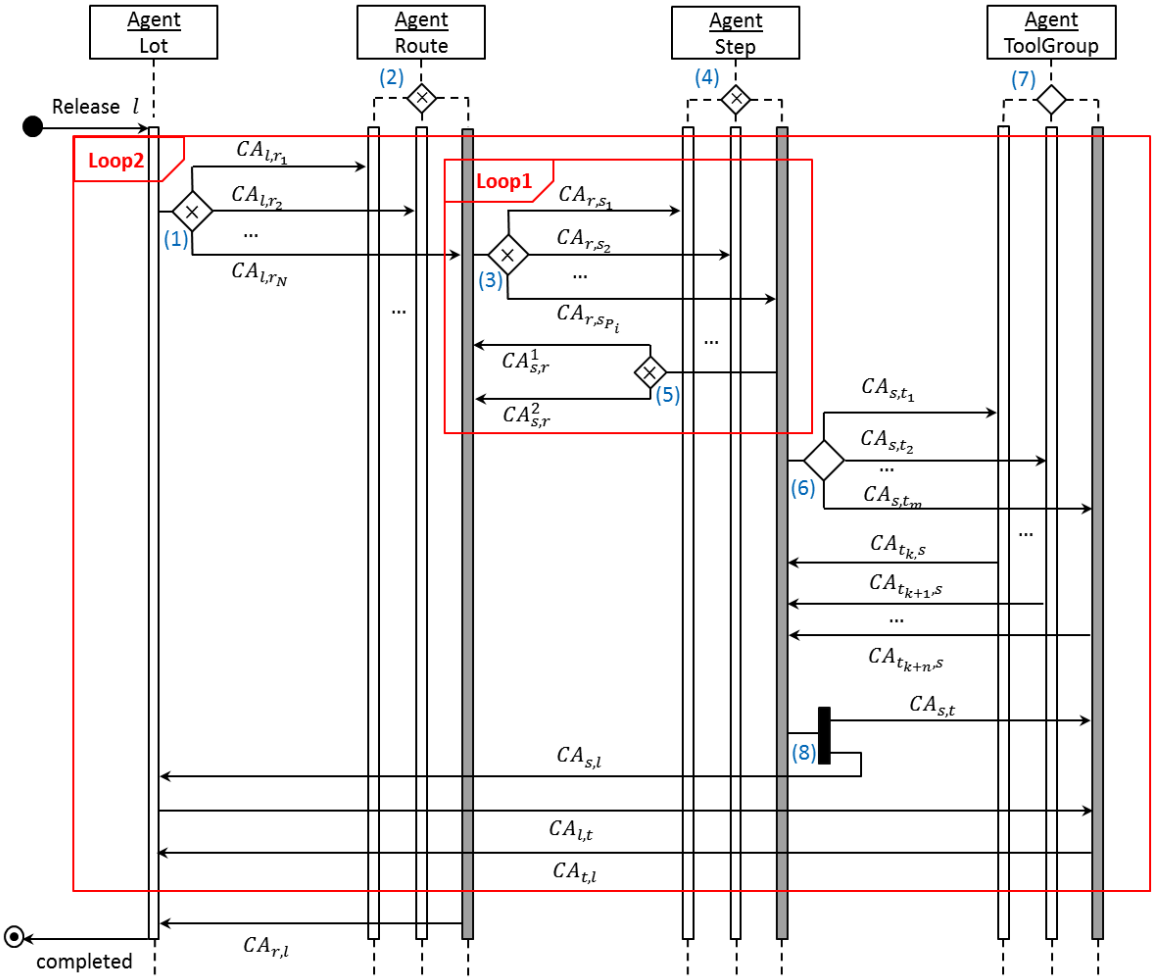

Figure 3.20: A sequence diagram that specifies the interactions between agents in the model

Assume that lot l is released into the fab, a *process request* must be sent to the related processing route agent among all the available route agents. The possible communication between lot l and route agents is displayed by a decision box (1). According to the product type of lot l, the decision box

decides which message will be sent. Assume that lot l is associated to route r. A process request message is transmitted to route agent r ($CA_{l,r}$). The activity diagram of route r is invoked by receiving this message to determine the proper process step which has to be performed for lot l. A decision box (2) is used to show all the possible communication acts between route agent r and step agents and indicates that only one step must be selected for communication. Assume that process step s is selected by a set of actions in the *route activity* diagram as the next step that must be performed for lot l (see Figure 3.22). A *process request* message must be sent to step s among all the step agents available in the model ($CA_{r,s}$). The activity diagram of step s is invoked by receiving this message to decide whether the process request can be accepted or not. A decision box displays these two possible response messages (3). The decision is made according to the sampling status of step s (see Figure 3.21). If the process request is rejected, a *rejection* message is transmitted to route r ($CA^1_{s,r}$). In this case, route r has to select the next process step and send a *process request* message. The interaction between route r and step agents will be continued until getting an *acceptance* message from a step. Loop 1 encloses these series of communications which are repeated. If the *process request* is accepted, an *acceptance* message is first transmitted to route r ($CA^2_{s,r}$) and then an *information* request message is transmitted to a set of tool groups which can perform step s of lot l (CA_{s,t_k},$CA_{s,t_{k+1}}$,...,$CA_{s,t_{k+n}}$). The decision box number (4) indicates that step s can communicate with multiple tool group agents. Step s asks for the current information of each tool group (e.g. length of waiting queue) to select the best tool group to process lot l. The activity diagram of a tool group is triggered by receiving this message (see Figure 3.22). After verifying the response messages from tool groups and then selecting tool group t as the best tool group, two concurrent messages are transmitted by step s, one to tool group t ($CA_{s,t}$) and another one to lot l ($CA_{s,l}$). Message $CA_{s,t}$ is transmitted to inform tool group t about the information of lot l which must be processed on t in the future. Message $CA_{s,l}$ informs lot l on tool group t and its location. Once lot l reaches the destination, a *process request* message is sent to tool group t ($CA_{l,t}$). The activity diagram of tool group t is invoked by getting this message to store lot l in the processing queue. When

the process of lot l is finished on tool group t, message $CA_{t,l}$ is transmitted to l to inform on the process completion. The production process of lot l must be continued on another process step. Thus, a *process request* message is sent again to route r and the same communication acts will be repeated to process lot l on a tool group. The communication between lot l and other agents will continue until its production process in the fab is completed. *Loop2* encloses these series of communications which are repeated. Once the production process is completed, message $CA_{r,l}$ is transmitted to inform lot l.

■ Internal behavior of a Lot agent

The internal behavior of a lot agent in our simulation model is specified using the activity diagram presented in Figure 3.21.

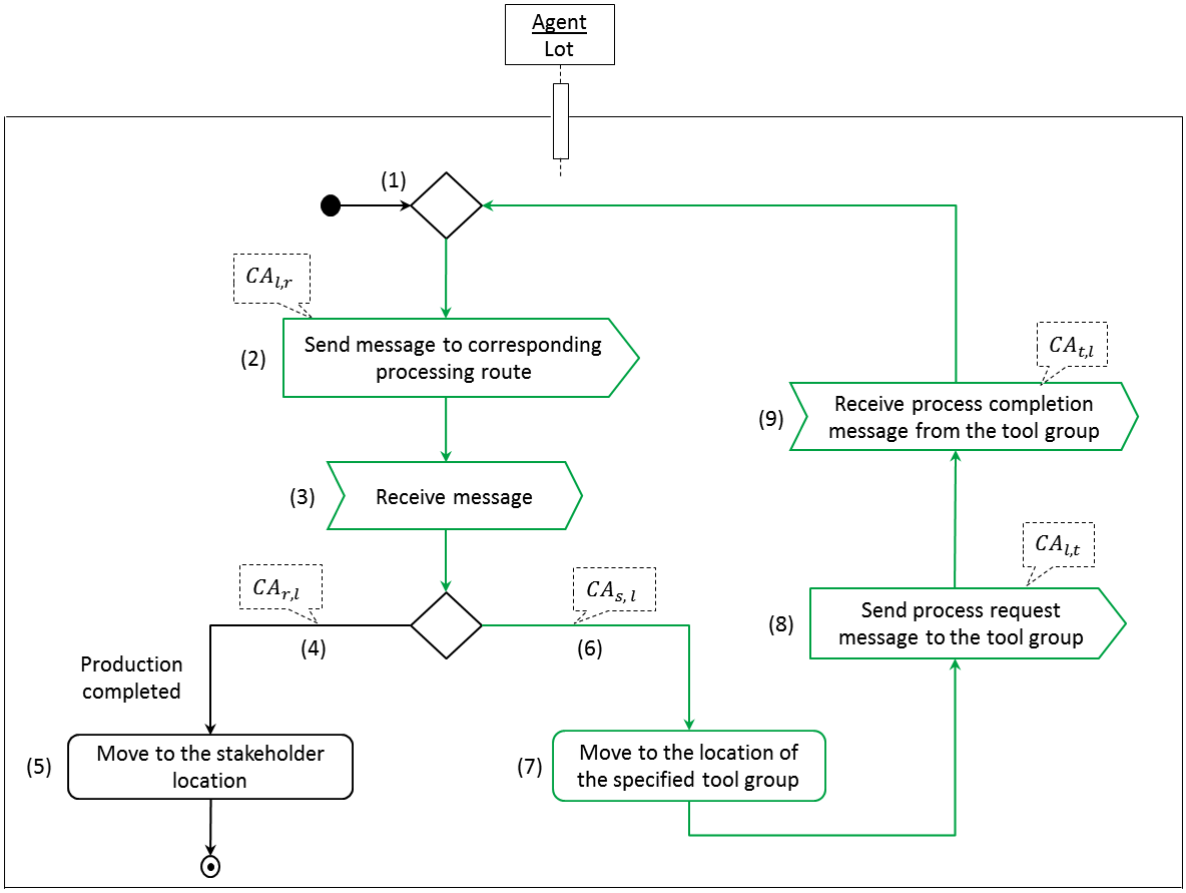

Figure 3.21: An activity diagram that specifies the internal behaviour of a Lot agent

The activity diagram of lot l is invoked when the time comes to release l into the fab (1). Assume that lot l is associated with route r. Route r contains a sequence of hundreds of process steps which must be performed for lot l. Thereby, when lot l is released into the fab, a *process request* message is transmitted to route r (2). This message can be replied by route r or process step s which is selected by route r and must be performed for lot l. Once a message is received (3), a decision node checks the sender of the message. A message from route r (4) notifies lot l on the completion of the production process and so l moves to the stakeholder location and the activity is terminated. Message from step s (5) contains the information of tool group t which will perform lot l in the next step. When lot l receives

this message, it starts to move to the location of tool group t. Once lot l reaches the location, a *process request* message is sent to tool group t. The *ToolGroup* activity is invoked by getting this message. This message is answered when the process of lot l is completed in tool group t (9). After receiving this message, the production process of lot l must be continued on another process step and so on another tool group. Thus a *process request* message has to be sent again to the corresponding route. A merge node is used to trigger this action as when lot l is released into the fab (1). The same process actions outlined in green are repeated until receiving the *production completion* message from route r. This repetition is specified by *Loop2* in the sequence diagram presented in Figure 3.20.

■ Internal behavior of a Route agent

The internal behavior of a route agent in our simulation model is specified using the activity diagram presented in Figure 3.22.

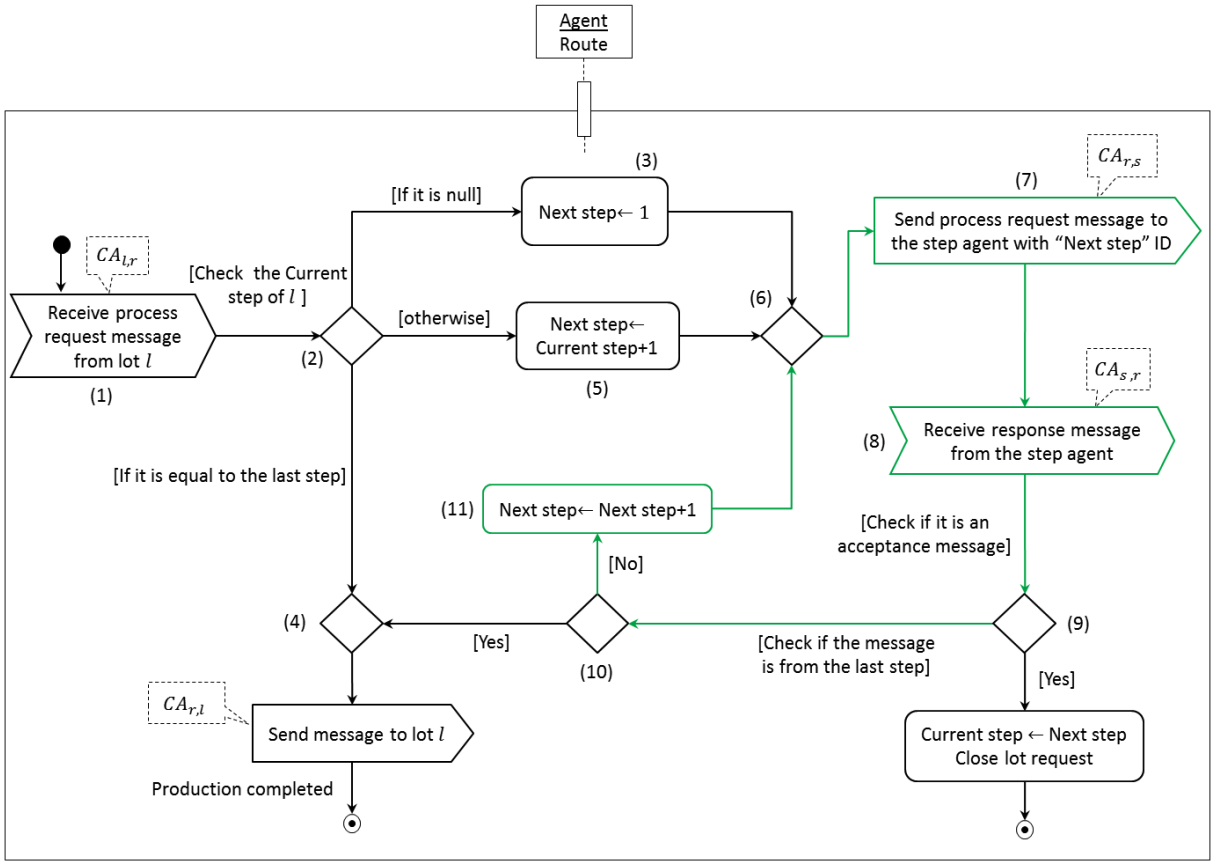

Figure 3.22: An activity diagram that specifies the internal behaviour of a Route agent

The activity diagram of route r will be invoked by receiving the process request message from a lot (these types of messages are shown with $CA_{l,r}$ at the Agent Interaction diagram in Figure 3.20). The request message contains the status of lot l of product type r which has to be processed in the fab. The status of lot l is characterized by the information on the process step which has recently been performed for l. This process step is called *Current step*, and has three different states for lot l in the fab. Lot l has right now been released into the fab if *Current step* is null. Its production process has been completed if *Current step* is equal to the index of the last process step corresponding to route r. Otherwise, l requires performing process steps to be completed. Thereby, once a process request message is received (1), the process method is decided conditionally. Based on the different possible

states of *Current step*, a decision node with one input flow and three outputs flows is used (2). Each output has a guard condition attached which expresses one state of *Current step*. Note that *Route activity* is executed to assign lot l to the proper process step. The index of the step which must be selected is specified by *Next step*. If *Current step* is null, the first process step of route r must be performed for lot l (3). If *Current step* is equal to the last process step, a message ($CA_{r,l}$) is sent to l to notify the completion of its processing(4). Otherwise, action (5) is executed to determine the next process step to be performed. The merge node (6) combines two decisions paths that were created by decision node (2).

The selected process step has to confirm that it can be processed for lot l (because of the sampling status). Thus, route r sends a *process request* message ($CA_{r,s}$), where s is the next in r (7). The activity diagram of s is invoked when receiving this message to answer the process request (see Figure 3.23). The response message (8) can be a rejection or an acceptance message according to the sampling status. If it is an acceptance message, lot l is assigned to s by modifying its status ($Currentstep \leftarrow Nextstep$) and the activity is terminated (9). If it is a rejection message, a decision node (10) checks the index of s. If it is the last process step of route r, a message specifying that the production process is completed is sent to l and the activity is terminated (4). Otherwise, the next step which has to be performed is selected (11). At merge node (6), this new decision path is combined with the paths that were created by a decision node (2) at the beginning and the actions outlined in green are executed until a final node is reached in the activity.

■ Internal behavior of a Step agent

The internal behavior of a step agent in our simulation model is specified using the activity diagram in Figure 3.23.

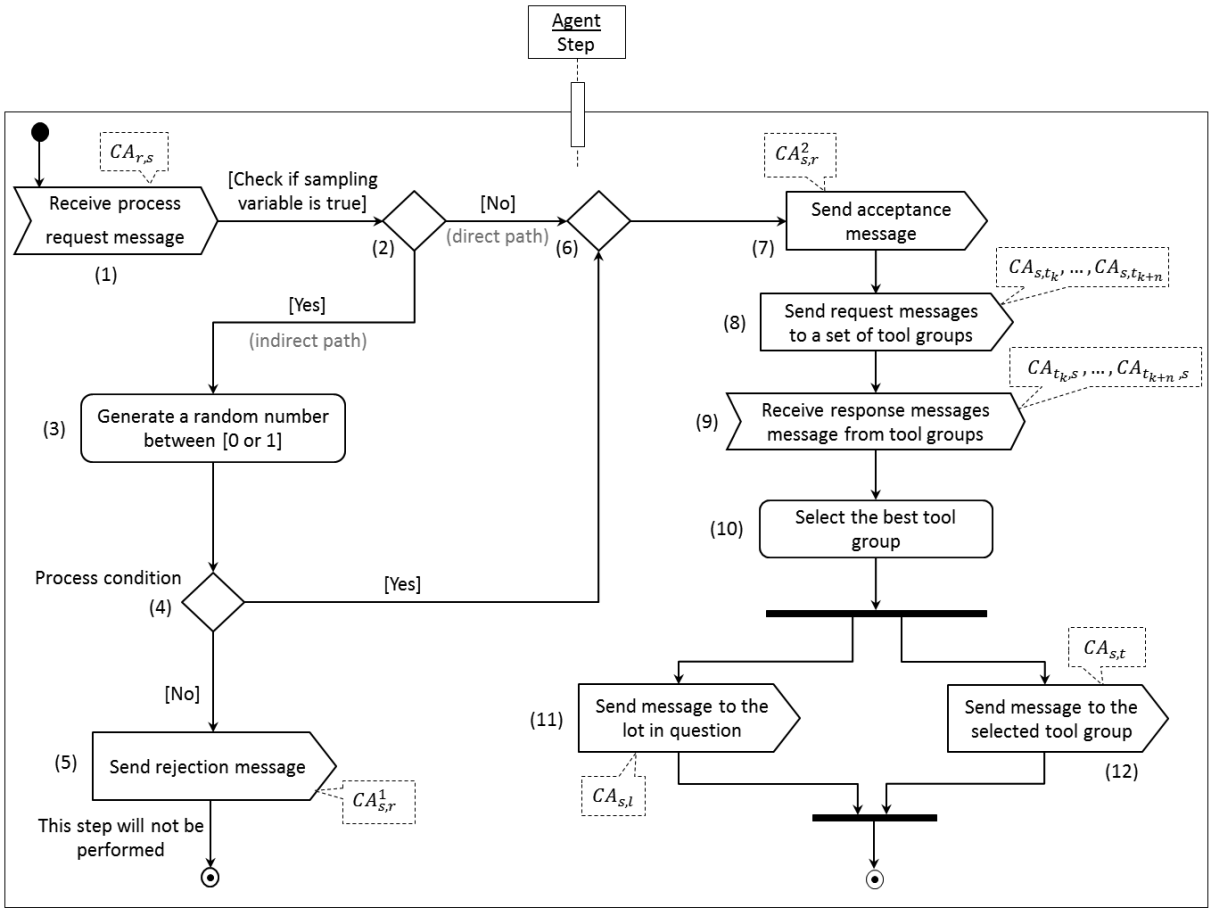

Figure 3.23: An activity diagram that specifies the internal behaviour of a Step agent

The step activity will be invoked by the *process request* message received from a route. The request message contains information of a lot to be processed in this step. Once the request message is received (1), the process is decided conditionally (2). Depending on the sampling status, a direct process path or an indirect process path is performed. The sampling status is specified by a boolean variable and a percentage. The boolean variable (True or False) indicates whether the process can be skipped or not. If the variable is True, the percentage indicates the probability of skipping the process. An indirect path is traversed if the sampling boolean variable is True (2). In this path, first a random number between [0, 1] is generated (3). Then, the process request condition check is performed to check if the generated

number is lower than the given percentage (4). If the condition is not met, the process actions are skipped and a rejection message is transmitted to the route agent (5) and the activity terminates. If the condition is met, merge node (6) brings back together two decisions paths that were created using a decision node (2). Once a flow (direct or indirect) reaches the merge node, it proceeds at the output. First an *acceptance* message is transmitted to the route agent (7). Recall that a process step can be performed by a set of tool groups. To select an appropriate tool group, their current status, which includes the length of their waiting queue, will be considered. To this end, a message which asks for this information is transmitted to each tool group (8). When response messages are received from tool groups (9), the tool group with the shortest waiting queue is selected (10).

Once the best tool is selected, the actions split into two concurrent actions. One action consists in updating the current step ID of the lot and sending a message to the lot (11). This message contains the information and location of the tool to which the lot should move. The other action consists in sending a message that contains the information of the lot to the selected tool group (12). Finally the concurrent actions combine to close the request for process.

■ Internal behavior of a ToolGroup agent

The internal behavior of a ToolGroup agent in our simulation model is specified using the activity diagram in Figure 3.24.

