Technische Universiteit
Eindhoven
University of Technology

Graduation Project Report

Development of a Simulation Model of a Workcell at KMWE

Master:	Mechanical Engineering
Research group:	Dynamics and Control
Student:	H.H.C.M. van Wesel
Identity number:	0615931
Thesis supervisor (TU/e):	I.J.B.F. Adan
Mentor (TU/e):	A.A.J. Lefeber
Mentor (KMWE):	K. Herps

Document number:	DC 2021.049
Date:	May 21, 2021

Summary

Looking at the future, KMWE would like to expand its production capacity with a minimal increase in the required resources. To this end they are looking to automate the logistical processes from shop floor level to the KMWE group and their suppliers level. To accomplish this the Advanced Manufacturing Logistics (AML) project was started to investigate and implement optimization and automation of the logistical processes on all levels. This includes developing simulation models for the various parts the KMWE Precision Components factory consists of. This graduation project is focussed on developing a simulation model representative of the workcells in the factory.

Each workcell at KMWE consists of two CNC milling machines and a robot servicing the machines with product materials and cutting tools. Product materials and tools are exchanged with the cell by an operator. The operator also oversees the automated milling process by the workcell and is responsible for implementing the scheduled production at the workcell. Production at the workcell is defined by workorders, containing information on the type of product to be milled, the number of products, and the required tools not part of the milling machine standard toolbelt.

In order to develop the workcell simulation model, a detailed understanding of the workcells at KMWE was gained first. In addition, AnyLogic was chosen as the software package to develop the workcell model and other models within the AML project. Next, the model was developed by starting with the most basic functionalities of the workcell model and building it out from there until almost all functionalities where included in the model. The most notable difference between the developed workcell model and the workcells at KMWE is the lack of any workorder scheduling and production planning, as this was expected to be part of a larger integrated model of the KMWE Precision Components factory. However, during the workcell model development this integrated model was deemed infeasible by the AML project. Some other notable features missing from the model are any form of maintenance or workcell component failures and repairs that would disrupt the workcell process.

After verifying the workcell model several simulation experiments with the model where performed. The experiment results showed that the relative item and tool capacity in the cell are a major limiting factor on the workcell performance. A common factor discovered during the experiments is that an increased relative variability in the workcell performance measurements has a significant negative impact on the workcell performance. Further investigation indicates that a large source of variability within the model is the random assignment of the milling machine and clamping location per workorder on workorder arrival. Based on the simulation experiment results it is recommended to further develop the workcell model to include workorder scheduling and planning for the workcell based on a workload per machine. Furthermore, validating the workcell model is recommended as well in order to confirm the model as an accurate representation of the workcells at KMWE Precision Components.

Acknowledgements

First, I would like to take this opportunity to thank Ivo Adan for setting up my graduation project, introducing me to KMWE and the AML project, and his support with the project.

Next, I would like to thank Erjen Lefeber for his support and guidance during this project. I would also like to thank both Erjen Lefeber and Marijke Creusen for their support, guidance and patience to allow and help me completing my graduation project after personal problems interfered with finishing the project initially.

Furthermore, I would like to thank Koen Herps, my mentor at KMWE, for his guidance, input into the project, and providing information related to the project and input. I would also like to thank Nitish Singh and the other members of AML project team for their input and collaboration with the simulation model development.

Finally, I would like to thank my parents and brothers for their continued support throughout my study and life in general.

Contents

Chapter 1

Introduction

In this chapter the context and objective of the graduation project is introduced. This graduation project is part of a collaboration between KMWE and Eindhoven University of Technology. Therefore, this chapter starts with an introduction into KWME and the AML project in which Eindhoven University of Technology is collaborating. In the second section a brief overview of the KMWE Precision Components factory is given. The third section contains the objective for the graduation project part of the AML project about the workcells part of the KMWE Precision Components factory. This chapter is concluded with an outline for the rest of the report in the fourth section.

1.1 KMWE and the AML Project

KMWE Group, founded in 1955, is specialized in high mix, low volume, and high complexity machining of stainless steel, aluminum and titanium product components. They also deliver high quality (cleanroom) assembled mechatronic modules and systems. KMWE has around 12000 products in their system of which approximately 8000 are produced with a volume of 50 to 500 per year in batch sizes between 5 and 15. KMWE delivers products, systems and solutions to customers in the medical, semi-conductor, aerospace and industrial automation sectors.

KMWE group consists of various company groups and joint ventures. Precision Components is responsible for machining products, and Precision Systems does system assembly. Both are present in both the Netherlands and Malaysia with 210 highly skilled specialists in the Netherlands and 90 in Malaysia respectively. In 2014 KMWE took over DutchAero, a supplier of aerospace structures and engine parts, giving KMWE sheet metal fabrication and thermal spraying capabilities. KMWE Projects, also located in the Netherlands, is responsible for engineering, prototyping and project management. In total, KMWE has over 550 employees.

In the Netherlands KMWE Precision Components is responsible for the machining of the high mix, low volume and often highly complex products. For products that require a lot of manual attention manually operated milling machines are used. The rest of the products are machined in a workcell, a robot servicing two milling machines. KMWE Precision components has several workcells in the factory. The factory also consists of a storage area, a quality control center and a tool service center.

Looking at the future, KMWE would like to expand its production capacity with a minimal

increase in the required resources. To this end they are looking to automate the logistical processes from the Workcell/shop floor level to the KMWE group and their suppliers level. To accomplish this the Advanced Manufacturing Logistics (AML) project is started to investigate and implement optimization and automation of the logistical processes on all levels. Eindhoven University of Technology is one of the partners collaborating with KMWE in this project.

The AML project focuses on company wide automation and optimization, as well as data gathering and integration by using simulation techniques. Simulation techniques can be used to investigate automation options and optimizing the production planning, logistical process and reducing tool and equipment redundancy.

1.2 KMWE Precision Components

As stated in Section 1.1, at KMWE Precision Components the milling of products that do not require a lot of manual attention is automated on workcell level. A workcell consists of a robot servicing two milling machines and is operated by an operator. Tools used by the milling machines are measured and made ready for use at the Tool Service Center (TSC), and at Quality Control (QC) milled products can be inspected. Most material is prepared and clamped on a pallet for milling in the workcell by the operator, although some material is clamped on a pallet in the workcell itself by an automated clamping machine. Products, materials and tools are transported between the workcell and other locations throughout the factory manually by material handling (MH).

Currently at KMWE, through the use of an ERP system, the production demand is generated and used to schedule the machine tasks through workorders (WO). Based on the generated production plan, cells are supplied with the right tools and materials by the cell operator. Finished products and used tools are retrieved by the operator from the cell as well. At the moment there is manual feedback from the workcell and factory shop floor level to the ERP system. Also, there is a certain amount of redundancy of milling, clamping and local measurement tools, as well as handling and loading equipment at the workcells.

At the shop floor level the utilization of tools and equipment could be improved by automation of the logistical processes. This might be done by reducing tool and equipment redundancy by sharing common tools and equipment between machines, and thus increasing the utilization of tools and equipment at each cell or machine. By collecting and sharing more data on a workcell, milling machine, and shop floor level with the ERP system the production process and planning might be further improved.

To gain better insight in the current production process, animation, simulation and optimization techniques can be used. They can further be used to analyze and compare automation solutions at the workcells/milling machines, and factory level with the current situation. In particular, animation, simulation and optimization techniques can be used to:

- predict and optimize the logistical process at all levels.
- investigate feasibility of automation of the logistical process at all levels.
- gain more insight in the relation between the production requirements on item level and the process requirements such as the machine, tool and material availability.
- analyze the effect of tool and equipment sharing between different cells and machines.

- optimize the use of milling, clamping and measurement tools as well as handling and loading equipment.
- optimize the production process through optimal planning/scheduling procedures.

To this end part of the AML project is focused on developing simulation models to accurately represent the production, planning and logistical processes at KMWE company wide. The simulation model development is started with models of the individual areas and centers at the Precision Components factory. These areas and centers are the workcells, TSC, and QC. After developing these models, integration into a single model on a factory level is investigated, including adding MH and production planning functionalities. This graduation project is focused on starting the development of a workcell model representing the workcells at KMWE Precision Components.

1.3 Objective

The objective of this graduation project is to develop a simulation model of a single workcell functioning as a digital twin of the workcells at KMWE, so the model can be used to simulate, analyze and predict the workcell performance at KMWE Precision Components. In order to develop the simulation model, first a detailed understanding of the workcells at KMWE is required. This includes gaining insight into the flow of material and tools within the workcell and between the workcells and the Precision Components factory.

Since this project is at the start of simulation model development within the AML project, a suitable simulation model development software package has to be chosen for this project and other planned simulation models within the AML project as well. This includes the capability to integrate several independently developed models into a single model. Intergrading the workcell model into a larger factory wide model has to be taken into account during the model development process as well.

Included in the objective of this project is to verify the developed workcell simulation model to ensure it functions as intended. Finally, determining the workcell performance for the verified simulation model and start experimentations to investigate what effects the workcell performance is included as well.

1.4 Outline

The workcells part of the KMWE Precision Components factory is described in detail in Chapter 2. Next, in Chapter 3, the developed workcell is described in detail after introducing the software package used to develop the model. The model verification process is described in Chapter 4, followed by an overview of the performed experiments with the simulation model in Chapter 5. Finally, in Chapter 6 the conclusion and recommendations are given.

Chapter 2

Workcell

In the previous chapter, KMWE, KMWE Precision Components, and the AML project were introduced. KMWE would like to better predict and optimize the logistical process and investigate automation solutions on all levels, from KMWE and its suppliers/customers to individual machines or workcells at KMWE Precision Components. In part, simulation models of all levels can be used to accomplish this. These models can then be used to analyze, investigate, predict and optimize both current processes and new (automation) solutions.

In order to develop the workcell simulation model, information on the workcells and other parts of the factory were provided by KMWE. This information was used to gain a better understanding of the workcell and its interactions and dependencies within the factory. This chapter further describes, analyses and discusses the workcells used at KMWE Precision Components. First, a general description and functionalities of the workcell and its individual components is given in Section 2.1, including how the workcell relates to other areas within the factory, and how production is defined and scheduled using workorders. Next, the flow of material and tools within and around the cell is shown and discussed in Section 2.2. The process flow and decisions related to processing and completing workorders within the factory from the perspective and responsibilities of the workcell are stated and discussed in Section 2.3. Finally, important performance measurements and indicators for the workcell are discussed in Section 2.4.

2.1 General Description

At KMWE Precision Components, products that do not require a lot of manual attention are milled by automated milling machines in a workcell. The milling machines can mill products from three material groups: stainless steel, titanium and aluminum. Before taking a closer look into the workcell containing these machines, it is important to briefly describe the planning of the products to be milled by these machines in the workcell through the use of workorders. In addition, an introduction of other areas in the factory and important interactions with the workcell is given.

2.1.1 Workorders and Factory Relations

Using production forecasts and the ERP system, factory planning creates workorders. A workorder (WO) holds the documentation that includes the product type to be milled and

corresponding CNC program, the product batch size, and the necessary toolset containing product specific tools not available in the standard tool inventory of the milling machine. When the raw material for the product is available, the workorder can be scheduled for production on one of the machines in a workcell. Apart from the workorder due date and workcell availability, the characteristics of the product are important to determine which machine in which workcell is allocated to the workorder. Some of these important characteristics are the product material group, clamping method and the milling process time on the milling machine.

After the workorder is scheduled, the raw material and product specific tools are prepared and sent to the designated workcell. Material Handling (MH) is responsible for purchasing, acquiring and delivering the raw material from the factory storage to the workcell. All tools used for milling are assembled, combined into toolsets, and disassembled or discarded at the Tool Service Center (TSC). Therefore, they are responsible for assembling and delivering the corresponding toolsets from the workorders to the designated workcell.

Part of processing the workorders at the workcell includes the first milled product of each workorder being subjected to a quality inspection at Quality Control (QC). After milling, the product is retrieved from the workcell and delivered at QC. Failing this inspection effects the remaining process at the workcell, but is discussed later. After passing the quality inspection and processing the remainder of the products, completing the workorder at the workcell, the finished products are collected by MH and either stored in the factory warehouse or immediately brought to QC for inspection.

2.1.2 Workcell Composition

To better understand how a workorder is processed at a workcell, a general description of the workcell composition is stated. A workcell consists of two milling machines serviced by an automated robot, and is handled by an operator. The milling machines, the automated robot and the operator can be defined as the three main aspects or domains of the workcell, each with different components, functions and responsibilities in the production process.

In the milling machine domain, a single milling machine can be divided into three components; the milling machine core, a toolbelt holding the machine tool inventory and the chip exit. In the automated robot domain, the physical components of the robot are a human interface for interactions with the operator and a robot arm in the center of the workcell. The robot arm is referred to as the robot, when a distinction between the robot arm specifically and the robot as an automated domain is not important. Beside the two milling machines, the robot arm is surrounded in a hexagonal layout by a clamping machine, inventory storage areas, and exchange areas for exchanging material and tools between the robot arm and the operator. Exchanging material and tools between the robot and milling machines occurs directly between the robot arm and the milling machine core. Given that these physical components surround the robot arm in a hexagonal setup, the robot domain is also referred to as inside the workcell, the inside of the workcell, or the cell. Everything part of the workcell around the cell and the milling machines, also referred to as the outside of the workcell or outside the workcell, is the domain of the operator. In this domain, the workplace for the operator, storage areas for chip containers at the milling machine chip exits, and coolant storage can be found. A schematic overview of the workcell components is shown in Figure 2.1.

In order for the workcell to process a workorder by milling the products defined in the workorder, material handling equipment, material, and tools are used. First, there is raw material from which the product is milled, resulting in the finished product after milling. For convenience, the

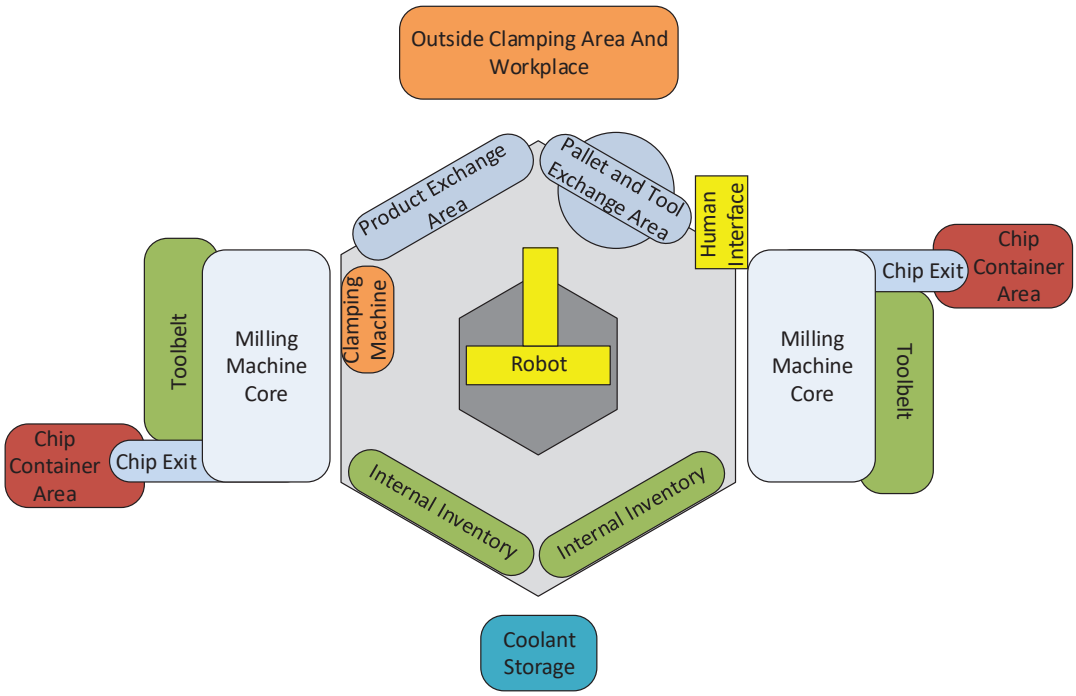

Figure 2.1: Schematic overview of the workcell components.

raw material and milled products are mostly referenced to as products or items throughout the rest of the report, where the state of said product or item is mentioned if necessary. Pallets and clamping equipment are used to clamp product material on pallets, and after milling unclamping finished products from pallets. Clamping and unclamping is performed by either the operator or the clamping machine depending on the product type.

Next, cutting and measuring tools are used by the milling machines to setup, execute and examine the milling process. The milling machine toolbelt contains a standard set of tools, while product specific tools are loaded into the toolbelt for the milling of that specific product only. The product specific tools are collected in a special toolset, specified for each individual workorder based on the product type and batch size. Throughout this report the standard tools in the milling machine toolbelt are referred to as standard tools and standard toolsets, while the product specific tools and toolset are referred to as special tools and special toolset, or tools and toolset. The distinction between cutting and measurement tools is not made anymore unless relevant. Material handling equipment is used by the robot arm to retrieve, move and deposit unclamped products, clamped products on pallets, empty pallets, and tools throughout the workcell. Finally, chip containers are placed at the chip exit of each milling machine to collect the chipped or cut material during the milling process.

With the workorders and factory areas introduced, the workcell components stated, and the material handling equipment, products and tools defined, a closer look is taken into each defined workcell aspect: the milling machine, robot and operator domain. For each domain, the components, functions, responsibilities, interrelations, interdependencies, as well as connections and dependencies with other areas within the factory, are expanded upon next.

2.1.3 The Milling Process: The Milling Machine Domain

The milling machine domain is responsible for executing the automated milling process. To execute this process, the two milling machines included in each workcell are standard 5-axis CNC milling machines capable of automatically handling, cutting and performing measurements on clamped product material according to a CNC program using cutting and measuring tools. In the workcells at KMWE, products are milled from three types of material; stainless steel, titanium or aluminum. Each machine is configured to process a single type of material, including a tool inventory containing a standard set of material type compatible tools and extra storage room for product specific special toolsets.

As defined earlier, a single milling machine consists of three components: the milling machine core, the toolbelt and the chip exit. The milling machine core consists of the spindle, capable of performing measurement and milling operations on the product material by retrieving, equipping, using and storing measurement and cutting tools, as well as a bed where clamped products are placed in a fixed location on the bed. Both the standard tools and the special tools are stored in the milling machine toolbelt. Tools from the special toolset are automatically retrieved from and stored in the toolbelt by the machine core spindle. The operator can manually retrieve and load tools into the toolbelt as well. Manually retrieved tools can only be manually loaded back into the toolbelt. During the milling process, chip, the cut material from the product, leaves the machine through the chip exit into a chip container. The coolant fluid used by the machine coolant system is stored outside the workcell. Both the chip container and coolant need to be available. An overview of the milling machine components can be found in Figure 2.2.

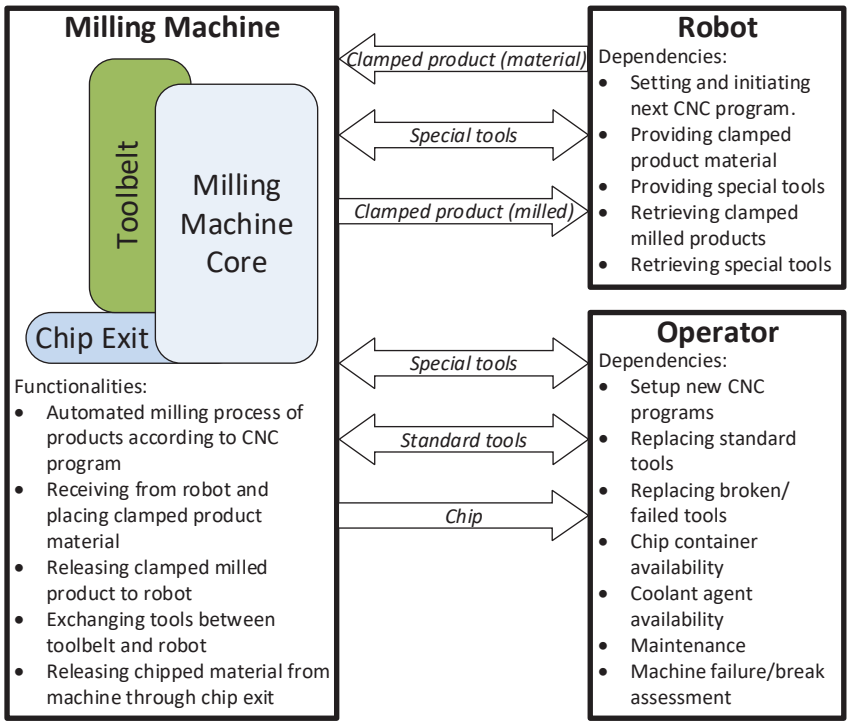

Figure 2.2: Overview of the components and functions of the automatic milling process domain, including product and tool flow between and dependencies on the other domains.

While the milling machine executes the automated milling of products, the robot domain initiates

and handles this process. Assuming the milling machine is operational and not reporting any errors, before the next milling process can be initiated by the robot, the milling machine has to be available. First, this includes the previous milling process being finalized, either through successfully completing or, after encountering complications, terminating this process. Subsequently, the CNC program for the next process has to be installed on the machine, and all required tools from the standard toolset have to be available in the toolbelt and approved for use by the robot. Additionally, the milling machine has to have access to the coolant fluid from the coolant storage, and a chip container has to be present at the chip exit. Finally, the milling machine cannot be reserved or scheduled for use by the operator.

After the robot has determined that the machine is operational and available, the robot initiates the next automated milling process. The required clamped product material and the special toolset, if not already loaded into the toolbelt, are provided to the milling machine by the robot. Clamped products on pallets are placed on the machine core bed by the robot arm. Product specific tools from the special toolset are also provided by the robot arm, retrieved by the machine spindle, and loaded into the toolbelt. When the product is fixed on the bed and all required tools are loaded into the toolbelt, the automated milling process is executed by starting the CNC program. Assuming no complications during the milling process, after completing the CNC program, the clamped product is retrieved by the robot arm from the bed. When required, the tools from the special toolset are retrieved from the toolbelt by the machine spindle and provided to the robot arm as well. Assuming the milling machine is still operational and available, the process can then be repeated for the next product.

When a product from a workorder is produced for the first time on a milling machine, the machine has to be setup with the new CNC program by the operator. After the setup and a successfully produced product that passed quality inspection, the machine can run the CNC program for subsequent products autonomously. When required, standard tools from the machine toolbelt are retrieved and replaced by the operator manually. The same applies when tools fail or break during production. In addition to replacing broken tools during production, the operator is required to abort the current milling process and inspect the milling machine. If there are no complications with the machine itself, the milling machine remains operational, allowing the robot to retrieve the product and tools, and initiating the automated milling process of the next product.

In addition to the milling machine components, Figure 2.2 shows the described automated milling process of the milling machine through the functions of this domain, the flow of products and tools between the milling machine domain and the other workcell domains, and the functions from the other domains the milling machine is dependent on. In the following subsection the robot domain is discussed.

2.1.4 Inside the Workcell: The Robot Domain

In this subsection, the domain of the robot is expanded upon. Taking a closer look at previously stated physical components of the cell domain, the robot arm is used for retrieving, moving and depositing products, empty pallets, clamped products on pallets, and tools between components within the cell and the milling machines. Handling equipment used by the robot arm is available within the cell inventory for assembly and disassembly from the arm. A percentage of the product types are clamped onto a pallet, and after milling unclamped from the pallet, inside the workcell by the clamping machine.

The raw material/product exchange area, or just product exchange area, is used to retrieve the

products in raw material form from this exchange to be delivered to the clamping machine. After unclamping, milled products are deposited at the product exchange area. This exchange is the only location in the cell where unclamped products can be temporarily stored, typically holding only one product batch at the same time. This leads to the internal inventory of the workcell being able to store tools and pallets, either empty or with a clamped product attached. The storage capacity of the cell inventory is 400 tools and 75 pallets. Tools and pallets are retrieved for internal use or storage at the pallet and tool exchange area, and are deposited there for retrieval from the cell as well. The pallet and tool exchange area, also referred to as the pallet exchange area, can only hold a single tool or pallet simultaneously. Both exchange areas are used to exchange products, pallets and tools between the robot and the operator. Additionally, at the pallet exchange area, the human interface is located. The human interface allows for interaction between the robot computer and the operator. All components within the robot domain are shown in Figure 2.3, where the human interface is displayed separately from the pallet exchange area.

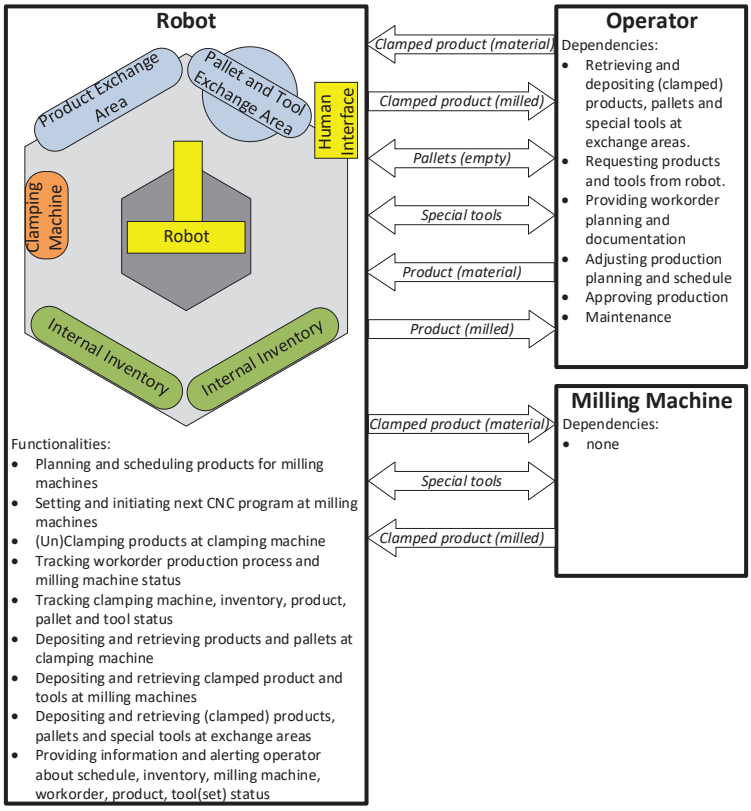

Figure 2.3: Overview of the components and functions of the robot domain, including product and tool flow between and dependencies on the other domains.