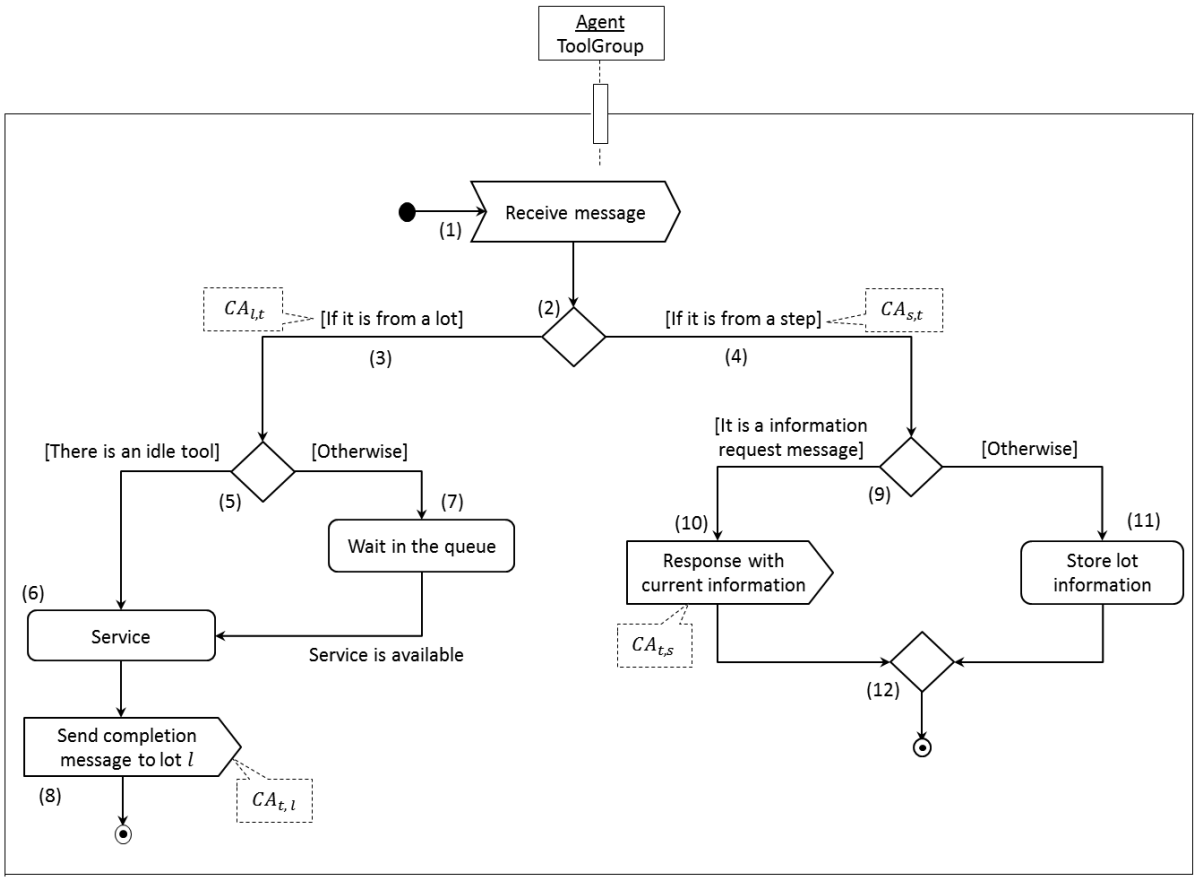

Figure 3.24: An activity diagram that specifies the internal behaviour of a Tool group agent

The activity diagram of tool group t is invoked by receiving a message (1). Depending on the sender of the message, process actions are divided into two different paths (3) and (4) by a decision node (2). If the message has been sent by a lot (e.g. lot l, $CA_{l,t}$), it is a *process request* message and tool group t must perform lot l on one of its tool (3). The process of lot l begins immediately if there is an idle tool (6), otherwise lot l waits in the *actual* waiting queue of tool group t until getting the authorization to start processing on a tool. Once processing of lot l is completed, a *completion message* is transmitted to l (8). This message informs lot l to continue the production process on another step. Note that a tool group has two different waiting queues: *Virtual* and *actual* waiting queues. The *virtual* waiting queue

stores information on two types of lots: Lots which will arrive in the future, and lots which are currently waiting for process. The *actual* waiting queue is a subset of the *virtual* waiting queue that only contains the lots that arrived at the location of the tool group. The *virtual* waiting queue is used to notify a tool group of lots which are on the way to its location.

If the message which invokes the activity has been sent by a process step (e.g. step s, $CA_{s,t}$) the process actions of path number (4) are executed. This message may be a *request message* which asks for the current information of tool group t or a *notification message* which informs tool group t about a lot that has to be processed in the future. If it is a *request message*, a response message which contains the length of the *virtual* waiting queue is transmitted to process step s (10). Otherwise, the information on the lot attached to the *notification message* is stored in the *virtual* waiting queue (11). The activity is terminated after executing one of these two process paths (12).

3.6 Model Coding

To develop the simulation model from the conceptual model presented in Sections 3.4 and 3.5, a multi-method modelling software which supports AB and DE simulation methods is required. Although there are several multi-method modelling software such as Repast/AnyLogic/Netlogo, AnyLogic is the only simulation software which combines DE, AB and SD simulation methods. Hence, in this study, AnyLogic is chosen as an appropriate tool to implement a simulation model of a semiconductor manufacturing facility. In the following we first provide a brief introduction to the basic structure of AnyLogic and some of its particular elements which are used in our study. We present the process of constructing a simulation model for a semiconductor manufacturing facility using AnyLogic.

3.6.1 An introduction to AnyLogic simulation tool

The first step is creating a new model (see Figure 3.25 (a)). It already has one agent called *Main*. Main is the top-level agent of the simulation model

which is the start point of execution. It contains the population of agents, environment agents (where the space and layout are defined), functions for initializing data, controlling variables, and collecting statistics. The next steps are to create agents with specific types (see Figure 3.25 (b)) and to define agents' components and behavior. In the project view, for each agent, choose New| Agent Type.

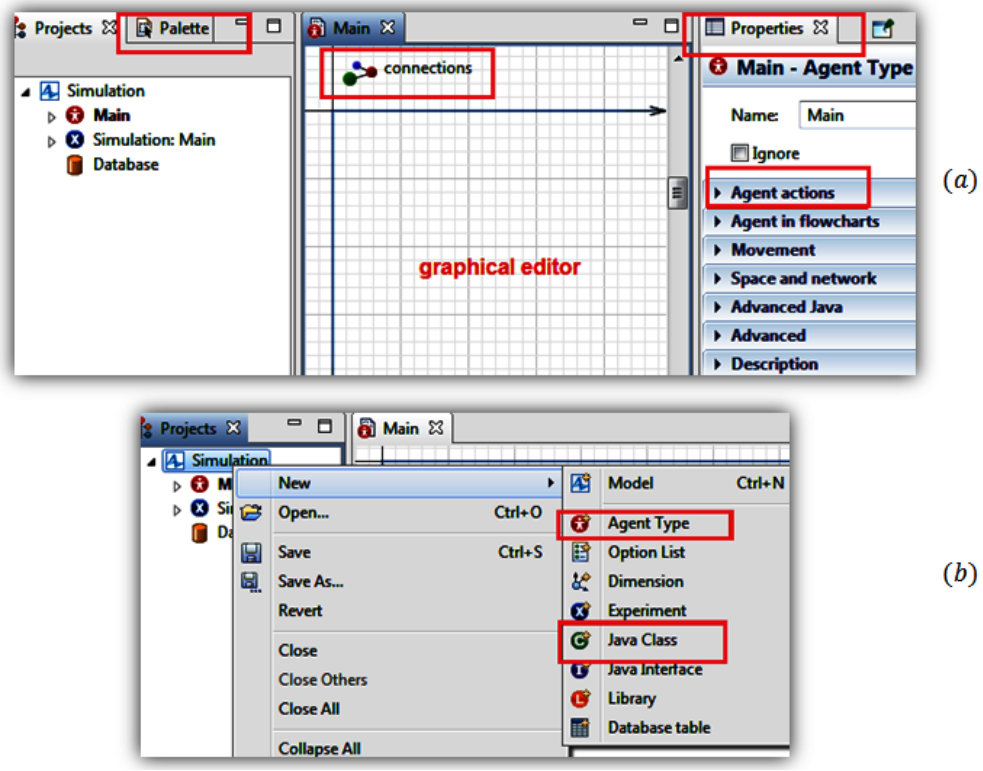

Figure 3.25: Creating a new model in AnyLogic simulation software

An agent type has a *graphical editor*, a place where the agent's internal structure elements such as components, behavior diagrams, graphical representation, etc. will be defined. AnyLogic software simplifies the process of modeling and drawing diagrams by providing the list of model elements grouped in palettes. Figure 3.26 represents a number of palettes that are used in this study. The *Agent* palette (Figure 3.26 (a)) contains elements for defining agents' characteristics (such as parameters, variables, collections, etc.) and their dynamic behavior (such as dynamic event). The *Presenta-*

tion palette (Figure 3.26 (b)) is used to define the visual representation of an agent. We can simply drag the required shapes from the Presentation palette view to the graphical editor of the agent. AnyLogic supports various ways of specifying the agent behaviors (actions) such as state charts, action charts, process flowcharts, DE diagrams, SF diagrams. We use action chart presented at *Actionchart* palette (Figure 3.26 (c))) and *Dynamic event* within the *Agent* palette to model the behavior of agents. Action charts visualize the algorithms which must be performed as agents' behaviors. Each block of the Action chart has a place to write Java code, which is helpful for those who are not familiar with the syntax of Java and makes it easier for other users to understand the implemented algorithm without going through the code. Dynamic events schedule any concurrent and independent events defined as agents' behaviors. We create an instance of a dynamic event by calling the method

$$create\_<DynamicEventName>$$

This method has two or more parameters. The first parameter is a timeout which specifies the time that an action occurs from the current model time. And the second parameter specifies the timeout unit. We can define other parameters which are required to perform the action.

$$create\_\ DynamicEventName(timeout,\ unit,\ \ldots,\ \ldots,\ldots)$$

(a) (b) (c) (d)

Figure 3.26: Some palettes in AnyLogic

Figure 3.27 (a) represents various sections of the *Properties* view of an agent which is set up to view and modify its properties. For example, the *Agent actions* section represents the behavior of the agent in different situations. For agent's actions in each situation, a box is set up where we can type Java code (e.g. adding the calls for the functions or activating an action chart). For example the code within *On arrival to target location* box executes when the agent arrives at the target destination.

AnyLogic supports agent communication (e.g. message passing). An agent can send a message to an individual agent or a group of agents. The content of the message can be an object of any type, including a text string, a value, or a structure with multiple fields (in this case we need to define a java class with required parameters as specified in Figure 3.25 (b). To send a message, multiple methods are provided in the software. The function below is used in our model:

Send (Object msg, Agent dest)

The first argument specifies the message that is send, and the second argument specifies the agent recipient. Reacting on the received message can be done in various ways. We use the *connections* element of the agent available in its graphical editor. Figure 3.27 (b) represents the *Properties* view of a *Connections* element. *On message received* box is set up where we can type a reaction code. Once the message is received, the reaction code within the box is executed.

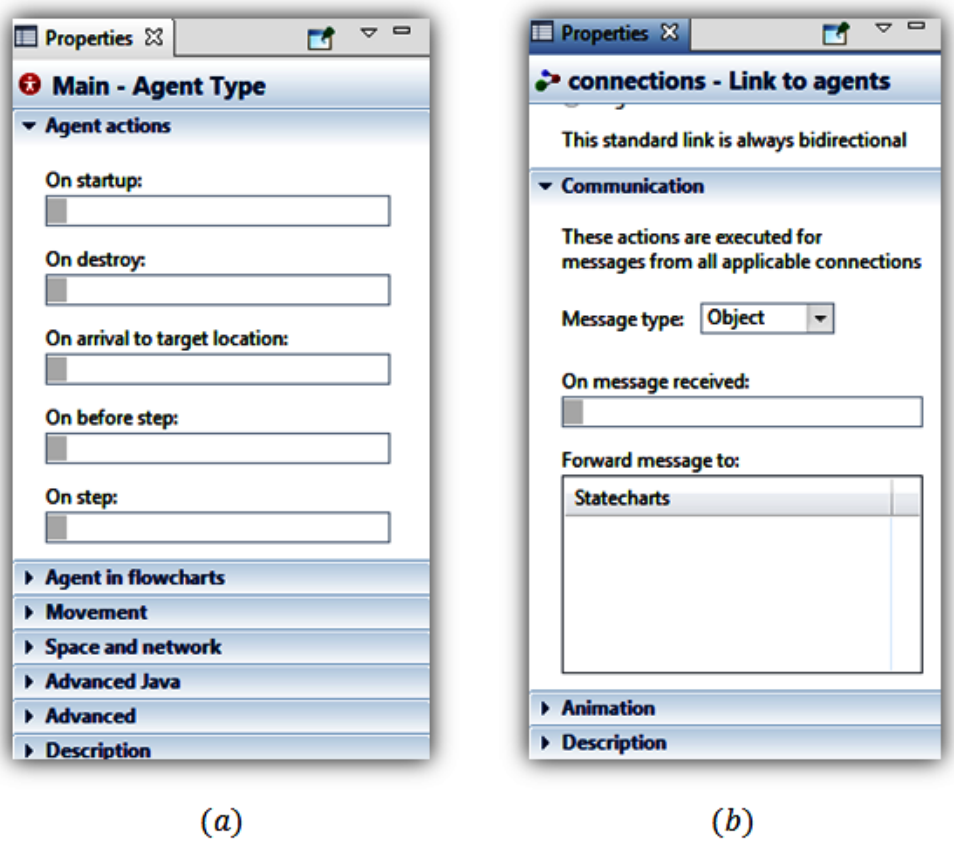

Figure 3.27: Agent *Properties* view (a) and connection *Properties* view (b)

After constructing the model (agents, connections, and their behavior, etc.), input data should be fed into the model by reading external files. The *Connectivity* palette (see Figure 3.26 (d)) has a set of tools enabling us to access external data. The *Excel File* and *Text File* connectivity tools are used in this study. Excel File provides access to Microsoft Excel files (.xsl, .xslx). It enables us to read and write to excel files using the specified API. Text File allows reading and writing to text files. According to the type of the external data file, we can simply drag an Excel File or Text File element from the Connectivity palette into the graphical editor of *Main* and then browse to add the source file. In this study, the Java I.O package has been used to process the input files and produce the output files.

3.6.2 Simulation model of a semiconductor manufacturing facility

Here we briefly describe how to construct a flexible simulation model of a semiconductor manufacturing facility using AnyLogic. First of all, as explained before, we create a new model. For each agent defined in the conceptual model (Lot, ToolGroup, Step and Route), a new Agent Type is created with the specified name. The next step is specifying the agents' characteristics. In the following, some of the most important characteristics of each agent are listed.

■ **Defining agents' characteristics**

- **Agent Lot** Create three different parameter types that will store the lot's ID (of type integer which is unique in the model), type (of type string and equal to the ID of a route) and priority value (of type integer). In the beginning, all lots have the same priority (the initial value is 300). For each parameter, drag the *Parameter* element from the Agent palette to the graphical editor of Lot agent type. As explained in the Model behavior section, a variable element *Current step* is required to store the index of the Step agent which is currently performed for the lot. Two variables *processing_time* and *arriving_time* are respectively defined to store the processing time and arriving time of lots at tool groups. So, drag three *Variable* elements from the *Agent* palette: One of type integer and two of type double.

  For visual representation of a lot, drag an oval shape from the *Presentation* palette to the editor of Lot. The color of the oval can be dynamically set up by each individual lot. Note that lots of the same type of product have the same color.

- **Agent ToolGroup** Create a parameter that stores the tool group's ID. Use the *Agent* palette to add tree *Collection* elements, name them *actual_waiting_queue*, *virtual_waiting_queue* and *list_Tools*. *actual_ waiting_queue* will store lots arriving at the location of

the tool group (set the collection class to ArrayList and set the Elements class to Lot). *virtual_waiting_queue* stores processing times of lots which will be processed in the tool group (set the collection to HashMap<Key k, Value v> where k is of type integer that represents lots ID, and v is of type double that represents lots processing time). Recall that a tool group has a set of tools with identical ID. As already explained, a Boolean variable must be defined for each tool to represent its availability. Since Anylogic enables us to create Java classes (see Figure 3.25 (b)), we create a java class called *Tool* with one parameter containing the tool's ID and a Boolean variable representing its availability. Then, the collection class of *list_Tools* is set to ArrayList and its elements class to Tool.

For visual representation of a tool group, drag a Rectangle shape and a Text shape from the presentation palette to the editor of ToolGroup. Each tool group will be visualized by rectangle shape that is labeled by a text containing the tool group's ID. The color of the rectangle as well as the content of the text can be dynamically set up by each individual tool group. As an example, the graphical editor of the ToolGroup agent type is presented in Figure 3.28.

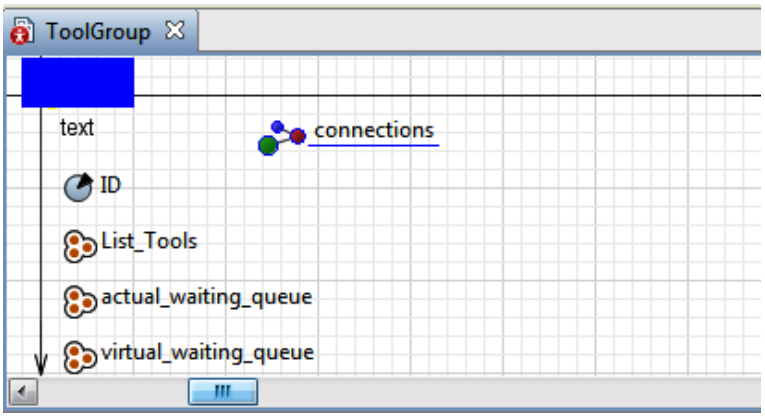

Figure 3.28: Graphical editor of ToolGroup agent type

- **Agent Step** Create a parameter of type string that stores the

step's ID. To store the sampling status of the step, create two different parameter types. The first parameter of type Boolean indicates whether the step agent has sampling status. The second parameter of type Double contains the sampling percentage. Drag all the parameters from the *Agent* palette into the graphical editor of the Step agent type. Recall that each process step can be performed by a set of tool groups and that the processing time depends on the tool group. To store this information, use a *Collection* element. Set the collection class to HashMap. Set the key elements class to String that represents the tool group ID and set the value elements class to Double that represents the processing time corresponding to tool groups.

- **Agent Route** Create a parameter that stores the route's ID. As already explained, a route contains a sequence of process steps. To create a population of Step agents inside a route agent, drag the agent Step from the project tree to the graphical editor of Route. To have multiple steps inside Route, select *Population of agents* in the properties of the dragged Step agent.

■ **Creating the structure of the model**

Once all the agent types and their characteristics are defined, we must create population of agent types in the model. A semiconductor fab contains a wide variety of products and a set of unrelated parallel tool groups. In our model, the Main agent acts as the fab. Thus, a population of tool group agents must be placed in Main. Drag the agent ToolGroup from the project tree to the graphical editor of Main. This way specifies that an instance of ToolGroup is embedded in Main. The icon and the presentation of tool group appear in the editor. To have multiple tool groups inside Main (agent of the same type), select *Population of agents* in the *Properties* view of the dragged ToolGroup agent and change the name to *TGs*. To consider production of various types of products in the model, create a population of Route agents (called *routes*) representing different processing flows and a population of Lot agents (called *lots*) Note that the information corresponding to Tool groups, various processing flows and product types are read from Excel files.

To visualize the layout of the fab, each tool group agent must be located in a specific area. AnyLogic provides a method to set the coordinates of the agent location *setXY(double x, double y)*
By using this method and the visual representation of tool group agents (a Rectangle shape and a Text shape), the layout of the fab will be created. Lot agents with oval shape will move among Tool group agent locations.

■ **Releasing Lot agents in the model**
After creating the structure of the model containing a population of Tool group agents and a population of processing route agents with specific characteristics, we have to define an approach for releasing Lot agents in the model at specified times. In contrast to other agent types that have been created at the beginning, a lot agent will be created when its release time comes.

First of all, two extra agents *Raw material Storehouse* and *Lots Storehouse* are created. The Raw material storehouse (RMS) is responsible for generating Lot agents of different types at various times. The Lots Storehouse (LS) agent is designed to keep the lots whose production process is completed (i.e. LS is the final destination of lots).

Drag a collection called *Lots_information* in the RMS graphical editor to store the information of lots which have to be released into the model during the simulation run (the process of feeding input data into the RMS agent will be explained later). Each element contains information corresponding to one type of product such as the product type, the total number of lots to be released, and the release rate. Since each Lot agent has an identical ID of type integer, drag a parameter of type Integer and name it *LastIndex* in order to keep the ID of the next lot to be released into the model. Its initial value is zero, thus the ID of the first released lot becomes zero and the second one is 1 and so on. Whenever a new Lot agent is created, the value of *LastIndex* will increase by one unit. Finally, the dynamic event *CreatingLots* is designed to schedule the release of lots:

CreatingLot(releaseRate, RouteID, MaxNblot)

The first argument is a timeout which specifies the time when a lot agent is released from the current model time. The second argument specifies the

type of the lot to be released and the last argument represents the number of lots of type *RouteID* that have to be released in the model in the future. These arguments are provided by the *Lots_information* collection. At the startup section of the RMS agent, we create an instance of the dynamic event *CreatingLot* for each type of product. Time counting begins when the dynamic event is created:

Create_CreatingLot<releaseRate, RouteID, MaxNblot>

When the timeout expires, AnyLogic executes the event's action and deletes this instance of the dynamic event. In the Action section, type Java code to create a new Lot agent of type RouteID with *LastIndex* ID. Then, update the value of *LastIndex* and *MaxNblot* respectively with one unit increase and with one unit decrease. To schedule the release time of the next Lot agent, in the event's action, create an instance of dynamic event with the same (and updated) arguments. In this way, whenever a lot agent is created, the release date of the next lot will be scheduled. The argument *MaxNblot* is setup as a stopping condition. An instance of dynamic event is created if MaxNblot>0.

■ **Specifying Agents interactions**

Agents' interaction is implemented based on the Model behavior stage of the conceptual modelling process presented in section 3.5. In the conceptual model, we use a sequence diagram to display the sequence of agents' communication and an activity diagram to display the internal agent's behavior. Since AnyLogic supports Action charts, the activity diagrams corresponding to the internal behavior of agents can be easily converted to the Action charts in the model coding stage. Accordingly, here we just mention the most important parts in the coding process.

The interaction between agents begins when a lot agent is created. In AnyLogic, once an agent is created, the code in the *On startup* section within its *Properties* view is executed before anything else. Thus, the code related to the sending *process request* message from a Lot agent to a Route agent types at *On startup* of Lot agent is executed. The process request message carries the ID and *Current step* of the lot . We need to define a java class with required parameters to represent these types of messages. The process request message has to be sent to a Route agent in the routes population with a specific ID. This ID is already specified in a parameter of Lot agent

and is equal to its type. In order to access an agent in a population that meets a condition, AnyLogic provides the *findFirst()* function:

findFirst(java.lang.Iterable<T> collection, java.util.function.Predicate<T> condition)

This function returns the first agent from the given collection which meets the given condition. The code to send a message to the route with ID equal to the Lot agent's type will look like:

Send (process request, main.routes, r -> r.ID.equals (Lot agent's type))

Once this message is delivered to the destination Route agent, its action code is executed. The agent's reaction on message arrival is defined in the *On message received* section of the *connections* element of Route agent (see Figure 3.27 (b)). In the *On message received* section, we forward the received message to the action chart which is designed based on the activity diagram of Route agent presented at section 3.5 (see Figure 3.22). As already explained, the process request from a Lot agent is treated according to the *Current step* value. The Route agent sends a process request message to a Step agent with particular index (the way of identifying this index is explained in section 3.5). The Step agent sends *information request* messages to a set of ToolGroup agents with particular IDs which are specified in a collection of its editor. Two functions *Send()* and *findFirst()* are used for coding these communications. We create the function *best_TG* to select the best Tool group agent based on the incoming messages. Drag the *Function* element from the *Agent* palette and type the required codes in its *Action body*.

Once all messages are received from ToolGroup agents containing their current information, we call the *best_TG* function. It returns the ID of the selected Tool group agent. The Step agent sends two message: One to the selected ToolGroup agent (containing information on the lot which to be processed) and one to the Lot agent (containing the location of the selected ToolGroup agent). Once the Lot agent receives this message, the function *moveTo* is called as a reaction. At this moment, the Lot agent starts moving to the specified location. ToolGroup stores the information received from the Step agent in *virtual_waiting_queue*.

Once the Lot agent arrives at the destination, a process request message will be sent to the ToolGroup agent. the code corresponding to this message is written in the *On arrival to target location* section of the *Prop-*

erties view of Lot agent. This message is treated by an Action chart designed based on the activity diagram in Figure 3.24. The incoming Lot agent waits in *actual_waiting_queue* and its information is removed from *virtual_waiting_queue*.

Various dispatching functions are implemented to schedule lots waiting in the queue. For each dispatching function, drag the *Function* element and type your Java code in the action body. You can change the dispatching rules and see how they affect the lot cycle times. To schedule the production time of lot agents in ToolGroups agents, we create a dynamic event:

$$Performing<p,\ l,\ t,\ tg>$$

The first parameter is timeout which specifies the processing time of a Lot agent with ID l, on a tool with ID t from the set of tools within a ToolGroup agent with ID tg. The process of the Lot agent begins immediately if there is a Tool with Boolean variable true in the ToolGroup agent. First, we remove the lot agent from the waiting queue and change the Boolean variable of tool t to False and then create an instance of the *Performing* dynamic event. The counting begins when the instance of *Performing* dynamic event is created. During this period which is equal to the processing time of Lot l, the Boolean variable remains equal to False. When the event timeout expires, AnyLogic executes the event's action and then deletes this instance of the dynamic event. In the *Action* section, a *completion* message is sent to lot l and the value of the Boolean variable of tool t changes to True. At this moment, ToolGroup tg has an available tool, thus the next Lot agent waiting in the queue must be scheduled. To select a Lot agent from the *actual_waiting_queue*, a dispatching function is called in the Action section. Once a lot is selected (if the waiting queue is not empty), the Boolean variable of tool t changes to False and an instance of the *Performing* dynamic event is created with new parameters (p and l). Note that once a Lot agent receives a *completion* message, a process request message has to be sent again to the corresponding route. This process is repeated until the production process of the Lot agent is completed.

■ **Feeding data into the model**

In the final phase, we read the semiconductor manufacturing facility's data available in multiple Microsoft Excel files and configure agents by assign-

ing information to them. The number, location, and parameters of Tool groups (such as ID and set of tools) are defined in the Excel file called *ToolGroups.xslx*. The user can control the number of tool groups (removing or importing) by modifying this file. To feed the data corresponding to the tool groups into the model, drag the Excel File element from the *Connectivity* palette and change the name to *ToolGroups_information*. To specify that this element will work with this file, browse the *ToolGroups.xslx* using the *Browse* button in its *Presentation* view. The path of the source file will appear.

The information corresponding to the products that must be produced in the fab such as product types, total number of released lots for each product, and release horizon are included in the Excel file called *Lots.xslx*. Each row contains the information for one product type. To feed the data corresponding to the product types, drag the Excel File element and call it *Lots_information*. In its *Presentation* view, specify the path of *Lots.xslx* file. This information will be fed into *Raw_Material_storehous* agent who is responsible for creating Lot the agents and releasing them in the fab.

Recall that each product type is associated with one route and each route is defined as a sequence of hundreds of process steps. Due to the large number of process steps, the information corresponding to each processing route is stored in a separate excel file. As explained earlier, for each input file of format .xslx, an Excel File element has to be dragged into the editor. Since our simulation model is a data-driven model, in the Model coding stage, we are not aware of the number of product types that must be considered in the model. Thereby, dragging the Excel File element for each route input file is not applicable. To overcome this issue, we create a text file called *Routes.text* which contains the path of all route excel files. Each row contains the path of the route corresponding to one product type. To feed the data corresponding to the routes, drag the Text File element from the *Connectivity* palette and change the name to *Routes_information* and browse the source text file using the *Browse* button in the *Presentation* view. Note that the user can control the number of product types that must be considered in the simulation model by modifying *Lots.xslx* and *Routes.text*.

The Java I.O. package is used to read the input files. Since Main agent is the start point of execution, all the functions for reading data are called in the *On startup* section of the *Properties* view of Main. Once we run the simulation model, the model's startup code executes and the model's agents are constructed and initialized. This is a place for starting agent activities. The graphical interface of our simulation tool is shown in Figure 3.29.

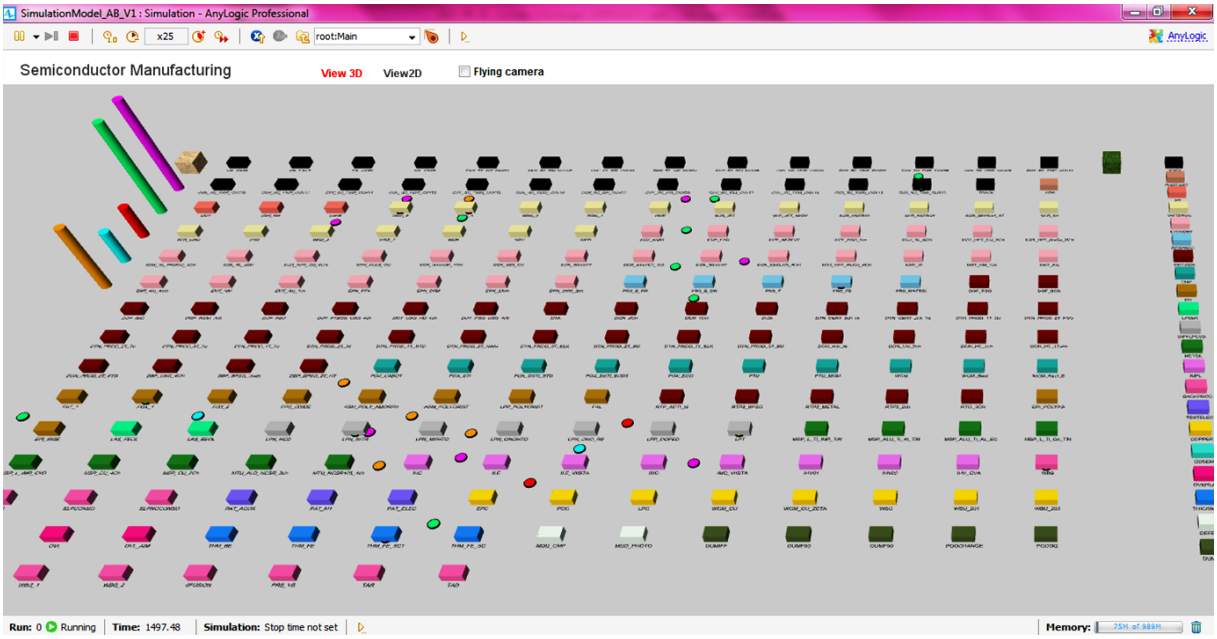

Figure 3.29: Graphical representation of the simulation model for a semiconductor manufacturing facility

3.7 Numerical experiments

In this section, simulation experiments are performed to evaluate the proposed simulation model. The experiments are conducted on an industrial instance which includes five types of products. In total, there are 449 tools in 203 tool groups in the fab which are shared between the process steps of various types of products. The basic information related to products is given in Table 3.1.

In the simulation, 260 lots with the same priority are released in the fab per week. The release horizon is set to 20 weeks. The total number of

released lots for each product during 20 weeks is shown in Table 3.1. The simulation runs until the processing of all released lots is completed. Even for a large system such as this one, the execution time of each simulation run is lower than 3 minutes.

Table 3.1: Characteristics of the products considered in the simulation model

Product Type	Nb.Process steps	Nb. Tool Groups	Nb. Lots
Type 1	969	126	1560
Type 2	831	131	1560
Type 3	667	124	540
Type 4	1100	144	540
Type 5	806	130	1040

tThere are two types of output files in the simulation model. One type includes the statistical data during the simulation and another type is created when the simulation is stopped.

Figure 3.30 provides the screenshot of the information that can be viewed during the simulation. It shows the simulation time versus the cycle time of lots, i.e. in the simulation, each time a lot is completed; its cycle time is shown on the graph. In this experiment, the FIFO (First In First Out) dispatching rule is used. Cycle times are in line with the actual cycle times in the factory.

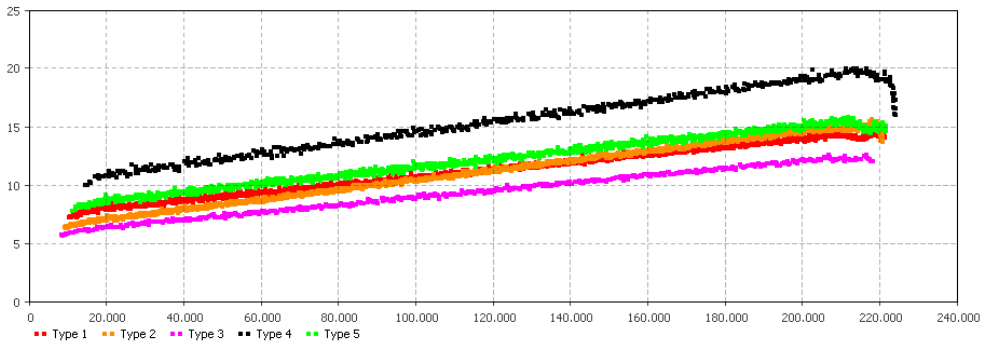

Figure 3.30: Cycle times of lots for all production types

The results show that product type 4 has the largest cycle time. Assume it is desirable to reduce the cycle time of product 4 because of its customer

priorities or high holding costs. The simulation model allows a new dispatching rule to be implemented with the objective of accelerating the production of product type 4. This dispatching rule is based on priorities where a higher priority is assigned to lots of product type 4. Figure 3.31 presents the simulation result using the new dispatching rule.

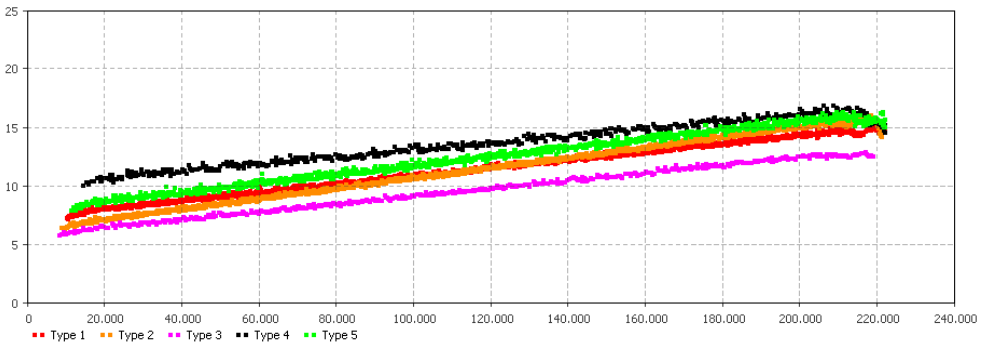

Figure 3.31: Cycle time when accelerating lots of product type 4

Table 3.2 compares the cycle times of lots in both simulation runs. When changing the priority of product type 4 and applying the new dispatching rule, a significant improvement is obtained on the cycle time of product type 4. However, there is no significant change in the total cycle time.

Table 3.2: Average cycle times without and with prioritization of product type 4

Product Type	Original	Average Cycle Time (days) When accelerating product type 4
Type 1	11.10	11.31
Type 2	10.99	11.27
Type 3	9.27	9.48
Type 4	15.35	13.56
Type 5	12.03	12.28
Total	11.47	11.54

In addition to the dynamic representation of statistical data, after stopping the simulation, a text file is created for each lot which contains its

waiting time and completion time in each process step. Using these files, various information can be extracted for each lot, e.g. the bottleneck process step. Also, the utilization percentage of each tool group is written in a text file. According to this information, bottleneck tool groups are specified and we can apply some optimization approach to reduce bottleneck tool groups, cycle times and also to balance the production process, e.g., apply some specific dispatching rules only for bottleneck tool groups.

3.8 Conclusion and Perspectives

This chapter presented a conceptual model and the development of a flexible multi-method simulation model for semiconductor manufacturing. The simulation model combines discrete event and agent based simulation methods. The model is developed based on the dynamic nature of semiconductor manufacturing processes, and thus can be easily re-used and reconfigured. In addition, the model can be extended to study various types of problems in semiconductor manufacturing such as scheduling, capacity planning and production control.

The simulation model has been evaluated on real-world data. Experiments illustrate how the simulation model can be efficient in extracting relevant information and so detecting critical problems. According to the extracted information, the user can set new objectives and apply new strategies using the simulation model. A simple example was used to illustrate the flexibility and efficiency of the simulation model in applying a new objective and a new dispatching rule.

Our future work includes applying and studying the simulation model for various scheduling and optimization problems and extending the model with extra information such as batching machines, machines breakdowns, etc. In addition, the model has to be tested on more industrial instances.

Chapter 4

A Framework for Consistency between Global and Local Scheduling Decisions

To deal with a large-scale complex semiconductor manufacturing system, the overall scheduling problem is often decomposed into a series of sub-problems in each work center (local level). Since scheduling decisions at the local level are discussed independently, the obtained local solution at each work center may not meet the global objectives of the fab (global level). In addition, scheduling decisions within a work center are influenced by decisions taken in other work centers. The goal is to propose a general framework which considers the interrelation and interaction between local scheduling decisions and ensures the consistency between global objectives and local scheduling decisions.

4.1 Introduction and Motivations

Effective production planning and scheduling is critical success factor to achieve the goals of manufacturing companies. Production planning techniques correspond to the medium-term planning level in the hierarchy which determines the quantities of products to be completed in each period of the planning horizon. The decisions are taken according to the resource capacity, supply and demand. These techniques attempt to fulfill the demand which maximizing the utilization of equipment and minimizing inventory, scraps and etc. Once the quantities and types of products are determined, scheduling decisions at the short-term planning level in the hierarchy have to be taken to allocate limited resources to the products and determine start and completion times. There is a clear correlation between planning and scheduling decisions. In fact, production planning decisions at higher level provide targets, constraints, and production quantities for scheduling decisions at lower level. This is not a one-side effect. Poor scheduling decisions can degrade performance and feasibility of planning decisions (Dauzère-Péres and Lasserre (2012) [12]). Developing and designing successful planning and scheduling techniques have been widely studied, either individually or in combination.

Scheduling itself is a very difficult problem which can be crucial for the overall performance of a facility. The focus of this thesis is on scheduling decisions in complex semiconductor manufacturing systems linked to the short-term planning level. Through the production planning decisions, a set of lots is released into the facility over time which have to be scheduled. A semiconductor manufacturing system consists of multiple work centers. A work center contains a set of machines with similar capabilities. All released lots into the fab must go through work centers one after another, according to their processing route. Each route is composed of hundreds of processing steps, and each of them must be performed in a particular work center. The same types of processing steps are executed multiple times which lead to visit the same work centers multiple times and so the reentrant feature for production flow. Scheduling in such a large-scale complex environment is one of the most challenging problem that the semiconductor industry faces.

Thereby, the overall scheduling problem is often decomposed into a series of sub-problems. Based on the physical decomposition of the semiconductor manufacturing process into work centers, scheduling problems are often addressed independently in each work center. The independent study of the scheduling problem in each work center is additionally motivated by the fact that each work center contains specific types of machines, constraints and objectives. For example, the photolithography work center contains bottlenecks machines and requires masks as auxiliary resources for processing an operation.

The scheduling decisions are made and sometimes optimized according to the local information at individual work centers, e.g. current locations of lots, waiting times, arriving times and machines statuses. Since arriving times of lots and scheduling decisions within each work center are influenced by decisions taken at other work centers, without considering global information (information at fab level), the obtained local solution at each work center may not meet the global objectives of the fab such as minimizing the overall cycle time of lots. The global information corresponds to the entire facility, such as release dates, arrival of future lots, current lots in progress and the status of all resources.

The interaction and interrelation between local scheduling decisions at work center level (called local level), as well as their effect on the overall fab (called global level), show the necessity of developing a global scheduling approach. As explained earlier, since fully integrated scheduling approaches involve untamable complexity, work center-based decomposition heuristic methods are extensively used for solving large-scale scheduling problems in semiconductor wafer fabrication (Monch and Drießel (2005) [50], Monch et al. (2007) [53], Sourirajan and Uzsoy (2007) [76]). For example, Monch and Drießel (2005) [50] suggest a two-layer hierarchical approach to decompose the overall scheduling problem into work center-based sub-problems. The upper layer as an aggregated model determines start dates and due dates for the jobs within each work center which are then fed to the lower layer. A modified shifting bottleneck heuristic is then applied in each work center in order to provide detailed schedules.

Global scheduling methods attempt to fill the gap between scheduling decisions at work center level and at fab level. The overall aim of this thesis is to fill this gap by guiding, supporting and coordinating local scheduling decisions that take global information and objectives into account. In other words, scheduling decisions at local level are managed globally to meet global objectives while ensuring the consistency of global and local scheduling levels (see Figure 4.1). After providing an overview of this context in section 4.3, we propose a general framework to ensure consistency between global and local scheduling levels.

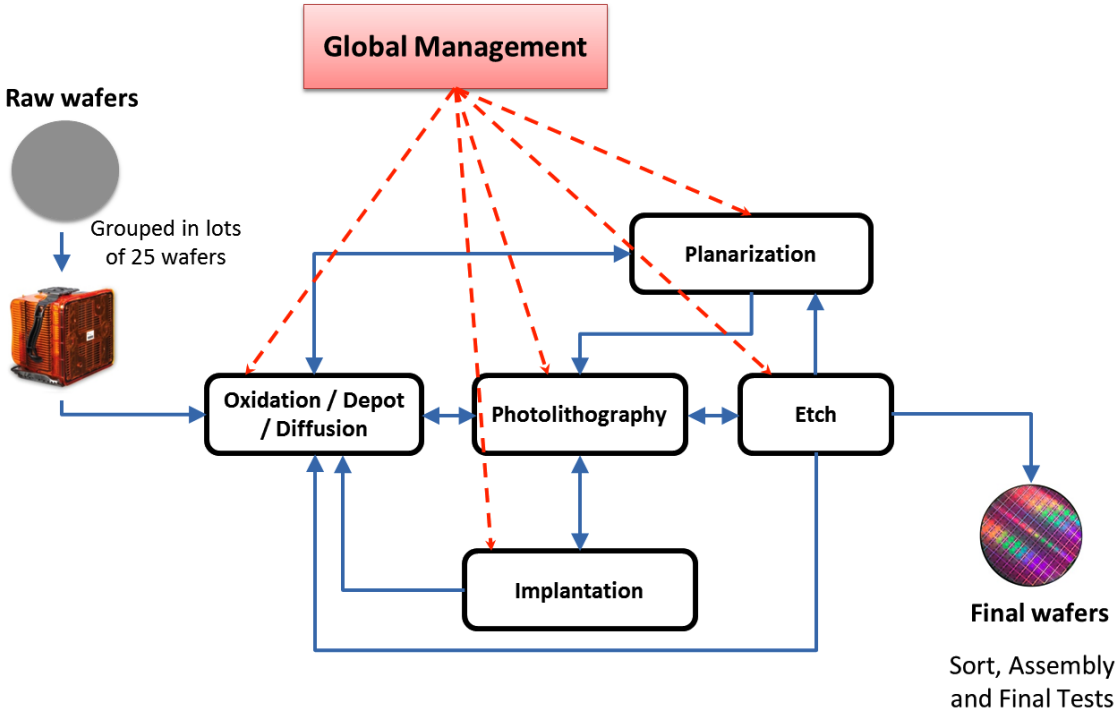

Figure 4.1: Global management of work centers within wafer fabrication

This research specifically deals with the following objectives at global level.

1. *Management of time constraints:* Time constraints between production steps are studied in detailed in chapter 2 of this thesis. An approach to control lots before starting in time constraints

is proposed. However, lots which have already started the production must also be managed to avoid reprocessing or scrap. A time constraint may cover multiple process steps within single or multiple work centers. Thus a global vision of the fab is required to manage lots within time constraints. A monitoring and evaluation approach to manage the lots within time constraints is proposed in section 4.5.

2. *Control of linearity constraints:* Linearity consists of smoothing differences that could appear between:

   - The WIP level of a block and its fixed target
   - The Cycle time in a block and its estimated target

   Blocks correspond to a logical separation of a route that allows intermediate controls on lot manufacturing. Thus this constraint should be managed globally. A monitoring and evaluation approach to control linearity constraints is proposed in section 4.6.

In section 4.7, we describe the simulation framework used to evaluate the performance of the proposed approach. It is an extended version of the simulation model presented in chapter 3 which includes time constraints and linearity. Numerical results on various industrial instances are presented and discussed in section 4.8.

4.2 Problem Statement

We consider the full semiconductor wafer fabrication system with high product mix and reentrant process flows. Among characteristics which cause uncertainty, sampling and lot priorities are taken into consideration. The wafer fabrication model is constructed under the following assumptions:

- The model consists of tools, tool groups and work centers,
- A tool group is a set of tools which can perform similar process steps,

- A buffer with infinite capacity corresponds to a tool group,
- Each product type has an identical processing route,
- Each route contains a sequence of process steps,
- Each process step can be performed by a set of tool groups with tool group-dependent process time,
- Each lot has a priority,

According to the physical structure of the semiconductor manufacturing system and thus the decomposition of the overall scheduling problem into sub-problems, scheduling decisions can be divided into global and local scheduling levels. The local level refers to scheduling decisions at work-center level or machine level which use local information (e.g. processing time, waiting time, queue length of machine at current work center or single machine) to optimize a performance measure. The global level corresponds to scheduling decisions in order to achieve global objectives by taking global information (information concerning all work centers) into account. These two scheduling levels are depicted in Figure 4.2.