The main responsibility of the robot domain is the automated execution by the robot computer of the production on the milling machines documented in the issued workorders for the workcell. The execution of this process consists of the robot planning, scheduling, preparing, initiating and handling the automated milling process of individual products for each milling machine. An overview of this process in terms of the robot functions, dependencies on the operator and milling machines, and exchange of products, pallets and tools can also be found in Figure 2.3.

In order to execute all parts of the automated production process the robot is responsible

for, the robot tracks the status of each workorder product batch, individual products, special toolsets, individual tools including the tools from the standard toolsets in the milling machine toolbelts, the cell inventory including empty pallets, and the milling machines. This information is available to the operator as well.

Based on the available status information tracked by the robot, the operator provides the robot with the workcell workorder planning and documentation. Using this workorder planning and information, the robot plans the production of the individual products from all workorders for both machines. This planning is shared with the operator and can also be adjusted by the operator. The production planning is used as the basis of the production schedule for each machine, but the schedule is subject to change and not final until a product is approved for production by the operator, and both the product and toolset are available and ready within the cell.

Before the automated milling process of an approved scheduled product on one of the milling machines can be initiated, the robot requires the product and tools from the special toolset available and ready inside the cell. Both are provided by the operator, unless the tools from the toolset are already available in the cell inventory. Products that are clamped inside the cell are provided at the product exchange area, while already clamped products are provided to the robot through the pallet exchange area. The operator communicates through the human interface to the robot when all unclamped products from a single workorder batch are available for clamping at the product exchange area. The robot arm retrieves the products for clamping at the clamping machine. At this machine, the robot provides both the product and an empty pallet from the inventory for clamping, and retrieves the clamped product for storage in the cell afterwards. The usage of the pallet exchange area is controlled by the operator. The operator instructs the robot to retrieve clamped products and tools from this exchange area for storage in the cell inventory. Clamped products or tools are generally exchanged sequentially as a complete batch or set. In addition, the operator instructs the robot to provide empty pallets for clamping products outside the cell, and retrieving empty pallets for storage in the cell.

With the product and tools available and ready in the cell, and assuming the milling machine is operational and available in accordance with the milling production schedule, the robot initiates the automated milling process at the milling machine. First, tools from the toolset are provided to the milling machine, if not already loaded in the machine toolbelt, followed by providing the product. When all are provided, the milling machine is instructed to begin the CNC program. After the milling process is finished, the product is retrieved from the milling machine, as well as the tools from the special toolset when required.

After production, the operator can instruct the robot to deposit milled clamped products at the pallet exchange, either a single product or the complete or remaining batch of a workorder in sequence. Products clamped with the clamping machine are unclamped at this machine as well, before being deposited at the product exchange area. This can also be either a single product, or the complete or remaining batch in sequence. Clamped products are deposited at the clamping machine, and after unclamping, the product and the empty pallet are retrieved by the robot arm. The product exchange area has to be available before products are unclamped at the clamping machine, meaning not full with products from another workorder or being used by the operator. Additionally, tools can also be instructed for removal from the cell by the operator through the pallet exchange, generally per toolset.

The automated execution of the production process on the milling machine can continue without the operator as long as there are approved scheduled products with the associated toolsets available in the cell and the milling machines stay operational and available. The robot can alert

the operator if there is no approved and available scheduled production for one or both machines, as well as any problems with the milling machines or the automated milling process on these machines. In similar fashion to the milling machines, the robot and other cell components are subjected to scheduled maintenance and depend on the operator for assessment and arrange replacement or repairs if any component fails or breaks.

2.1.5 Outside the Workcell: The Operator Domain

Outside the workcell, the domain of the operator, consists of the remaining components and functionalities of the workcell not covered by the robot and milling machines. The main responsibility or functionality of the operator is initiating, handling, supervising and finalizing the production process from the assigned workorders according to the factory planning. The aforementioned responsibility includes the operator investigating and handling any complication effecting the capability of the workcell to execute and successfully complete the production process from the assigned workorders according to the factory planning. Given this responsibility, the operator domain can be regarded as responsible for executing the main purpose of the workcell, processing workorders through the automated milling of products defined in the workorder. Unsurprisingly, the workcell is connected to the factory through the interactions and dependencies between the operator domain and other factory areas, while the robot and milling machine domain only have interactions and dependencies within the workcell between each other and the operator.

The operator domain includes a workplace area, chip container storage areas at the chip exit of each milling machine and the coolant storage. The workplace is used by the operator to store and prepare products and tools for processing in the cell or other areas in the factory. Additionally, the workplace serves as the delivery and retrieval area for products, tools and equipment for exchange between the workcell and other factory areas. At the chip container areas, chip containers are used to collect chip from the milling machines. The coolant storage holds coolant fluid used by both machines during the milling process. An overview of the operator domain components, functionalities and dependencies within the workcell and other factory areas can be found in Figure 2.4. All aspects of the operator domain functionalities, as well as interactions and dependencies between the operator and the robot, milling machines and factory areas are expanded upon next.

After workorders are planned and assigned to a workcell, the operator is provided with the workorder production planning and documentation by the team leader. Although the workorder planning and assignment to workcell is performed with detailed knowledge of the workcell production schedule and availability, the operator uses the workorder documentation and the information tracked by the robot to perform a final assessment on the feasibility of processing the workorders according to the planning. Besides the already scheduled production workload on the workcell, the operator determines for each workorder if the required special toolset is available in the cell and usable for this workorder production. If not available, a toolset request is issued at the TSC by the team leader.

An important factor of tool information tracked by the robot, is the used milling time for each tool and how long a tool can still be used before reaching its maximum life time and has to be replaced. Using this information from each tool in the toolset, and the milling time for each tool per product from the CNC program for the workorder, the robot calculates the number of products from the workorder can be produced with the current toolset. If the toolset is available in the cell, this toolset is used for the workorder production. If the toolset is available, but the workorder can only be partially completed with this toolset, the operator has the option to split

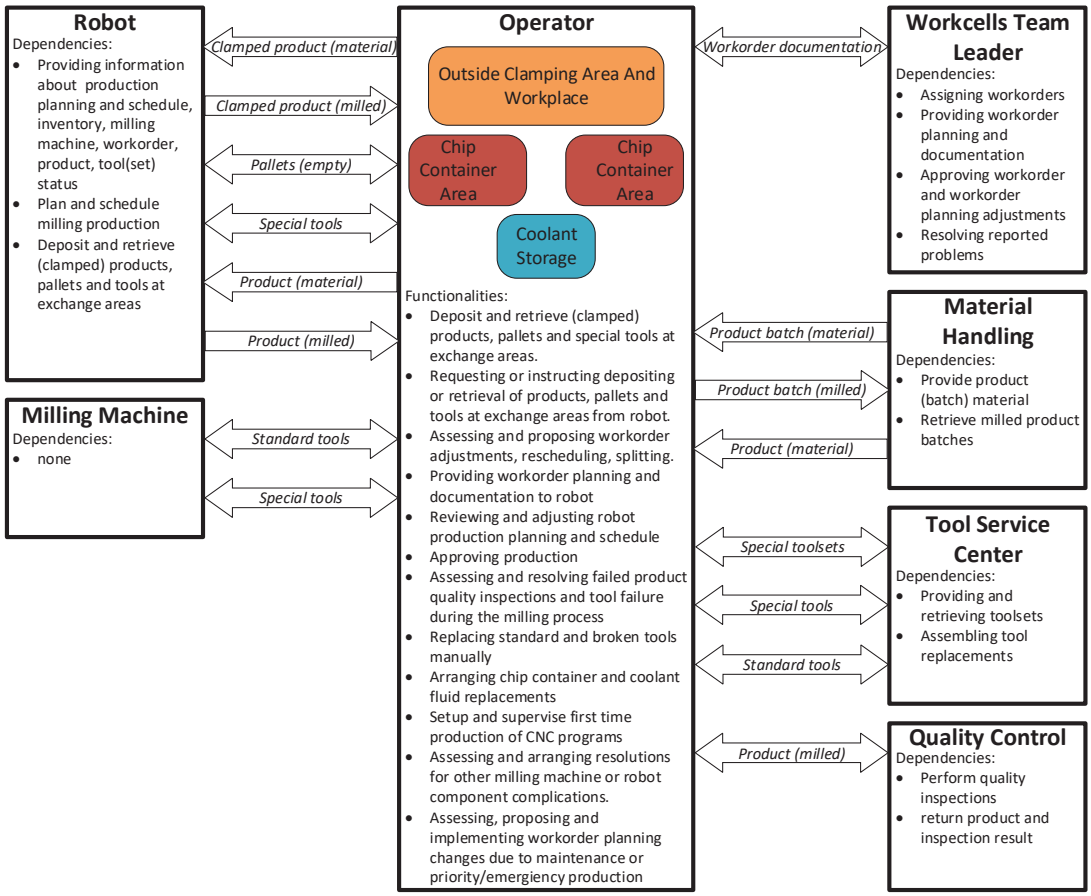

Figure 2.4: Overview of the components and functions of the operator domain, including product and tool flow between and dependencies on the other domains.

the workorder into two workorders. The first workorder contains an adjusted batch size based on the provided number of products the toolset is allowed to be used for production. The second workorder contains a batch size for the remaining products, for which a new toolset has to be requested at the TSC.

Any planning issues, proposed adjustments, workorder splits, or other problems causing production delays are reported to the team leader, which can lead to a reevaluation and adjustment of the workorder planning and workcell assignment. If the workorder planning is feasible, required adjustments or workorder splits are approved, and the toolset is available or requested, the operator inserts the workorder planning into the robot. The robot adds the products from the workorders to the milling machine production planning and adjusts the milling production schedule accordingly. Both the planning and schedule are reviewed by the operator. The operator can adjust both when deemed necessary.

Before the scheduled production on the machines by the robot can be executed, the cell requires the products and tools available. The products in raw material form are delivered per batch to the workcell from factory storage by MH according to the workorder planning. Most products are clamped by the operator. For each product clamped by the operator, the operator instructs the robot to provide an empty pallet if needed and available in the cell inventory, and retrieves it from the pallet exchange when provided by the robot.

After all products from the workorder batch are clamped, or if products are clamped inside the cell, the operator can provide the products to the cell. Based on information on the production schedule, product exchange area availability and cell inventory capacity from the robot, the operator decides when and which workorder products to provide to the cell. When required, finished products, empty pallets or unused toolsets are first removed from the cell. When the product exchange area is available, unclamped products are deposited there. After loading the complete batch, the operator informs the robot the products from that workorder are ready for clamping. Clamped products are provided to the robot through the pallet exchange area. The operator checks the robot arm availability, loads one product in the pallet exchange area and instructs the robot to retrieve and store the product. This process is repeated until the complete batch is retrieved by the robot. In similar fashion, toolsets are provided per tool through the pallet exchange.

In addition to the product and toolset availability, products have to be approved by the operator before milling on the machine is allowed. For each workorder, the milling of the first product is always approved. For each unique workorder product type, special toolset, CNC program, milling machine, and required standard tools from the machine toolbelt combination, a single milled product requires inspection at QC. Milled products requiring inspection are unclamped and retrieved from the cell, and subsequently delivered to QC by the operator. After conducting the inspection, QC returns the product and informs the operator on the result. Assuming the product passed the quality inspection, the operator approves all subsequent products of the workorder until the special toolset, any individual tool from this toolset, or any of the required standard tools need replacement due to reaching life time use or failure during milling. In that case only the next product is approved for production, requiring a new quality inspection pass in order to approve subsequent milling of products from the unique combination.

Products requiring inspection after milling, part of a completed batch, or otherwise needed by the operator, are retrieved from the cell through the inverted process they were provided to the cell. For products clamped outside the cell, the operator instructs the robot to provide the clamped products at the pallet exchange. After the robot arm deposits the product in the exchange, the product is retrieved by the operator. Products clamped inside the cell are requested from the robot for retrieval. If the product exchange area is available, products are first unclamped before being deposited in the exchange area by the robot arm. Deposited products in the exchange are retrieved by the operator. Either a single product or each product from a workorder batch in sequence can be retrieved from the cell. Individual tools or all tools per toolset required from the cell are retrieved through the pallet exchange area using the same process as products retrieved through this exchange. Products clamped by the operator are unclamped at the workplace. Products from completed workorders are combined into a batch, ready for retrieval by MH for further processing. Tools from toolsets are combined into a single transportable toolset. Toolsets are retrieved by the TSC or delivered by the operator for disassembly. Processed workorders are transferred to the appropriate factory area by MH.

During the production process, the milling machines have to be available before the automated milling process of the next product can proceed. Most machine availability conditions depend on the operator. Starting with tool replacement, similar to the tools of the special toolset, the used time and remaining milling time of the standard tools from the milling machine inventory is tracked by the robot. Using this and milling times per tool from CNC programs for scheduled production, the operator regularly evaluates when standard tools require replacement. Standard tools are replaced manually by the operator directly at the machine toolbelt. The machine toolbelt cannot be used by the milling machine and operator simultaneously. The operator has to wait, pause or abort the active milling machine CNC program or any other retrieving/loading tool process before using the toolbelt. In return, the automated processes of neither the milling

machine nor the robot can start any process requiring toolbelt access.

In addition to replacing tools, the operator is responsible for arranging replacement of full chip containers and dirty coolant fluid from the coolant storage. The milling machines cannot mill without access to the coolant fluid and are not allowed to mill without chip containers at the chip exit. All replacements can potentially delay the production schedule.

For each machine, new CNC programs are installed on the machine and executed under close supervision when milling the first product with this program by the operator. A workorder containing a new CNC program is processed similar to other workorders, including providing the product and toolsets through the robot. However, the first product is scheduled in the robot planning by the operator with extra time for the setup process. After production, the product is inspected at QC. Assuming the product passed inspection, the CNC program is approved for automatic production.

If a product fails quality inspection, the product is removed from the workorder and the operator assesses the milling production process for the unique combination used to mill the product. The operator starts the assessment at the milling machine, investigating the CNC program, used tools and any other relevant components or processes when necessary. Depending on the discovered problems in the process, resolutions can include replacing tools, adjusting and reconfiguring the CNC program, and repairing the milling machine. All assessments and subsequent implementation of resolutions cause delays in the effected milling machine schedule and workorder production process. Depending on the resolutions, the current workorder is cancelled, or continued with possible adjustments and delays. Continued workorders are rescheduled in the workcell, or removed from the workcell planning for reassignment by the team leader. For rescheduled workorders and based on raw material availability, the removed product is replaced with a new product delivered by MH. Alternatively, the workorder batch size is adjusted for the removed product, with a future workorder issued by factory planning including the removed product in the batch size. The workorder process continues accordingly with the new first product approved for production and inspection at QC after milling. Workorders with a new or adjusted CNC program require the operator to install the CNC program and supervise the first milling process again. Significant delays caused by the assessment and resolution implementation in the milling machine production schedule or the workcell workorder planning requiring adjustments or rescheduling of workorders, are reported to and reviewed by the teamleader.

Tools breaking or failing during the automated milling process are detected by the milling machine. The automated milling process is paused, requiring assessment by the operator. Assuming the tool failure did not cause any damage to the milling machine, the operator retrieves the broken tool manually trough the toolbelt, while the product is retrieved via the robot through the cell if further inspection of the product is required. The broken tool is replaced at the TSC and the product is evaluated at the operator workplace when applicable. Assuming a replacement tool is available and prepared immediately, the replacement tool is manually loaded into the toolbelt. If the product has no damage, the milling process can be resumed, after reentering the product into the machine through the cell if removed for inspection. If the product is damaged, the product is removed and the paused milling process is aborted. The workorder and workorder planning is adjusted following the same process for product removal due to failing inspection. The finished product or next milled product after product removal is inspected at QC for production approval. In case the replacement tool is not immediately available at the TSC, the paused milling process is terminated and the partially milled product removed from the workorder. If appropriate, the operator proposes the workorder to be split, with a workorder for the finished milled products and a workorder for the remaining products, with

the batch size adjusted and scheduled based on material availability for a replacement product and the expected replacement tool availability date. The remaining toolset and products are retrieved from the milling machine and cell through the robot when required. When available, the replacement tool is manually loaded through the toolbelt into the machine. The tool is retrieved by the robot for storage or removal from the cell, depending on the associated toolset location.

In addition to production process assessments and replacing broken tools, the operator is responsible for assessing other milling machine and cell components complications, including failures or damage. The operator resolves the complications, arranges resolutions, and/or reports the complications to the team leader. Finally, the operator has to assess, propose possible adjustments to the teamleader and implement workorder planning changes due to scheduled maintenance or priority and emergency production assigned to the workcell.

2.2 Product and Tool Flow

After describing the workcell components, functionalities and factory relations from the perspective of the operator, robot and milling machine, a closer look is taken into the flow of products, tools and other parts throughout the workcell. Using the information from the previous section, the flow and status of products, tools and workorder documentation between the workcell domains and other factory areas are combined to provide a global overview of the product, tool and workorder documentation flow. This overview is shown in Figure 2.5, including the flow of coolant fluid and chip containers.

As shown in Figure 2.5, the flow of workorder documentation, chip containers and coolant fluid is straightforward. The workorder planning is provided to the operator by the team leader and, after assessment and possible adjustments, provided to the robot. Processed workorders are returned to the teamleader. The replacement of coolant fluid and chip containers is arranged by the operator and performed by MH. When required, empty containers and new coolant fluid are provided, while full containers and used fluid are retrieved by MH.

Alternatively, the flow of products and tools is more complicated, warranting a more detailed description on component level. Figure 2.6 illustrates a complete overview of the product and tool flow. In addition to exchanging products and tools between the workcell domains and other factory areas, this overview depicts the status, the movement between components, and the processing at each component of products and tools. In Figure 2.6, the workcell components, domains and factory areas are shown according to a schematic layout. Since the product and tool flow is identical for both milling machines in the workcell, only one machine is included to represent both. The product and tools are color coded according to their status, while the movement of products and tools is illustrated by arrows, with other symbols used to represent the various processes, including clamping and milling, storage and exchange between domains.

2.2.1 Product Flow

In order to describe the product flow, important characteristics and product states are stated, since the flow is partly defined and dependent on both. First, a distinction is made between pre and post milled products. The status of pre milled products is stated as material or referenced as product material, while the status of post milled products is stated as milled or referenced as milled products. Products can be transported individually or as a batch per workorder. Next,

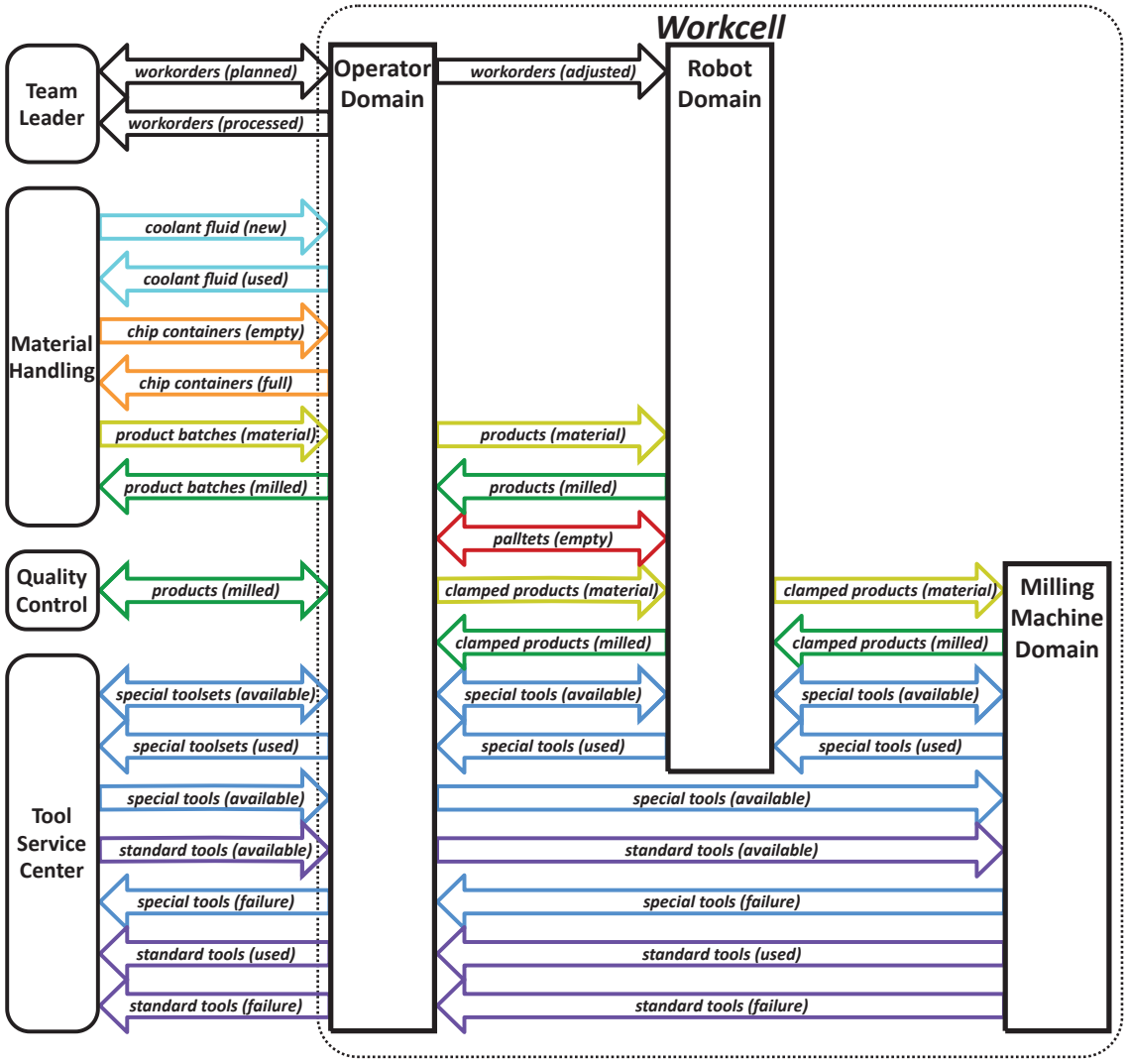

Figure 2.5: Flow of products, tools and other parts between the workcell domains and factory areas.

before milling, products are clamped on a pallet, referenced to as clamped products. The flow of empty pallets in the cell is considered as a separate flow and subsequently included in the product flow overview shown in Figure 2.6. Finally, the product flow in the workcell is dependent on the clamping and unclamping location in the workcell, as well as the milling machine assigned to mill the product. In the flow description and overview from Figure 2.6, the different flow depending on clamping location is shown. However, as mentioned earlier, no distinction is made between the milling machines in the flow description.

For each workorder, the product flow starts with the arrival at the workcell of the product material batch from MH, stored at the operator workplace. At the workplace, for products clamped inside the cell, product material batches are separated into individual products and placed in the product exchange by the operator. Alternatively, for products clamped by the operator, the product material batches are separated, clamped on a pallet, and provided to the robot through the pallet exchange. Inside the cell, clamped products are retrieved from the pallet exchange and stored in the pallet inventory by the robot arm. Products provided through the product exchange are retrieved and delivered to the clamping machine. After products are clamped inside the cell, they are retrieved and stored in the pallet inventory as well.

When scheduled, clamped products are retrieved from the cell inventory and provided to the milling machine at the milling machine core. In this location the products are milled. After milling, the milled product is retrieved from the milling machine core by the robot arm and stored. Milled clamped products are provided to the operator through the pallet exchange if clamped by the operator, or provided to the clamping machine if clamped inside the cell instead. After unclamping, the milled product is retrieved from the clamping machine and deposited at the product exchange.

After retrieving clamped products from the pallet exchange, the operator unclamps the product from the pallet at the workplace. If a product requires inspection at QC, the operator retrieves the product from his workplace, or the product exchange if clamped inside the cell, and delivers the product to QC. After inspection, the product is returned to the workplace. When all products of a processed workorder are either retrieved from the pallet exchange and unclamped, or provided at the product exchange instead, the products are merged into a batch. Milled product batches are retrieved by MH, finishing the product flow at the workcell.

As mentioned earlier, the flow of empty pallets is considered a separate flow. Empty pallets are stored inside the cell inventory. Before clamping, the robot arm retrieves and delivers a pallet to the clamping machine, or the pallet exchange where it is retrieved by the operator. After clamping, empty pallets are either provided through the pallet exchange by the operator and subsequently retrieved and stored inside the cell by the robot arm, or retrieved from the clamping machine and stored instead.

The described, and shown in Figure 2.6, product flow does not include products removed from workorders after failing inspection at QC, partially milled products after complications during the milling process, or replacement products (material). Partially milled products, or product material requiring (temporary) removal from the cell, leave the milling machine and cell, and are unclamped according to the milled product flow. Damaged or otherwise rejected products retrieved from the cell or returned from QC to the operator workplace, are retrieved by or delivered to MH. Replacement products (material), or otherwise not included in the product batch delivery, are delivered by or retrieved from MH as well. Finally, (partially) milled products are clamped, and provided to the robot and milling machine according to the product material flow when applicable.