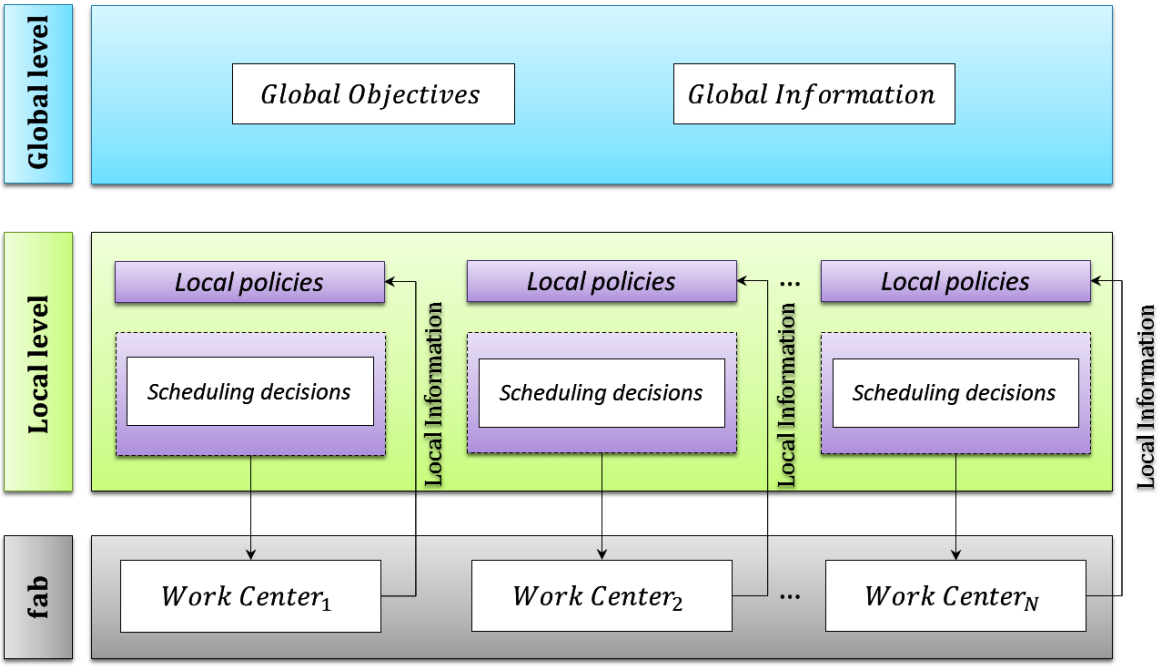

Figure 4.2: Scheduling levels in semiconductor manufacturing processes

Scheduling decisions are often locally taken in each work center for a time horizon ranging between several days and several minutes. Thus, in this study, we use the term "local policies" for scheduling policies at work center level. For a set of lots that have to be processed in a particular work center, scheduling decisions typically decide which machine is used to process a lot, which lots are grouped in the same batch, and in which order lots are processed on their assigned machines over a period of time. The scopes of these policies differ in their decision making horizon and the level of modelling details. The horizon of scheduler is several days or hours. It provides a detailed plan over considered time horizon in order to satisfy the demand from the planning level such as product quantity and due date; maximize the use of the resources and minimize the average waiting time. Usually, the scheduler uses optimization methods to calculate the solution such as mathematical models and meta-heuristic algorithms. Dispatcher takes a shorter time horizon into account and usually is known as a tool to assign the next jobs to be processed from a set of waiting jobs. However, it may deal with

multiple resources, in which, the decisions are made to determine the next machine to process a job. Local scheduling policies are shown in Figure 4.2.

Although scheduling decisions are taken locally based on local information in order to achieve local objectives and satisfy the constraints at work center level (or machine level), global objectives and scheduling decisions at global level may provide objectives or constraints for local decisions. To take these interaction and interrelation into account, we present a general framework which aims at ensuring that local scheduling decisions are consistent with global objectives. The general idea is to provide the local policies a set of extra information in order to achieve global objectives while ensuring consistency between global and local decisions.

4.3 Literature Study

Scheduling decisions greatly impact the overall performance of a fab. This study deals with the scheduling decisions taken at the global and local levels. Local scheduling decisions use local information (e.g. processing times, waiting times and queue length of machines) to optimize one or more performance measures, e.g. the Shortest Process Time (SPT) dispatching rule uses the processing time of lots to maximize throughput. Global scheduling decisions use information at the global level such as the arrival of future lots, customer requests, etc., e.g. the Cost Over Time (COVERT) and Apparent Tardiness Cost (ATC) rules use information on future lots to minimize tardiness (Lu et al. (1994) [47], Vepsalainen and Morton (1988) [81]).

Scheduling decisions at global and local levels are widely discussed for optimizing single performance measures, while a successful semiconductor manufacturing facility requires scheduling rules which are capable of optimizing multiple performance measures simultaneously. Hence, in recent years, advanced scheduling rules are introduced for multi-objective requirements. Most of the multi-objective rules use linear weighting methods to combine multiple scheduling rules into a single rule. Dabbas and Fowler (2003) [11] propose a scheduling approach which combines multiple local dispatching rules and global algorithms using Design Of Experiments (DOE) and mul-

tiple response optimization. Zhang and Jiang (2007) [91] develop a general dynamic scheduling approach which incorporates various global and local dispatching rules based on fuzzy logic and DOE. In this approach, in particular, the global Dynamic Bottleneck Dispatching (DBD) rule is designed for controlling bottlenecks.

Although combining multiple global and local scheduling rules into a single rule can be a successful methodology for a multi-objective optimization problem, the interactions between global and local levels are neglected, i.e. scheduling rules at the global level provide objectives or constraint for decisions at the local level. To deal with this problem, Bureau et al. (2007) [8] present a simulation based scheduling approach which aims at ensuring that local scheduling decisions are consistent with global objectives.In this approach, global objectives are met by dynamically adapting parameters used at the local level. Studying this approach in more details is given as a future research direction by Mönch et al. (2011) [52]. Li and Jiang (2012) [41] propose a pull Virtual Production Line (VPL) based release policy and dispatching rule. The multi-objective optimization is achieved during the process of selecting release and dispatching rules. Yao et al. (2011) [87] propose a decentralized multi-objective scheduling algorithm in which global objectives are decentralized into local ones. This decentralized policy controls VPLs and machine workload. The parameters used in the proposed methodology in [87] and [39] are determined by simulation experiments. Lee et al. (2009) [37] describe a multiple-objective scheduling and real-time dispatching approach (MSRD) based on timed extended object-oriented Petri nets (EOPNs) to optimize various performance measures with real-time response for complex semiconductor manufacturing. The real-time dispatching rule is applied at each allocation of lots on a machine.

4.4 General Framework

In this study, we present a general framework which aims of supporting and controlling the decisions taken at the local level regarding global information and objectives to deal with consistency problems between global and

local scheduling decisions. The goal is to guide the production process at the local level to ensure that global objectives are achieved.

The framework is composed of two layers that are depicted in Figure 4.3. The bottom layer includes local policies used in each work center depending on their features to meet local objectives, e.g. the local dispatching rule SPT can be used to maximize throughput at the local level. The top layer consists of global objectives, global information and the global strategy which is the core of this framework. The global strategy is proposed to control local policies as well as production processes at the local level by taking into account global information, global objectives and consistency issues between the global and local decision levels. The idea is to periodically run the global strategy during the production to guide the production process towards achieving global objectives.

The reentrant nature of the production flow, the uncertainty in demand and uncertainty factors like lot sampling and machine breakdowns motivate the use of a periodic global strategy, so that changes in the process flow can be perceived. Since the overall aim of the global strategy is to improve the ongoing production process, historical information is required in order to understand what has happened during the production and why things happened that may cause the failure of achieving global objectives such as a given overall cycle time. Moreover, managing some constraints which are linked to a particular portion of the production flow (e.g. time constraints which may cover multiple process steps or work centers) reveals the importance of capturing and using the historical information. Thereby, the framework relies on historical and actual information of the entire fab.

The processes within and among bottom and top layers of the framework can be categorized into three main processes: Monitoring, Evaluating/Optimizing and Reporting processes.

1. *Monitoring* is the process of extracting relevant information from the past and ongoing activities in the entire fab over a period of time. This information helps to evaluate the performance of local policies towards achieving expected objectives at the global level. The information which is key element in the performance

assessment is collected according to the global objectives in question and the global strategy at the top layer of the framework. The historical information which is continuously captured can be completion time and waiting time of lots in each process step of its processing route. The current position (process step) of lots, throughputs of machines and their current status (busy/idle) can be considered as real-time information.

2. *Evaluating/Optimizing* is the process which is performed by the global strategy. The global strategy analyzes the production process during the last period according to the global information collected through monitoring. It determines whether the production process at the local level is going in the right direction to achieve the global objectives and how the production process at the local level might be improved. The global strategy can be an evaluation-based algorithm which aims at improving the production process with no guarantee for the optimal solution, or an optimization-based algorithm. Based on the global strategy, the improvement in the local level can be done by

   - Making change in local policies or supply them with extra information; e.g. applying new local dispatching rules when a bottleneck machine is detected,
   - Imposing new constraints on the production process; e.g. specifying a production length for machines
   - Modifying parameters; e.g. updating priorities of lots,
   - ...

3. *Controlling* is the process of feeding the information, constraints and policies by the global strategy into decision-making processes at the local level.

Monitoring, Evaluating/Optimizing, and Reporting are ongoing processes that ensure that sufficient progress is being made toward achieving global

objectives while local and global decisions stay consistent. Recall that monitoring is a continuous process of collecting information of the entire fab at periodic intervals to measure the performance of production at local level towards achieving the objectives at the global level. At the end of each time interval, Evaluating/ Optimizing (or global strategy) and then Reporting processes are performed to guide the production process in the next time periods.

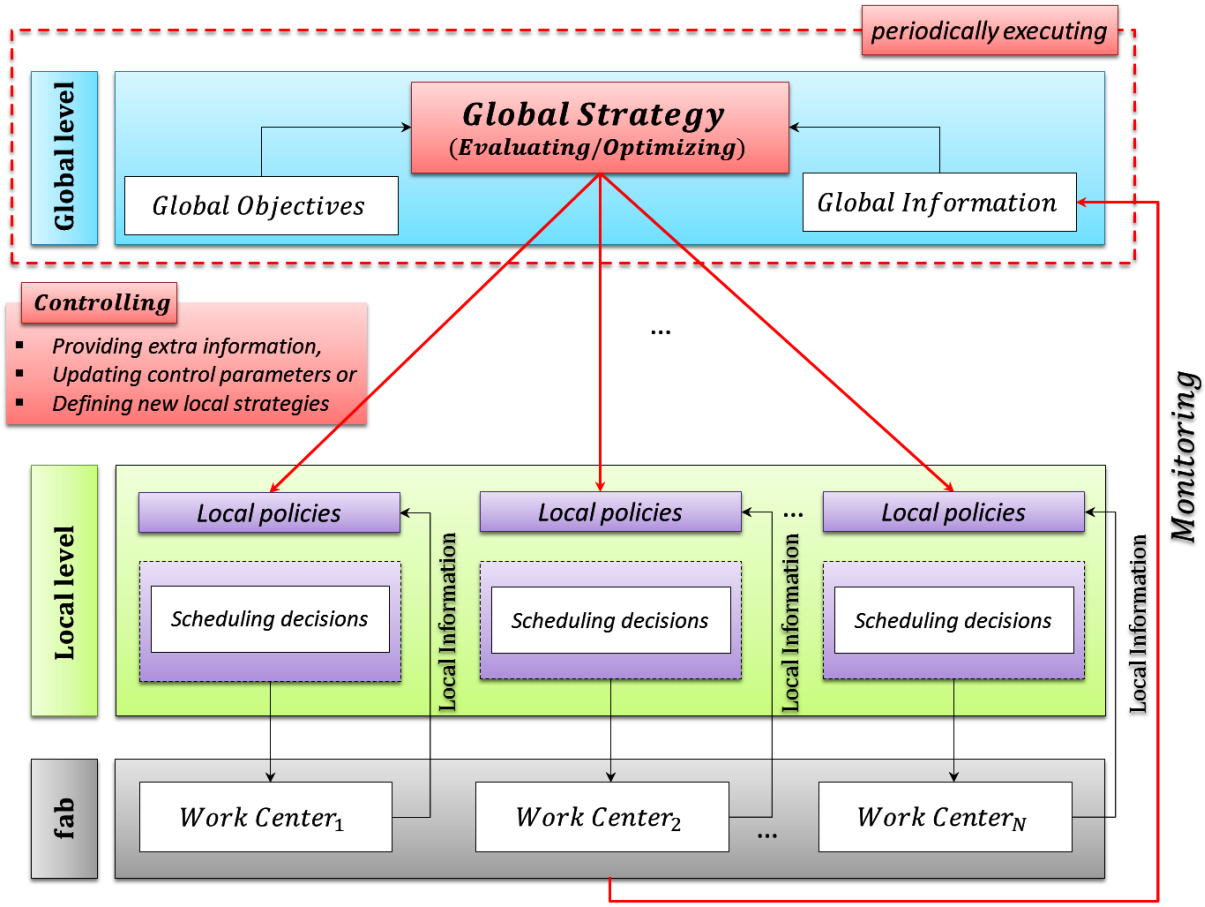

Figure 4.3: Overall structure of the proposed framework

The framework presented in this study focuses on the global strategy. The global strategy will be designed and developed according to the specification of the global objectives , now the global objectives can be satisfied and the information which can be extracted from the local level. To avoid capturing

and collecting unnecessary information, the monitoring process is designed based on the data requirements of the global strategy. The outcome of the global strategy determines the reporting policies and procedures.

The global objectives considered in this study are: Management of Time Constraints and Control of Linearity Constraint. First, we investigate each objective individually and propose an evaluation-based global strategy for each case in the following sections. Then, we study the combination of these two global objectives.

4.5 First Global Objective: Management of Time Constraints

A time constraint corresponds to a maximum time that a lot can spend between two consecutive or non-consecutive process steps in its manufacturing route. Lots that violate a time constraint have to be scrapped or reprocessed which is a waste of cost and time. In this study, management of lots within time constraints in semiconductor manufacturing is considered as a global objective. More precisely, we want to guide the real-time local scheduling decisions to minimize the number of lots which violate the time constraints and to minimize the average time deviation of the lots from their time constraints.

Using the notation presented in Chapter 2, lot l_O can satisfy a time constraint $TC_{g,h}$ if its completion time or sum of waiting times and processing times between two process steps PS_g and PS_h is lower than the maximum allowed time $MaxTime_{g,h}$. Assume that l_O is currently waiting within process step PS_k covered by time constraint $TC_{g,h}$ with $g < k < h$. The goal is to guide the production process of l_O within PS_k and the remaining process steps PS_i with $k+1 \leq i < h$ if l_O may violate its time constraint at the end of production. Lot l_O may violate $TC_{g,h}$ if it has waited a long time within process steps (or corresponding tool groups) already performed. Therefore, the required information to evaluate l_O within a time constraint are: Completion time in each process step PS_i with $g \leq i < k$, arrival time of l_O at current process step PS_k (historical information), number of remain-

ing process steps and remaining allowed time in the time constraints (actual information). This information must be captured through the monitoring process.

Since a time constraint limits the sum of the waiting times and the processing times between two process steps, and each process step can be performed by a set of tool groups, the satisfaction of time constraints strongly depends on the lots buffered in the tool groups. Thus, one way to avoid the violation of time constraints is to accelerate the production of lots by modifying local scheduling rules or updating their parameters. Most of scheduling policies are based on priorities of lots where lots with higher priority will be scheduled before the others. Thereby, by increasing the priority of the necessary lots, their production process will be accelerated and so the possibility of violating time constraints will decrease.

In this study, we focus on lot priorities as parameters to be updated at the global level using the global strategy given in Algorithm 4.1. The basic idea is to accelerate lots which may not satisfy time constraints. Recall that the proposed framework runs periodically. At the end of each time interval, the monitoring process provides the global strategy with historical and actual information of whole fab. The global strategy predicts whether a lot may satisfy its time constraints or not, and then accelerates this lot if it may violate its time constraints. Based on the historical information, the remaining time constraint that a lot is allowed to spend in the remaining process steps is calculated in line 4. The idea is to estimate the allowed time in the time constraints in each remaining process step. The time allowed by a given time constraint in the current step is estimated by dividing the remaining time in the time constraints by the number of remaining process steps in line 5. A lot should be accelerated if its actual waiting time at the current process step is strictly larger than the allowed waiting time (see lines 14 and 15). The allowed waiting time is the time difference between the estimated allowed time constraints and the processing time at the current step (see line 9). Finally, the controlling process feeds new parameters into the local level and the production process continues with updated parameters. The global strategy and the associated notation are presented below.

Parameters

TCT	set of time constraints
l_i	lot i limited by a time constraint
PS_k	current process step of lot i
$CT_{i,j}$	Completion time of lot l_i in process step j
$ATime_i$	Arrival time of l_i at PS_k
$PTime_i$	Processing time of l_i at PS_k
$CTime$	Current time
RTC	Remaining time in time constraints
$AT_{i,TC_{m,n}}$	Allowed time for l_i in time constraint $TC_{m,n}$ at PS_k
AT_i	Final allowed time in time constraints for l_i at PS_k
$WTime_i$	Waiting time of l_i at PS_k
$ATime_i$	Allowed waiting time for l_i at PS_k

Algorithm 4.1. An Evaluation-based Global Strategy: Time Constraint Management

1: Initialization for each lot l_i :$AT_{i,TC_{m,n}} = \infty$ and $AT_i = \infty$
2: **for** all $TC_{g,h} \in TC$ **do**
3: **for** each l_i limited by $TC_{m,n}$ **do**
4: $RTC = TC_{g,h} - (CTime - CT_{i,k})$
5: $AT_{i,TC_{g,h}} = \frac{RTC}{\text{number of remaining steps}}$
6: **if** $AT_i > AT_{i,TC_{g,h}}$ **then**
7: $AT_i = AT_{i,TC_{g,h}}$
8: **end if**
9: $WTime_i = CTime - ATime_i$
10: $ATime_i = AT_i - PTime_i$
11: **end for**
12: **end for**
13: **for** each limited l_i **do**
14: **if** $WTime_i > ATime_i$ **then**
15: accelerate l_i by increasing w_i
16: **end if**
17: **end for**

Note that overlapping may occur between time constraints, i.e.

$$\exists TC_{g,h}, TC_{v,w} \in TC \quad \text{where} \quad q \leq v < \min(h, w) \tag{4.1}$$

In this case, a process step is covered by multiple time constraints, so we consider the minimum allowed time constraint among the estimated allowed time constraints (see lines 6 and 7 of the algorithm). In this study, we assume that all lots at the beginning have the same priority with a standard value of 300. The acceleration is performed by increasing local priorities with two different methods. The first method for updating priorities is presented in Algorithm 4.2 and is visualized in Figure 4.4 which has some similarity with the method proposed in Bureau et al. (2007) [8]. The allowed waiting time at the current process step is assumed as target tolerances. Target tolerances indicate that under 10%, we consider that lots can spend some time in a waiting queue without violating time constraints. Between 10% and 20%, lots are accelerated by changing their priorities to 400. Finally, when the gap from the allowed waiting time is larger than 20%, the priority of the lot

is set to 500. Then, new priorities are sent to the local level and the process continues with new parameters. At the end of each time interval, the global strategy is launched to update the priorities of lots.

Algorithm 4.2. Priority setting (I)

```
1: for each limited l_i do
2:     if WTime_i > ATime_i then
3:         if WTime_i > ATime_i + ATime_i * 0.2 then
4:             w_i = 500
5:         end if
6:         if ATime_i + ATime_i * 0.1 < WTime_i < ATime_i + ATime_i * 0.2
   then
7:             w_i = 400
8:         end if
9:         if WTime_i < ATime_i + ATime_i * 0.1 then
10:            No change
11:        end if
12:    end if
13: end for
```

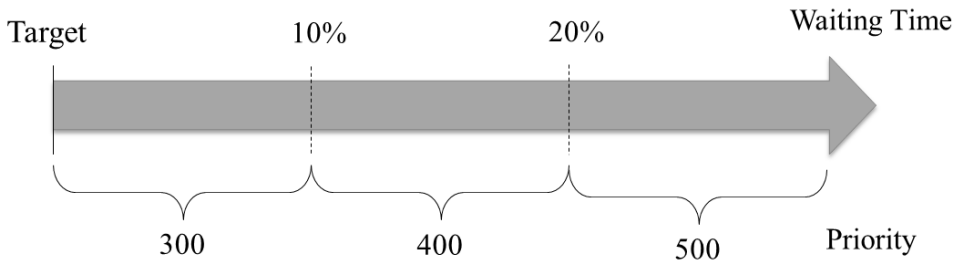

Figure 4.4: Example of priority setting to manage time constraints

The second method for accelerating priorities is presented in Algorithm 4.3. In this method, instead of considering gaps to attain priorities of 400 and 500 for all limited lots by time constraints, we consider percentage of lots, e.g. at most 5% of the lots with the largest deviation from their time constraints are accelerated by changing their priority to 500 and at most 20% of the remaining lots with the largest deviation are accelerated by setting their

priority to 400. For this purpose, lots must be placed in a list in decreasing order of deviation of time constraints (called *SortedList*). The algorithm and the associated notation are presented below.

Parameters

DTC_i	Deviation of time constraint for l_i
SortedList	List of lots with $DTC > 0$
P_1	percentage of lots which should be accelerated with priority 500
P_2	percentage of lots which should be accelerated with priority 400

Algorithm 4.3. Priority setting (II)

```
1: for each limited l_i do
2:     DTC_i = WTime_i − ATime_i
3:     if DTC_i > 0 then
4:         Add l_i in SortedList
5:     end if
6:     for P_1% of current lots within SortedList do
7:         w_i = 500
8:     end for
9:     for P_2% of remained lots within SortedList do
10:        w_i = 400
11:    end for
12: end for
```

4.6 Second Global Objective: Control of Linearity Constraint

Because of the long and complex production process in semiconductor industry, linearity constraint is defined in order to have intermediate controls on lots manufacturing process and to balance processing routes. To setup

this constraint, processing route r_α of product type α with q_α process steps should be divided into B_α sub-sequence of process steps which called *blocks*. The set of blocks corresponds to the route r_α can be formalized as follow:

$$block_\alpha^1 = \{PS_{\alpha,i} | i = 1, \ldots, n_1\}$$

$$block_\alpha^2 = \{PS_{\alpha,i} | i = n_1 + 1, \ldots, n_2\}$$

...

$$block_\alpha^{B_\alpha} = \{PS_{\alpha,i} | i = b_\alpha + 1, \ldots, q_\alpha\}$$

The i^{th} block of route r_α contains b_i process steps from process step $PS_{\alpha,m}$ till process step $PS_{\alpha,n}$ with $m < n$. The number of existing process steps (b_i) in the block called length.

The division can be done in two ways: first, by dividing a route into blocks with the same level of activity and so with identical target which may differ in the length, secondly, by dividing a route into blocks with the same length but with distinct target. According to the division, the linearity constraint can be expressed as

- Smoothing differences that could appear between activity levels of all blocks.
- Smoothing differences that could appear between activity level of a block and its fixed target

In this study, routes are divided into a specified number of blocks where activities within a block are considered as WIP level to avoid WIP accumulation in the fab and cycle time of individual lots to avoid consuming long processing time within some portion of the route. The goal is to control the production process within each block in order, for instance, to maintain the WIP levels close to predetermined WIP level targets. The control of linearity constraint in semiconductor manufacturing is considered as a global objective. More precisely, we want to guide the real-time local scheduling decisions to minimize the deviation of the WIP level or the cycle time within each block from their desired targets. Estimating the target of each block

itself is a challenge when controlling linearity constraint in a semiconductor fab. Since the WIP level and the partial cycle time can be obtained from historical data of a real fab, we have to conduct a simulation study to estimate these targets. In the following, an evaluation-based global strategy is presented independently for each type of activity.