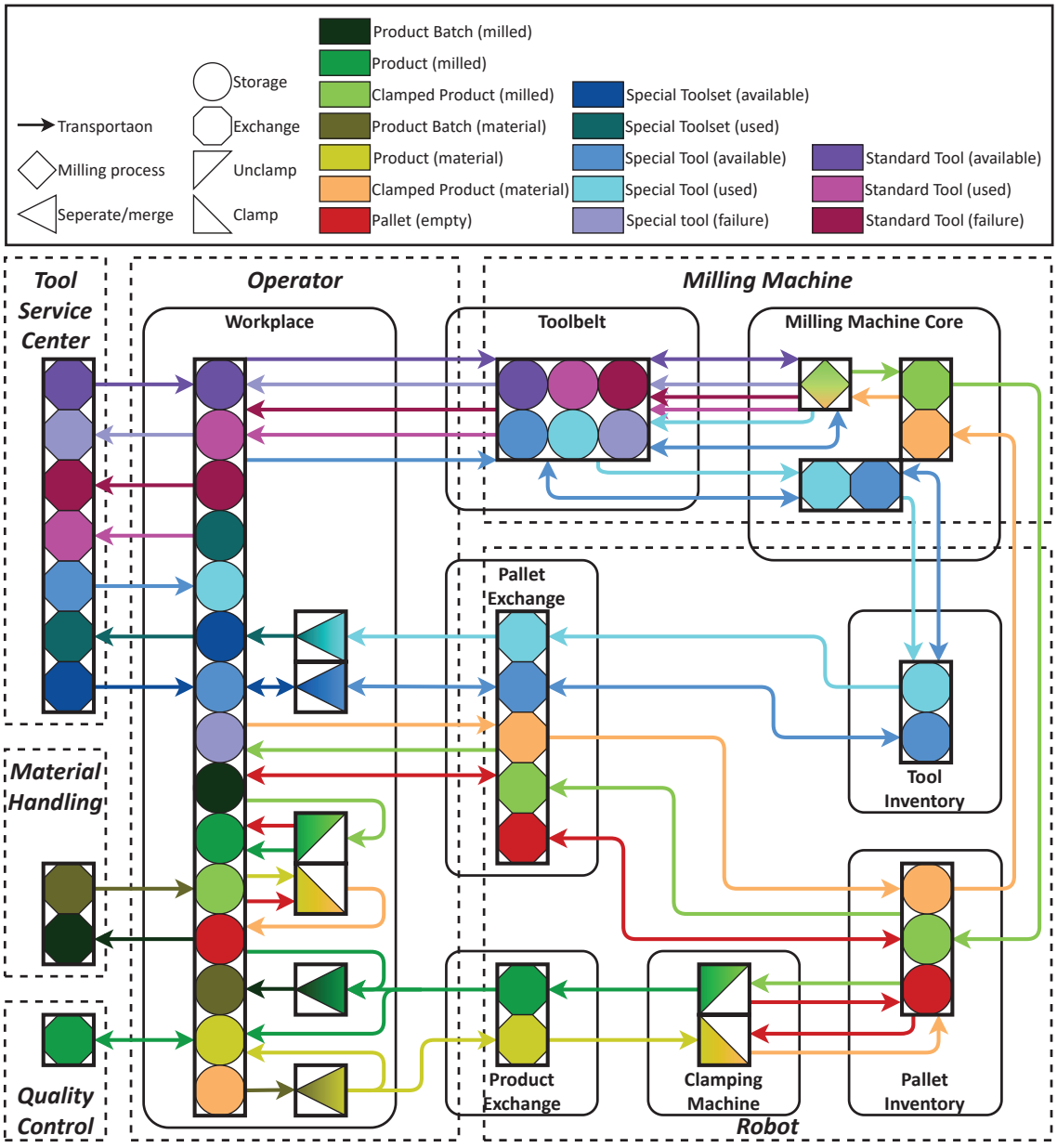

Figure 2.6: Flow of products and tools throughout the workcell.

2.2.2 Tool Flow

Similar to the product flow description, before describing the tool flow, important tool characteristics and states are stated. The most important distinction between tools is whether the tool is part of the standard milling machine toolset or a product specific special toolset. Each tool is referred to as a standard or a special tool respectively and the flow for each is described separately. Continuing, a distinction between three states is made for all tools. The tool status is available, or referred to as an available tool, when the tool, or the special toolset it is included in, has not reached the maximum user life time yet, and can still be used for at least one complete product milling process. Otherwise, the tool state is referred to as used or a used tool. In case of a failed or damaged tool during the milling process, the tool state is referred to as failure, or a failed tool. Finally, standard tools are transported individually and manually exchanged with the milling machine. In contrast, special tools can both be transported as a set or individually and are exchanged with the milling machine automatically, with the exception of replacing failed tools.

The flow of standard tools, also shown in Figure 2.6, starts at the TSC. Available tools are usually retrieved by the operator, or delivered at the workplace otherwise, and manually placed into the machine toolbelt. When required during the milling process, available tools are retrieved from the toolbelt, used, and placed back by the machine spindle. Used and failed tools are retrieved from the toolbelt by the operator and delivered at the TSC for further processing, finishing the standard tool flow. Not included in the product and tool flow overview in Figure 2.6 is the placement or removal of (partial) standard toolsets in the case of a new, reconfigured or decommissioned milling machine for the workcell.

The special tool flow also starts at the TSC. Tools are delivered as a set to the workplace at the workcell. When appropriate, the toolset is separated into individual tools by the operator. Each tool is placed into the pallet exchanged and subsequently retrieved and stored inside the cell inventory by the robot arm. When required, available tools are retrieved from the inventory and provided to the milling machine. The milling machine spindle retrieves the tools from the robot arm and loads them into the toolbelt. During the milling process, tools are retrieved, used, and placed back by the machine spindle according to need. Both available and used special tools are retrieved from the toolbelt and provided to the robot arm by the machine spindle, followed by the robot arm retrieving and storing the tool in the cell inventory. If requested, both available and used tools are provided to and retrieved by the operator through the pallet exchange. If all tools from the special toolset are retrieved, the tools are merged into a toolset. In some cases available toolsets are stored at the workplace for reentry into the cell. Otherwise, available and all used tools are delivered to the TSC as toolsets for further processing, finishing the tool flow. However, similar to failed standard tools, failed special tools during the milling process are manually retrieved and delivered to the TSC individually. New available replacement special tools are provided by the TSC to the operator and manually loaded into the milling machine toolbelt accordingly.

2.3 Process Flow

During the development of the workcell simulation model, various process flowcharts showing the product realization process at KMWE Precision Components where developed by Koen Herps for the AML project. These process flowcharts, especially the process flowchart from the workcell perspective, where used as a blueprint for the workcell simulation model. Based on this

flowchart and the workcell information described in the previous sections, the workcell process flow is shown and described next.

The main workcell functionality or process, consisting of the automated milling process of products in the workcell through processing workorders, can be shown visually in a process flowchart. This complete workcell flowchart can be found in Figure A.1 in Appendix A. In this flowchart, the workcell process is depicted as a sequence of actions and decisions, arranged according to domain or area where these actions and decisions are taken. Actions are represented by blue rectangles, and decisions by green diamond shapes. In the process flowchart context, decisions are defined as yes or no questions, with the subsequent occurring action(s) dependent on the answer. These decisions relate to possible outcomes of actions or events, the state of components, workorders, products, tools, or any other factors or characteristics. The domains and areas where the actions and decisions are taken or are responsible for taking place have already been introduced and discussed in the previous sections. For the workcell process flow it is assumed that the workcell components remain operational, meaning maintenance and component failure are not included. Given the complete workcell process flow is extensive, the process is divided into four phases: planning, preparation, production and finalization. Each phase is shown and discussed separately in the remainder of this section.

2.3.1 Planning Phase

In the planning phase of the workcell process flow, shown in Figure 2.7, the workorders are scheduled and assigned to the workcell according to a daily planning by the team leader, based on product material availability through MH. The operator assesses the workorder planning feasibility. If there are no issues, the workorder planning and documentation is provided to the robot and the workorder production preparation can be started. In case of workorder planning issues, the operator assesses the workorder production process for the workcell. Any workorder planning issues and required workorder adjustments are reported and proposed to the team leader. The team leader approves workorder adjustments and reschedules the daily workorder planning if necessary. Any milling production schedule issues or conflicts in the workcell are assessed and adjusted by the operator as well, including additional workorder planning or production process issues resulting from the milling production schedule problems.

2.3.2 Preparation Phase

The preparation phase of the workcell process flow, shown in Figure 2.8, consists of preparing and providing the products and tools to the cell. The team leader initiates the toolset preparation at the TSC if the toolset is not already available in the workcell. After assembling all tools from the toolset, the toolset is delivered to the workcell. If tools cannot be assembled on time due to unavailable tool parts at the TSC, the team leader adjusts and reschedules the workorders according to the process in the planning phase if necessary. Meanwhile, MH delivers the product material to the workcell, while the operator replaces any standard tools from the milling machine if required. If the standard tools cannot be replaced on time, the operator assesses the workorder production schedule and planning for adjustments according to the process in the planning phase if necessary as well.

After the product material is delivered, the operator clamps the products if required and provides them to the cell. The robot retrieves the products, clamps the product at the clamping machine if required and stores the clamped product material. When applicable, the operator provides

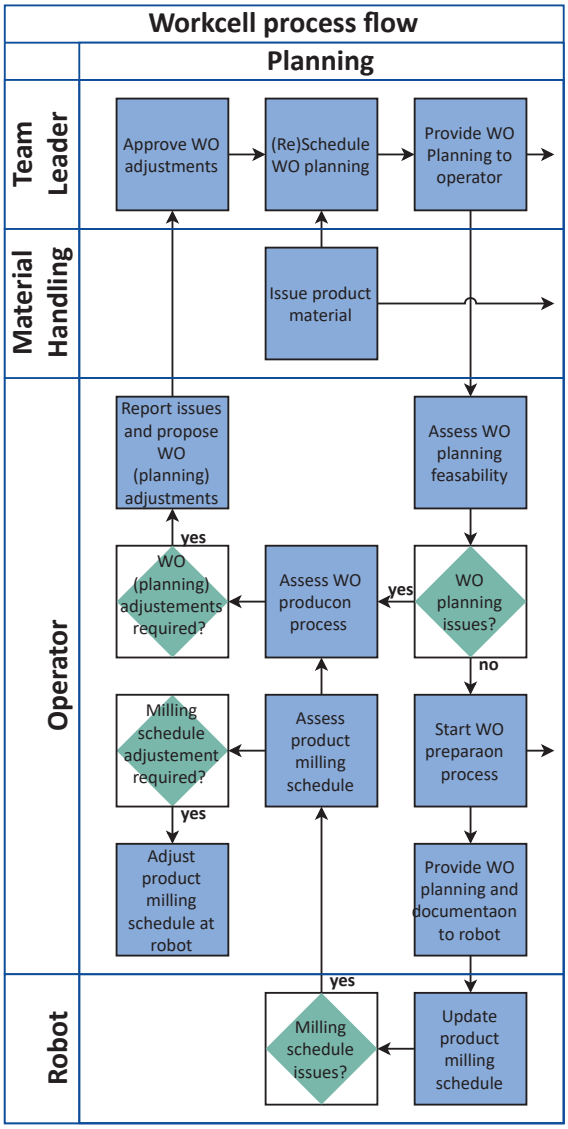

Figure 2.7: Planning phase of workcell process flow.

the tools from the standard toolset to the cell, where the robot retrieves and stores each tool. In case of insufficient storage space for new tools in the cell, the operator determines which toolsets are removed from the cell based on the toolset statuses, experience and estimates of use for future workorders. The chosen toolsets are provided by the robot and retrieved by the operator before loading the new toolset into the cell. Removed toolsets are delivered at the TSC for disassembly. If both the clamped product material and toolsets are available in cell, the workorder production process is started. In case the product material and toolsets are not available according to the milling schedule, any consequent milling schedule and subsequent workorder process issues are assessed according to the workcell process in the planning phase.

2.3.3 Production Phase

The production phase of the workcell process flow consists of the product milling process. The production phase process flow is shown in Figure 2.9, assuming products pass quality inspection

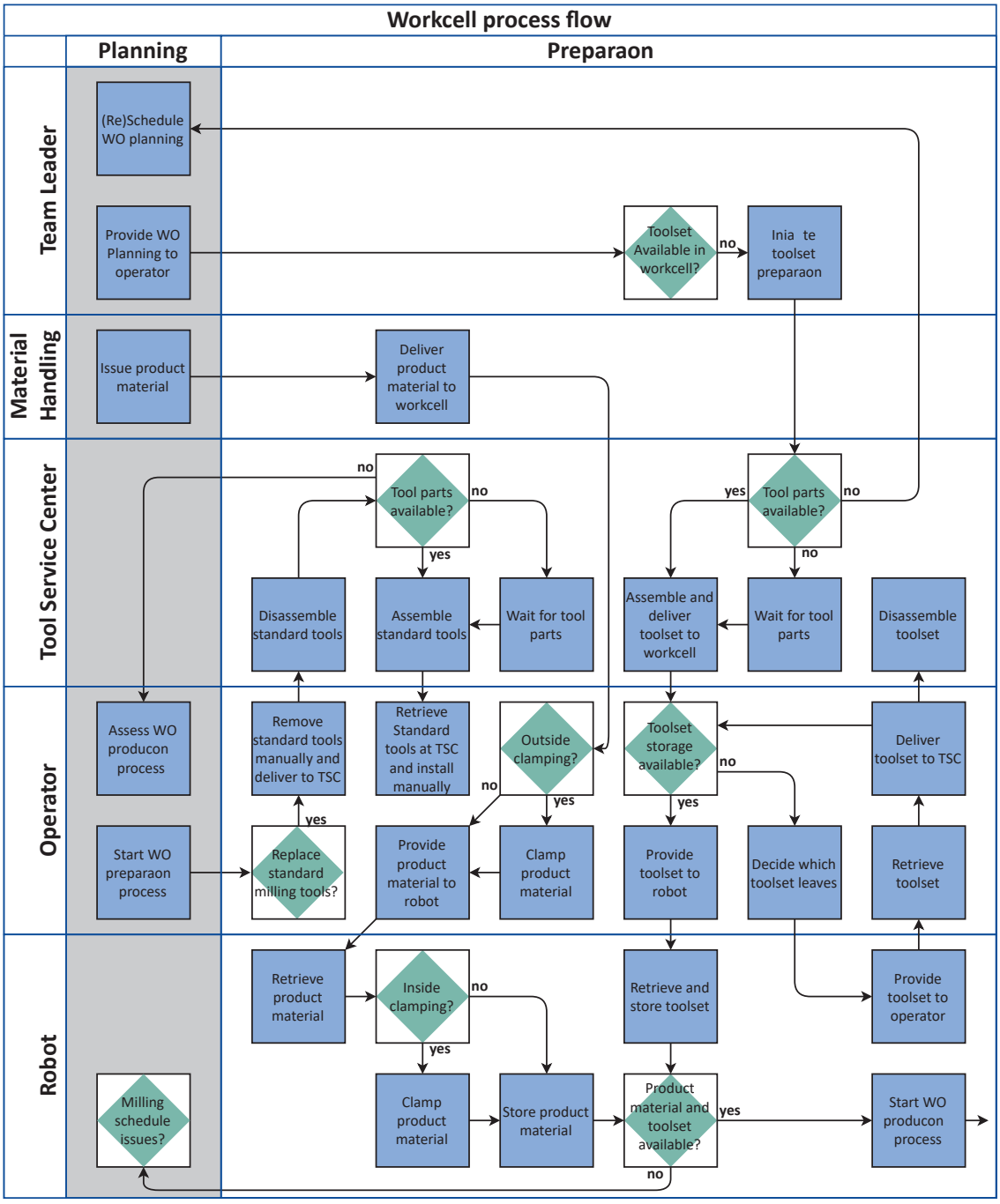

Figure 2.8: Preparation phase of workcell process flow.

and no tool failure during milling process. After the workorder production process is started, the products from the workorder are milled according to the milling production schedule for each machine, with the first product of a workorder always approved for production. Per machine, after the current milling process is finished and the machine is available, the milling process for the next scheduled product is started. If the milling machine is not available according to schedule, any consequent milling schedule and subsequent workorder process issues are assessed according to the workcell process in the planning phase. If the next scheduled product type is milled for the first time, the operator installs the CNC program on the milling machine and supervises the first milling process. If not in the milling machine, the tools from the special toolset are provided by the robot and loaded into the toolbelt by the milling machine spindle, before the product is provided to the milling machine by the robot and the milling process is started. Assuming no tool failures or other problems, the milling process is completed.

After completing the milling process, the product is retrieved by the robot. If the special toolset is not used in the next milling process, all tools from the toolset are provided to the cell for storage by the robot as well. The milling process is finished, allowing the next process to start according to schedule. If the milled product completes the workorder production process, the workorder finalization process is started.

If the product requires inspection at QC, the product is provided to the operator by the robot, and either unclamped inside the cell before or unclamped by the operator after. The operator delivers the product at QC. After inspection, the product and result of the inspection is delivered to the workcell. Assuming the product passed inspection, the operator approves the further production of all products using the same special toolset, CNC program and milling machine, and the workorder production process continues. If the product is removed from the workorder and to be discarded, the product is removed from the cell using the same process before being discarded by the operator.

Figure 2.10 shows the production phase process workcell process flow after a tool fails during milling or a milled product fails inspection. If a tool breaks or fails during the milling process, the process is paused and the failed tool has to be replaced by the operator. Any problems with the milling schedule due to the tool failure are assessed according to the workcell process in the planning phase. The operator removes the tool manually from the milling machine and delivers it to the TSC where a replacement tool is immediately assembled, assuming all tool parts are in stock. The replacement tool is manually loaded into the machine toolbelt and the milling production is continued. After the milling process is finished, the milled product is inspected at QC. Since a tool is replaced, all approved production using this tool is revoked until a product passes inspection at QC.

In case a milled product fails quality inspection or a failed tool cannot be immediately replaced, the operator assesses and solves subsequent product and workorder process issues. Paused milling programs are aborted if they cannot continue due to a missing tool, finishing the milling program. Required adjustments to the milling production schedule are made. Workorder adjustments, including removing failed or partially milled products when appropriate from workorders and splitting workorders, are proposed, and workorder planning issues are reported to the team leader. For (adjusted) workorders consisting of only milled products, or for which further production is cancelled or suspended, the finalization process is started. For (adjusted) workorders that continue the production process, the next product is approved for production. Any products removed from workorders are first retrieved from the cell when applicable and discarded. When the delayed replacement tool for a failed tool is ready, it is retrieved by the operator and manually loaded into the milling machine. The milling machine provides the robot with the tool and is stored in the workcell completing the toolset for further use, or removal by the operator from

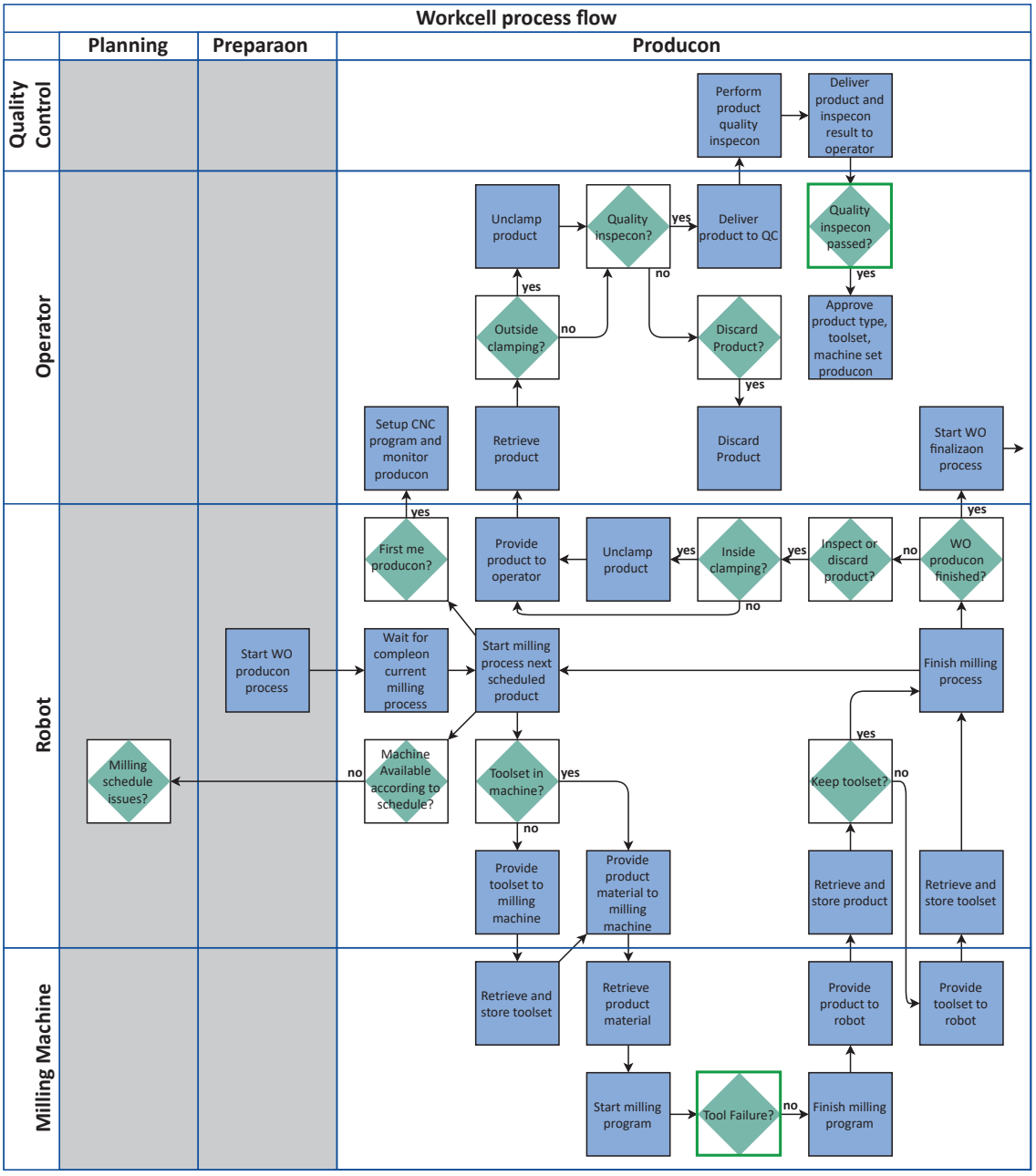

Figure 2.9: Production phase of workcell process flow, assuming no tool failure and quality inspection fail.

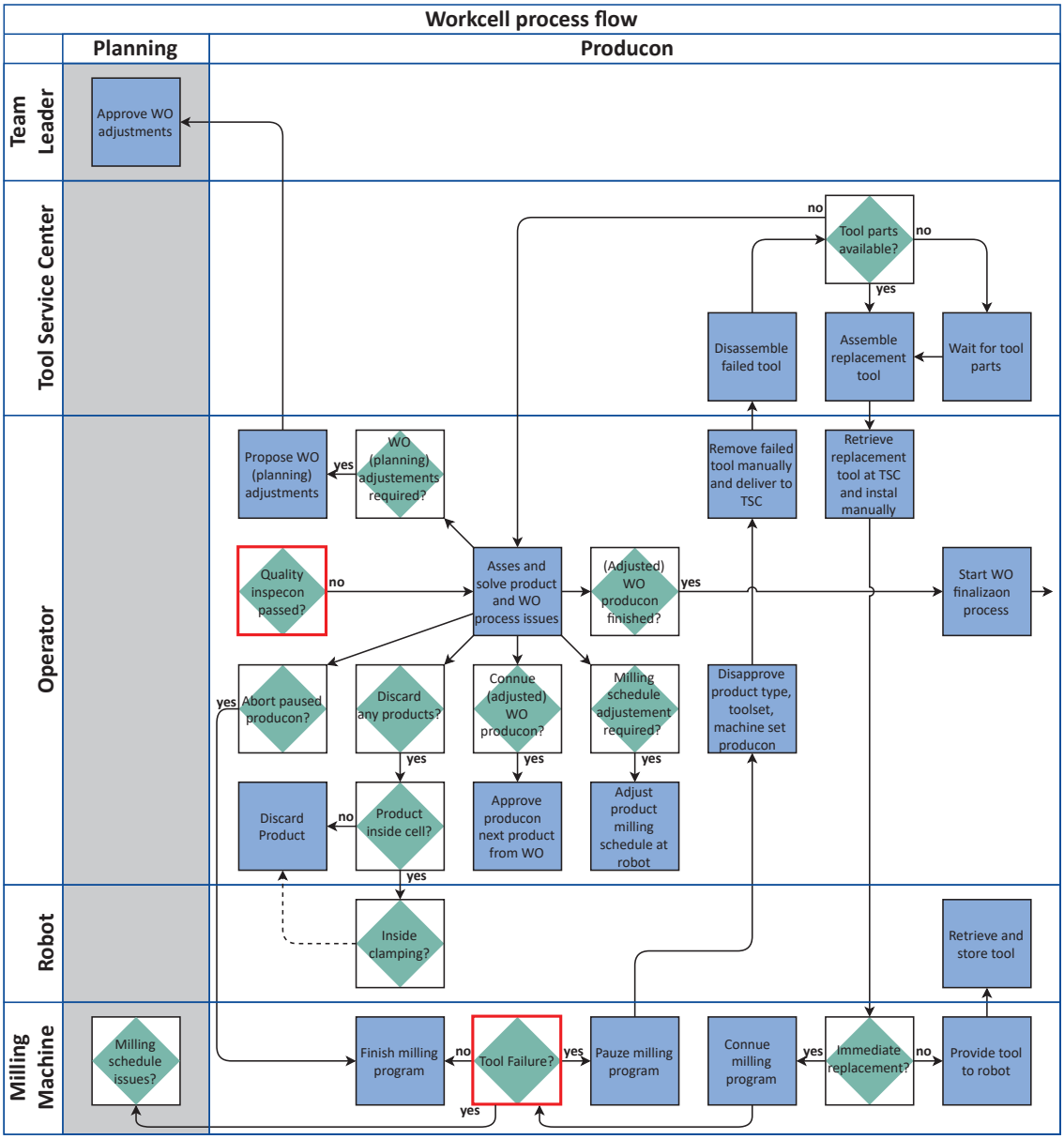

Figure 2.10: Tool failure and quality inspection fail process in the production phase of workcell process flow.

the cell when tool storage space is required for new toolsets according to the process in the preparation phase.

2.3.4 Finalization Phase

The finalization phase of the workcell process flow, shown in Figure 2.11, consists of the workorder finalization process. The workorder products are provided by the robot, retrieved by the operator, and unclamped at the appropriate location. Milled products are inspected by the operator, discarding faulty products and adjusting the workorder accordingly. The workorder product batch is retrieved by MH for further processing, ending the workorder process flow.

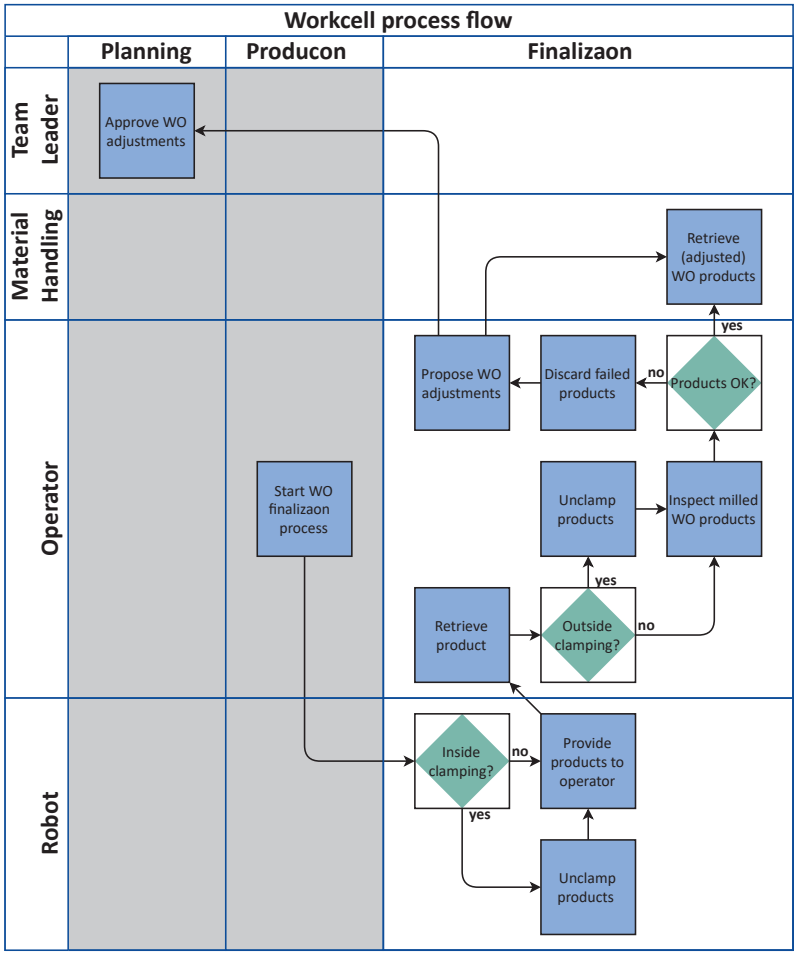

Figure 2.11: Finalization phase of workcell process flow.

2.4 Performance Measurements

In order to compare the current workcell performance with automation solutions of the logistical process at all levels in the context of the AML project, as well as workcell optimization solutions, through simulation models, performance measurements are used. In this section, these performance measurements and their importance are stated.

Given the primary function of the workcell is the milling of products, the utilization of the milling machines is an important performance measurement. The utilization of the milling machines is defined as the fraction of time the machine is executing milling production through CNC programming. Ideally, the utilization of the machines is as high as possible and close to 100 percent. The utilization of the milling machines can be improved by reducing CNC program setup times, minimizing the number of tool exchanges before and after milling, waiting on the operator or robot to provide or retrieve required tools, waiting on the robot to provide or retrieve products, minimizing tool or machine failures during milling, and reducing maintenance and repair times.

Directly related to the machine utilization is measuring the number of tool and toolset changes compared to the number of milled products per machine. Where a reduction in the number of tool and toolset changes would increase the machine utilization, given other factors remain equal.

Other important performance measurements for the workcell are the throughput and lead times of processing workorders. The lead times of workorders can be measured from the moment of scheduling until processed at the workcell and transferred over for further processing in the factory, or the time between arrival of the product batch material at the workcell and departure of the milled product batch from the workcell. The throughput and lead times can be affected by multiple factors, including the workcell workorder planning, milling machine production schedule, milling production and (un)clamping times, workorder batch size, operator availability, robot and machine availability, as well as machine, robot arm or other workcell component failure, maintenance or repair.

After stating important workcell performance measurements, the information and description of the workcell and its components, functionalities, relations and dependencies both within the workcell and between the workcell and the factory, the flow of products and tools throughout the workcell, and an overview of the workcell process flow in this chapter, is used to develop the simulation model of a workcell. The simulation method, assumptions made, and the developed workcell simulation are described in the next chapter.

Chapter 3

Simulation Model

In the previous chapter, the workcells at KMWE Precision Components where described and discussed. At the workcell, products are milled through processing workorders. To gain insight in all aspects of the workcell, the workcell components, functionalities, internal relations and dependencies, and relations and dependencies within the factory where detailed. The flow of products and tools throughout the workcell was described as well, before providing an overview of the workcell process flow, showing the workorder processing. Finally, important performance measurements of the workcell were introduced.

The workcell information, described in Chapter 2, was used as a blueprint to develop a simulation model of the workcell. In this chapter, the modeling methods, software package, modeling choices, and the developed workcell simulation model are described. In Section 3.1, the modeling methods and software packages used to develop the simulation model are introduced. Next, in Section 3.2, the model design choices and assumptions are discussed. In the remainder of this chapter, the developed workcell simulation model is described. Section 3.3 describes the global structure of the model and defines the agents the model consists of. Section 3.4 details the structural agents that form the foundation of the simulation model, followed by the incorporated process flowcharts in Section 3.5 and the rest of the agents in Section 3.6 and Section 3.7. Section 3.8 provides an overview and discussion of the functions and algorithms used throughout the model. Finally, an overview of the simulation methods and options is provided in Section 3.9, followed by an overview of the workcell model performance measurements and other statistics collected during simulation in Section 3.10.

3.1 Modeling Method and Software

The development of a simulation model of the workcell is part of the AML project introduced in Chapter 1. Within the context of the AML project, the use of simulation models can be used to gain better insight in the current production process and investigate automated logistical solutions. In order to develop simulation models, software packages can be used to provide a platform, including tools and pre build libraries, to develop models and run simulations. For the development of a workcell simulation model, as well as other simulation models within the AML project, a suitable software package was chosen. Important requirements used to determine the software package included discrete event simulation capabilities with supporting tools for manufacturing and logistical systems, the capability to integrate multiple models or the

independent development of submodels within a model, the option to include custom algorithms or coding to add or alter functionalities included in the software tools and libraries. Other factors taken into consideration in determining the software package where the import and export capabilities of data, the capability to visually display simulation models in 3D and import custom 3D models, user friendliness and previous experience with the software package and licensing costs.

After investigating, reviewing and comparing several software packages, AnyLogic was chosen. One important reason for choosing AnyLogic is the integration of multiple methods for developing simulation models, combining discrete event, agent based and system dynamics methods within one software package. Other important factors for choosing AnyLogic include the capability to integrate multiple models into one model, extensive capabilities of adding functionalities and customizing included tools and libraries within AnyLogic through java based coding and algorithms, and previous experience with AnyLogic at the Eindhoven University of Technology.

Using AnyLogic, multiple modeling and simulation methods can be used when developing simulation models. According to AnyLogic [1], the idea of this multimethod modeling is to seamlessly integrate different modeling and simulation methods to overcome the drawbacks of individual methodologies and get the most from each one. As mentioned before, the three major methods for building dynamic simulation models are discrete event modeling, agent based modeling and system dynamics [1].

With the discrete event method, processes are modeled as a sequence of discrete events. Most system processes can be described as such, and discrete event modeling and simulation is widely used in manufacturing and logistic fields [2]. Agent based modeling focusses on the active components of a system. These active entities, or agents, must be identified and their behavior, connections and environment defined in order to model and simulate the system [3]. System dynamics is an abstract method of modeling. It ignores the details of a system, but models and simulates a general representation of the system instead [4].

Again, according to AnyLogic [1], combining the three methodologies into one multimethod software package allows for modeling different parts of complex systems using different methods, and integrate all parts into one simulation model of the system. If a system consists of many independent objects or entities, the agent based method can be used. If the available information about a system mainly contains global or high level dependencies and relations, the system dynamics approach can be used. If a system can be easily described as a process, the discrete event approach can be used. If a system has two or all three aspects, a combination of the modeling and simulation methods can be considered [1]. The workcell simulation model is developed using a combination of the discrete event and agent based methodology.

3.2 Modeling Choices

As defined in Chapter 1, the purpose of a simulation model of a single workcell is to function as a digital twin in order to simulate, analyze and predict workcell performance. Using AnyLogic, a workcell simulation model is developed with this purpose in mind. The model development started with modeling the core components and basic functionality of a workcell, milling products at the milling machines. During the model development, more functionalities and capabilities where added to increasingly better represent the real workcell at KMWE Precision Components. The modeling choices, assumptions and differences between the workcells and the developed simulation model of a workcell are stated next.

Before describing the modeling choices, a few general modeling choices made during the development process that are different in the workcell model compared to a KMWE workcell are stated. First, the workcell model does not simulate any maintenance or workcell component failure and repairs. This includes both milling machines, the clamping machine and the robot arm. However, tool failure during milling and item quality inspection failure are included in the simulation model. Second, material handling tools and equipment, chip containers and coolant agent are assumed to be always available. The number of pallets available in the workcell is equal to the pallet storage capacity within the cell. Finally, the first time setup for a new item type, including its CNC program installation is not included in the simulation model.

As introduced in the previous section, the developed simulation is based on a combination of discrete event and agent based modeling methods. In the simulation model, process flow modeling, the discrete event method, is used for entities that are physically processed or transported within the workcell. These entities are items (product material or finished products), batches of items referenced as itembatches, tools, toolsets, pallets, and chip containers. The operator, robot(arm), milling machines and clamping machine are used as resources within the process flowcharts of the model. The clamping and material handling tools and equipment, and the coolant agent are not modeled. State charts, using the agent based method, are used to track the status of resources, workorders, itembatches, toolsets and both exchange areas. The Tool Service Center and Quality Control are represented in the model as locations outside the workcell, where the TSC location is used to replace standard and broken tools and the QC location for inspecting milled items for further production approval.

In contrast with the workorder and workorder planning process as detailed in Chapter 2, created workorders in the workcell model are processed on a itembatch arrival first come first serve basis. The workorders in the model contain information about the itembatch and toolset needed to process the workorder. This information, representing the workorder characteristics, is randomly picked from databases or determined through distribution functions. After an optional time delay, the itembatch from the workorder is created and arrives at the workcell. Itembatches are split into individual items, clamped and entered into the cell when the required empty pallets are available for the whole batch in the cell. If an available toolset can only mill part of the workorder, the workorder is split into two workorders or suborders, one workorder with a batch size to be completed with the available toolset and a second workorder for which a new toolset is needed. If the toolset is not present and available in the cell, after an optional delay the toolset is created and arrives at the workcell as well. After all items of a workorder are loaded into the cell, the toolset is split and each tool is loaded into the cell, provided the cell tool inventory has the required storage space for the whole toolset. If not, unused toolsets are removed from the cell first until enough storage space is available. If both the items from the itembatch and tools from the toolset are stored inside the cell, the milling process can begin.

In the workcell model, production is approved per workorder after milling the first item of the workorder and successfully passing the quality inspection. This means the first item of a workorder is always approved for production, with the rest of the batch approved after passing inspection. Approved items are milled at the assigned milling machine on a first come first serve basis as well. If a tool fails during milling, the workorder items do not have to be approved again if the broken tool can be immediately replaced. Should an item fail quality inspection, the item is removed from the workcell model. If the batch size of the workorder is equal to one, the workorder is removed from the workcell as well. If not, the workorder batch size is adjusted, the operator performs a milling production assessment at the milling machine, and the next item is milled and inspected. This second inspection is always passed. No new workorders are created, or new items added to existing workorders, to account for failed and removed items from workorders at the workcell.

Besides performing milling assessments, the operator also replaces standard tools when required and broken tools after tool failure during milling. The milling machine can continue milling, and exchanging tools and items with the robot, while the operator is performing an assessment or exchanging tools at the machine. However, the milling machine requires all tools from the standard toolset and the workorder specific toolset available in the milling machine inventory before starting or resuming milling the item. The operator replaces standard and broken tools at the TSC location per tool. Standard tools are always immediately replaced. Broken tools can be immediately replaced or replaced after a significant delay, simulating missing tool parts. If a tool fails during milling, the milling process is paused and resumed after the operator has replaced the broken tool, or is aborted if the broken tool cannot be replaced immediately. If the milling process is aborted, the workorder can be split in two suborders if one or more of the workorder items are already milled. The first workorder contains the milled items, while the second workorder contains the rest of the items, including the item from the aborted process. The aborted item is milled again from start after the broken tool is replaced, and counts as the first item of the new workorder, requiring a quality inspection before further production is approved. In either case, there is no inspection by the operator of the milling machine and the item in question when a tool fails during milling.

3.3 Model Structure and Agent Definitions

Before describing the workcell model structure, a brief overview of the structure and modeling components for models created in AnyLogic is given. As introduced in Section 3.1, AnyLogic combines the discrete event, agent based and system dynamics modeling methods into one multimethod simulation modeling software package. Regardless of which modeling method, or combination of methods, is used to develop simulation models, the basic structure a simulation model in AnyLogic consists of is agents. Agents are created and defined by the user in AnyLogic as agent types. The agent type that represents the top level or main agent is used as the starting point of running the simulation model, and therefore contains the starting environment of the model. This agent type, and all other created agent types, contain the complete model. In the main agent type, agents can be embedded as single agents or populations of agents. Consequently, each embedded agent can embed other agents or populations of agents as well to any desired depth. A population of agents can be used to dynamically add and remove agents defined by the same agent type during simulation at the agent in which the population of agents is embedded.

Each agent type defines agents that represent a part of the model and, besides embedding other agents, can contain any of the model objects, components and functionalities provided by AnyLogic. These include parameters, variables, events, functions, 2D and 3D presentation and animation, control buttons, space markups, statecharts, stock and flow diagrams, process flowcharts, as well as data collections, statistics and graphical charts for analysis. All these various components are graphically displayed in the AnyLogic model development environment, where the properties of each agent type and each component it contains can be set, viewed and altered when selected.

Agent types can contain an environment for other agents, process flowcharts and system dynamics diagrams, as well as to model the behavior and track the state of an entity or object in an environment, to be used as an entity or resource in a process flowchart, or anything else required in the model. Besides agent types and anything it can contain, databases can be created or imported into the model. Finally, the model is simulated by creating a simulation experiment. The standard simulation allows the user to create a simulation start screen and set options for the simulation, including setting or changing the parameters of the main agent.

The workcell simulation model is developed using both the agent based and discrete event modeling method, using a combination of mostly process flowcharts, statecharts, functions and events to model the behavior of the workcell. Most of the created and developed agent types the model consists of can be divided into three groups, structural agent types, material agent types and resource agent types. The structural agent types define the agents that contain the environment of the workcell and the process flowcharts. Material agent types are used to define agents that model entities that are transported, handled and processed throughout the workcell. Resource agent types define the agents that model entities, objects or components that are used as resources to transport, handle or process the material agents.

Next, all agent types, the agents they define and how the agents are connected to create the simulation model are introduced. In AnyLogic, agent types start with a capital letter, while agents and populations of agents start with a small letter. In addition, all agent types and agents in the workcell model start with *"WCE_"* and *"wCE_"* respectively, a naming convention adopted in earlier development to avoid naming conflicts for integration with other models in the AML project.

3.3.1 Agent Types and Global Model Structure

The workcell model contains three structural agent types to model the environment of the workcell based on the operator, robot and milling machine domains as defined in Section 2.1 of the previous chapter. The first structural agent type, *WCE_OutsideCell*, is used as the top level or main agent of the model, referenced as *wCE_OutsideCell*. This agent contains the environment of the operator domain, or outside the cell, including representations for the TSC and QC locations. The second structural agent type, *WCE_Cell*, contains the environment for the robot domain inside the cell, while the environment for both milling machines is contained in the third structural agent type, *WCE_MillingMachine*. The environment of the workcell domains in each structural agent type is schematically represented by space markups. The agent type *WCE_Cell* is used to embed the agent *wCE_Cell* in the agent type *WCE_OutsideCell*, while *WCE_MillingMachine* is used to embed the agents *wCE_MillingMachine1* and *wCE_MillingMachine2* in *WCE_OutsideCell*.

Besides the workcell environment, all three structural agent types contain process flowcharts, using material and resource agents to model the transportation, handling and processing of items, tools, pallets and chip containers by the operator, robot, both milling machines and clamping machine. A more detailed description of the structural agents can be found in the next section, Section 3.4, while the process flowcharts are described and shown in Section 3.5.

The material agent types are used to define the material agents that are created during simulation. As discussed and shown in Section 2.2 of the previous chapter, these agents are modeled after the entities and objects that flow through the workcell. The *WCE_Workorder* agent type defines the *wCE_Workorder* agents in the simulation that are modeled after the workorder documentation. Workorder agents hold all information of the items to be produced and the special toolset required for milling, are used to trigger the creation of agents modeling the item batches and toolsets, and tracks through a statechart if the item batch is in the workcell or not.

In the model, batches of items are represented by unique agents, instead of being a collection or population of agents representing single items. The *WCE_ItemBatch* agent type defines the *wCE_ItemBatch* agents that model batches of items. Itembatch agents track the state of the production process of the entire batch and can trigger functions or other processes in the model when changing states through a statechart. In addition, the agents are used in a process flowchart to simulate the flow of itembatches before splitting and after combining in the workcell.

Finally, itembatch agents trigger the creation of agents representing the single items the batch consists of when the itembatch agent is split during simulation. The *WCE_Item* agent type defines the *wCE_Item* agents that model the items created during simulation. Item agents are used in the process flowcharts and the state of each item agent is tracked through variables.

Similar to itembatches, toolsets are represented as unique agents in the model as well. The agent type *WCE_Toolset* defines the *wCE_Toolset* agents created during simulation. Equivalent to the item batch agents, the toolset agents track its state throughout the workcell using a statechart, can trigger functions or other processes on state changes, are used in process flowcharts before splitting and after combining, and trigger the creation of the individual tools of the toolset when the toolset is split as well. The *WCE_Tool* agent type defines the *wCE_Tool* agents that model the tools created during simulation. Tool agents are used in the process flowcharts, and the state of each Tool agent is tracked through variables. Finally, similar to tools and items, the agent types *WCE_Pallet* and *WCE_ChipContainer* define the agents *wCE_Pallet* and *wCE_ChipContainer*, that model pallets and chip containers when created during simulation.

Since material agents are created and oftentimes destroyed during simulation runtime, a population of agents for each material agent type is embedded in the structural agent type *WCE_OutsideCell*. For example, the agent type *WCE_Workorder* is used to create the population of agents *wCE_Workorders* that can contain, create and destroy agents *wCE_workorder* defined by *WCE_Workorder*. The other material populations of agents use equivalent naming. Created material agents can be used in the appropriate process flowcharts of all structural agents, as the other structural agents are also embedded in *WCE_OutsideCell* as well. The material agents are further described in Section 3.7

The resource agent types define the resource agents that model the operator, robot arm, clamping machine, and both milling machines. Resource agents are used by resource pools in the process flowcharts, limiting the use per resource agent to one material agent used in the process flowcharts concurrently. Besides being used by resource pools, each resource agent contains a statechart to track its state, and most have more functionalities beyond serving as agents in resource pools. This is especially true for both milling machine agents, as they also serve as structural agents by containing the milling machine environment as well. As mentioned before, both milling machine agents, *wCE_MillingMachine1* and *wCE_MillingMachine2*, are embedded in *WCE_OutsideCell*. The operator agent *wCE_Operator*, defined by the agent type *WCE_Operator*, is also embedded in *WCE_OutsideCell*. Since the robot arm and clamping machine are only used inside the cell, or the robot domain, they are modeled by the agents *wCE_Robot* and *wCE_ClampingMachine*, defined by the agents types *WCE_Robot* and *WCE_ClampingMachine* respectively, and embedded in the structural agent type *WCE_Cell*.

The final two agent types, used to define the agents modeling both exchange areas, are similar to resource agent types, except they are not used as resources by resources pools in the process flowcharts. The agent type *WCE_PalletExchange* defines the agent *wCE_PalletExchange* that models the exchange of pallets, clamped items and tools at the pallet exchange area, while the agent type *WCE_ItemExchange* defines the agent *wCE_ItemExchange* that models the exchange of items at the item exchange area, called the product exchange area in the previous chapter. Both agents use statecharts and functions to track the state of the exchange areas and control the exchange of the relevant material agents between the operator and robot domain environments in the *wCE_OutsideCell* and *wCE_Cell* agents. Both agents are embedded in *WCE_OutsideCell*. A more detailed description of these agents and the resource agents can be found in Section 3.6.

Besides the agent types the workcell model consists of simulation experiments, databases and resources. The simulation experiments are used to setup and run simulations of the workcell

model. The standard simulation experiment is used to run the model with specified parameters and supports animation, real time mode, and debugging. The parameters variation experiment is used to run multiple simulations, varying one or more parameters per iteration. Databases are used to provide global data that can be accessed by all agents. Data can be imported or directly entered in created databases in the model. Finally, resources contain links to external files used by the workcell simulation model, consisting of excel and text files to import and export data, images and 3D models. A more detailed description of the simulation experiments, simulation options and the use of databases can be found in Section 3.9.

Figure 3.1 illustrates the global structure of the workcell simulation model, listing the agent types and other components that constitutes the model introduced in this section. The figure further shows where all agents, defined by their respective agent types, are embedded. For each agent it lists the environment it contains when applicable, whether it contains process flowcharts and which agents are used in these flowcharts, whether it contains a statechart, and whether it is used in process flowcharts.

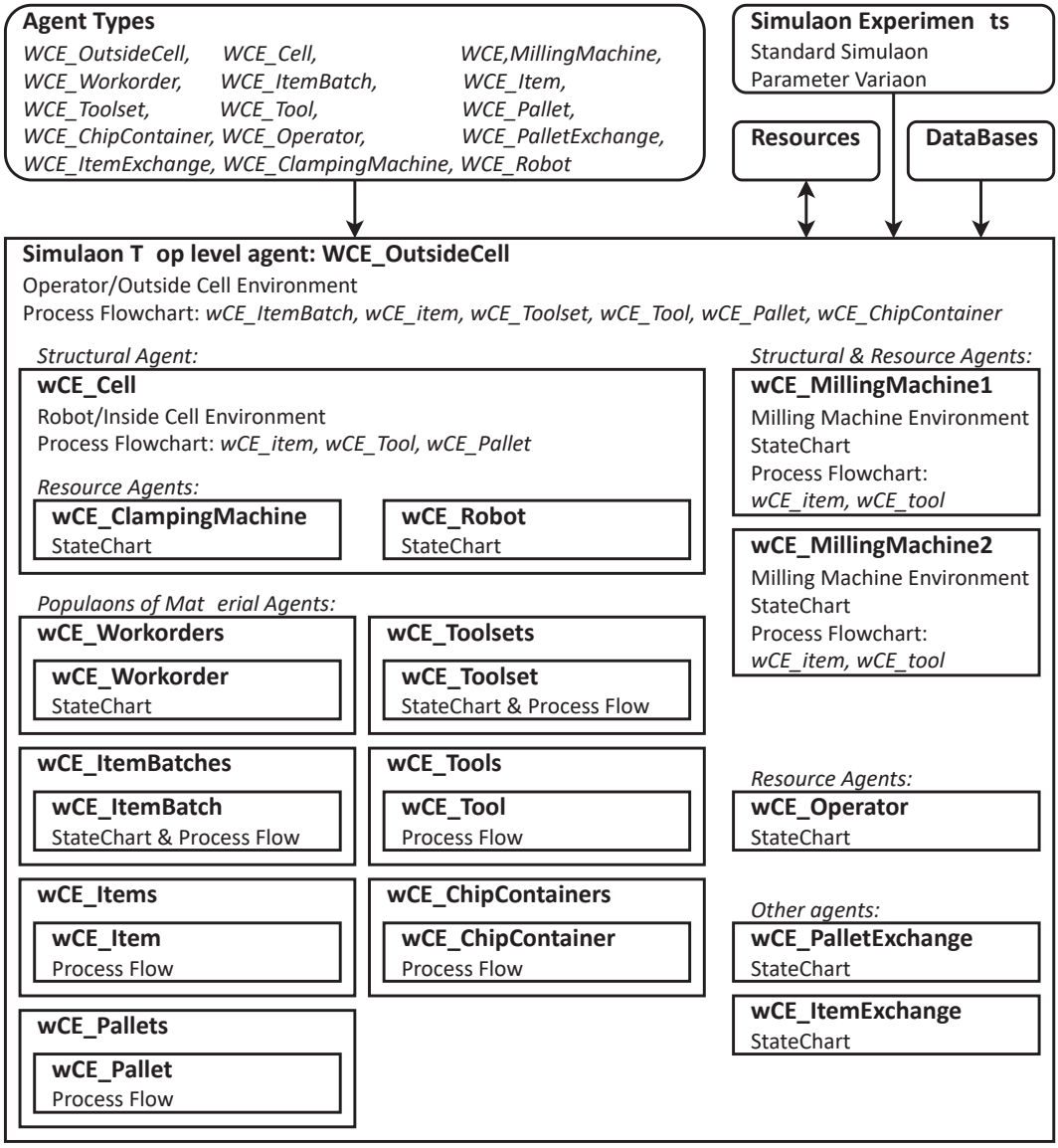

Figure 3.1: Global structure of the workcell simulation model.

3.3.2 General Agent Content and Java Based Programming

Before detailing each agent type and the other components of the workcell model in the remainder of this chapter, an overview is given of general contents common among most or all agents, after the introduction of Java based programming used throughout the model to define and determine the behavior of agents and their components during simulation.

Java is a high level object oriented programming language that uses a class structure to define objects, where objects contain variables and methods, called functions in AnyLogic. A simulation model developed in AnyLogic model is based on the Java class structure and completely mapped into java code. The Java class structures are automatically modeled by AnyLogic. Almost all objects or elements a simulation model consists of are instances of predefined Java classes and no custom classes are created for the workcell model. Therefore, Java code programming used throughout the model consists of expressions, function calls and statements, and is written in the model object properties.

Each agent type created in the workcell model is an instance of the Java class agent, and each agent type is used as a class structure itself for the agents based on each agent type. The parameters and variables contained in the agent types can be defined as an instance of a primitive data type or any Java class, including the agent types the model consists of. In the workcell model both are almost always defined as an instance of one of four primitive data types. These data types are `int` (representing integer numbers), `double` (representing real numbers), `boolean` (representing the boolean values true or false), and `String` (representing text strings). Parameters are typically used to represent agent characteristics and can be used to hold values for user control methods during simulation setup and runs including setting simulation options through sliders and boxes. Variables are typically used to represent agent states. A collection part of an agent type is used as a list of instances of the defined data type or class in the collection properties. Instances of the defined type or class can be added to, sorted, and removed from the collection during simulation runtime.

Variables or lists of a variable can be declared locally in a block of Java code statements, for example in the body of a function, or in an action defined in the properties of an agent type or event. These variables and lists only exist during the execution of the block of statements they are contained in. These variables or list of variables are commonly declared to assign one or more agents currently existing in the material agent populations to alter their parameters or variables, execute one of their functions or events, trigger their statechart state transitions, or progress them further in the process flowchart blocks through Java code programming.