General Parameters

$block_\alpha^k$	k^{th} block of route r_α
FID_α^k	ID of the first process step in k^{th} block of r_α
LID_α^k	ID of the last process step in k^{th} block of r_α
$CT - Target_\alpha^k$	Cycle time target in k^{th} block of r_α

a) An evaluation-based global strategy: Control of Linearity Constraint with Cycle time target

Linearity consists of smoothing differences that could appear between cycle time in a block and its estimated target. Lot l_O can respect linearity constraint within block $block_\alpha^k$ if its completion time between two process steps $PS_{FID_\alpha^k}$ and $PS_{LID_\alpha^k}$ is equal to the predetermined cycle time target $CT - Target_\alpha^k$. Assume that l_O of type α is currently waiting at process step PS_i covered by k^{th} block with $FID_\alpha^k < i < LID_\alpha^k$. The goal is to guide the production process of l_O within PS_i and the remaining process steps PS_j with $i < j < LID_\alpha^k$ if it is ahead or behind the schedule corresponding to $block_\alpha^k$ at its cycle time target. Lot l_O is ahead of its schedule if the forecast results show that its completion time within $block_\alpha^k$ will be lower than $CTtarget_\alpha^k$. It express the fact that l_O has caused long waiting time for other lots in the block, thus, the production process of l_O should be slow down in order to accelerate the production process of late lots. The same argument applies to speed up the production process of l_O if it is behind the schedule. Therefore, the required information to evaluate l_O within a block which have to be provided by monitoring process are: completion time at

the previous steps, arrival time of l_O at the current process step, remaining allowed cycle time and number of remaining process steps.

Since most of local scheduling policies are based on priority, we focus on lot priorities as parameters which should be updated by global strategy which is presented in Algorithm 4.4 to speed up or slow down the production process of the necessary lots at the local level. The procedures that are operated in the presented framework to control linearity constraint as global objective are as follows: At the end of each time period, historical and actual information collected by monitoring process are feed to the global strategy. The global strategy predicts whether a lot is ahead or behind the schedule corresponding to its current block. It estimates allowed waiting time within current process step based on the reaming cycle time and process steps (see line 6,7,8). The production process of a lot will be speed up (slow down) if its actual waiting time at the current process step is larger (smaller) than the allowed time (line 10 till 24). Finally, the reporting process updates the production process with new parameters. The global strategy and the associated notation are presented below:

Parameters

$AverageCT_\alpha$	Average cycle time of route r_α
PT_α	Processing time in route r_α
PT_α^k	Processing time in k^{th} block of route r_α
ID_i	ID of current process step of lot l_i
w_i	Priority of lot l_i
$WTime_i$	Waiting time of l_i at the current process step
$PTime_i$	Processing time of l_i at the current process step
$CT_{i,j}$	Completion time of lot l_i at process step PS_j
$ArTime_i$	Arrival time of lot l_i at the current process step
$CurrentTime$	Current time

RCT	Remaining cycle time
RPS	Remaining process step
$AT^k_{i,\alpha}$	Allowed waiting time for lot l_i within its current process step in k^{th} block of route r_α

Algorithm 4.4. An Evaluation-based Global Strategy: Control of Linearity Constraint with Cycle Time Target

for each route α **do**
  if r_α is limited by linearity constraint **then**
    for each $block^k_\alpha$ of r_α **do**
      for each l_i within $block^k_\alpha$ **do**
        $RCT = CTtarget^k_\alpha - (CurrentTime - CT_{i,FID^k_\alpha})$
        $RPS = LID^k_\alpha - ID_i$
        $AT^k_{i,\alpha} = \frac{RCT}{RPS}$
        $WTime_i = CurrentTime - ArTime_i + PTime_i$
        if $WTime_i > AT^k_{i,\alpha} + AT^k_{i,\alpha} * 0.2$ **then**
          $w_i = 500$
        end if
        if $WTime_i < AT^k_{i,\alpha} - AT^k_{i,\alpha} * 0.2$ **then**
          $w_i = 100$
        end if
        if $AT^k_{i,\alpha} - AT^k_{i,\alpha} * 0.2 < WTime_i < AT^k_{i,\alpha} - AT^k_{i,\alpha} * 0.1$ **then**
          $w_i = 200$
        end if
        if $AT^k_{i,\alpha} + AT^k_{i,\alpha} * 0.1 < WTime_i < AT^k_{i,\alpha} + AT^k_{i,\alpha} * 0.2$ **then**
          $w_i = 400$
        end if
        if $AT^k_{i,\alpha} - AT^k_{i,\alpha} * 0.2 < WTime_i < AT^k_{i,\alpha} + AT^k_{i,\alpha} * 0.1$ **then**
          $w_i = 300$
        end if
      end for
    end for
  end if
end for

The acceleration (deceleration) of the production process of lots is performed by increasing (decreasing) the lots priorities. The method for updating priorities is visualized in Figure 4.5 which is an extended version of

the priority setting method presented in Algorithm 4.2. In fact, in addition to the acceleration process, the deceleration process is also included in this method.

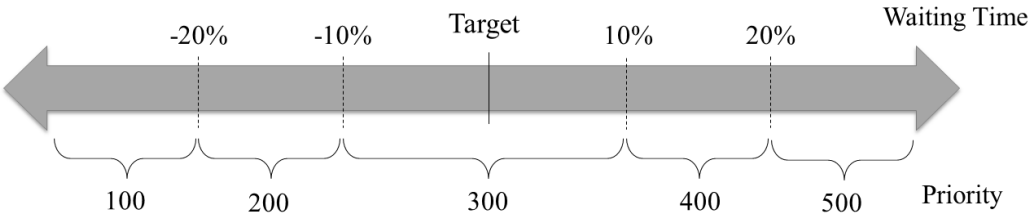

Figure 4.5: Example of priority setting to control linearity constraint with cycle time target

b) An evaluation-based global strategy: Control of Linearity Constraint with WIP level target

Linearity consists of smoothing differences that could appear between the WIP level of a block and its fixed target. The goal is to balance the processing route and avoid WIP accumulation in a portion of fab. Algorithm 4.5 presents the global strategy to control linearity constraint with WIP level target. The production process of lots waiting at process steps covered by a block will be speed up (slow down) if current WIP level within the block is larger (smaller) than the target level (see lines 8 until 23).

Parameters

TW_{α}^{k}	WIP target in k^{th} block of route r_{α}
$CurrentWIP_{\alpha}^{s}$	Current WIP level within s^{th} process step of router$_{\alpha}$
CW_{α}^{k}	Current WIP level within k^{th} block of router$_{\alpha}$

Algorithm 4.5. An Evaluation-based Global Strategy: Control of Linearity Constraint with WIP Target Level

for each route α **do**
  if r_α is limited by linearity constraint **then**
    for each $block_\alpha^k$ of r_α **do**
      for $s = FID_\alpha^k; s < LID_\alpha^k; s++$ **do**
        $CW_\alpha^k = CW_\alpha^k + CurrentWIP_\alpha^s$
      end for
      for each l_i within $block_\alpha^k$ **do**
        if $CW_\alpha^k > TW_\alpha^k + TW_\alpha^k * 0.2$ **then**
          $w_i = 500$
        end if
        if $CW_i < TW_\alpha^k - TW_\alpha^k * 0.2$ **then**
          $w_i = 100$
        end if
        if $TW_\alpha^k - TW_\alpha^k * 0.2 < CW_\alpha^k < TW_\alpha^k - TW_\alpha^k * 0.1$ **then**
          $w_i = 200$
        end if
        if $TW_\alpha^k + TW_\alpha^k * 0.1 < CW_\alpha^k < TW_\alpha^k + TW_\alpha^k * 0.2$ **then**
          $w_i = 400$
        end if
        if $TW_\alpha^k - TW_\alpha^k * 0.2 < CW_\alpha^k < TW_\alpha^k + TW_\alpha^k * 0.1$ **then**
          $w_i = 300$
        end if
      end for
    end for
  end if
end for

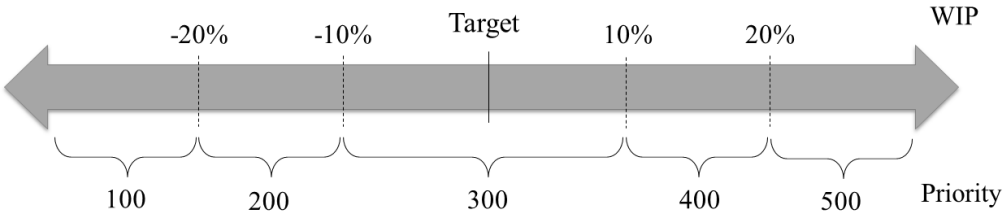

Figure 4.6: Example of priority setting to control linearity with WIP level target

4.7 Experimental Study

This section deals with the evaluation and analysis of the proposed framework. The goal is to show the performance of the framework and to verify its applicability in a real semiconductor fab. Since conducting experiments in a complex and expensive semiconductor manufacturing system require an inordinate cost and length of time, simulation experiments must be performed to assess the effectiveness of the proposed framework. The framework is designed to cope with scheduling decisions at global level (whole fab) and local level (work centers) of a range of semiconductor systems which have structural similarities. Thus, experiments must be conducted on a data-driven generic simulation model of a semiconductor manufacturing system. For this end, Chapter 3 is dedicated to the modelling and development process of such a simulation model.A simulation model was developed based on the general structure of a semiconductor manufacturing system which can measure cycle times of multiple types of products by applying various dispatching rules. In order to design the simulation experiments, the simulation model must be extended by considering structure and elements of the framework. First of all, an appropriate dispatching rule is selected based on the global strategy in the framework. Then, the simulation model is extended by considering time constraints and linearity constraints as new characteristics of production routes. The global strategy is then developed in the model and incorporated during the simulation run. Figure 4.7 shows how the global strategy (an evaluation algorithm) is applied during the simulation run.

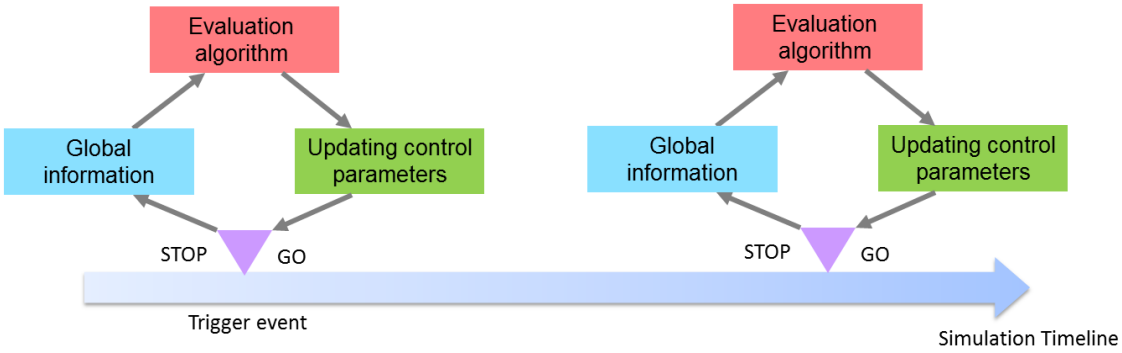

Figure 4.7: Simulation Experiments (applying an Evaluation algorithm)

4.7.1 Selecting an appropriate dispatching rule

Several dispatching rules such as FIFO, SPT, EDD, etc. have been developed in the simulation model to sequence the lots queued in front of tool groups in real time. According to the purpose of the experiments, an appropriate dispatching rule must be applied in the simulation. Since the lot priority is taken into account as a control parameter in our global strategy, it is considered as the main parameter in the required dispatching rule. The second parameter included in the dispatching rule is the waiting time in front of tool groups which heavily influences the delivery date and causes the delay of lots. The corresponding dispatching rule is expressed in the following algorithm:

Parameters

TG_j	Tool group j which has an idle tool
$Priority_i$	Priority of l_i
$ArrivingTime_i$	Arrival time of l_i at TG_j
EP_i	Estimated priority for lot l_i
l_O	Lot which will be assigned at the idle tool of TG_j to be processed
$CurrentTime$	Current time in the simulation run

Algorithm 4.6. Dispatching rule

1: **for** each lot l_i queued in front of tool group TG_j **do**
2: $\quad EP_i = (CurrentTime - ArrivingTime_i) * Priority_i$
3: **end for**
4: Select l_O with the largest priority as a lot to be dispatched first

4.7.2 Extending the Simulation Model

■ **Integrating Time Constraints**

In order to consider time constraints as new components in the simulation model, it is necessary to go through the model content phase of the conceptual model. As explained in section 3.4 of Chapter 3, create a Class for time constraints and call it *Time constraint*. A time constraint covers a set of process steps and limits the processing times and waiting times within these steps by a maximum time. Thus, the class *Time constraint* four attributes: ID of time constraint, ID of the first step covered by time constraint, ID of the last step covered by time constraint, and the maximum time that can be spent within the covered process steps. The relationships between the *Time constraint* class and other classes is specified in Figure 4.8. A route may contain zero or more time constraints. To specify this relationship in the diagram, a solid line which represents an *association* relation is established between the *Route* class and *Time constraint* class. A process step may be covered by zero or more time constraints (i.e. when time constraints are overlapping) and a time constraint needs the information of only two process steps (first and last steps). This relationship is depicted by a solid line between the *Process step* class and *Time constraint* class.

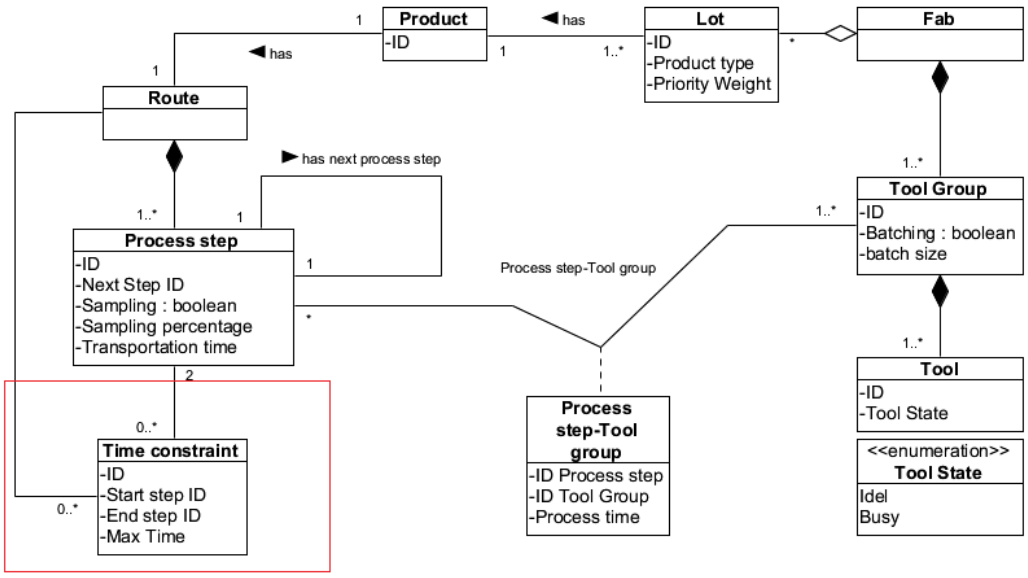

Figure 4.8: Integrating time constraints in the simulation model (UML class diagram)

■ Integrating Linearity Constraints

In order to consider linearity constraints in the simulation model, we can follow the same procedure for the integration of time constraints. The linearity constraint is specified by a set of blocks and a WIP target or a cycle time target corresponding to each block. Thus, blocks are the main component of the linearity constraint which must be included in the model. Create a class for blocks and call it *Block*. A block covers a set of process steps and limits the number of waiting lots or processing time within these steps. Thus, the *Block* class contains four attributes: ID of the block, ID of the first step covered by the block, ID of the last step covered by block, the maximum time that can be spend within process steps, and the maximum number of lots can be accumulated within the covered process steps. Figure 4.9 represents the relationships between the Block class and other classes. A processing route may be into zero (when it does not have linearity constraints) or more blocks. To represent this relationship, a solid line is established between the *Route* class and *Block* class. When a route is limited by linearity constraints, the corresponding process steps are only covered by one block. A block requires the information of the first process step and the last process step. This relationship is depicted by a solid line between the *Block* class and the *Process step* class. To extend the developed simulation model with time constraints and linearity constraints follow the procedure presented in the Model coding phase in section 3.6 of Chapter 3.

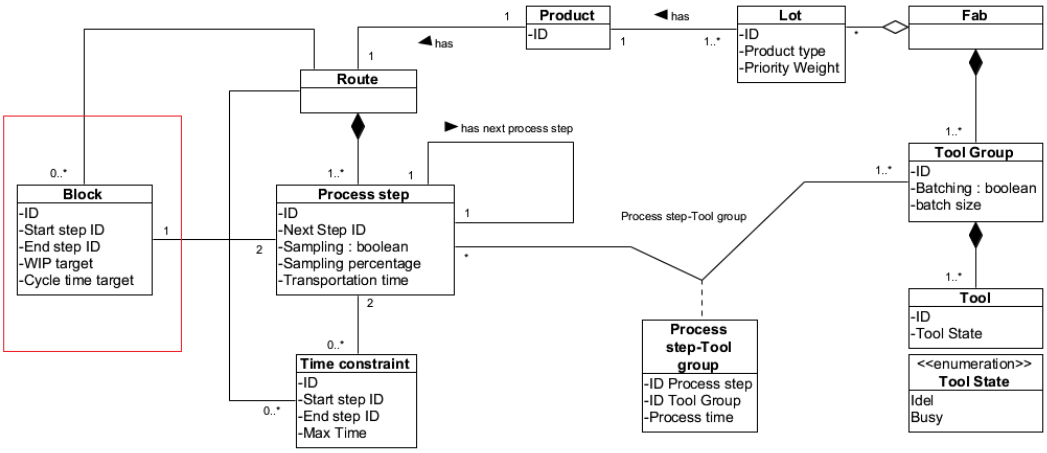

Figure 4.9: Integrating blocks in the simulation model (UML class diagram)

■ Incorporating Global Strategy

We use Function and Dynamic event presented at the Agent palette to incorporate the global strategy within simulation (see section 3.6 of Chapter 3). Function has a place to write Java code. Drag a function from the Agent palette, the graphical editor of the Main agent, and call it *Evaluation_ Strategy*. The Evaluation-based Global strategy presented in Algorithm 4.1 and priority setting algorithm presented in Algorithm 4.2 are implemented in the function body. In order to periodically apply the global strategy during the simulation run, create a dynamic event and call *Applying_ Evaluation_ Strategy*. Recall that dynamic events schedule any concurrent and independent events. We create an instance of the dynamic event by calling the method

create_ Applying_ Evaluation_ Strategy_<DynamicEventName>

The first parameter of this method is a timeout which specifies the time at which the evaluation strategy must be applied from the current model time. The function *Applying_ Evaluation_ Strategy* is called in the event's action and is executed when the timeout expires. To schedule the execution of the *Evaluation_ Strategy* in the next time period, in the event's action, create an instance of the dynamic event with the same parameters. An instance of the dynamic event is created until the production process of all released lots is completed.

4.8 Experimental Results

In this section, simulation experiments are performed to evaluate the proposed approach. A simulation model combining discrete-event models and agents was developed using AnyLogic version 7.1, based on the structure and characteristics of a real semiconductor manufacturing facility (see Chapter 3).

To illustrate the performance of our approach in managing lots within time constraints, the percentage of lots violating time constraints and the average time deviation of the lots from their time constraints are considered. Note that the time deviation from a time constraint by a lot is only counted when the lot violates the time constraint, i.e. the deviation is only positive.

The experiments are conducted on an industrial instance which includes five types of products. The basic information of products is given in Table 4.1 In the simulation, 260 lots are released in the fab per week. The release horizon is set to 20 weeks. The total number of released lots for each product during 20 weeks is shown in Table 4.1. The simulation runs until the processing of all released lots is completed.

Table 4.1: Characteristic of the 5 products in the simulation model

Product Type	Nb.Process steps	Nb. Tool Groups	Nb. Lots
Type 1	969	126	1560
Type 2	831	131	1560
Type 3	667	124	540
Type 4	1100	144	540
Type 5	806	130	1040

The basic input files of the simulation model are “ToolGroups”, “Routes” and “Lots”. The file “Lots” includes the information of the lots planned to be released in the fab. The processing information of each product type is provided in the file “Routes”. Finally, the file “Tool Groups” contains the information related to the tools in the fab.

To validate the effectiveness of the approach to manage lots within time constraints using simulation experiments, we extended our simulation model

by considering an additional file "Time Constraints" and an evaluation-based global strategy. The "Time Constraints" file includes information on the time constraints for each type of product. The global strategy is presented in section 4.5.

The simulation model is set to run on the instance of Table 4.1 where the route of product type 1 includes 5 time constraints, each covering a different number of process steps. The general information of each time constraint is shown in Table 4.2.

Table 4.2: Time constraints for product type 1

Time Constraint	Nb.Covered steps	Maximum Time (Min.)
TC 1	27	1250
TC 2	27	1500
TC 3	50	2400
TC 4	50	2500
TC 5	70	1550

To apply our approach in the simulation model, we chose a dispatching rule based on lots priorities to sequence lots on each machine. This rule consists in sorting lots according to their priorities and waiting times in front of the machine using the following formula:

$$w_i * (CTime - ATime_i) \tag{4.2}$$

We define the trigger events using a constant time interval, which is equal to 10 minutes in our experiments, i.e., every 10 minutes, the simulation is stopped and the algorithm is applied to change lot priorities if necessary. Other time intervals were tested (5 minutes and 20 minutes) and, although more analysis is required, the results were not significantly different. The simulation starts again with the new lot priorities. We thus guide the production of lots during the simulation to minimize the time deviation of lots from their time constraints and to minimize the number of lots which violate time constraints.

The simulation model is executed without and with our approach. The results are shown in two tables. Table 4.3 compares the percentage of lots

violating time constraints in both simulation runs. Note that all time constraints are much better satisfied when lot priorities are updated at the global level.

Table 4.3: Percentage of lots violating time constraints

Time Constraint	Without changing lot priorities	When changing lot priorities
TC 1	26.9%	0.3%
TC 2	27.2%	0.0%
TC 3	30.4%	0.0%
TC 4	30.1%	2.5%
TC 5	18.4%	8.9%

When considering the average time deviation from time constraints in Table 4.4, the results show that very large improvements are also obtained for all time constraints.

Note that Tables 4.3 and 4.4 show that our approach allows TC2 and TC3 to be satisfied for all lots, and TC1 for most of the lots. However, close to 9% of the lots are still violating TC5. This probably means that less lots should be released in this time constraint. A perspective could be to use the static approach proposed in Sadeghi et al. (2015) [72] to control the release of lots in time constraints.