Functions in AnyLogic are available for system wide use or part of Java classes. The system wide available functions include mathematical operations, probability distributions, writing text output to console or text files, and returning the current simulation time. Functions part of any of the objects used in the model include adding and removing variables from collections, restarting events, releasing agents from process flowchart blocks, and adding new agents to agent populations. Custom functions can be added to agent types as well, with many of these included in the workcell model.

Functions are called in Java code by the function name followed by parenthesis: `functionName()`. Some functions require one or more input arguments written between the function call parentheses: `functionName(argument1, argument2, ...)`. Some functions return a data type or class value as well, which can be assigned to a variable in the Java code body. For example, the statement `double currentTime = time()` declares the double variable *currentTime* and assigns the current simulation time returned by the function as the initial value of the *currentTime*

variable. Functions that return a value can be called without assigning the return value to a variable, in which case the return value is discarded.

For functions graphically added to the agent types throughout the model, the optional function arguments, the Java code body defining what actions the function executes when called, and the optional return value type or class are set through the function properties. Java code statements can be added to many of the objects or classes added to the agent types through actions part of the object or class properties. Actions defined in the workcell model include actions for agent types, process flowchart blocks, schedules, events, as well as statechart states and transitions.

Events execute their action when triggered, with their trigger conditions defined in the properties. Most events in the workcell model are triggered through user controlled timeouts, where a default timeout time can be defined in the event propertied or entered as an argument in the event restart function to be used instead. When the event restart function is called the event action is scheduled for execution at exactly the timeout time after the event is started by the restart function. Instead of user controlled, events can also be scheduled cyclicly at the first specified occurrence time from simulation start and at exactly the specified recurrence time after execution afterwards. Some events in the workcell model are executed at occurrences by a specified rate as well.

Schedules are used in the model to specify on and off moments or intervals on a weekly time schedule basis. The schedule action is executed each time the specified on or off moments or the interval start and end are scheduled. Agents can execute actions defined in the agent type properties on agent startup and removal from the simulation. The actions for process flowchart blocks and statechart states and transitions are described later in this chapter.

Each agent type contains one or more viewareas to view the agents status, statechart, process flowcharts and model statistics during simulation if included in the particular agent. Each agent type also contains one or more navigation bars and functionalities to navigate between viewareas of agents during simulation as well.

Throughout many of the agent type elements, especially in the process flowchart blocks and statecharts, data is collected about the model. This data is used to calculate and show the model performance and other statistics during simulation. In addition, text files are setup and can be used by all agents during simulation to write data into these text files for use after simulation runs. Both shown data during simulation and exported data are used in the model verification process and experiments. A more detailed description of the collection of data, calculating performance measurements and exporting data can be found in Section 3.10.

Throughout the remainder of the chapter, The agent types and other components that constitute the model, as well as how the agents interact during simulation runtime, are further described. After describing all agents and the process flowcharts, Section 3.8 expands upon the more complex functions and algorithms that determine the progression of processing workorders by the workcell model during simulation.

3.4 Structural Agents

This section will further detail the structural agents defined by their respective agent types introduced in the previous section. In the remainder of this report, both agents and agents types will be referred to as agents, unless the distinction between the agents and agent types is

relevant. For each structural agent all aspects of the model the agent contains is introduced and the environment these agents provide is expanded upon.

3.4.1 Top Level and Operator Domain Agent

The first structural agent, *wCE_OutsideCell*, serves as the top level or main agent, besides providing the environment of the operator domain in the model. Therefore, this agent contains many other elements besides the outer workcell environment and process flowcharts. These contents of *wCE_OutsideCell* are briefly introduced before describing the workcell environment of the agent.

Since each simulation run of the workcell model starts at this agent, *wCE_OutsideCell* contains parameters that hold values used throughout the model to determine the model behavior based on the simulation options. The agent also contains variables holding values retrieved from the model databases containing general processing and handling times for use throughout the model. Text file links, variables, functions and viewareas to calculate, display and export the workcell performance measurements and other statistics are included in the main agent as well.

The top level agent also contains various process flowcharts. First, it contains process flowcharts for itembatch and toolset agents, modeling the flow of both from arrival to splitting in individual items and tools, and from combining into itembatches and toolsets to departure. Second, it contains process flowcharts for item agents in the operator domain, modeling the outside clamping and unclamping process, loading into and retrieving items from the cell through the exchange areas, and the quality inspection of items at QC as well. Next, it contains process flowcharts for tool agents in the operator domain, modeling loading into and retrieving tools from the cell through the pallet exchange area, retrieving standard and broken tools from the machine toolbelts, replacing standard and broken tools at TSC, and loading replacement tools back into the machine toolbelts. It also contains a process flowchart for pallet agents in the operator domain, modeling the retrieval of pallets from the cell for clamping and loading pallets into the cell after unclamping through the pallet exchange. Finally, it contains the process flowchart for chip container agents, modeling the arrival of empty containers and departure of full chip containers at each milling machine. The resource pools used in the various process flowcharts, using the resource agents modeling the operator and both milling machines, are contained in *wCE_OutsideCell* as well.

The agent *wCE_OutsideCell* contains various collections, schedules, events and functions used to determine many aspects of the workcell model behavior, including the workcell initialization at the start of each simulation run, creating workorders and the setup of newly added itembatches and toolset, and partially controlling the progression of the item milling process.

Finally, as introduced in the previous section, all material agent populations are embedded in *WCE_OutsideCell*. For each agent population a variable is included to assign a unique ID number to each agent added to the respective population during simulation. The cell and both milling machine agents are also embedded in this agent, as well as the operator and both exchange area agents.

In the workcell model, each environment provided by the structural agents is defined by space markups. The space markups consists of nodes and paths. Nodes represent a location or area part of the workcell, while paths define the links or routes between the nodes. The nodes and paths, representing the workcell environment, are used by the model to animate the material agents during simulation, visually showing the agents in 2D or 3D at their current location

(node) or moving between locations along the paths connecting the nodes. Nodes representing workcell locations or areas contain attractors to define the possible locations and orientations for the agent presentation when located at the node. Nodes are also used to split or combine paths between the nodes representing workcell areas.

The top level and structural agent *wCE_OutsideCell* provides the outer cell, or operator domain, environment. Since the other structural agents, are embedded in this agent, all environments are combined and presented as the complete workcell environment at this agent as well. An image with a schematic layout of the workcell for the model, shown in Figure 3.2, is used to place the nodes and paths of this agent, and the visual representation of space markups of the other structural agents, according to this workcell layout.

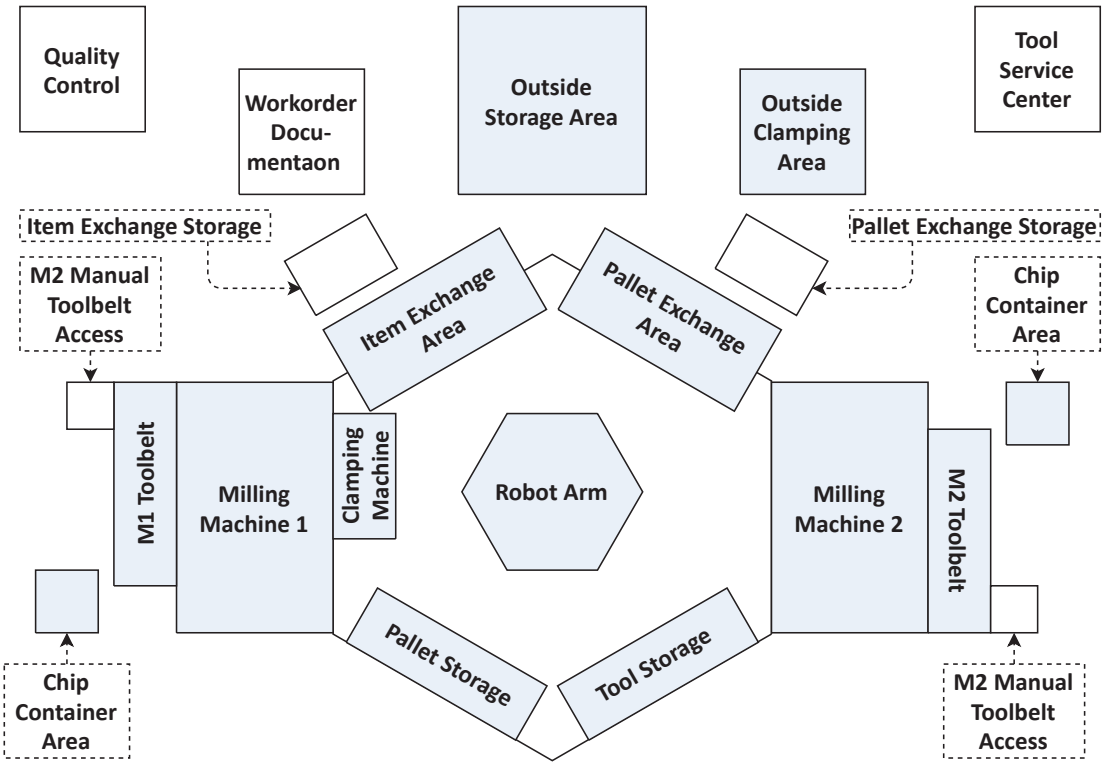

Figure 3.2: Image of workcell layout used in model with labels.

The network of nodes, attractors and paths representing the environment of the operator in the main agent is shown in Figure 3.3. The node *localStorage*, representing the outside storage area, is used as the arrival and departure point for toolsets and itembatches in the workcell model. The node *documentArea* is used to visually display the workorders being processed at the workcell. At the node *outsideClampingArea*, used for workorders requiring clamping and unclamping by the operator, itembatches are separated into individual items, items are clamped and unclamped, and items are combined into batches after production. Clamped items and empty pallets are combined and separated, a separate process from clamping and unclamping items outside the cell in the model, at the node *palletExchangeStorage*.

Toolsets are separated into tools and tools are combined into toolsets at this node as well. The node *palletExchange* represents the outside cell or operator side of the pallet exchange area. Pallets and clamped items mounted on pallets are loaded into and retrieved from the pallet

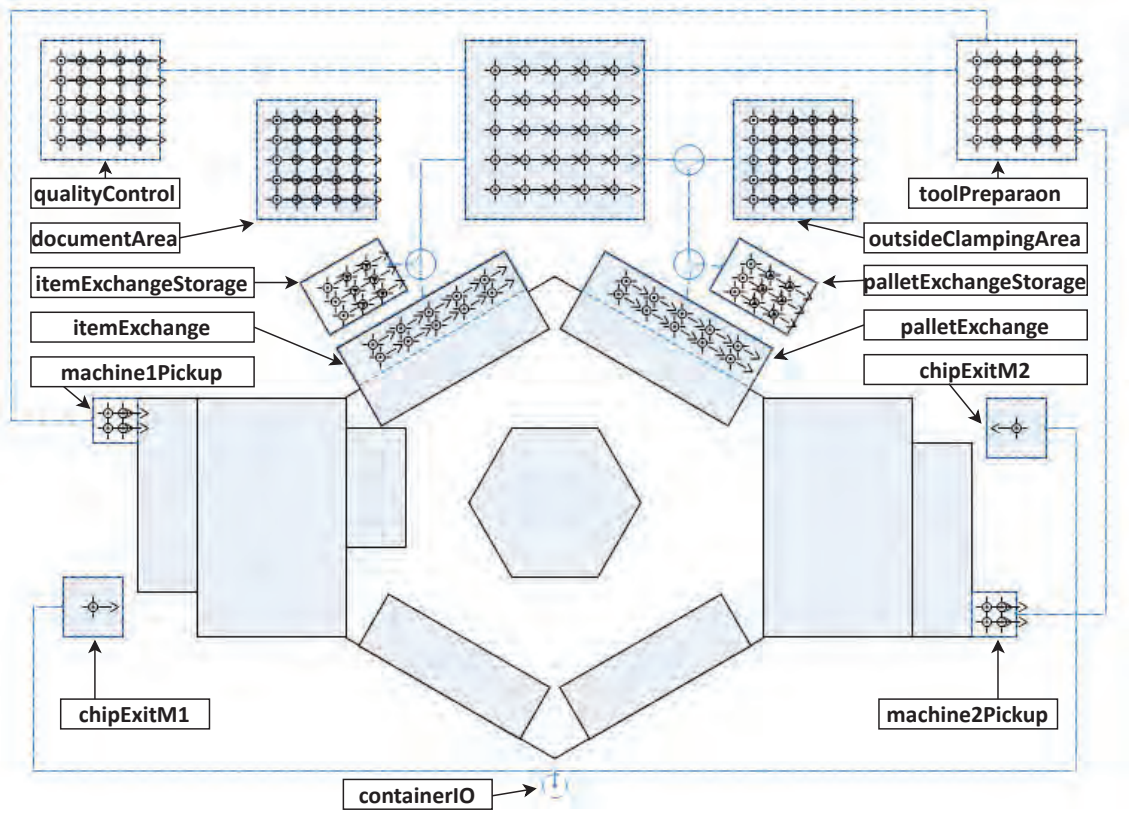

Figure 3.3: Image of the operator domain workcell environment space markups.

exchange by the operator here. Similarly, the node *itemExchangeStorage* is used to separate and merge itembatches from workorders requiring (un)clamping by the clamping machine inside the cell, while the node *itemExchange* is used to load into and retrieve items from the item exchange area by the operator.

The nodes *machine1Pickup* and *machine2Pickup* are used as areas to exchange tools between each milling machine toolbelt and the operator. The node *qualityControl*, representing QC, is used to inspect milled items, while the node *toolPreparation*, representing the TSC, is used to replace standard and broken tools. Finally, chip containers are placed at the milling machines at nodes *chipExitM1* and *chipExitM2*, while containers enter and exit the model at the node *containerIO*.

As mentioned before, most agents are embedded in *wCE_OutsideCell*, including the structural agents *wCE_Cell*, *wCE_MillingMachine1* and *wCE_MillingMachine2*. Their respective environments are visually represented in *wCE_OutsideCell*. Since both milling machines are used as resources in the model process flowcharts, each machine is visually represented with a 2D/3D model inside the milling machine environment space markups. Similarly, the robot and clamping machine agents, *wCE_Robot* and *wCE_ClampingMachine*, embedded in the cell agent, are visually represented with a 2D/3D model inside the cell environment space markups. These four resource agent presentations are placed in fixed locations in the workcell model environment.

The final resource agent, *wCE_Operator* representing the operator, includes a 2D/3D model presentation as well, but moves throughout the operator domain environment during simulation runtime. The operator model is placed inside the environment at the start of a simulation run.

Besides the structural and resource agents, both agents modeling the exchange areas behavior are embedded as well. These agents are visually presented in the workcell environment by a colored circle and a label indicating the respective exchange area status next to the respective exchange area nodes in the environment. The status of each machine and the operator is presented with colored circles and labels in the environment as well. Figure 3.4 shows the complete workcell model environment, including the fixed resource models and status indicators.

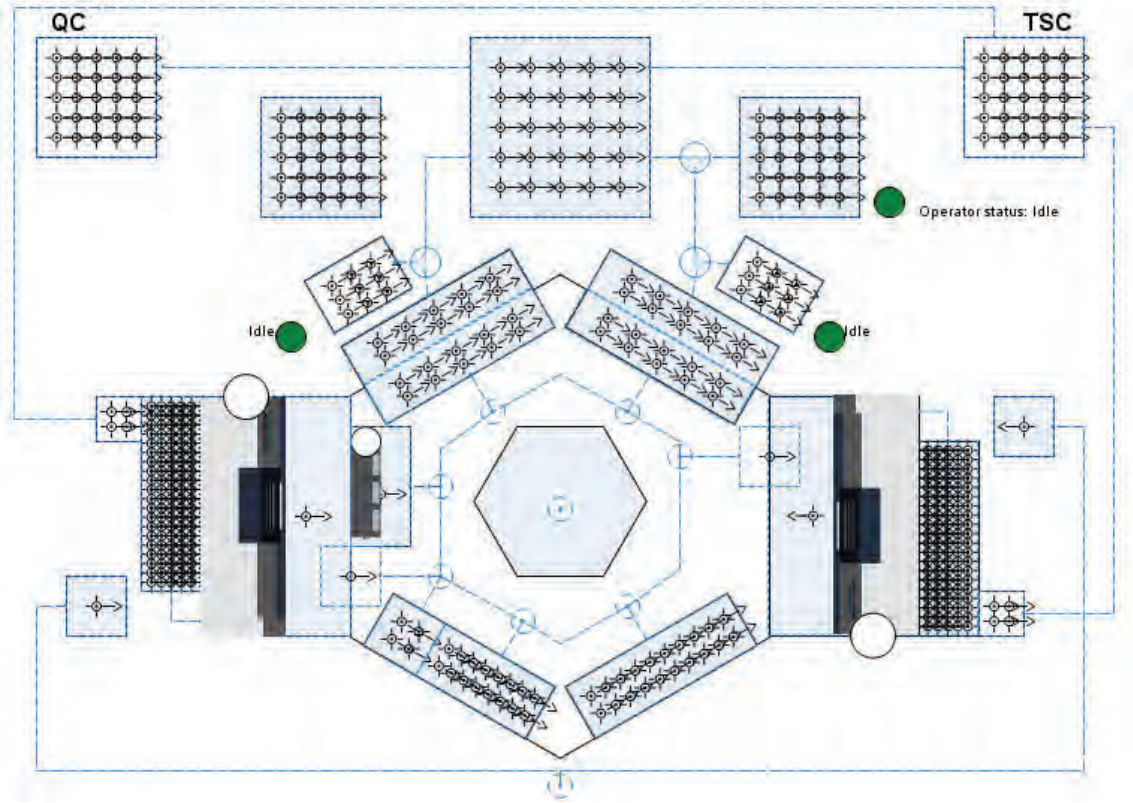

Figure 3.4: Image of the complete workcell environment space markups and presentation.

3.4.2 Robot Domain Agent

The agent *wCE_Cell*, representing the cell or operator domain, is embedded in the top agent and includes the robot domain environment and various process flowcharts. In addition, the cell agent contains variables to define robot handling times, which are imported from databases. The robot priorities are also defined by variables, their values set to zero by default, and changed through an event at the start of a simulation run if the robot priorities option is chosen.

The process flowcharts included in *wCE_Cell* are used to model the item, tool and pallet flow throughout the robot domain. The item process flowchart models the retrieval from the exchange areas, clamping when required, and storing of items. Additionally, the flowchart models delivering clamped items to the milling machines and retrieving milled items from the milling machines, and it models unclamping items when required and loading items in the exchange areas as well.

Similarly, the tool process flowchart models the retrieval from the pallet exchange area and storing of tools, delivering at and retrieving tools from the milling machines, and loading tools into the pallet exchange area. The pallet process flowchart models providing empty pallets for

clamping at the clamping machine or by the operator, and delivering pallets from storage at the clamping machine or the pallet exchange area from storage. After unclamping, the pallet flowchart models retrieving pallets from the pallet exchange area or clamping machine back to storage. The resource agents *wCE_Robot* and *wCE_ClampingMachine*, representing the robot or robot arm and clamping machine, are embedded in *wCE_Cell*. Each agent is used by the resource pools for the robot and clamping machine, which are used throughout the process flowcharts included in *wCE_Cell*. The process flowcharts use the milling machine resource pools for both milling machine from *wCE_OutsideCell* as well.

Similar to the outer cell agent, the environment of the robot domain consists of a network of nodes and paths shown in Figure 3.5. The nodes *itemExchangeCell* and *PalletExchangeCell* represent the inner cell or robot side of the exchange areas. The *itemExchangeCell* node is used to retrieve items for clamping from and load items after unclamping into the item exchange. Similarly, the *palletExchangeCell* node is used to retrieve from and load pallets, clamped items and tools into the pallet exchange. The nodes *palletInventory*, *itemInventory* and *toolInventory* represent the cell storage areas at which empty pallets, clamped items and tools are stored and retrieved from respectively.

At the clamping area, represented by the node *insideClampingArea*, items and empty pallets are delivered for clamping, while clamped items are delivered for unclamping. After clamping, clamped items are retrieved, while items and empty pallets are retrieved after unclamping. The nodes *machine1Cell* and *machine2Cell* represent the area at each milling machine where items and tools are delivered to and retrieved from each machine. Finally, the node *clampingMachineHome*, inside the area of *insideClampingArea* node, defines the spot where the visual presentation of the clamping machine agent is placed, while the node *robotHome* in the center of the cell defines the location for the robot presentation.

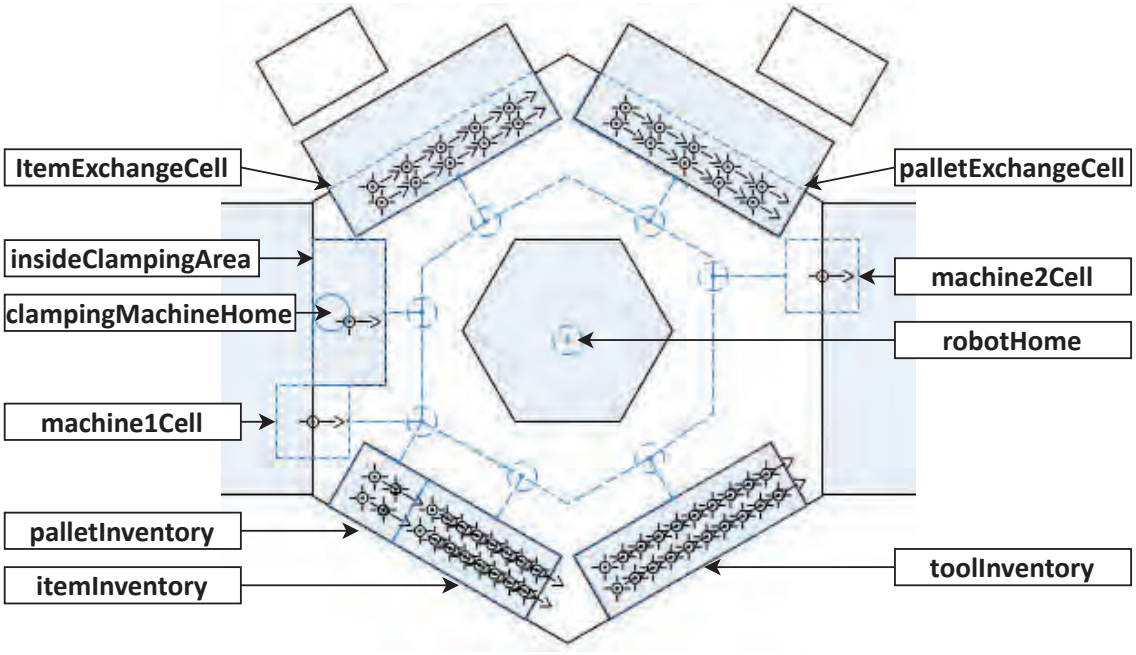

Figure 3.5: Image of the robot domain workcell environment space markups.

3.4.3 Milling Machine Domain Agents

The agents *wCE_MillingMachine1* and *wCE_MillingMachine2* represent the milling machine domain and are both instances of the agent type *WCE_MillingMachine*. Both agents are embedded in the top agent. Since both agents are instances of the same agent type, they are identically modeled, with *wCE_MillingMachine1* representing the first milling machine and *wCE_MillingMachine2* the second. In addition to representing the milling machine environment, the agents are used as resources throughout the workcell model process flowcharts. The milling machine agents contain the milling machine environment space markups, a 2D/3D presentation for the milling machine, and a state chart to track the state of the milling machine.

The Process flowcharts included in the milling machine agents are used to model the item and tool flow throughout each machine. The item process flowchart models the milling of items. The tool process flowchart models storing tools in the machine toolbelt after receiving them from the robot arm, retrieving tools from the toolbelt for retrieval by the robot, and exchanging tools between the toolbelt and the operator.

The milling machine agents contains collections, functions and events to model the milling process behavior at the milling machine domain. This includes requesting the next toolset and item for milling, determining tool failure during milling and initiating repairing failed tools, as well as initiating the removal of special tools after milling from the machine and replacing standard tools when required. These functionalities are further detailed in Section 3.8.

The milling machine environment, shown in Figure 3.6, consists of the nodes machine and inventory, representing the milling machine itself and its toolbelt respectively, with a path connecting both nodes. At the *machine* node items are deposited and retrieved by the robot, and milled by the machine. Tools are delivered and retrieved here by the robot as well, while delivered tools are stored at the node *inventory* and retrieved from *inventory* by the milling machine for retrieval by the robot. Tools are directly stored and retrieved at the *inventory* node by the operator. As the milling machine agents also serve as resources, the agent presentation includes a 2D/3D model representing the machine and a status indicator as well. Although not included in Figure 3.6, the complete presentation of both milling machines is included in Figure 3.4, showing the complete workcell model environment and presentation. The resource aspect of the milling machine agents is further discussed in Section 3.6.

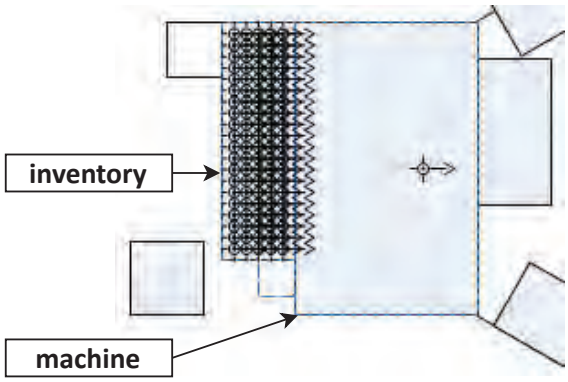

Figure 3.6: Image of the milling machine domain workcell environment space markups for milling machine 1.

3.5 Process Flowcharts

In the workcell model, process flowcharts are used to model the transportation and processing of batches of items, items, toolsets, tools, pallets and chip containers by the operator, robot, clamping machine and milling machines. Process flowcharts consists of blocks and connectors linking the blocks together. The material agents are used within the flowcharts, while resource agents are used by the flowcharts through resource pools whenever the operator, robot, clamping machine or milling machines are required to further process the material agents throughout the process flowcharts. The process flowcharts are modeled based on the product and tool flow of the workcell discussed and shown in Section 2.2 in the previous chapter.

In AnyLogic, the process modeling library provides the blocks to create process flowcharts in addition to the space markups used to create the workcell environment in the structural agents described earlier. Through the blocks the process flowcharts use the space markups to model the material agents transportation and processing throughout the workcell environment. An example of an AnyLogic process flowchart including all blocks used in the workcell model process flowcharts is shown in Figure 3.7. The functionality and usage of each block used in the model process flowcharts is described next.