Table 4.4: Average time deviation from time constraints

Time Constraint	Without changing lot priorities	When changing lot priorities
TC 1	17.02%	0.0
TC 2	47.46%	0.0
TC 3	99.34%	0.0
TC 4	151.91%	0.06
TC 5	15.62%	0.79

Average cycle times are changing per product type, but remain the same for all products.

Table 4.5: Average Cycle Time

Product Type	Without changing lot priorities	When changing lot priorities
Type 1	-	-5.7%
Type 2	-	+2.5%
Type 3	-	+2.4%
Type 4	-	+2.2%
Type 5	-	+1.2%
Total	-	0.0%

4.9 Conclusion and Perspectives

In this chapter, we first introduced a general framework to take more consistent decisions between the global and local scheduling levels. This framework aims at controlling production in real time at the local level to ensure that global objectives are reached.

After describing the principle of the proposed approach, the management of time constraints was defined as a first global objective. We developed an algorithm based on the updating of the priorities of lots that are within time constraints. The control of linearity constraints was defined as a second global objective. The linearity constraints are expressed in two ways and an algorithm based on updating the priorities of lots is developed. To evaluate the performance of our framework while managing of time constraints is considered as a global objective, a simulation model was developed which is an extended version of the simulation model presented in chapter 3. Experiments on real-world data illustrate how our approach improves performance measures related to the satisfaction of time constraints.

The first goal of our future research is to evaluate the performance of our approach for controlling linearity constraints by conducting experiments on real-world data. In reality, it is necessary to optimize multiple global objectives simultaneously. Thus, proposing an algorithm which optimizes multiple performance measures by considering more information and constraints at the global and local levels, e.g. lot release policies, cycle times of

lots, batching machines, machines breakdowns, etc., is one of the main goals of our future research. Also, we would like to study the right trigger events (e.g. the right time interval to update global decisions for time constraints). In addition, our approach has to be tested on more scenarios of our industrial instance and on additional instances.

Chapter 5

An Approach for Consistency between Global and Local Scheduling Decisions

Based on the framework presented in Chapter 4, an optimization (Linear Programming) model with various extensions is proposed to take decisions at the global scheduling level. Cycle times are first minimized. Time constraints are then taken into account. Finally, a linearity objective is considered. The control mechanism of the local level is different than in the previous chapter. Priorities of lots are not updated but quantities of products to complete in process steps in each period are sent to and imposed at the local level. Computational results with our framework using the optimization model are presented and discussed.

5.1 Introduction and Motivation

Figure 5.1, which is similar to Figure 4.3, recalls the structure of the proposed framework. In Chapter 4, "simple" rules are proposed at the global level to consider global objectives and determine parameters to control the local level.

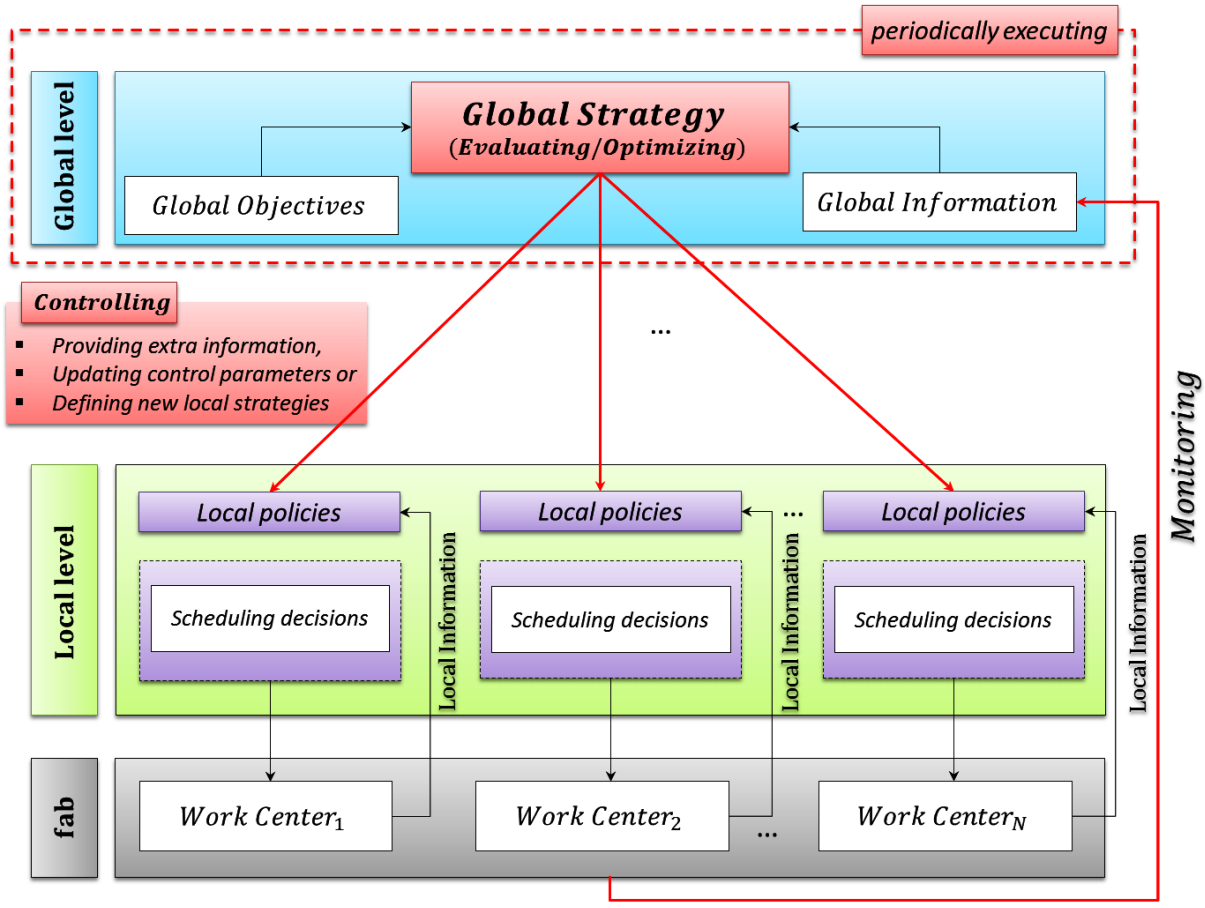

Figure 5.1: Overall structure of the proposed framework

In this chapter, a different control mechanism than in Chapter 4 is used. At the global level, instead of updating lot priorities, the quantities of products to complete in each process step and in each period are optimized. These quantities become objectives to attain, but also constraints to satisfy, at the local level. We introduce in Section 5.2 a Linear Programming model to optimize global scheduling decisions taken at the global level. Linear Pro-

gramming is necessary to be able to solve instances of industrial sizes with hundreds of machines and process steps.

5.2 An Optimization (Linear Programming) Model

An important question is related to the objectives and constraints that should be considered in the optimization model. A base Linear Programming (LP) model is presented in Section 5.2.1, which is extended to consider time constraints in Section 5.2.2, and then linearity in Section 5.2.3.

Figure 5.2 shows how the LP model is applied in a rolling horizon in the framework of Figure 5.1. A trigger event, e.g. after a given number of time periods, induces the resolution of the LP model after collecting parameters from the current situation in the system, in particular inventory levels in workcenters. The solution of the LP model is then used to impose objectives and constraints at the local level in terms of quantities of each product (associated to a single route) to be completed in a given process step in each period (variables Y_{glp}). The control mechanism of the local level is detailed in Section 5.3.

5.2.1 Base Linear Programming model

The following notation is used to write our base Linear Programming model. First, let us introduce the sets and indices:

G: Set of all routes (products),
g: Route index,
K: Set of all workcenters (station families),
k: Workcenter index,
$L(g)$: Number of process steps in route g,
l: Process step index,
$LK(k)$: Set of process steps and routes that must be processed on workcenter k, i.e. $(g, l) \in LK(k)$ if

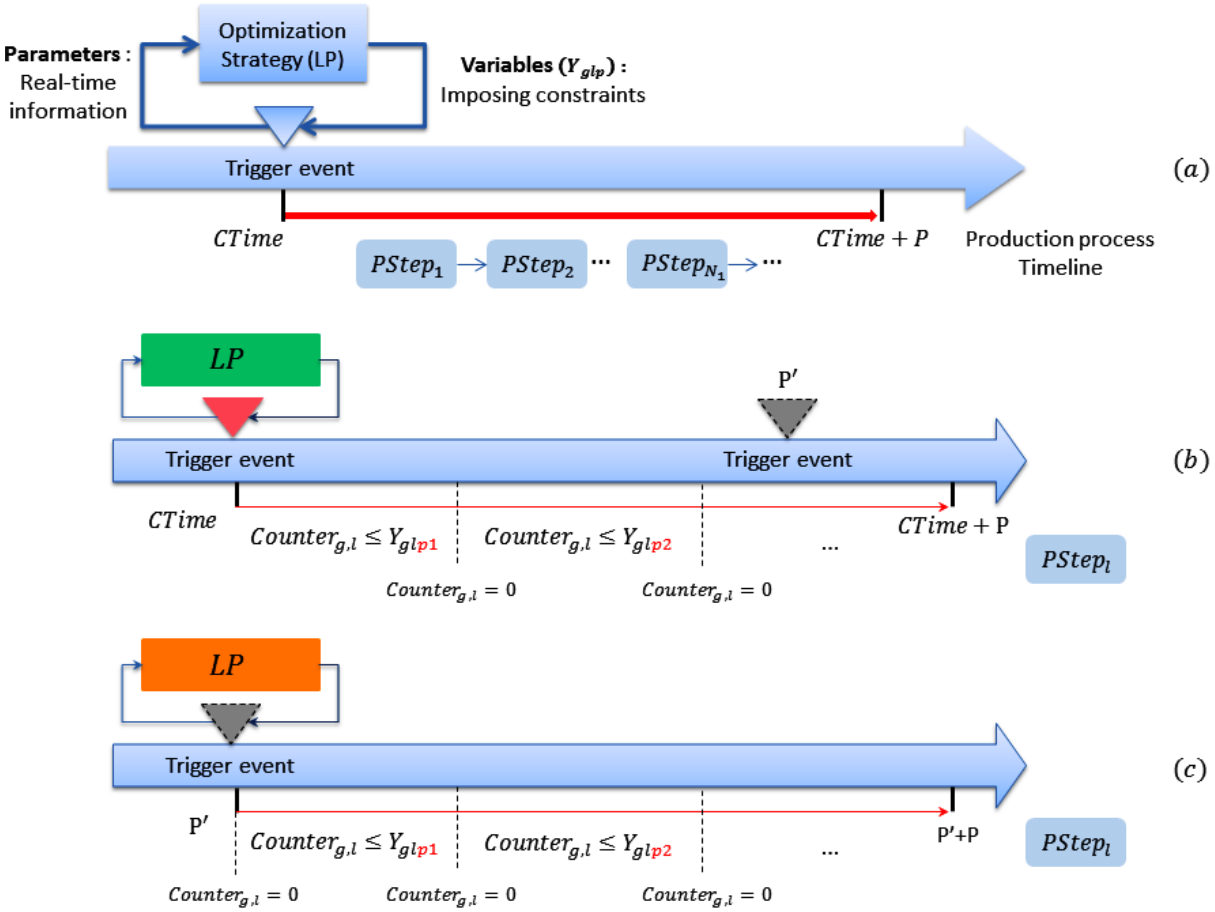

Figure 5.2: Applying the optimization based global strategy in a rolling horizon

process step l of route g must be processed on k,
P: Number of periods in the planning horizon,
p: Period index.

The parameters below are necessary:

ω_{glp}: Unit WIP holding cost at process step l of product in route g in period p,
IW_{gl}: Initial WIP in process step l of product in route g,
R_{gp}: Release quantity of products in route g in period p,
α_{gl}: Processing time for process step l of product in route g,
C_{kp}: Capacity of workcenter k in period p.

Note that, because this is a scheduling model and not a production planning model, release quantities in the fabs are given and not determined.

The following decision variables are used:

Y_{glp}: Quantity of products in route g completing process step l in period p,
X_{glp}: Quantity of products in route g arriving in process step l in period p,
W_{glp}: WIP of product in route g at process step l at the end of period p.

The base Linear Programming model is:

$$\min \sum_{g \in G} \sum_{l \in L(g)} \sum_{p=1}^{P} \omega_{glp} W_{glp} \tag{5.1}$$

Subject to:

$$X_{glp} = Y_{g(l-1)p} \qquad \forall g \in G, \forall l \in L(g), l \geq 2, \forall p \tag{5.2}$$

$$W_{g11} = IW_{g1} + R_{g1} - Y_{g11} \qquad \forall g \in G \tag{5.3}$$

$$W_{gl1} = IW_{gl} - Y_{gl1} \qquad \forall g \in G, \forall l \in L(g), l \geq 2 \tag{5.4}$$

$$W_{g1p} = W_{g1(p-1)} + R_{gp} - Y_{g1p} \qquad \forall g \in G, p = 2, \ldots, P \tag{5.5}$$

$$W_{glp} = W_{gl(p-1)} + X_{glp} - Y_{glp} \qquad \forall g \in G, \forall l \in L(g), l \geq 2, p = 2, \ldots, P \tag{5.6}$$

$$\sum_{(g,l) \in LK(k)} \alpha_{gl} Y_{glp} \leq C_{kp} \qquad \forall k \in K, p = 1, \ldots, P \tag{5.7}$$

$$W_{glp}, Y_{glp}, X_{glp} \geq 0 \qquad \forall g \in G, \forall l \in L(g), p = 1, \ldots, P \tag{5.8}$$

The objective function (5.1) aims at minimizing the WIP, which corresponds to pushing products to their last process step, i.e. minimizing cycle times. Note that ω_{glp} must be chosen such that $\omega_{glp} \leq \omega_{gl-1p}$, $\forall g \in G$, $\forall l \in L(g)$, $l \geq 2$, $\forall p = 1, \ldots, P$, i.e. the unit WIP holding cost is decreasing when the product is advancing in its route. Although this is not natural since the value of a product is actually increasing after each process step, ω_{glp} is chosen so that it is preferable to process products than keeping them in the inventory. Constraints 5.2 are the coupling constraints between consecutive process steps, i.e. the output of process step $l-1$, $Y_{g(l-1)p}$, is the input of the next process step l, X_{glp}. Constraints (5.3)-(5.6) are inventory balance equations linking the WIP of each product at each process step and in each period with the quantity completed in period p (Y variables) and the quantity arriving in period p (X variables). Constraints (5.3) correspond to the first process step in the first period where the initial WIP and the release quantity must be considered. Constraints (5.4) correspond to the remaining process steps in the first period where the initial WIP must be considered. Constraints (5.5) correspond to the first process step in the remaining periods where the release quantity must be considered. Constraints (5.6) correspond to the remaining process steps in the remaining periods. Constraints (5.7) are the capacity constraints. Constraints (5.8) are the non-negativity constraints.

5.2.2 Considering Time Constraints

The following additional parameters are used:

TC_g: Number of time constraints in route g ($TC_g = 0$

if g has no time constraint),
t: Time constraint index,
S_{gt}: First process step of time constraint t in route g ($t \leq TC_g$),
E_{gt}: Last process step of time constraint t in route g ($t \leq TC_g$),
MP_{gt}: Maximum number of periods for time constraint t in route g ($t \leq TC_g$).

The following constraints are added to the base Linear Programming model:

$$\sum_{p'=1}^{p+MP_{gt}} Y_{gE_{gt}p'} \geq \sum_{p'=1}^{p} Y_{gS_{gt}p'} \qquad \forall g \in G, \forall t \leq TC_g, p = 1, \ldots, P - MP_{gt} \tag{5.9}$$

Constraints (5.9) ensure that quantities of product in route g cannot remain longer than MP_{gt} periods for time constraint $t \leq TC_g$. More precisely, for a given route g and time constraint $t \leq TC_g$ and at each period p, the quantity of products in route g that completed the last process step E_{gt} of t from the first period to the end of the time constraint (period $p + MP_{gt}$), i.e. $\sum_{p'=1}^{p+MP_{gt}} Y_{gE_{gt}p'}$, is larger than or equal to the quantity of products in route g having started in the first process step S_{gt} of t from the first period to the start of the time constraint (period p), i.e. $\sum_{p'=1}^{p} Y_{gS_{gt}p'}$.

5.2.3 Optimizing Linearity

Additional parameters are required:

B: Number of blocks in a route,
b: Block index,
S_b: First process step of block b,
E_b: Last process step of block b,
MW_{gb}: Percentage of allowed WIP in block b of route g, $\sum_{b=1}^{B} MW_{gb} = 1$, $\forall g \in G$.

The following additional decision variables are used:

$WD_{b-1,b,p}$: WIP deviation between block $b-1$ and b (where $b \geq 2$) at the end of period p,
WD_{max}: Maximum WIP deviation between each pair of consecutive blocks.

One of the following objective functions is used:

$$\min \sum_{b=1;b>=2}^{B} \sum_{p=1}^{P} WD_{b,b-1,p} \tag{5.10}$$

or

$$\min WD_{max} \tag{5.11}$$

The objective function is (5.10) if one wants to minimize the sum of WIP deviations, and (5.11) if one wants to minimize the maximum deviation.

The following constraints are added to the base Linear Programming model: :

$$WD_{b-1,b,p} \geq \sum_{g \in G} (\sum_{l=S_{b-1}}^{E_{b-1}} W_{glp} - \sum_{l=S_b}^{E_b} W_{glp}) \qquad \forall b \geq 2, \forall p \tag{5.12}$$

$$WD_{b-1,b,p} \geq \sum_{g \in G} (\sum_{l=S_b}^{E_b} W_{glp} - \sum_{l=S_{b-1}}^{E_{b-1}} W_{glp}) \qquad \forall b \geq 2, \forall p \tag{5.13}$$

$$WD_{max} \geq WD_{b-1,b,p} \qquad b \in B(g), b \geq 2, \forall p \tag{5.14}$$

$$WD_{b-1,b,p} \geq 0 \qquad \forall b \in B(g), b \geq 2, \forall p \tag{5.15}$$

$$WD_{max} \geq 0 \tag{5.16}$$

Constraints (5.12) and (5.13) are used to compute $WD_{b-1,b,p}$, the WIP deviation between blocks $b-1$ and b at the end of period p. Note that $WD_{b-1,b,p}$ is computed using Constraints (5.12), respectively Constraints (5.13), if the WIP in block $b-1$ is larger than the WIP in block b, respectively if the WIP in block b is larger than the WIP in block $b-1$. WD_{max} is determined through Constraints (5.14).

5.3 Experimental Design

In the global strategy, the quantities of products to complete in each process step and in each period of the planning horizon are defined as control parameters. These quantities optimized using the models presented in Section 5.2 become objective that we want the system to produce and constraints to satisfy at the local level.

Figure 5.3 provides a general overview of how the simulation is performed. The steps in red correspond to the call of the optimization approach, applied periodically, that updates the control parameters. When the optimization approach is applied, historical and actual information are first collected and then fed into the LP model. The solution of the LP model, variables Y_{glp}, imposes objectives in each process step and in each period of the next planning period in the simulation run.

The standard solver IBM ILOG CPLEX is used to solve the optimization model. To ensure a smooth running of the simulation, it was necessary to integrate IBM ILOG CPLEX in AnyLogic. This was possible because IBM ILOG CPLEX provides a set of Java class libraries allowing the IBM ILOG CPLEX optimizer to be embedded in Java applications. The file that contains the libraries is cplex.jar, and the packages ilog.concert and ilog.cplex have to be imported in the application. Because AnyLogic is based on Java, we imported the IBM ILOG CPLEX libraries in the simulation model and then constructed and solved the optimization models in AnyLogic (see Figure 5.4).

Once IBM ILOG CPLEX is integrated in AnyLogic, optimization models can be constructed and solved in the simulation model. Since AnyLogic allows the users to create their own Java classes with any required functionality, we followed the IBM ILOG CPLEX documentation for Java to create optimization models. For example, a Java Class is created for the base linear programming model and is called *LP_Basic*. This class contains a method called *model* which constructs the model and then solves it (see Figure 5.5).

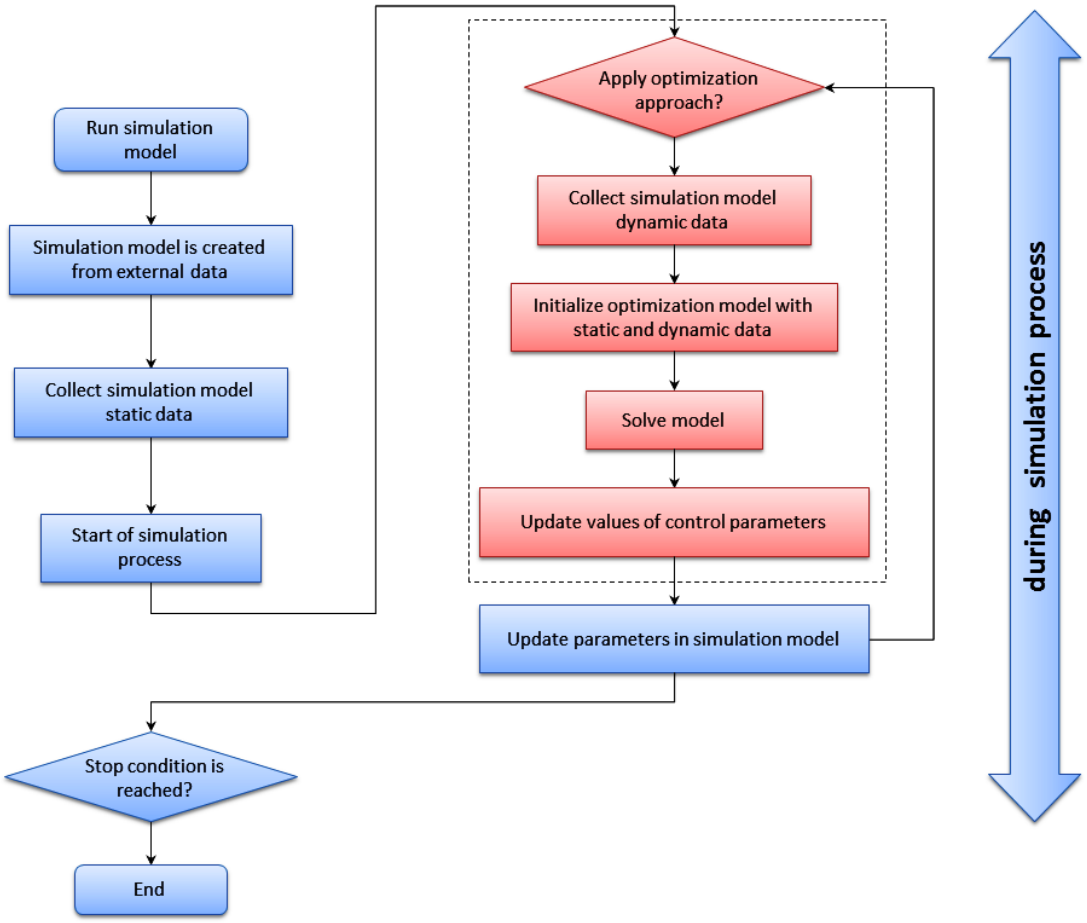

Figure 5.3: General overview of the simulation

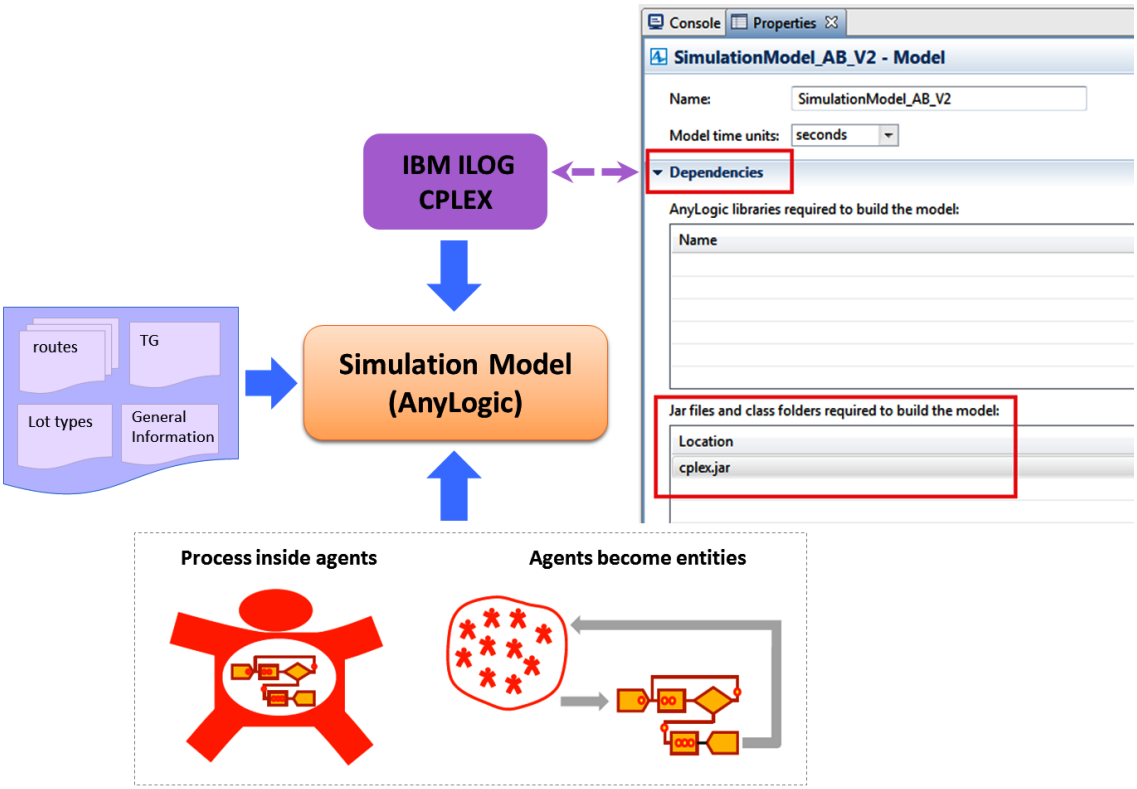

Figure 5.4: Integrating AnyLogic with IBM ILOG CPLEX

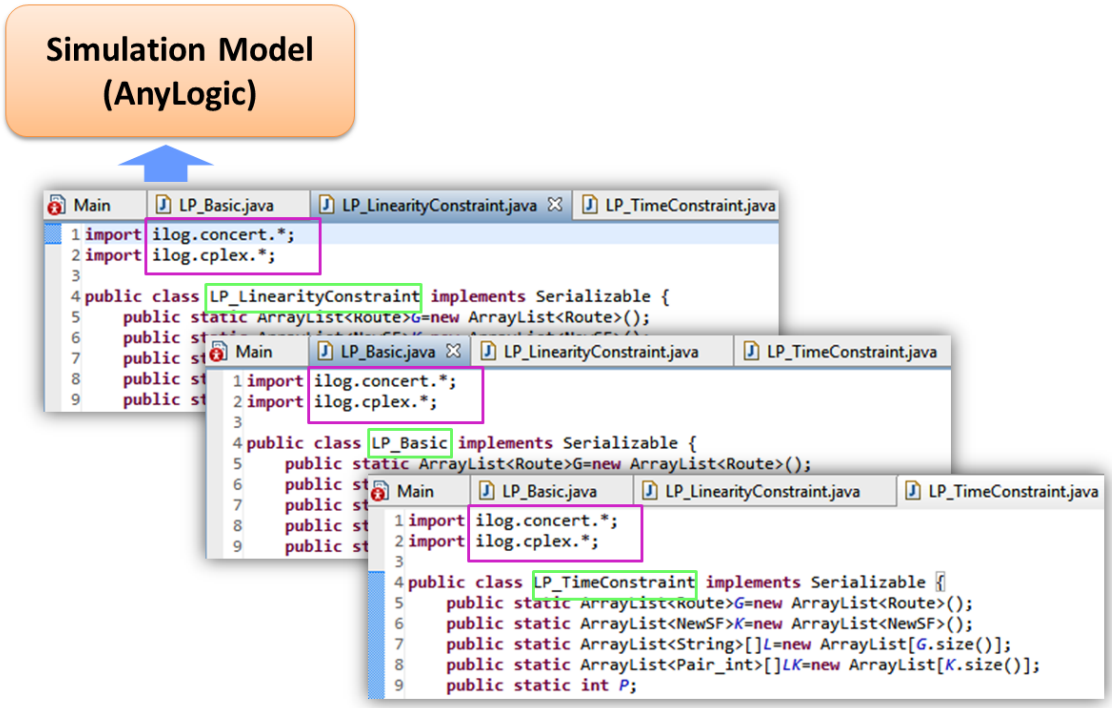

Figure 5.5: Integrating optimization algorithms with AnyLogic

In the optimal solution determined by IBM ILOG CPLEX, the variables Y, associated to the quantity of products completed for each route and process step in each period, are fed in the simulation model so that the control parameters are updated before the simulation continues.

In order to guide dispatching decisions at the local level so that the optimal values of Y_{glp} are considered as objectives, a variable $counter_{gl}$ of type Integer is defined for each process step. Every time a product of type g is performed in process step l in period p, the value of $counter_{gl}$ is increased. The dispatching rule in Algorithm 4.6 is modified so that only lots such that $counter_{gl} < Y_{glp}$ in front of a tool group are considered for dispatching. If $counter_{gl} \leq Y_{glp}$, $\forall g$, $\forall l$, for all lots waiting in front of a tool group, then all lots are considered for dispatching.

5.4 Computational Experiments

5.4.1 Industrial instances

Table 5.1 provides the main characteristics of the products in the industrial instance we are using. There are three products, whose number of steps in the route ranges from 352 to 501. In the simulation, 260 lots are planned to be released and processed in the fab per week. The release horizon is set to 15 weeks. Among the lots released, 40% of lots are of product type 1, 40% of product type 2, and 20% of product type 3. The total number of lots to be released for each type of product during the specified horizon is given in Table 5.1.

Table 5.1: Characteristics of the products

Product Type	Nb. Process Steps	Per.of Lots	Nb. Lots
Type 1	501	40%	1560
Type 2	440	40%	1560
Type 3	352	20%	780

5.4.2 Impact on average cycle times

The objective of the experimental study is to compare the average cycle times of different types of products with and without applying the optimization algorithm in the production process. We run the simulation model with several scenarios to analyze what happens if the optimization algorithm is applied. Table 5.2 provides the details of the scenarios.

Scenario 1: Run the simulation model without applying the optimization algorithm.

Scenario 2: Run the simulation model when applying the optimization algorithm in a rolling horizon. The length of the planning horizon is 15. The optimization algorithm is applied during the simulation run at periodic intervals where the length of the time interval is 10. The time unit is 3 hours.

The result shows that, in scenario 2, although the cycle time of product type 3 is increased compared to scenario 1, the cycle times of product type 1 and type 2 are decreased. In addition, the average cycle time in scenario 2 is lower than in scenario 1 (basic simulation run).

In order to analyze the performance of the optimization algorithm, three more scenarios are set up and run. The details of these scenarios are similar to scenario 2, but vary in the priority of lots depending on the product types. The production process of lots with higher priority is accelerated during the simulation run. *Weight* in Table 5.2 represents the priority of each type of product.

Scenario 3: Run the simulation model when applying the optimization algorithm where products types 1 and 2 have the same priority and the priority of product type 3 is increased.

Scenario 4: Run the simulation model when applying the optimization algorithm where product type 3 has the highest priority and product type 2 has the lowest priority.

> **Scenario 5**: Run the simulation model when applying the optimization algorithm where products types 2 and 3 have the same priority and the priority of product type 1 is increased.

The result shows that, by increasing the priority of products, corresponding cycle times are decreased. Moreover, average cycle times in all scenarios in which the optimization algorithm is applied, is lower than in the basic simulation run.

Table 5.2: Performance of optimization algorithm (LP) on first industrial instance

	Scenario1 (Simulation)		**Scenario2** (Simulation+LP)	
ProductType	Weight	CycleTime	Weight	CycleTime
1	1	33.10	1	28.18
2	1	30.47	1	**19.95**
3	1	25.49	1	42.72
Average Cycle Time		30.52		**27.79**

	Scenario3 (Simulation+LP)		**Scenario4** (Simulation+LP)		**Scenario5** (Simulation+LP)	
ProductType	Weight	CycleTime	Weight	CycleTime	Weight	CycleTime
1	1	38.77	2	19.27	2	**10.80**
2	1	27.83	1	51.44	1	40.33
3	3	**8.51**	3	**7.98**	1	38.90
Average Cycle Time		28.34		29.88		28.23

5.4.3 Considering Time Constraints

In these experiments, we consider the products presented in Table 5.1 with modified percentages of the lots released into the fab. The new characteristics of the products are presented in Table 5.3.

Table 5.3: Characteristic of the products when the percentage of released lots is modified

Product Type	Nb. Process Steps	Percent. Lots	Nb. Lots
Type 1	501	35%	1365
Type 2	440	35%	1365
Type 3	352	30%	1170

The objective of these experiments is to analyze the performance of the optimization algorithm in controlling time constraints. To this aim, various time constraints are set up for product type 1. The general information of each time constraint is presented in Table 5.4.

Table 5.4: Time constraints for product type 1

Time constraint	Nb. covered Steps	Maximum Time (Min.)
TC 1	27	550
TC 2	50	1370

In order to analyze the performance of the optimization algorithm on controlling time constraints, three scenarios are set up and run.

Scenario 1: Run the simulation model without applying the optimization algorithm.

Scenario 2: Run the simulation model when applying the optimization algorithm (base Linear Programming model) in a rolling horizon.

Scenario 3: Run the simulation model when applying the base linear programming model and considering time constraints.

The simulation model is executed with three scenarios. The results are shown in three tables. Table 5.4 compares the cycle times of the products in three scenarios. Table 5.5 and Table 5.6 respectively compare the percentage of lots violating time constraints and the average time deviation from time constraints in the scenarios. Although cycle times are not improved by applying optimization algorithms, time constraints are much better satisfied. The results in Table 5.6 show that, even by applying the base linear programming model (Scenario 2), the percentage of lots violating time constraints is decreased. The results is improved by considering time constraints in the base linear programming model (Scenario 3).

Table 5.5: Performance of optimization algorithm on cycle times

	Scenario1 (Simulation)		**Scenario2** (Simulation+LP)		**Scenario3** (Simulation+LP+TCs)	
ProductType	Weight	CycleTime	Weight	CycleTime	Weight	CycleTime
1	1	30.6	1	30.6	1	30.6
2	1	29.4	1	29.4	1	29.4
3	1	32.3	1	32.3	1	32.2
Average Cycle Time		30.7		30.7		30.7

Table 5.6: Performance of optimization algorithm on percentage of lots violating time constraints

	Percentage of lots violating time constraints		
	Scenario1 (Simulation)	**Scenario2** (Simulation+LP)	**Scenario3** (Simulation+LP+TCs)
TC 1	28.20%	23.34%	19.94%
TC 2	29.45%	21.30%	21.20%

When considering the average time deviation from time constraints in Table 5.7, the results show that very large improvements are also obtained for all time constraints.

Table 5.7: Performance of optimization algorithm on average time deviation from time constraints

	Average time deviation from time constraints (Min.)		
	Scenario1 (Simulation)	**Scenario2** (Simulation+LP)	**Scenario3** (Simulation+LP+TCs)
TC 1	190.13	21.95	19.23
TC 2	204.40	38.19	36.35

5.5 Conclusion and Perspectives

In this chapter, an optimization (Linear Programming) model to minimize cycle times is first presented to take decisions at the global scheduling level within the framework proposed in Chapter 4. These decisions are used as inputs at the local scheduling level. Two extensions of the model are proposed: the first one is related to ensuring the satisfaction of time constraints and the second extension aims at considering the linearity objective.

Numerical experiments on instances constructed from industrial data and using the initial model and the first extension associated to satisfying time constraints are presented and discussed. The results show that the global scheduling level guides the local scheduling level to ensure that the global objectives are met.

The perspectives of this work, that will be discussed in more details in the last chapter of this thesis, include experimenting on the extension of the model that optimizes the linearity, but also and more importantly considering multiple objectives simultaneously.

Chapter 6

Conclusion and Perspectives

In this thesis, a general framework to ensure the consistency between global and local scheduling decisions in a semiconductor manufacturing facility has been developed. A multi-method generic simulation model has been developed to evaluate the performance of the proposed framework. The simulation model can be applied to a wide range of problems in semiconductor manufacturing. In addition, the problem of managing time constraints as critical constraints in wafer fabrication is studied and various approaches have been developed.

6.1 Managing time constraints

The first part of this thesis studies the management of time constraints between production steps in wafer fabrication as a critical constraint to ensure the quality of the final product. We propose three approaches to prevent lot scraps and reprocessing: WIPMax calculation approach, Deterministic approach, and Probability estimation approach. Since a time constraint limits the sum of the waiting times and the processing times between two process steps, the first approach is based on the concept of capacity and estimates the maximum allowed number of lots (WIPMax) within a time constraint. A time constraint is satisfied by a given lot, if the current number of lots in the time constraint is lower than the estimated WIPMax. The main drawback of this approach is that the impact of the position of lots in the time constraints is ignored in the computation of WIPMax. In order to deal with this limitation, we propose a deterministic approach which estimates whether a given lot can respect time constraints by considering the current status of the fab such as the current position of lots and the current status of machines. This approach is composed of three steps: Modeling the problem through a disjunctive graph, scheduling lots using a list scheduling algorithm, and evaluating time constraints. Using this approach, the production times of a given lot in time constraints are estimated and then compared with the maximum allowed times of time constraints. The advantage of the approach is its low computational complexity which can handle problems of industrial size. Multiple time constraints even with overlapping can be evaluated using this approach. However, the drawback of this approach is that only one schedule is computed to estimate the production time of a given lot in the time constraints. Since the way of dispatching and scheduling lots on machines has a significant impact on the production time of the given lot; the probability estimation approach is proposed to consider many schedules. It is an extended version of the deterministic approach that the second step is applied multiple times with a random dispatching rule. The outcome of this approach is a probability of time constraint satisfaction. Numerical experiments on industrial instances shows the advantage of the probability estimation approach compared to the deterministic approach.

One of the main advantages of the probability estimation approach is its ability to extract relevant information to support decisions and to identify the root causes of violations. Some examples are: Deviation from time constraints, critical process steps, and bottleneck machines. In addition, the approach can be extended to evaluate multiple lots before time constraints.

A recommendation for future research is to estimate the right number of runs in the probability estimation approach to balance between the computational time and the quality of the results. Analyzing and managing time constraints to support decisions to allow lots to enter time constraints based on the computed probability is one of the main goals of our future research. Another interesting future direction is to perform experiments on industrial instances to evaluate multiple lots before time constraints. And the last future direction is to show the application of identifying critical steps and bottleneck machines, and to estimate the deviation of time constraints on industrial problems.

6.2 Implementing a flexible simulation model

The second part of this thesis studied the development of a flexible simulation model for a semiconductor manufacturing facility. The main purpose of the simulation model is to evaluate the performance of the proposed approaches in this thesis.

We focus on the conceptual model process in the simulation study. Conceptual modelling is the process of abstracting a simulation model from the real world system without considering the impact of computer programming. We present a new conceptual modelling framework which is a revised version of the frameworks presented by Robinson (2011) [69] and Furian et al. (2015) [21] which are used for discrete event systems. Our conceptual model is generic and can be applied for any type of systems because the process of selecting an appropriate simulation method depending on the system is placed through the process of conceptual modelling. A combination of discrete event method and agent based method is selected to model a semiconductor manufacturing system. A data-driven generic simulation model

has been developed for complex semiconductor manufacturing facilities.

One of the main advantages of our simulation model is its flexibility and extendibility. Our simulation model is automatically generated from external files and thus can be applied to a range of semiconductor facilities which have structural similarities. In addition, it can be extended to a wide range of problems in semiconductor manufacturing.

It would of great interest in the future to consider batching machines, machine breakdown, and to integrate the transportation system in the simulation model. To extend the developed simulation model with required information, we just need to follow the process of conceptual modelling. For example, to consider batching machines, it is necessary to go through the model content phase of the conceptual model and add two extra attributes to the class *Tool*. The first attribute is of type Boolean which indicates whether the tool is batching. The second attribute identifies the batch size. New dispatching methods must be developed to decide which lots are grouped in the same batch.

6.3 Ensuring the consistency of global and local scheduling decisions

The third part of the thesis studied the consistency of global and local scheduling decisions in semiconductor manufacturing which is the core of this thesis. We present a general framework which aims at supporting and controlling local decisions by considering global objectives and information. In fact, the framework guides the production process towards achieving global objectives. The framework contains three main processes: Monitoring, Evaluating/Optimizing, and Controlling. The monitoring process extracts relevant information from the past and ongoing activities at the global level. The evaluating/optimizing process is designed to analyze the production process according to the information provided by the monitoring process. It determines whether the production process at the local level is going in the right direction to achieve global objectives and according to the analysis improves the local strategies. The controlling process feeds the information

provided by the evaluating/optimizing process at the local level. The evaluating/optimizing process is the core of the framework. We define a global strategy to perform analysis which can be an evaluation-based algorithm or an optimization-based algorithm. The framework runs periodically during the production process. In Chapter 4, we propose an evaluation-based global strategy which aims at improving the production process with no guarantee for the optimal solution. Two global objectives are studied: Management of time constraints and control of linearity constraints. For each objective, we propose an evaluation-based global strategy. The simulation model developed in the second part of the thesis is used to evaluate the performance of the framework. Simulation experiments are performed to evaluate the evaluation-based algorithm which considered the management of time constraints as the global objective. In chapter 5, we propose three optimization models as optimization-based global strategies: A base Linear Programming model which aims at minimizing cycle times, an extension of the model that considers time constraints, and another extension of the model that considers linearity constraint. Simulation experiments are performed to evaluate the performance of the two first linear programming models. The results show that the global strategies guide the local scheduling level to ensure that the global objectives are met.

A key challenge for future research is to conduct experiments on the evaluation-based global strategy that considers linearity constraint as the global objective and on the extension of the linear programming model that optimizes linearity. In this thesis, linearity constraints are expressed as: Smoothing differences that could appear between the activity of a block and its fixed target. In this study, routes are divided into a specified number of blocks where activities within a block are considered as WIP level to avoid WIP accumulation in the fab and cycle times of individual lots to avoid consuming long processing times within some portions of the route. In order to evaluate the proposed global strategies to optimize linearity, WIP levels and cycle times of block must be given as input parameters. Since these parameters depend on the number of routes considered in the experiments and also on the number of released lots in the fab, determining these parameters is challenging. A possible way if by conducting simulation runs (without ap-

plying the framework) and then estimating the parameters according to the results. These estimations can be used as input parameters and the framework can then be applied during the simulation run to evaluate the proposed global strategies.

Another interesting future direction is to study the right duration for the periods in which the framework is applied. Extending the global strategies by considering multiple global objective and extra information such as batching machines, machines breakdowns, lot release policies is also set as a future research direction.