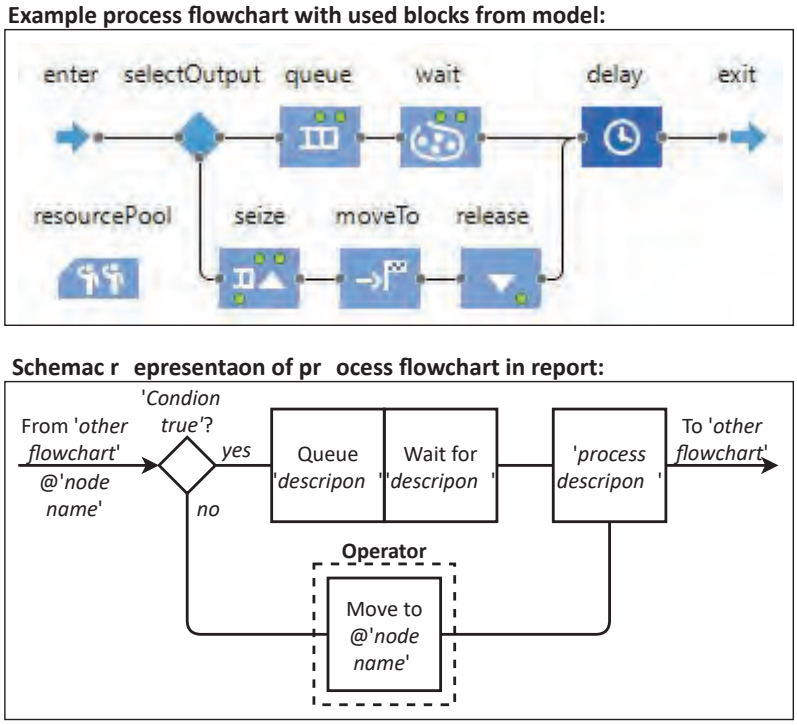

Figure 3.7: Example of process flowchart from model and schematic representation used in report.

In the model, all process flowcharts begin with *Enter* blocks and end with *Exit* blocks, connected to other blocks through connectors in between within the flowchart. As such, *Enter* blocks only have an exit port connected with the entry port of the next block, and *Exit* blocks only have an entry port connected to the exit port of previous blocks. An entry port can be connected with multiple exit ports, but an exit port can only connect to one entry port. Unless mentioned otherwise, the other block types have at least one entry and exit port. The *Enter* and *Exit* blocks are used to enter already existing agents into and exit agents from process flowcharts. In

the *Enter* block the agent type for this process flowchart is defined, as well as the node where the entering agent presentation is placed in the workcell model environment. Entered agents leave the *Enter* block through the exit port as soon as the next block is free to receive agents. Agents are immediately removed from the process flowchart after entering an *Exit* block. In all blocks actions can be used to program model behavior when interacting with agents in the process flow, most commonly used in the workcell model when an agent enters or exits a block.

Wait, *Delay* and *Queue* blocks are used to hold agents in these blocks at their current or a specified location in the environment until released. Agents at a *Queue* block are released when the next block in the process flow is available for the next agent, most commonly on a first in first out basis. Agents at a *Wait* block are released through java code from functions and events, or through actions from other parts of the model including statecharts or other blocks in process flowcharts. Agents at a *Delay* block are released after a predefined time present in the block. All three block types have an adjustable queue or holding capacity for the number agents that can be simultaneously present in a block, ranging between one and maximum, where maximum means all agents that want to enter the block. In the model, most *Wait* and *Queue* blocks use maximum capacity or the inventory/storage size they represent, where most *Delay* blocks have a capacity of one, as they are most commonly used to model the processing of an agent by a resource.

Seize and *Release* blocks are used to seize resource agents from a *ResourcePool* block for use by agents in the process flowchart. Similar to the *Wait* and *Delay* blocks, the *Seize block* has a queue to hold agents arriving at a *Seize* block until a resource agent is available. In each *Seize* block agents can be given a priority. Multiple *Seize* blocks in multiple process flowcharts can use the same resource pool. Available resource agents are seized and linked to the waiting agent in any of the *Seize* blocks with the highest priority or that arrived first when priorities are equal. Seized resource agents can be sent to a destination and attached to the linked agent when required. Attached resources move together with the agent in the model environment. *Release* blocks release specified resource agents from the linked agents in the process flow. *ResourcePool* blocks are used to create or connect existing agents to be used as resources in process flowcharts. Therefore, *ResourcePool* blocks have no entry or exit ports, but are used by *Seize* and *Release* blocks to seize and release the connected resource agents. In the model, all resource pools consist of one existing resource agent embedded in the structural agents.

MoveTo blocks are used to transport agents, and any attached resource agents, from their current position to a new destination in the model environment. Agents can jump or travel to the destination location, where traveling agents can move to the destination at a specified trip time or based on the travel distance and agent speed. In the workcell model locations are always nodes and *MoveTo* blocks are used to instantaneously jump agents to the destination node or travel over the paths between the nodes according to a trip time. Similar to *MoveTo* blocks, *ResourceSendTo* blocks are used to send seized resource agents to a new destination independent of the linked agent.

Finally, *Select* blocks are used to send agents to a different block based on wether a condition is true or false. Therefore, each *Select* block has two exit ports, one exit used if the condition is true, the other when false. Conditions can be based on functions, boolean variables, or equations between variables that result in a true or false outcome. In the workcell model, the function `randomTrue(double p)` or conditions based on the parameters and variables representing the properties and state of the agent in question are mostly used. The function `randomTrue(p)` generates a random number between zero and one, and returns true if the chance p is smaller than this number and false if not.

In the model, the structural agents contain at least one process flowchart per structural agent and per applicable material agent type used in flowcharts in each structural agent. In the remainder of this section the process flowcharts throughout the model are shown and described per material agent type. The figures used to illustrate these flowcharts use the schematic presentation shown in Figure 3.7.

3.5.1 Itembatch Process Flowcharts

At the workcell, items arrive in batches as defined in the workorders. In the model, each batch of items is represented by a single agent *wCE_ItemBatch*. Itembatch agents are used in process flowcharts after being added to the model during simulation until split into items, and after items are combined into a batch again until departure from the model. A schematic overview of these process flowcharts can be found in Figure 3.8. Both flowcharts are part of the *wCE_OutsideCell* agent. After entering the first flowchart, itembatches follow a different path through the flowchart depending on whether the items from the batch will be clamped outside the cell by the operator or inside the cell by the clamping machine. Itembatches clamped inside the cell have to wait until the operator is on shift, after which itembatches have to wait until available pallets are reserved for the items of the batch for clamping, after which the itembatches have to wait until the item exchange is free for use by the operator. Then, each itembatch is moved next to the item exchange by the operator, as soon as the operator is seized for this transportation. Moved itembatches then exit the process flowchart and initiate the `split()` function part of the itembatch agents themselves. Itembatches clamped outside also have to wait for the operator to be on shift, after which each itembatch is moved to the clamping area by the operator before leaving the flowchart and initiating the batch `split()` function.

After items are combined into batches again, the itembatches enter the second process flowchart. Itembatches unclamped inside the cell enter the flowchart trough the *Enter* block linked to the node next to the item exchange, where itembatches unclamped outside the cell enter at the outside clamping area. Each itembatch is then moved by the operator to the *localStorage* node, representing the general storage area outside the cell. Moved itembatches exit the flowchart and are removed from the simulation model.

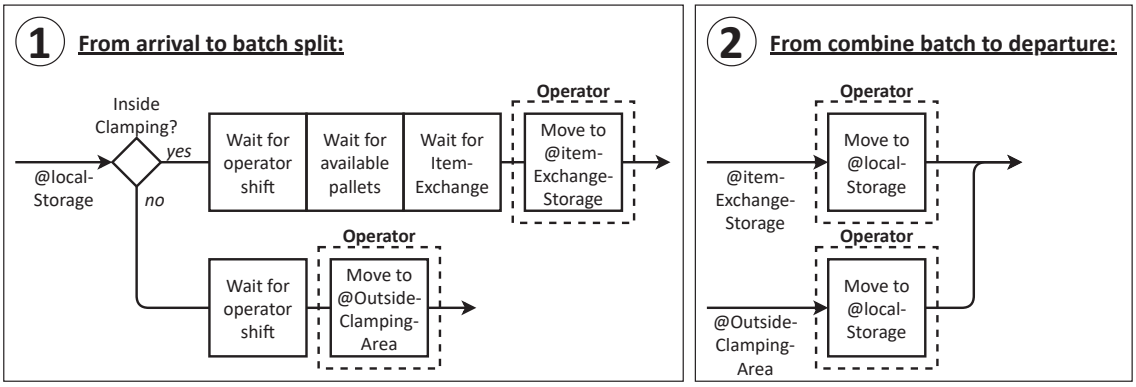

Figure 3.8: Schematic overview of the itembatch process flowcharts at *wCE_OutsideCell*.

3.5.2 Item Process Flowcharts

In the workcell model, items are created and added to the simulation by the *wCE_Itembatch* agent `split()` function. Items are used in process flowcharts in all structural agents representing the three domains of the workcell. In the *wCE_OutsideCell* agent, item agents are used in five item process flowcharts. After an itembatch is split and the items are added to the model, the items enter one of the two process flowcharts shown in Figure 3.9. Items clamped inside the cell enter the first flowchart, and items clamped by the operator enter the second flowchart.

Each item entered into the first process flowchart is moved to and subsequently loaded into the item exchange by the operator. There, the items wait until the item exchange is closed before exiting the flowchart to enter the item process flowchart at *wCE_Cell*. Each item entered into the second flowchart is clamped by the operator. After clamping, items have to wait until all items of the same batch are clamped, wait until the operator is on shift, wait until available pallets for all items of the same batch are reserved, and wait until the reserved pallets are retrieved from the cell by the operator. Then, each item is moved next to the pallet exchange and mounted on the retrieved pallet, completing the clamping process, by the operator. Next, as soon as the pallet exchange is available, each item is moved to and loaded into the pallet exchange by the operator. After the pallet exchange is closed, items exit the flowchart to enter the item process flowchart at the *wCE_Cell* agent.

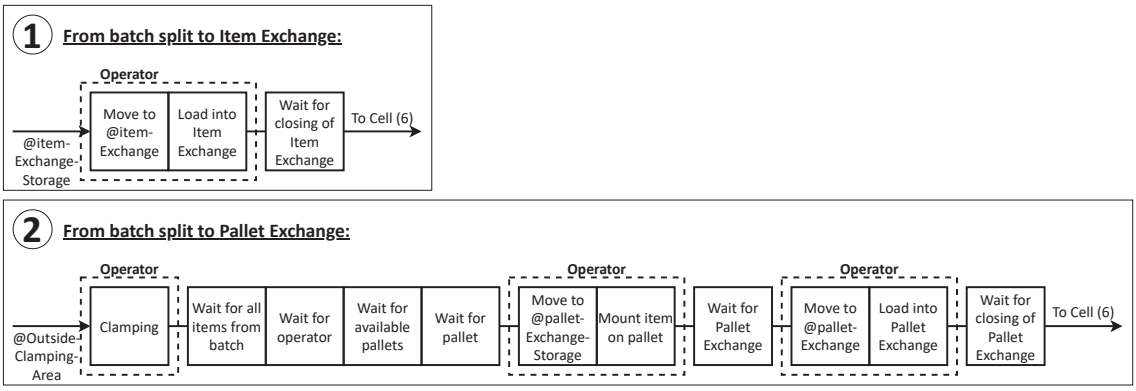

Figure 3.9: Schematic overview of item process flowcharts at *wCE_OutsideCell* before milling.

After exiting the cell process flowchart, items are entered into the process flowcharts, shown in Figure 3.10, at the *wCE_OutsideCell* agent again. Items unclamped inside the cell enter the third flowchart, and items to be unclamped outside the cell enter the fourth. In the third process flowchart, each entered item is retrieved from the item exchange and moved to the storage area next to the exchange by the operator. Then, if an item requires a quality inspection, the item exits the flowchart and is entered in the fifth item process flowchart modeling the quality inspection process. If an item is to be discarded, the item exits the flowchart and is discarded. All other items, as well as items reentering the process flowchart after the quality inspection, wait until all items from the same batch are ready to be combined into a batch again through the `combine()` function of the corresponding *wCE_Itembatch* agent, before exiting the flowchart and removal from the simulation.

Each item entered in the fourth process flowchart is retrieved from the pallet exchange and moved to the storage next to the exchange by the operator. There, the operator dismounts each item from the pallet, and moves each item to the outside clamping area. After all items from the same batch that where retrieved from the pallet exchange are available and the operator is on shift, each item is unclamped by the operator. Similar to the third process flow, after

unclamping, items requiring inspection or to be discarded exit the flowchart. All other items, as well as items reentered after inspection, wait until all items from the same batch are ready to be combined into a batch again through the `combine()` function of the itembatch, before exiting the flowchart and removal from the simulation as well.

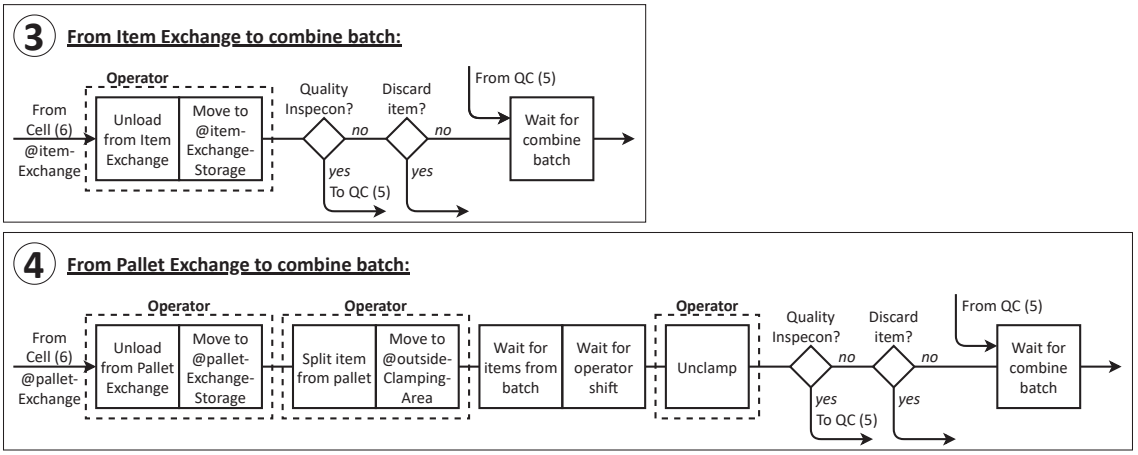

Figure 3.10: Schematic overview of item process flowcharts at *wCE_OutsideCell* after exiting the cell.

Items requiring a quality inspection enter the fifth process flowchart, shown in Figure 3.11. In this flowchart, each entered item is moved to QC by the operator, after which the quality inspection is performed on the items at QC. Items that passed inspection are moved back to the workcell and exit the process flowchart for reentry into their previous flowchart. Items clamped inside the cell are moved to the storage next to the item exchange and reentered into the third flowchart, where items clamped outside the cell are moved to the outside clamping area and reentered into the fourth flowchart. Items that failed inspection are discarded. Failed items from a workorder with a batch size of one immediately exit the flowchart, triggering the removal of these items and their respective itembatches and workorders. For all other items that failed inspection the operator is send to the machine each item was milled to perform a quality assessment. After this assessment the item exits the flowchart and is removed from the simulation.

Items entering the cell through either exchange enter the sixth item process flowchart, which is shown in Figure 3.12 and part of the *wCE_Cell* agent. Each item clamped outside the cell is retrieved from the palled exchange, moved to the cell inventory for items, and placed in this inventory by the robot. Although modeled by three blocks in the model process flowchart, In Figure 3.12 and other process flowchart figures this is depicted as a single block starting with the text *'Transport to'* when applicable. For each item clamped inside the cell, the clamping machine is seized before the robot transports each item to the clamping machine. Here, each item waits until their respective pallet is transported to the clamping machine as well, before each item is clamped by the clamping machine. The robot has to be seized again before the clamping machine is released and available to clamp the next item. Each clamped item is then moved to the inventory by the robot. The stored items wait at the item inventory until they can be milled at the assigned milling machine or are to be discarded from the cell.

Each item ready to be milled is seized by the appropriate milling machine, transported by the robot to that machine and exits the process flowchart in order to enter the milling machine process flowchart. This process flowchart is identical for both machines and shown in Figure 3.12 as well, indicated by number seven. In the milling machine flowchart, each entered item is milled

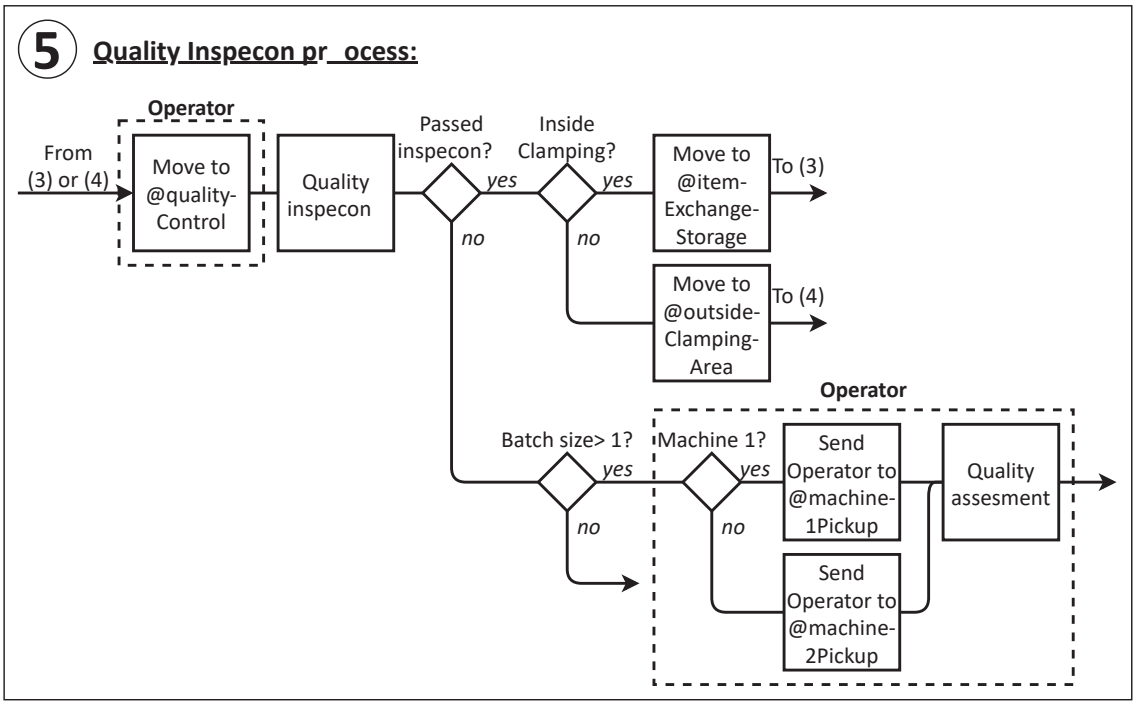

Figure 3.11: Schematic overview of the item quality inspection process flowchart at *wCE_OutsideCell*.

before exiting the flowchart. After reentering the cell process flowchart again, the robot retrieves each item from the milling machine before the milling machine is released. Then, the operator moves and places each item in the inventory. Items for which the milling process was aborted, wait at the inventory for milling again.

Successfully milled items transported to the inventory have to wait until the operator is on shift, wait until all other items from the same batch are milled unless the item requires a quality inspection, and wait until either the pallet exchange or item exchange is available. Each item clamped outside is transported by the robot to the pallet exchange once available, and exits the flowchart after the pallet exchange is opened for entry into the fourth item process flowchart described earlier. For each item clamped inside the cell the clamping machine is seized, after which each item is transported to the clamping machine area by the robot and unclamped by the clamping machine. The robot is seized before the clamping machine is released, before each item is transported to the item exchange by the seized robot. At the clamping exchange the items wait until the item exchange is opened before exiting the flowchart for entry in the third item process flowchart described earlier as well.

3.5.3 Pallet process flowcharts

In the workcell model, empty pallets are created at the start of the simulation and entered into the pallet process flowchart at *wCE_Cell*, shown schematically as the first flowchart in Figure 3.13. The pallets are entered at the pallet inventory node inside the cell and wait there until required for clamping either inside or outside the cell. Requested pallets for clamping inside the cell have to wait until all required pallets are available for clamping of all items in the batch, and then wait until the items are ready for clamping. Each pallet is then transported by

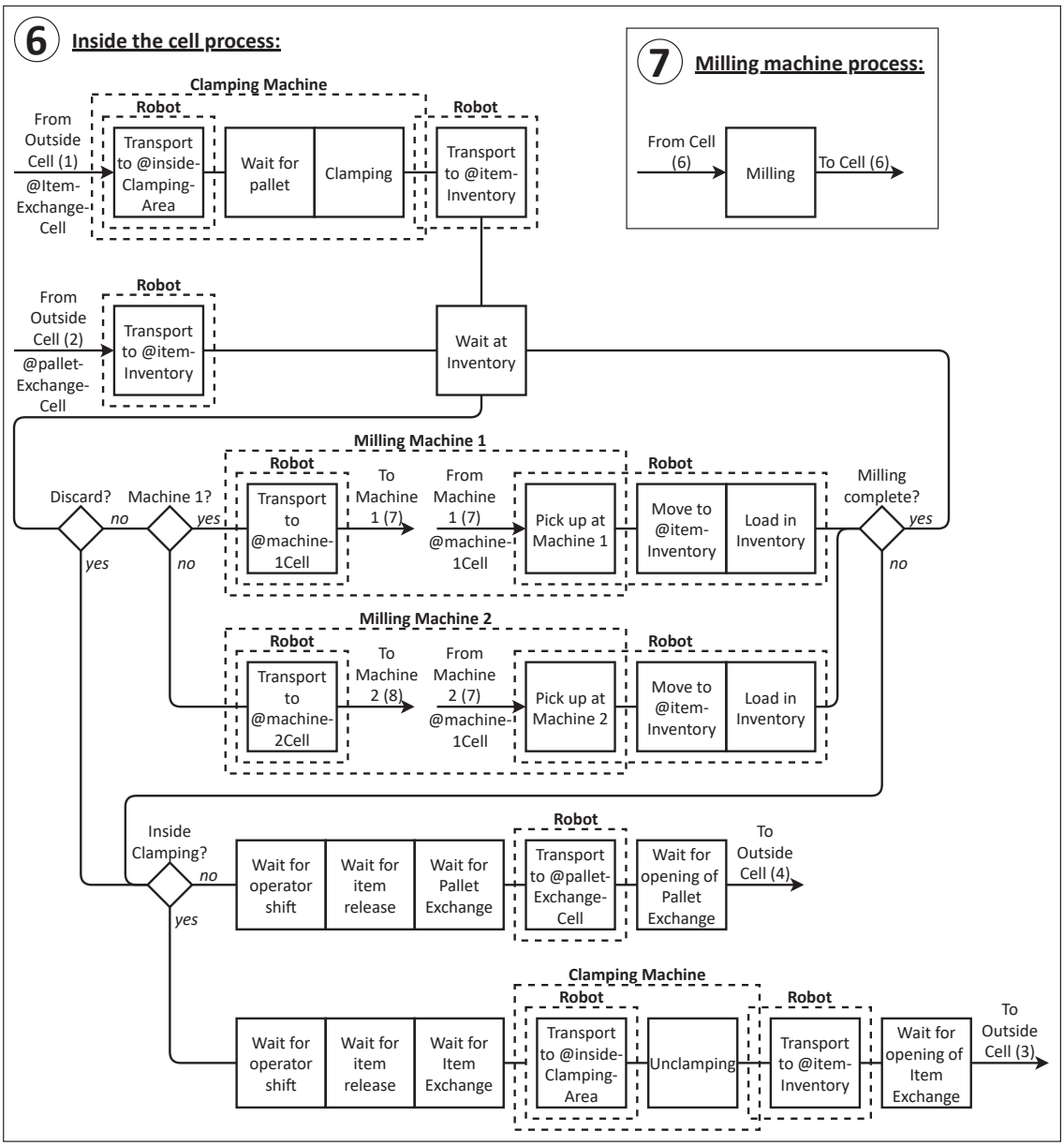

Figure 3.12: Schematic overview of the item process flowcharts at the cell and both machine agents.

the robot to the clamping machine, where arrived pallets wait until they are unclamped from their items. The clamping and unclamping process itself is part of the item process flowcharts described earlier. After unclamping the robot transports each pallet back to the pallet inventory where they wait until required for clamping again.

Pallets requested for outside clamping also wait until all required pallets are available and the items are ready for clamping as well. After the pallet exchange is available, the robot transports each pallet to the pallet exchange, where each pallet waits until the exchange is opened before exiting the first flowchart and entering the second. This second pallet process flowchart is part of the *wCE_OutsideCell* agent. In this flowchart, also shown in Figure 3.13, each entered pallet is retrieved from the pallet exchange and moved to the storage next to the exchange by the operator. There, the pallets wait until unclamped from their items modeled through the item process flowcharts. The unclamped empty pallets wait until the pallet exchange is available, after which each pallet is moved to the pallet exchange and loaded into the exchange by the operator. After the pallet exchange closes each pallet leaves the second flowchart and is entered into the first flowchart again. Each entered pallet is transported by the robot to the pallet inventory where they wait until required for clamping.