Bibliography

[1] From sand to silicon making of a chip illustrations. 2009. (cited on pages 9 and 6)

[2] R. Anthony. *Planning and control systems: A framework for analysis*. Harvard University, Boston, 1965. (cited on page 8)

[3] S. Attar, M. Mohammadi, R. Tavakkoli-Moghaddam, and S. Yaghoubi. Hybrid flexible flowshop scheduling problem with unrelated parallel machines and limited waiting times. In *The International Journal of Advanced Manufacturing Technology*, volume 68, page 1583–1599, 2013. (cited on page 20)

[4] M. Bartusch, R. H. Möhring, and F. J. Radermacher. Scheduling project networks with resource constraints and time windows. *Annals of operations Research*, 16(1):199–240, 1988. (cited on page 20)

[5] A. Bitar. Ordonnancement sur machines parallèles appliqué à la fabrication de semi-conducteurs : Atelier de photolithographie. 2015. (cited on page 13)

[6] A. Borshchev. *The big book of simulation modeling: multimethod modeling with AnyLogic 6*. AnyLogic North America, 2013. (cited on pages 10, 86 and 88)

[7] V. Botta-Genoulaz. Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness. *International Journal of Production Economics*, 64(1):101–111, 2000. (cited on page 20)

[8] M. Bureau, S. Dauzère-Pérès, C. Yugma, L. Vermariën, and J.-B. Maria. Simulation results and formalism for global-local scheduling in semiconductor manufacturing facilities. In *Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come*, pages 1768–1773. IEEE Press, 2007. (cited on pages 127 and 134)

[9] A. Caumonda and N. Lacommea, P.and Tcherney. A memetic algorithm for the job-shop with time-lags. In *Computers and Operations Research*, volume 35, page 2331–2356, 2008. (cited on page 19)

[10] L. Cho, J. Park, H.M.and Ryan, and T. Sharkey. Production scheduling with queue-time constraints : Alternative formulations. In *Proceedings of the 2014 Industrial and System Engineering Research Conference*, 2014. (cited on page 21)

[11] R. M. Dabbas and J. W. Fowler. A new scheduling approach using combined dispatching criteria in wafer fabs. *IEEE Transactions on semiconductor manufacturing*, 16(3):501–510, 2003. (cited on page 126)

[12] S. Dauzère-Péres and J. B. Lasserre. *An integrated approach in production planning and scheduling*, volume 411. Springer Science & Business Media, 2012. (cited on page 120)

[13] F. Deppner and M. Portmann. A hybrid decomposition approach using increasing clusters for solving scheduling problems with minimal and maximal time lags. In *In Proceedings Tenth International Workshop on Project Management and Scheduling (PMS 2006)*, pages 69–73, 2006. (cited on page 19)

[14] E. Dhouib, J. Teghem, and T. Loukil. Minimizing the number of tardy jobs in a permutation flowshop scheduling problem with setup times and time lags constraints. *Journal of Mathematical Modelling and Algorithms in Operations Research*, 12(1):85–99, 2013. (cited on page 20)

[15] U. Dorndorf, E. Pesch, and T. Phan-Huy. A time-oriented branch-and-bound algorithm for resource-constrained project scheduling with generalised precedence constraints. *Management Science*, 46(10):1365–1384, 2000. (cited on page 20)

[16] G. Fishman. *Discrete-event simulation: modeling, programming, and analysis*. Springer Science & Business Media, 2013. (cited on page 87)

[17] J. Fondrevelle, A. Oulamara, and M. Portmann. Permutation flowshop scheduling problems with time lags to minimize the weighted sum of

machine completion times. In *International Journal of Production Economics*, volume 112, pages 168–176, 2008. (cited on pages 16 and 20)

[18] J. Fondrevelle, A. Oulamara, and M.-C. Portmann. Permutation flowshop scheduling problems with maximal and minimal time lags. *Computers & Operations Research*, 33(6):1540–1556, 2006. (cited on page 20)

[19] J. W. Fowler and O. Rose. Grand challenges in modeling and simulation of complex manufacturing systems. *Simulation*, 80(9):469–476, 2004. (cited on page 62)

[20] M. Fowler. *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Professional, 2004. (cited on page 70)

[21] N. Furian, M. O'sullivan, C. Walker, S. Vössner, and D. Neubacher. A conceptual modeling framework for discrete event simulation using hierarchical control structures. *Simulation Modelling Practice and Theory*, 56:82–96, 2015. (cited on pages 10, 66, 67, 68, 69 and 175)

[22] A. K. Gupta and A. I. Sivakumar. Simulation based multiobjective schedule optimization in semiconductor manufacturing. In *Simulation Conference, 2002. Proceedings of the Winter*, volume 2, pages 1862–1870. IEEE, 2002. (cited on page 63)

[23] A. C. Hax and H. C. Meal. Hierarchical integration of production planning and scheduling. 1973. (cited on page 8)

[24] J. Hurink and J. Keuchel. Local search algorithms for a single-machine scheduling problem with positive and negative time-lags. *Discrete Applied Mathematics*, 112(1):179–197, 2001. (cited on page 21)

[25] B.-J. Joo and Y.-D. Kim. A branch-and-bound algorithm for a two-machine flowshop scheduling problem with limited waiting time constraints. *journal of the Operational Research Society*, 60(4):572–582, 2009. (cited on page 20)

[26] A. A. Juan, J. Faulin, S. E. Grasman, M. Rabe, and G. Figueira. A review of simheuristics: Extending metaheuristics to deal with stochastic

combinatorial optimization problems. *Operations Research Perspectives*, 2:62–72, 2015. (cited on page 61)

[27] C. Jung, D. Pabst, M. Ham, M. Stehli, and M. Rothe. An effective problem decomposition method for scheduling of diffusion processes based on mixed integer linear programming. *IEEE Transactions on Semiconductor Manufacturing*, 27(3):357–363, 2014. (cited on page 12)

[28] C. Jung, D. Pabst, and M. Stehli. An effective problem decomposition method for scheduling of diffusion processes based on mixed integer linear programming. In *Advanced Semiconductor Manufacturing Conference (ASMC)*, pages 35–40, 2013. (cited on page 20)

[29] R. Keeney. Value-focused thinking. *Journal of Quality*, 15(6):409–423, 1992. (cited on page 71)

[30] Y.-D. Kim, B.-J. Joo, and S.-Y. Choi. Scheduling wafer lots on diffusion machines in a semiconductor wafer fabrication facility. *IEEE Transactions on Semiconductor Manufacturing*, 23(2):246–254, 2010. (cited on page 12)

[31] S. Kitamura, K. Mori, and A. Ono. Capacity planning method for semiconductor fab with time constraints between operations. In *SICE-ICASE International Joint Conference*, pages 1100–1103, 2006. (cited on page 22)

[32] A. Klemmt and L. Monch. Scheduling Jobs with Time Constraints between Consecutive Process Steps in Semiconductor Manufacturing. In *Proceedings of the 2012 Winter Simulation Conference*, pages 2173–2182, 2012. (cited on pages 17 and 20)

[33] S. Knopp, S. Dauzère-Pérès, and C. Yugma. A batch-oblivious approach for complex job-shop scheduling problems. *European Journal of Operational Research*, 2017. (cited on page 12)

[34] R. Kohn, O. Rose, and C. Laroque. Study on multi-objective optimization for parallel batch machine scheduling using variable neighbourhood

search. In *Proceedings of the 2013 Winter Simulation Conference*, pages 3654–3670, 2013. (cited on page 20)

[35] P. Kumar. Scheduling semiconductor manufacturing plants. *IEEE Control Systems*, 14(6):33–40, 1994. (cited on page 6)

[36] C.-J. Kuo, C.-M. Liu, and C.-Y. Chi. Standard wip determination and wip balance control with time constraints in semiconductor wafer fabrication. *Journal of Quality*, 15(6):409–423, 2008. (cited on page 22)

[37] Y. Lee, Z. Jiang, and H. Liu. Multiple-objective scheduling and real-time dispatching for the semiconductor manufacturing system. *Computers & Operations Research*, 36(3):866–884, 2009. (cited on page 127)

[38] Y.-Y. Lee, C. Chen, and C. Wu. Reaction chain of process queue time quality control. In *Semiconductor Manufacturing, 2005. ISSM 2005, IEEE International Symposium on*, pages 47–50. IEEE, 2005. (cited on page 21)

[39] L. Li, Y. Li, and Z. Sun. Dispatching rule considering time-constraints on processes for semiconductor wafer fabrication facility. In *Automation Science and Engineering (CASE), 2012 IEEE International Conference*, volume 2, pages 407–412, 2012. (cited on page 127)

[40] T. Li and Y. Li. Constructive backtracking heuristic for hybrid flowshop scheduling with limited waiting times. In *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, pages 6671–6674, 2007. (cited on page 20)

[41] Y. Li, Z. Jiang, and W. Jia. A pull vpls based release policy and dispatching rule for semiconductor wafer fabrication. In *Automation Science and Engineering (CASE), 2012 IEEE International Conference on*, pages 396–400. IEEE, 2012. (cited on page 127)

[42] J. Lin and Q. Long. Development of a multi-agent-based distributed simulation platform for semiconductor manufacturing. *Expert systems with applications*, 38(5):5231–5239, 2011. (cited on page 63)

[43] J. T. Lin and C.-M. Chen. Simulation optimization approach for hybrid flow shop scheduling problem in semiconductor back-end manufacturing. *Simulation Modelling Practice and Theory*, 51:100–114, 2015. (cited on page 63)

[44] J. T. Lin and C.-J. Huang. A simulation-based optimization approach for a semiconductor photobay with automated material handling system. *Simulation Modelling Practice and Theory*, 46:76–100, 2014. (cited on page 63)

[45] S. Liu, J. Cui, and Y. Li. Heuristic tabu algorithm for hybrid flowshop scheduling with limited waiting time. In *International Symposium on Computational Intelligence and Design*, volume 2, pages 233–237, 2008. (cited on page 20)

[46] M. Y. H. Low, K. W. Lye, P. Lendermann, S. J. Turner, R. T. W. Chim, and S. H. Leo. An agent-based approach for managing symbiotic simulation of semiconductor assembly and test operation. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 85–92. ACM, 2005. (cited on page 63)

[47] S. C. Lu, D. Ramaswamy, and P. Kumar. Efficient scheduling policies to reduce mean and variance of cycle-time in semiconductor manufacturing plants. *IEEE Transactions on Semiconductor Manufacturing*, 7(3):374–388, 1994. (cited on page 126)

[48] R. Maidstone. Discrete event simulation, system dynamics and agent based simulation: Discussion and comparison. *System*, 1(6), 2012. (cited on page 87)

[49] A. Maria. Introduction to modeling and simulation. In *Proceedings of the 29th conference on Winter simulation*, pages 7–13. IEEE Computer Society, 1997. (cited on page 68)

[50] L. Mönch and R. Drießel. A distributed shifting bottleneck heuristic for complex job shops. *Computers & Industrial Engineering*, 49(3):363–380, 2005. (cited on page 121)

[51] L. Monch, J. Fowler, and S. Mason. Production planning and control for semiconductor wafer fabrication facilities, 2013. (cited on page 11)

[52] L. Mönch, J. W. Fowler, S. Dauzère-Pérès, S. J. Mason, and O. Rose. A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. *Journal of Scheduling*, 14(6):583–599, 2011. (cited on pages 9, 5 and 127)

[53] L. Mönch, R. Schabacker, D. Pabst, and J. W. Fowler. Genetic algorithm-based subproblem solution procedures for a modified shifting bottleneck heuristic for complex job shops. *European Journal of Operational Research*, 177(3):2100–2118, 2007. (cited on page 121)

[54] A. Negahban and J. S. Smith. Simulation for manufacturing system design and operation: Literature review and analysis. *Journal of Manufacturing Systems*, 33(2):241–261, 2014. (cited on page 62)

[55] A.-T. Nguyen, S. Reiter, and P. Rigo. A review on simulation-based optimization methods applied to building performance analysis. *Applied Energy*, 113:1043–1058, 2014. (cited on page 61)

[56] K. Nonobe and T. Ibaraki. A metaheuristic approach to the resource constrained project scheduling with variable activity durations and convex cost functions. In *Perspectives in Modern Project Scheduling*, pages 225–248. Springer, 2006. (cited on page 21)

[57] J. J. Odell, H. V. D. Parunak, and B. Bauer. Representing agent interaction protocols in uml. In *Agent-oriented software engineering*, pages 121–140. Springer, 2001. (cited on pages 10, 75 and 78)

[58] S. Onggo. Methods for conceptual model representation. *Conceptual modelling for discrete-event simulation*, pages 337–354, 2010. (cited on page 70)

[59] M. E. Pfund, H. Balasubramanian, J. W. Fowler, S. J. Mason, and O. Rose. A multi-criteria approach for scheduling semiconductor wafer fabrication facilities. *Journal of Scheduling*, 11(1):29–47, 2008. (cited on page 13)

[60] M. Pidd. Guidelines for the design of data driven generic simulators for specific domains. *Simulation*, 59(4):237–243, 1992. (cited on page 63)

[61] T. Ponsignon and L. Mönch. Heuristic approaches for master planning in semiconductor manufacturing. *Computers & Operations Research*, 39(3):479–491, 2012. (cited on page 11)

[62] W. H. Raaymakers and J. Hoogeveen. Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing. *European Journal of Operational Research*, 126(1):131–151, 2000. (cited on page 21)

[63] H. Richter and L. Marz. Toward a standard process: The use of uml for designing simulation models. In *Simulation Conference, 2000. Proceedings. Winter*, volume 1, pages 394–398. IEEE, 2000. (cited on page 70)

[64] J. Robinson and R. Giglio. Capacity planning method for semiconductor fab with time constraints between operations. In *Proceedings of the 1999 Winter Simulation Conference*, pages 880–887, 1999. (cited on page 21)

[65] J. K. Robinson. *Capacity planning in a semiconductor wafer fabrication facility with time constraints between process steps.* PhD thesis, University of Massachusetts Amherst, 1998. (cited on page 21)

[66] S. Robinson. *Simulation: the practice of model development and use.* Chichester, UK: Wiley, 2004. (cited on pages 10 and 65)

[67] S. Robinson. Conceptual modeling for simulation: issues and research requirements. In *Proceedings of the 38th conference on Winter simulation*, pages 792–800. Winter Simulation Conference, 2006. (cited on page 65)

[68] S. Robinson. Conceptual modelling for simulation part ii: a framework for conceptual modelling. *Journal of the Operational Research Society*, 59(3):291–304, 2008. (cited on page 68)

[69] S. Robinson, R. Brooks, K. Kotiadis, and D.-J. V. Der Zee. *Conceptual modeling for discrete-event simulation.* CRC Press, 2010. (cited on pages 10, 66, 67, 68 and 175)

[70] J. Rohde, H. Meyr, M. Wagner, et al. Die supply chain planning matrix. Technical report, Darmstadt Technical University, Department of Business Administration, Economics and Law, Institute for Business Studies (BWL), 2000. (cited on pages 9 and 10)

[71] F. Rossi, A. Sperduti, K. B. Venable, L. Khatib, P. Morris, and R. Morris. Learning and solving soft temporal constraints: An experimental study. In *International Conference on Principles and Practice of Constraint Programming*, pages 249–264. Springer, 2002. (cited on page 21)

[72] R. Sadeghi, S. Dauzère-Pérès, C. Yugma, and G. Lepelletier. Production control in semiconductor manufacturing with time constraints. In *Advanced Semiconductor Manufacturing Conference (ASMC), 2015 26th Annual SEMI*, pages 29–33. IEEE, 2015. (cited on page 150)

[73] W. Scholl and J. Domaschke. Implementation of Modeling and simulation in semiconductor wafer fabrication with time constraints between wet etch and furnace operations. In *IEEE Transactions on Semiconductor Manufacturing*, volume 13, pages 273–277, 2000. (cited on page 19)

[74] C. Schwindt and N. Trautmann. Batch scheduling in process industries: an application of resource–constrained project scheduling. *OR Spectrum*, 22(4):501–524, 2000. (cited on page 21)

[75] R. Siegfried. *Modeling and simulation of complex systems: A framework for efficient agent-based modeling and simulation*. Springer, 2014. (cited on page 87)

[76] K. Sourirajan and R. Uzsoy. Hybrid decomposition heuristics for solving large-scale scheduling problems in semiconductor wafer fabrication. *Journal of Scheduling*, 10(1):41–65, 2007. (cited on page 121)

[77] H. Stadtler and C. Kilger. Supply chain management and advanced planning. 2000. (cited on page 8)

[78] L. Su. Ahybrid two-stage flowshop with limited waiting time constraints. In *Computers and Industrial Engineering*, volume 44, page 409–424, 2003. (cited on page 20)

[79] Y. Tu and C. Chen. Model to determine the capacity of wafer fabrications for batch-serial processes with time constraints. In *International Journal of Production Research*, volume 49, pages 2907–2923, 2011. (cited on page 22)

[80] Y. Tu and H. Chen. Capacity planning with sequential two-level time constraints in the back-end process of wafer fabrication. In *International Journal of Production Research*, volume 47, page 6967–6979, 2009. (cited on page 22)

[81] A. Vepsalainen and T. Morton. Improving local priority rules with global lead-time estimates: A simulation study. *Journal of Manufacturing and Operations Management*, 1(1):102–118, 1988. (cited on page 126)

[82] S. Werner, S. Horn, G. Weigert, and T. Jähnig. Simulation based scheduling system in a semiconductor backend facility. In *Proceedings of the 38th conference on Winter simulation*, pages 1741–1748. Winter Simulation Conference, 2006. (cited on page 63)

[83] C. Wu, Y.-C. Cheng, P.-J. Tang, and J.-Y. Yu. Optimal batch process admission control in tandem queueing systems with queue time constraint considerations. In *Simulation Conference (WSC), Proceedings of the 2012 Winter*, pages 1–6. IEEE, 2012. (cited on page 21)

[84] C. Wu, J. T.Lin, and W. Chien. Dynamic production control in a serial line with process queue time constraint. In *International Journal of Production Research*, volume 48, page 3823–3843, 2010. (cited on page 21)

[85] C. Wu, J. T.Lin, and W. Chien. Dynamic production control in parallel processing systems under process queue time constraints. In *Computers and Industrial Engineering*, pages 192–203, 2012. (cited on page 21)

[86] D. Yang and M. Chern. A two-machine flowshop sequencing problem with limited waiting time constraints. In *Computers and Industrial Engineering*, volume 28, pages 63–70, 1995. (cited on page 19)

[87] S. Yao, Z. Jiang, N. Li, N. Geng, and X. Liu. A decentralised multi-objective scheduling methodology for semiconductor manufacturing. *International Journal of Production Research*, 49(24):7227–7252, 2011. (cited on page 127)

[88] C. Yugma, S. Dauzere-Péres, C. Artigues, A. Derreumaux, and O. Sibille. A batching and scheduling algorithm for the diffusion area in semiconductor manufacturing. *International Journal of Production Research*, 50(8):2118–2132, 2012. (cited on pages 12 and 19)

[89] T. Yurtsever, E. Kutanoglu, and J. Johns. Heuristic based scheduling system for diffusion in semiconductor manufacturing. In *CProceedings of the 2009 Winter Simulation Conference*, pages 1677–1685, 2009. (cited on pages 12 and 19)

[90] H. Zhang, Z. Jiang, and C. Guo. Simulation-based optimization of dispatching rules for semiconductor wafer fabrication system scheduling by the response surface methodology. *The International Journal of Advanced Manufacturing Technology*, 41(1):110–121, 2009. (cited on page 63)

[91] H. Zhang, Z. Jiang, and H. Hu. Multi-criteria dynamic scheduling methodology for controlling a semiconductor wafer fabrication system. In *Automation Science and Engineering, 2007. CASE 2007. IEEE International Conference on*, pages 213–218. IEEE, 2007. (cited on page 127)

[92] X. Zhang and S. van de Velde. On-Line Two-Machine Open Shop Scheduling with Time Lags. In *European Journal of Operational Research, 204 (2010)*, pages 14–19, 2010. (cited on page 21)

École Nationale Supérieure des Mines de Saint-Étienne

NNT : 2017 EMSE 0784

Rezvan SADEGHI

CONSISTENCY OF GLOBAL AND LOCAL SCHEDULING DECISIONS IN SEMICONDUCTOR

Speciality: Industrial Engineering

Keywords: Semiconductor manufacturing, time constraint, global and local decision levels, scheduling, simulation, Multi-method modelling, agent-based method, discrete event method, AnyLogic.

Abstract: The operational level in semiconductor manufacturing can be divided into a global level and a local level. The global level refers to the scheduling decisions and production control for the whole manufacturing facility (fab), while the local level deals with those issues in each work area. The global level provides objectives or constraints for the local level. In this thesis, we propose a general framework which aims at supporting and controlling the decisions taken at the local level to deal with consistency problems between global and local scheduling decisions. The framework is composed of two layers. The bottom layer includes local policies used in each work center. The top layer consists of global objectives, global information and a global strategy which is the core of this framework. The proposed global strategy aims at controlling local policies as well as production processes. The idea is to periodically run the global strategy while production is performed to guide the production process towards achieving global objectives, and thus ensuring consistency between decisions taken at the global and local levels. We propose two types of global strategy: (1) An evaluation-based strategy which aims at improving the production process with no guarantee to determine an optimal solution and (2) An optimization-based strategy, based on a Linear Programming model.

In order to evaluate the performance of the proposed framework, we develop a data-driven generic simulation model for semiconductor manufacturing facilities.

The simulation model is a combination of Agent-Based and Discrete Event modeling methods developed with the software AnyLogic. Since the standard solver IBM ILOG CPLEX is used to solve the linear programming model, we describe its integration with AnyLogic. A set of experiments on industrial instances are presented and discussed.

In addition, this thesis deals with the management of time constraints. In a semiconductor manufacturing facility, time constraints are associated to two process steps to ensure the yield and quality of lots. A time constraint corresponds to a maximum time that a lot can spend between the two steps. If a time constraint is not satisfied by a lot, this lot will be scrapped or reprocessed. Therefore, because manufacturing equipment is expensive and cycle times must be minimized, efficiently controlling the start of lots in time constraints is important. We propose an approach which estimates the probability of satisfying a time constraint before starting a lot in the first step of the time constraint. This approach was implemented and validated on industrial constraints.

École Nationale Supérieure des Mines de Saint-Étienne

NNT : 2017 EMSE 0784

Rezvan SADEGHI

COHÉRENCE DES DÉCISIONS D'ORDONNANCEMENT DANS LES SYSTÉMES DE FABRICATION DE SEMI-CONDUCTEURS

Spécialité: Génie Industriel

Mots Clés : Industrie des semi-conducteurs, contrainte de temps, Niveaux de décision mondiaux et locaux, Planification, simulation, Modélisation multi-méthodes, Méthode fondée sur l'agent, Méthode d'événement discrète AnyLogic .

Résumé: Le niveau opérationnel dans la fabrication de semi-conducteurs peut être divisé en un niveau global et un niveau local. Le niveau global est associé aux décisions d'ordonnancement et de contrôle de la production pour l'ensemble de l'unité de fabrication (fab), tandis que le niveau local traite de ces problèmes dans chaque atelier. Le niveau global établit des objectifs ou des contraintes au niveau local. Dans cette thèse, nous proposons un cadre général qui vise à contrôler les décisions prises au niveau local pour assurer la cohérence entre les décisions d'ordonnancement aux niveaux global et local. Le cadre est composé de deux niveaux. Le niveau inférieur comprend les politiques locales utilisées dans chaque atelier. Le niveau supérieur comprend les objectifs globaux, les informations globales et une stratégie globale qui est au cœur de ce cadre. La stratégie globale proposée vise à contrôler les politiques locales ainsi que les processus de production. L'idée est de gérer périodiquement la stratégie globale, en même temps que que la production, pour guider le processus de production vers la réalisation des objectifs globaux et assurer ainsi une cohérence entre les décisions prises aux niveaux global et local. Nous proposons deux types de stratégie globale : (1) une stratégie basée sur l'évaluation qui vise à améliorer le processus de production sans garantie de déterminer une solution optimale et (2) une stratégie d'optimisation basée sur un modèle de programmation linéaire.

Afin d'évaluer la performance du cadre proposé, nous avons développé un modèle de simulation générique basé sur les données pour les systèmes de fabrication de

semi-conducteurs. Le modèle de simulation, développé avec le logiciel AnyLogic, est une combinaison de méthodes de simulation multi-agents et de simulation à événements discrets. Étant donné que le solveur standard IBM ILOG CPLEX est utilisé pour résoudre le modèle de programmation linéaire, nous décrivons son intégration avec AnyLogic. Un ensemble d'expérimentations sur des instances industrielles sont présentées et discutées.

En outre, cette thèse traite de la gestion des contraintes de temps. Dans une usine de fabrication de semi-conducteurs, les contraintes de temps sont associées à deux étapes du processus pour assurer le rendement et la qualité des lots. Une contrainte de temps correspond à un temps maximal qu'un lot ne doit pas dépasser entre les deux étapes. Si une contrainte de temps n'est pas satisfaite, le lot sera mis au rebut ou traité à nouveau. Par conséquent, parce que les équipements de fabrication sont onéreux et que les temps de cycle doivent être minimisés, il est important de contrôler efficacement le démarrage des lots dans les contraintes de temps. Nous proposons une approche qui estime la probabilité de satisfaire une contrainte de temps avant de démarrer la première étape de la contrainte. Cette approche a été mise en œuvre et validée sur des données industrielles.