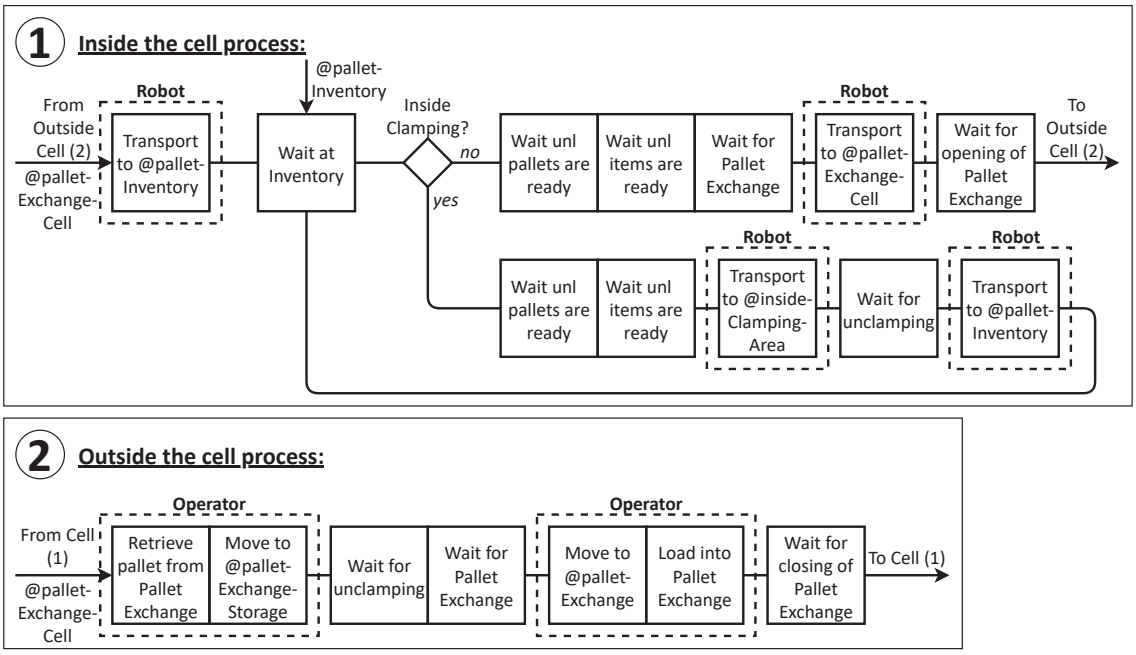

Figure 3.13: Schematic overview of the pallet process flowcharts inside and outside the cell.

3.5.4 Toolset Process Flowchart

The special toolsets required per workorder are represented by the *wCE_Toolset* agents in the model. After creation during simulation runs, the toolset agents modeling the special toolsets enter the process flowchart at the local storage node part of *wCE_OutsideCell* and shown in Figure 3.14. There, the toolsets wait until the operator is on shift, the items to be milled by the toolsets are available inside the cell, and the cell has enough tool storage capacity left for all tools of each toolset. Each toolset ready to enter the cell is moved next to the pallet exchange by the operator. There, the toolsets leave the process flowchart after triggering the toolset `split()` function to create the *wCE_Tool* agents modeling the individual tools of the toolsets.

After toolsets are combined into a toolset again, the toolset agents reenter the toolset process flowchart at the storage area next to the pallet exchange, from where the operator moves each entered toolset to the local storage exchange. If a toolset is reserved for milling a batch of items, it returns to the start of the process flowchart to wait for reentry into the cell, and if not, the toolset exits the process flowchart for removal from the simulation instead.

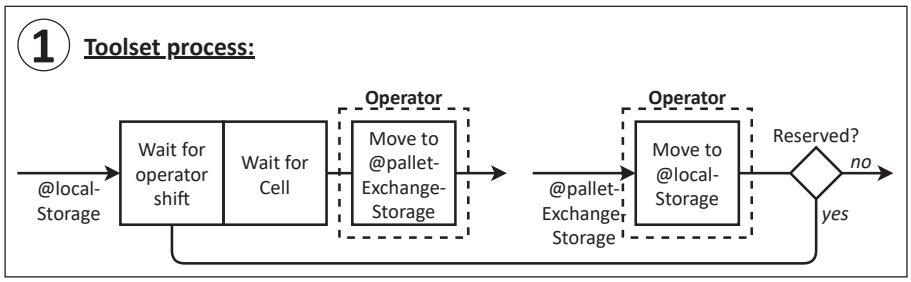

Figure 3.14: Schematic overview of the toolset process flowcharts at *wCE_OutsideCell*.

3.5.5 Tool Process Flowcharts

Similar to items, tools are created by the `split()` function part of the *wCE_Toolset* agent. Created tools part of a special toolset after the toolset is split are entered into the first process flowchart, shown in Figure 3.15, at the storage area next to the pallet exchange part of the *wCE_OutsideCell* environment. Entered tools wait until the pallet exchange is available, after which the operator moves each tool to the pallet exchange and loads each tool into the exchange. After the pallet exchange closes, each tool exits the process flowchart to enter the tool process flowchart at the *wCE_Cell* agent.

Tools leaving the cell enter the second process flowchart, also shown in Figure 3.15, at the pallet exchange node. Each entered tool is retrieved from the pallet and moved to the storage area next to the pallet exchange by the operator. There, the tools wait until all tools from the same toolset are present to combine into toolsets. Tools ready to combine exit the process flowchart and are removed from the simulation model, after triggering the `combine()` function at their corresponding *wCE_Toolset* agent.

Tools entering the cell through the pallet exchange enter the third tool process flowchart, shown in Figure 3.16, located in *wCE_Cell*. Each entered tool is transported by the robot to the tool inventory of the cell. Tools can also be entered into the process flowchart directly at the tool inventory, for example to enter tools into the flowchart from special toolsets already present in the workcell at the start of a simulation run. Tools wait at the tool inventory until released for further processing.

Each tool ready for use by one of the milling machines is transported to the corresponding milling machine by the robot after seizing the milling machine. Each tool exits the process flowchart to enter the process flowchart at the milling machine. Tools returning from the milling machine are entered back into the process flowchart at the cell with the machine already seized. The robot retrieves each tool, after which the machine is released and the robot moves and places each tool into the tool inventory.

Tools at the inventory required to leave the cell wait until the operator is on shift, wait until all required tools from the toolset are ready to exit the cell, and wait until the pallet exchange is available. When the pallet exchange is available, each tool is transported to the pallet exchange

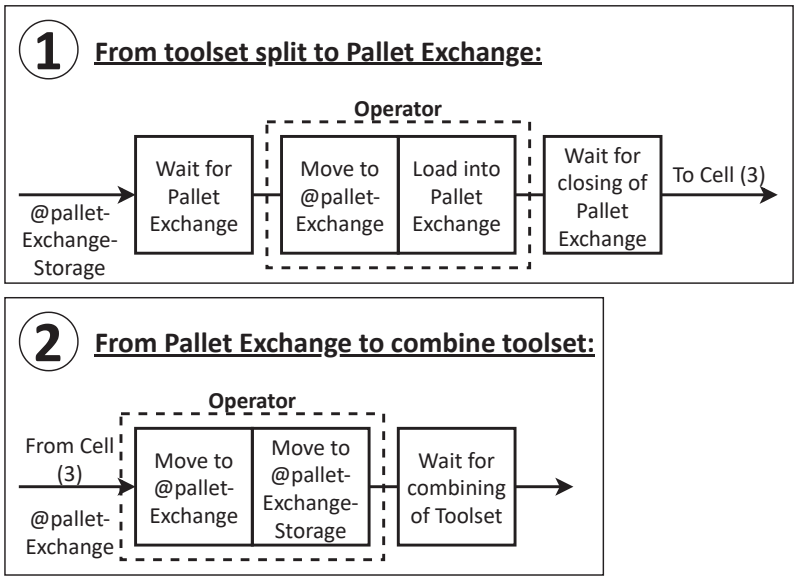

Figure 3.15: Schematic overview of the tool process flowcharts at *wCE_OutsideCell*.

by the robot and exits the flowchart after the pallet exchange is opened to enter the second process flowchart at *wCE_OutsideCell*.

Each tool entered into the milling machine process flowchart, the fourth tool process flowchart shown in figure 3.17, from the cell is moved to the machine inventory or toolbelt by the machine. At the machine inventory tools wait until required to leave the machine. The standard tools from the standard toolset for each machine are entered directly into to the machine inventory at the start of each simulation run. For each tool returning to the cell the machine is seized, after which each tool is moved from the toolbelt to the machine by the machine before exiting the flowchart for entry into the process flowchart at *wCE_Cell* again. Tools requiring replacement, either standard or special tools, exit the flowchart to enter the fifth tool process flowchart, while replaced tools are entered into the machine flowchart directly at the inventory.

Tools requiring replacement are entered into the fifth tool process flowchart, shown in Figure 3.18 and modeling the tool replacement process at the TSC part of the *wCE_OutsideCell* agent. Here, the operator retrieves each tool from the milling machine toolbelt, moves each tool to the TSC, and waits for each replacement tool assembled by the TSC. For each tool that is immediately replaced, the operator moves each tool back to the local storage area outside the workcell, then moves each tool to the appropriate milling machine and loads each tool into the machine inventory. Loaded tools then exit the flowchart for entry back into the milling machine tool process flowchart.

For each tool that cannot be immediately replaced the operator is released. The tools have to wait at the TSC area until they are replaced. Each replaced tool is moved back to the storage area outside the cell, where tools wait until required for milling at one of the milling machine again and the operator is on shift. Each tool is then moved and loaded into the machine inventory by the operator. The loaded tools then exit the flowchart for entry back into the milling machine flowchart.

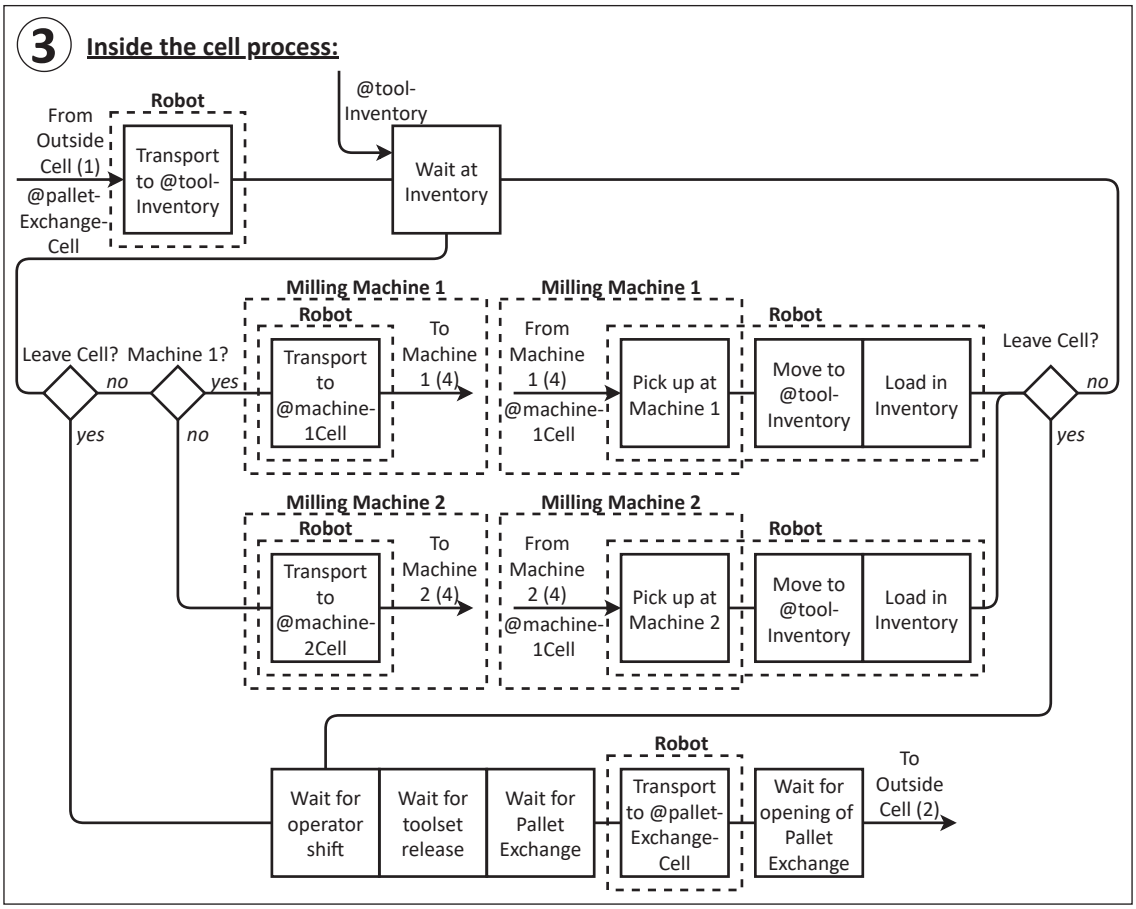

Figure 3.16: Schematic overview of the tool process flowcharts at *wCE_OutsideCell*.

3.5.6 Container Process Flowchart

Chip container agents are used to model the storage of chipped material during the milling process at each milling machine. At the start of the simulation or when requested through the milling machine to replace full containers, chip containers are created and entered into the process flowchart shown in Figure 3.19 at the *containerIO* node part of the *wCE_OutsideCell* environment. Entered containers are instantly moved to the chip exit of their assigned machine to enter the container queue at the machine. If there is no container being filled at the chip exit, the first container in the queue waits at the exit until filled. Full containers are moved to the *containerIO* node to exit the flowchart and removal from the simulation.

3.6 Resource Agents

In this section the agent mechanics of the agents used as resources and representing the exchange areas in the process flowcharts are described. More specifically, these agents contain statecharts used to track and display the status of the agents, and in some cases to control parts of the material agent progression throughout the process flowcharts as well. Similar to the structural agents, these resource agents contain navigation bars and viewareas, used to display the agents and navigate throughout the model during simulation runs, and further described in Section 3.9.

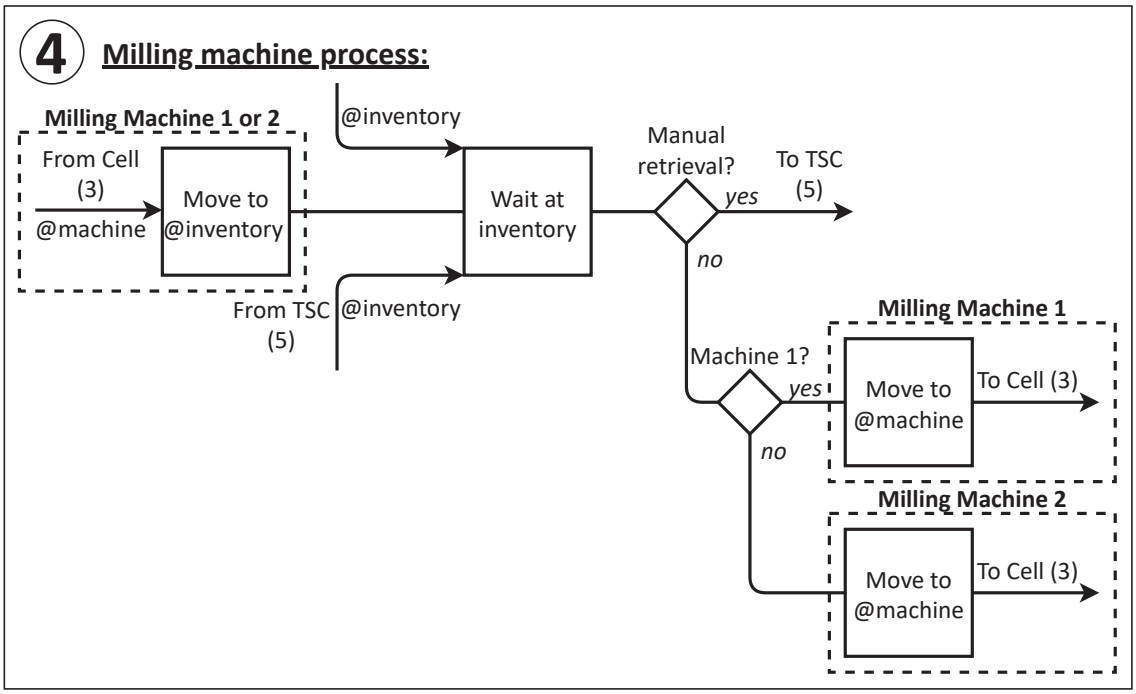

Figure 3.17: Schematic overview of the tool process flowcharts at both milling machines.

Before detailing each resource agent, the statechart mechanics used in AnyLogic models is introduced.

In AnyLogic, statecharts are used to determine agent behavior and interactions with other agents during simulation based on the state and state transitions defined in the statechart. In the workcell model, an agents statechart consists of an entry point, states, transitions, and in some cases one or more final states. An example AnyLogic statechart containing all used elements of the workcell model statecharts is shown in Figure 3.20. During simulation at the time of the creation of agents containing a statechart, the statechart enters its first state through the entry point. Each state in the statechart can execute a java code based action on state entry and exit. A final state can be transitioned into and the statechart will stay in this final state indefinitely. The final state can execute an action upon entry.

Statecharts change states through transitions. Each transition has a trigger mechanism, can perform an action on state transition, and can contain a guard condition. A state transition occurs between the origin state and the destination state of the transition when the transition is triggered and, if specified, the guard condition is true. The three types of trigger mechanisms used throughout the model are the timeout, message and condition trigger mechanisms.

A timeout transition is triggered after a specified amount of simulation time after the statechart entered the origin state elapses. A message transition is triggered when the statechart receives the message from any source in the model specified in the transition and the statechart is in the origin state of the transition. In the model, messages are specified as a particular string value for each transition. A condition transition is triggered when the specified java code based expression holds true. The guard condition is an optional expression that must result in either true or false. As the entry point is a guaranteed transition into the first state of the statechart at agent creation, it can also perform an action.

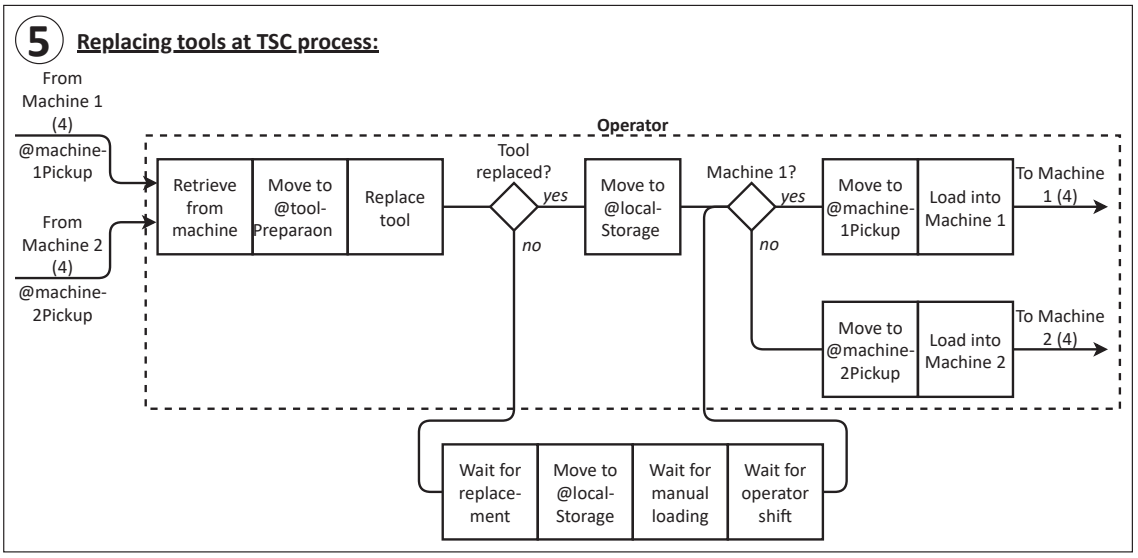

Figure 3.18: Schematic overview of the tool replacement process flowchart at the TSC.

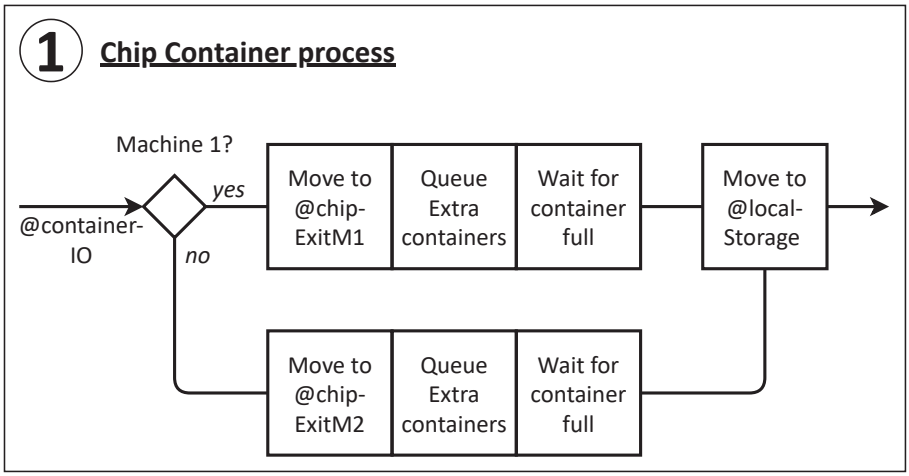

Figure 3.19: Schematic overview of the chip container process flowchart at *wCE_OutsideCell*.

3.6.1 Operator

The agent *wCE_Operator* is used to model the handling of tools, items and pallets outside the cell the operator of the workcell is responsible for. Besides being used as a resource through its resource pool in the process flowcharts contained in *wCE_OutsideCell*, the operator agent tracks the status of the operator through its statechart and models the operator being on or off shift through its statechart, events and schedules. The operator agent is embedded in *wCE_OutsideCell* and contains an object as its 2D/3D presentation used throughout the workcell model environment. Figure 3.21 shows the operator statechart and 2D presentation, as well as the cylinder and text to display the operator status part of the *wCE_OutsideCell* environment.

The operator statechart contains five states and is used to track whether the operator is working or idle, and on or off shift. The first state, *Initialize*, is entered upon the creation of *wCE_Operator* at the simulation runtime start and is used to determine if the operator is on or off shift. After a delay of 1 millisecond the statechart transitions to the state *Idle* if the operator is on shift, or the

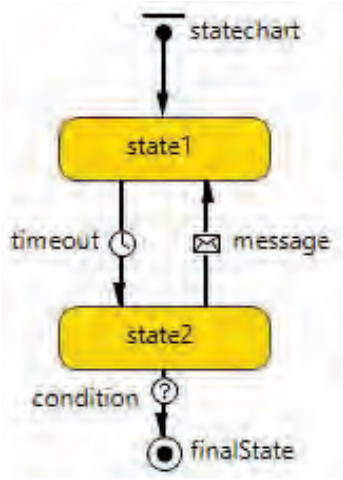

Figure 3.20: Example of a statechart used in AnyLogic models.

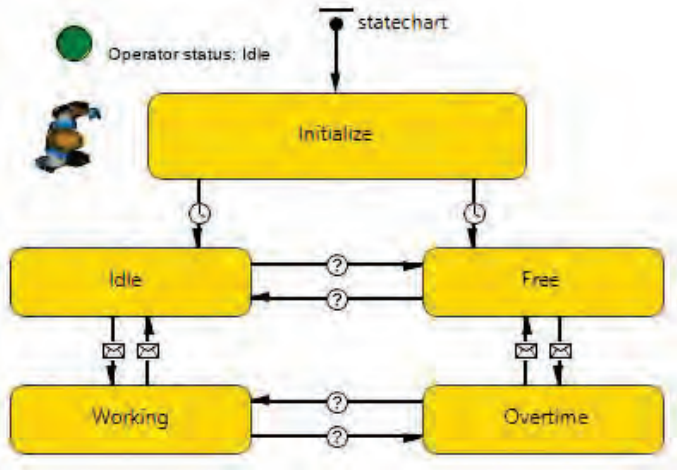

Figure 3.21: Statechart and presentation of the *wCE_Operator* agent.

state *Free* if off shift. The boolean variable *OperatorActive* is used to determine if the operator is on shift or not. The state *Idle* represents the operator being idle during its shift, while the state *Free* represents the operator being idle when off shift. The final two states, *Working* and *Overtime*, represent the operator working when on or off shift respectively.

If the operator is seized by one of the process flowcharts Seize blocks through the operator resource pool *Operators*, the on seize action of the resource pool sends a message to the operator statechart to trigger a state transition from either the state *Idle* to the state *Working*, or from the state *Free* to the state *Overtime*. Whenever the operator is released through a *Release* block, the operator resource pool, through its on release action, sends a message to the operator to trigger the reverse state transition, either from *Working* to *Idle* or *Overtime* to *Free*. When the operator shift starts the condition transitions becoming true evaluating the *OperatorActive* boolean trigger a state transition, from either the state *Free* to the state *Idle* or from the state *Overtime* to the state *Working*. Inversely, when the operator shift ends the condition to trigger the reverse state transition becomes true, triggering the transition either from *Idle* to *Free* or from *Working* to *Overtime*.

Through the entry and exit actions of the four states representing the operator status, the time

working and being idle while on or off shift is tracked for statistical and analytical purposes, further discussed in Section 3.10 as mentioned before. The color and text of the operator status indicator part of the workcell environment is adjusted accordingly by the state entry actions as well.

Set through the simulation options, the operator is either always on shift, on shift for 17 hours from 6:00 to 23:00 every day, or on shift for 17 hours from 6:00 to 23:00 from Monday to Saturday. Through the entry action of the *Initialize* state of the statechart, an algorithm checks whether the operator is on or off shift at the simulation start time and sets the operator to be on or off shift accordingly. Throughout the workcell model process flowcharts, *Wait* blocks are used to control the agent progression through the flowcharts based on the operator shift status. Agents entering one of these *'waitForOperator'* blocks are immediately released if the operator is on shift, checked through the *Wait* block on enter action. If the operator is off shift, the entered agents wait until released.

Based on the chosen simulation option, schedules are used to switch the operator between on or off shift through triggering events at the start and end time of the operator shift defined by these schedules. The events are used to change the operator statechart to the appropriate state and to release all waiting agents in the *'waitForOperator' Wait* blocks when the operator shift starts.

Finally, the operator prioritizes certain actions or tasks if chosen through the simulation options through the operator *Seize* blocks part of the process flowcharts. Since multiple *Seize* blocks can seize the operator, the operator is seized by the *Seize* block for the agent with the highest task priority of all waiting agents in all operator *Seize* blocks, while the operator is seized for waiting agents with equal priority on a first in first out basis. By default, all agents have the same task priority, but if the operator priorities option is chosen each *Seize* block gives entering agents a predefined priority per block, prioritizing certain operator actions over others.

3.6.2 Robot

The agent *wCE_Robot* is used to model the robot arm handling tools, items and pallets inside the cell. Besides being used as a resource by the process flowcharts through the robot resource pool *Robots* at *wCE_Cell*, the robot agent tracks the robot arm status and rotates the robot presentation to model the robot arm transporting items, tools and pallets between locations through a function and an event. The robot is embedded in *wCE_Cell* and its 2D/3D presentation consists of a cylinder and a robot arm object on top of the cylinder. The robot statechart and presentation are shown in Figure 3.22.

The robot statechart consists of four states representing the status of the robot arm. Upon the creation of *wCE_Robot*, the robot statechart enters into the *Idle* state, representing the robot being idle. In the *wCE_Cell* process flowcharts, the handling of items, tools and pallets by the robot is modeled by a *Delay*, *MoveTo* and another *Delay* block while the robot is seized. The first *Delay* block represents retrieving or picking up material and is represented in the robot statechart by the state *Pickup*. The *MoveTo* block represent transporting the retrieved material from its origin to destination and is represented by the state *Move*. The second *Delay* block represents the depositing or loading of the retrieved material at the destination and is represented by the state *Load* in the robot statechart. The statechart transitions are triggered through the on enter actions of the relevant process flowchart blocks, where the transition from state *Idle* to the state *PickUp* is triggered by the first *Delay* block, the transition from *Pickup* to *Move* by the *MoveTo* block, and the transition from *Move* to *Load* by the second *Delay* block. The final transition from the state *Load* to the state *Idle* is triggered through the on release

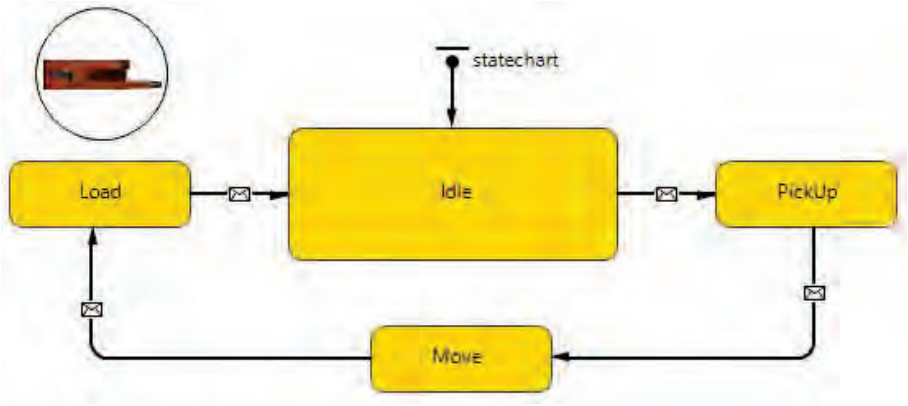

Figure 3.22: Statechart and presentation of the *wCE_Robot* agent.

action of the robot *Release* blocks.

Similar to the operator, the statecharts state entry and exit actions are used to track the time each state is active for statistical and analytical purposes, while the entry actions are used to change the cylinder color of the robot presentation to visually display the robot status in the workcell environment. If chosen through the simulation options, the robot can prioritize actions through the task priority assigned to arriving agents at each robot *Seize* block similar to the operator as well.

3.6.3 Clamping Machine

The agent *wCE_ClampingMachine* is used to model the clamping and unclamping of items to pallets. The agent is embedded in *wCE_Cell*, is used as a resource by the process flowcharts through the resource pool *ClampingMachines* at *wCE_Cell*, tracks the clamping machine status through its statechart, and contains an object and cylinder as its 2D/3D presentation to visually display the clamping machine and its status in the workcell model environment. The clamping machine statechart and presentation are shown in Figure 3.23.

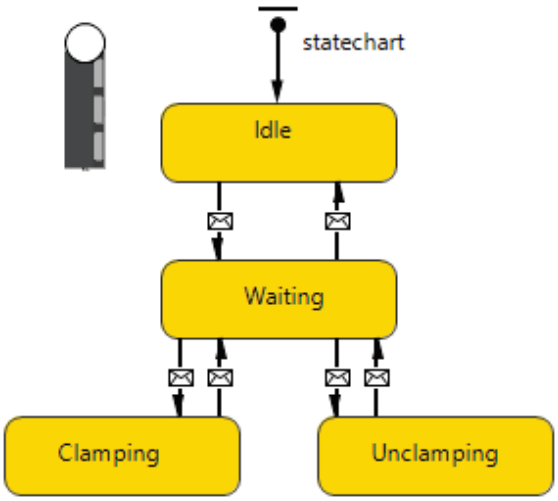

Figure 3.23: Statechart and presentation of the *wCE_ClampingMachine* agent.

The clamping machine statechart consists of four states representing the status of the clamping machine: *Idle*, *Waiting*, *Clamping* and *Unclamping*. The state *Idle* represents the machine being idle, the state *Waiting* represents the machine being seized or reserved while not actually clamping or unclamping items, the state *Clamping* represents the machine clamping items on pallets, and the state *Unclamping* represents the machine unclamping items from pallets. Similar to other resource agents, the statecharts state entry and exit actions are used to track the time each state is active for statistical and analytical purposes, while the entry actions are used to change the cylinder color of the clamping machine presentation to visually display the robot status in the workcell environment.

Upon the creation of *wCE_ClampingMachine* the statechart enters the state *Idle*. The state transitions are triggered through messages send to the clamping machine statechart through actions from process flowchart blocks. The transition from *Idle* to *Waiting* is triggered by on seize actions from clamping machine *Seize* blocks. The transition *Waiting* to *Clamping* is triggered by the on enter action and the transition from *Clamping* to *Waiting* by the on exit action of the *Delay* block used to model the clamping process. Similarly, the transition from *Waiting* to *Unclamping* is triggered by the on enter action and the transition from *Unclamping* to *Waiting* by the on exit action of the *Delay* block used to model the unclamping process. At the creation of *wCE_ClampingMachine* the statechart enters the state *Idle*. Finally, the state transition from the state *Waiting* to the state *Idle* is triggered by the on release action of clamping machine *Release* blocks.

3.6.4 Milling Machines

The milling machine agents *wCE_MillingMachine1* and *wCE_MillingMachine2* are already introduced in Section 3.4, as they also serve as structural agents providing the milling machine environment for the workcell model. Both machines also serve as resources used throughout the workcell model process flowcharts through their respective resource pools *Machine1* and *Machine2*. As both machines are based on the same agent type, the statechart used to track the machine status and the presentation are identical. The milling machine 2D/3D presentation consists of an object to represent the physical milling machine and an cylinder to show the milling machine status. The statechart and presentation for both milling machines is shown in Figure 3.24.

The milling machine statechart consists of five states and is used for similar purposes as the previous resource agents statecharts, to track the active time of each state for statistical and analytical purposes through the state entry and exit actions, and to change the presentation cylinder color through state entry actions to display the milling machine status in the model environment. The statechart is used to track the milling machine status when it is exchanging tools from special toolsets with the cell and when the machine is used to exchange and mill items. Manual tool exchanges between the milling machine toolbelt and operator are not included as they do not require the machine being seized or reserved and can occur while the machine is being used for other purposes as well.

Upon the creation of each milling machine agent, their respective statecharts enter the state *Idle*, representing the machine being idle. Through the tool process flowchart milling machine *Seize* block on seize actions, the statechart transition from the *Idle* to the state *ToolChange* is triggered. The *ToolChange* state represents the milling machine being seized or reserved to either retrieve a tool from the cell and store it in the toolbelt, or retrieve a tool from the toolbelt for pickup by the cell. Through the on release action of milling machine *Release* blocks in the tool process flowcharts, the reverse transition from *ToolChange* to *Idle* is triggered.

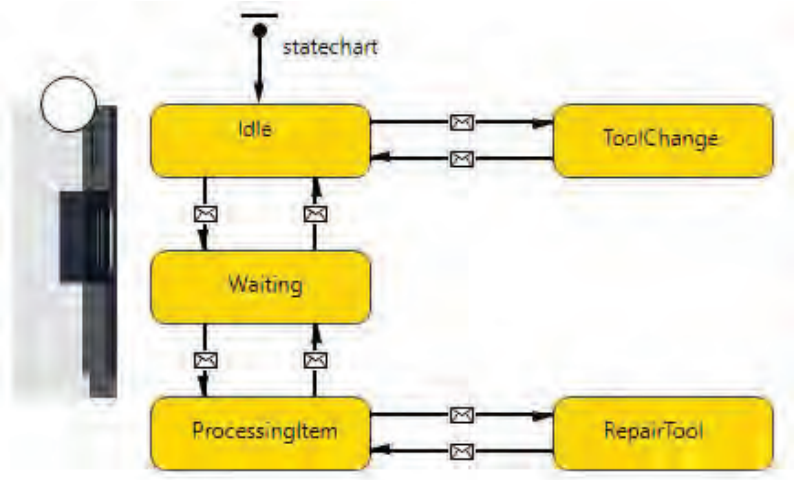

Figure 3.24: Statechart and presentation of the *wCE_MillingMachine1* and *wCE_MillingMachine2* agents.

The on seize action from milling machine *Seize* blocks in the item process flowcharts triggers the transition between the state *Idle* and the state *Waiting*. The state *Waiting* represents the milling machine being seized for milling while not actually milling or waiting on a tool repair during the milling process. Through the on release action of milling machine *Release* blocks part of the item process flowcharts the reverse transition from *Waiting* to *Idle* is triggered. The on enter action of the *Delay* block modeling the item milling part of each milling machine item process flowchart triggers the state transition from *Waiting* to *ProcessingItem*, while the on exit action of the same block triggers the reverse transition from *ProcessingItem* back to *Waiting*. The *ProcessingItem* state represents the milling machine milling items.

The final state, *RepairTool*, represents the interruption of the milling process due to tool failure. Through an event part of the milling machine functionalities to determine tool failure and initiating tool repairs the transition between *ProcessingItem* and *RepairTool* is triggered upon tool failure. If the tool is repaired, the repaired tool arriving back at the toolbelt inventory *Wait* block of the milling machine tool process flowchart triggers the transition from *RepairTool* to *ProcessingItem* through the on enter action. If the tool cannot be replaced immediately the same transition is triggered through the milling machine `AbortProcess()` function instead.

3.6.5 Item and Pallet Exchange

Both the *wCE_ItemExchange* and *wCE_PalletExchange* agents are used to control the process flowchart progression of items, pallets and tools to model their exchange between the operator and the robot through the item exchange and pallet exchange area respectively. Even though both exchange agents are not used as resources through resource pools in the process flowcharts, they act similar to resources agents through functions and an event determining the release of agents in relevant process flowchart *Wait* blocks. Both exchanges contain the `OpenExchange()` function to open the exchange, the `CloseExchange()` function to close the exchange, and the `CheckStatus()` function to control the exchange behavior and related process flowchart progression based on the status of the exchange and the relevant process flowchart blocks.

Similar to the resource agents described previously, both exchange agents contain a statechart to track the exchange area status and a 2D/3D presentation consisting of a cylinder and text

to display the exchange area status in the workcell model environment. The state entry and exit actions of both statecharts are used to track the time each state is active for statistical and analytical purposes, and the state entry actions are used to change the presentation text to the appropriate status. Since the statechart of each exchange tracks what each exchange is being used for, if anything, a variable and the presentation cylinder is used to track and display whether the exchange is open or not. An open exchange represents it can be used by the operator, while a closed exchange represent it can be used by the robot. The item exchange statechart and presentation are shown in Figure 3.25, while the pallet exchange statechart and presentation are shown in Figure 3.26.

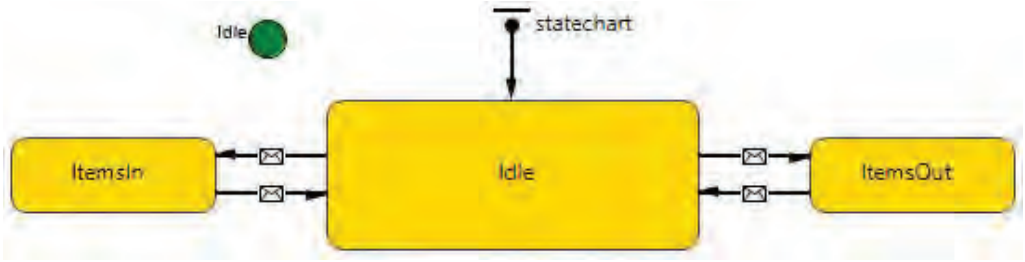

Figure 3.25: Statechart and presentation of the *wCE_ItemExchange* agent.

The item exchange area can be used to exchange unclamped items between the operator and the robot from one itembatch or workorder, and either loaded into the cell or retrieved from the cell. Therefore, the item exchange statechart consists of three states: *Idle*, representing the exchange not being used; *ItemsIn*, representing items being loaded into the cell; and *ItemsOut*, representing items being retrieved from the cell. Upon the creation of *wCE_ItemExchange* the statechart enters the *Idle* state. The entry action of the *Idle* state executes the `CheckStatus()` function. If one or more items from the same itembatch are waiting to unclamp and exit the cell in the *wCE_Cell* item process flowchart through the item exchange, the items are released from the *Wait* block and the transition from *Idle* to *ItemsOut* is triggered, closing the item exchange through the statechart transition action as well. After the exiting items are processed through the item exchange by the process flowchart blocks, opening the item exchange before retrieval by the operator in the process, the next batch of waiting items to exit the cell are released and the exchange is closed again. If no items are waiting to exit the cell the state transition from *ItemsOut* to *Idle* is triggered, closing the item exchange through the statechart transition action.

The item exchange prioritizes items exiting the cell over entering the cell. If the item exchange is idle and only itembatches are waiting for the item exchange in the itembatch process flowchart, the `CheckStatus()` function releases the longest waiting itembatch from the *Wait* block and triggers the transition from *Idle* to *ItemsIn*, opening the item exchange through the transition action. After the itembatch is split and all items processed through the item exchange into the cell, closing the exchange before retrieval by the robot, the next waiting itembatch is released for entry into the cell and the item exchange is opened again. If no itembatches are waiting the transition from *ItemsIn* to *Idle* is triggered instead, closing the item exchange through the transition action.

The pallet exchange area is used to exchange tools, empty pallets or clamped items between the operator and the robot individually, and either into the cell or from the cell. Therefore, the item exchange statechart, shown in Figure 3.26, consists of seven states: *Idle*, *ToolIn*, *ToolOut*, *PalletIn*, *PalletOut*, *ItemIn*, and *ItemOut*. These states represent the pallet exchange not being used, used to enter tools into the cell, exit tools from the cell, enter pallets, exit pallets, enter items and exit items respectively. The pallet exchange prioritizes pallets over items, tools over both, as well as exiting tools over entering tools, exiting pallets over entering pallets, and exiting

items over entering items.

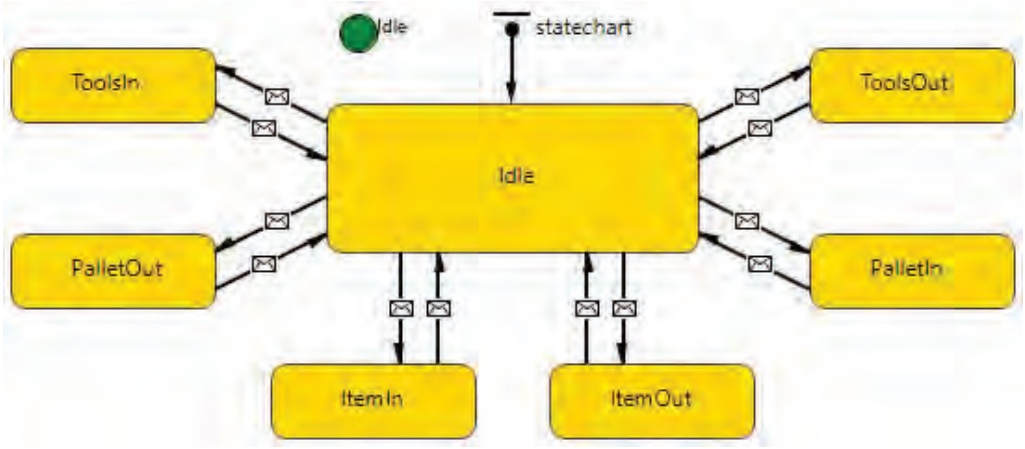

Figure 3.26: Statechart and presentation of the *wCE_PalletExchange* agent.

Upon the creation of *wCE_PallletExchange* the statechart enters the *Idle* state, executing the `CheckStatus()` function through the *Idle* state entry action. If tools are waiting in the *wCE_Cell* tool process flowchart, the longest waiting tool is released from the *Wait* block and the state transition from *Idle* to *ToolOut* is triggered, closing the pallet exchange through the transition action. After the tool is process by the process flowcharts through the pallet exchange, opening the exchange before retrieval by the operator from the exchange in the process, the next waiting tool to exit is released from the *Wait* block and the exchange is closed again. If no tools are waiting to exit the cell, the state transition from *ToolOut* to *Idle* is triggered instead, closing the exchange through the transition action.

if the exchange is idle and tools waiting in the *wCE_OutsideCell* tool process flowchart to enter the cell are the highest priority waiting agents for the pallet exchange, the longest waiting tool is released from the *Wait* block, the state transition from *Idle* to *ToolIn* is triggered, and the exchange is opened through the transition. After the tool is processed through the flowcharts, closing the exchange before retrieval by the robot in the process, the next waiting tool to enter the cell is released and the exchange opened again. If no more tools are waiting to enter the cell, the transition from *ToolIn* to *Idle* is triggered instead and the exchange is closed through the transition.

Similar to the tool exchange process, if the exchange is idle and pallets waiting in the *wCE_Cell* pallet process flowchart to exit the cell are the highest priority waiting agents for the pallet exchange, the longest waiting pallet is released from the *Wait* block, the state transition from *Idle* to *PalletOut* is triggered, and the exchange is closed through the transition. After the tool is processed through the flowcharts, opening the exchange before retrieval by the operator in the process, the next waiting pallet to exit the cell is released and the exchange closed again. If no more pallets are waiting to exit the cell, the transition from *PalletOut* to *Idle* is triggered instead and the exchange is closed through the transition.

if the exchange is idle and pallets waiting in the *wCE_OutsideCell* pallet process flowchart to enter the cell are the highest priority waiting agents for the pallet exchange, the longest waiting pallet is released from the *Wait* block, the state transition from *Idle* to *PalletIn* is triggered, and the exchange is opened through the transition. After the pallet is processed through the flowcharts, closing the exchange before retrieval by the robot in the process, the next waiting pallet to enter the cell is released and the exchange opened again. If no more pallets are waiting

to enter the cell, the transition from *PalletIn* to *Idle* is triggered instead and the exchange is closed through the transition.

Similar to both the tool and pallet exchange processes, if items waiting in the *wCE_Cell* item process flowchart to exit the cell are the highest priority waiting agents for the exchange, the longest waiting item is released from the *Wait* block, the state transition from *Idle* to *ItemOut* is triggered, and the exchange is closed. After the item is processed through the flowcharts, opening the exchange before retrieval by the operator, the next waiting item to exit the cell is released and the exchange closed again. If no more items are waiting to exit the cell, the transition from *ItemOut* to *Idle* is triggered instead and the exchange is closed.

If only items waiting in the *wCE_OutsideCell* item process flowchart to enter the cell are waiting for the exchange, the longest waiting item is released from the *Wait* block, the state transition from *Idle* to *ItemIn* is triggered, and the exchange is opened. After the item is processed through the flowcharts, closing the exchange before retrieval by the robot, the next waiting item to enter the cell is released and the exchange opened again. If no more items are waiting to enter the cell, the transition from *ItemIn* to *Idle* is triggered instead and the exchange is closed.

3.7 Material Agents

In this section the agents representing the workorders, itembatches, items, toolsets, tools, pallets, and chipcontainers are described. As introduced earlier, these agents are referred to as material agents and are added and removed from the model during simulation runtime as needed through their respective agent population embedded in the main agent *wCE_OutsideCell*. All material agents, except for *wCE_Workorder* agents, are used in process flowcharts, while the *wCE_Workorder*, *wCE_Itembatch* and *wCE_Toolset* agents contain a statechart, functions, and events to track and control the progression of the material agents throughout the model.

Similar to the resource agents, the material agents track the duration their states from statecharts are active through the state entry and exit actions and track the duration of their progression in the process flowcharts through the flowchart block actions, including on enter and on exit actions, for statistical and analytical purposes when applicable. The material agents also contain navigation bars and viewareas to view the agent states and navigate through the model during simulation runtime. In the remainder of this section the functionalities of each material agent type are described.

3.7.1 Workorder Agents

The *wCE_Workorder* agents, representing the workorder documentation in the workcell, are used to determine the characteristics of the batch of items to be milled and the special toolset required for the milling process, in order to create the *wCE_Itembatch* and *wCE_toolset* agents representing the itembatches and toolsets accordingly. Workorder agents are created through functions and events modeling the arrival of workorders included in the main agent *wCE_OutsideCell* based on the simulation options. The *wCE_Workorder* agents are created by adding a new workorder to the *wCE_Workorders* agent population and executing the `setupOrder()` function of the added workorder with the appropriate arguments.

After the workorder creation the workorder characteristics of the itembatch, the corresponding special toolset PL number, and the arrival times of the itembatch agent and toolset are determined

through the `setupOrder()` function based on the function arguments and simulation options. The itembatch characteristics include the batch size, the number of the milling machine, the milling or process time, whether the items are clamped inside or outside the cell, as well as values used to determine variable clamping and unclamping times per item upon item agent creation.

Workorder agents also contain a statechart to execute functions and events to initiate the creation of the *wCE_ItemBatch*, to either reserve an existing or initiate the creation of a corresponding toolset with the appropriate PL number. During simulation runtime the statechart also shows if the associated itembatch and/or toolset have arrived at the workcell. The statechart consists of five states and an end state. The 2D/3D presentation of the workorder, consisting of a rectangular box and text showing the unique ID number of the workorder, are instantly moved to the workorder area in the workcell environment. Figure 3.27 shows the workorder statechart and presentation.

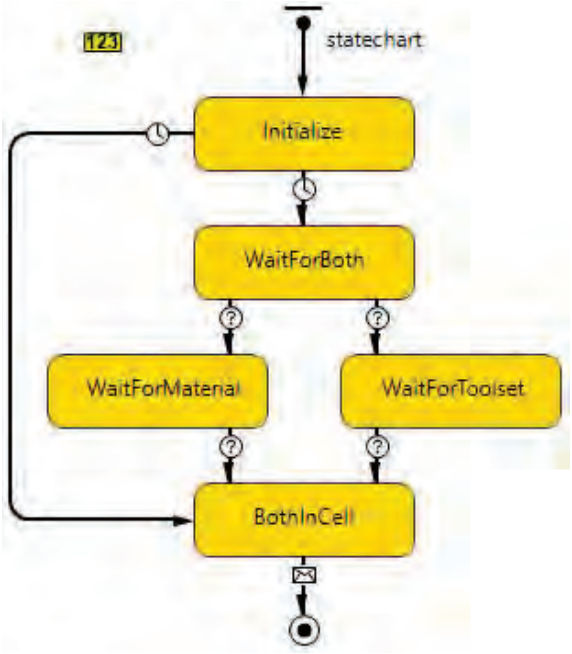

Figure 3.27: Statechart and presentation of the *wCE_Workorder* agents.

The first state, *Initialize*, is entered upon the workorder creation and is used to record the first statistical workorder data. Assuming the workorder was not created as a suborder of a current workorder already in production, after a 1 millisecond delay the statechart transitions from *Initialize* to *WaitForBoth*. Through this state entry action the `CheckToolset()` function to either reserve or initiate the creation of the corresponding toolset agent is executed. The event *ItemArrival* to initiate the creation of the itembatch agent is scheduled according the itembatch arrival delay as well.

Through condition transitions using boolean variables part of the workorder, which are set to true upon the creation of the itembatch or reservation or creation of the special toolset, the statechart state transitions are triggered. When the *ItemBatchInSystem* boolean variable is true, the state transition from *WaitForBoth* to *WaitForToolset* is triggered, and when the *ToolsetInSystem* boolean variable is true, the state transition from *WaitForBoth* to *WaitForMaterial* is triggered. The state transition from *WaitForToolset* to *BothInSystem* is triggered when *ItemBatchInSystem* equals true, while the state transition from *WaitForMaterial* to *BothInSystem* is triggered when

WaitForToolset equals true.

After the milling process controlled through the itembatch agent is completed, the state transition from *BothInSystem* to the end state is triggered through the removal process of the itembatch agent from the simulation model. The action of the end state is used to update the workorder statistical data and remove the workorder agent from the simulation as well.

The `CheckToolset()` function, executed through the entry action of the *WaitForBoth* state, determines if a toolset with the appropriate PL number is present in the workcell and available for use. If no toolset is available, the event *RequestToolsetTimer* is scheduled to initiate the creation of a new toolset according to the toolset arrival delay. If a toolset is available in the workcell, this toolset is reserved by the workorder. If this toolset cannot mill the entire batch of the workorder, the workorder is split into two suborders. The current workorder batchsize is adjusted to the number of items that can be milled by the reserved toolset, and through the workorder `splitStart()` function a second workorder is created. This second workorder inherits all the itembatch characteristics and PL number from the original workorder with the batchsize adjusted to the remaining items not milled by the first workorder. The arrival times for the itembatch and toolset from the new workorder are delayed by 24 hours compared to the arrival times determined by the original workorder. The color of the rectangle presentation of both workorders is changed to show these orders are suborders.

Finally, a workorder is also split into suborders if a failed tool during milling cannot be immediately replaced at the TSC, and if one or more of the items of the workorder are already milled. Through the `SplitCell()` function a new workorder is created to inherit all the workorder characteristics for the remaining items to be milled. This function also executes the `SplitBatchWOCell( WCE_Workorder workorder)` function at itembatch of the workorder, with the created suborder as the input argument, to create a second itembatch and transfer the remaining items of the itembatch accordingly. Tracked and controlled through the itembatch agent, the original workorder now only contains milled items and is further processed for completion and removal from the workcell.

3.7.2 ItemBatch Agents

The *wCE_ItemBatch* agents, representing the batches of items as a singular entity, are used to track and control the progression of the batch of items throughout the workcell model The are also used in process flowcharts when items are handled as a batch, and are used to create the *wCE_Item* agents when splitting the batch. Itembatches are created by the `SetupItemBatch (WCE_Workorder workorder)` function at *wCE_OutsideCell*, when executed by the workorder with itself as the input argument. The new itembatch inherits the itembatch characteristics from the workorder determined previously, including the batch size, the milling process time, milling machine number, and clamping location.

Each itembatch consists of a statechart, various functions and events to track and control of its milling process progression, and a 2D/3D presentation consisting of a container object, rectangle, and text to display its unique ID number. The presentation is visible in the workcell environment before the itembatch is split and after the items are combined into a batch again. Both the itembatch statechart and presentation are shown in Figure 3.28.

After creation the itembatch is used in the first itembatch process flowchart described previously in Section 3.5 and shown in Figure 3.8. In addition to the itembatch process flowcharts progression, the progression of the workorder milling process is tracked and partially controlled through

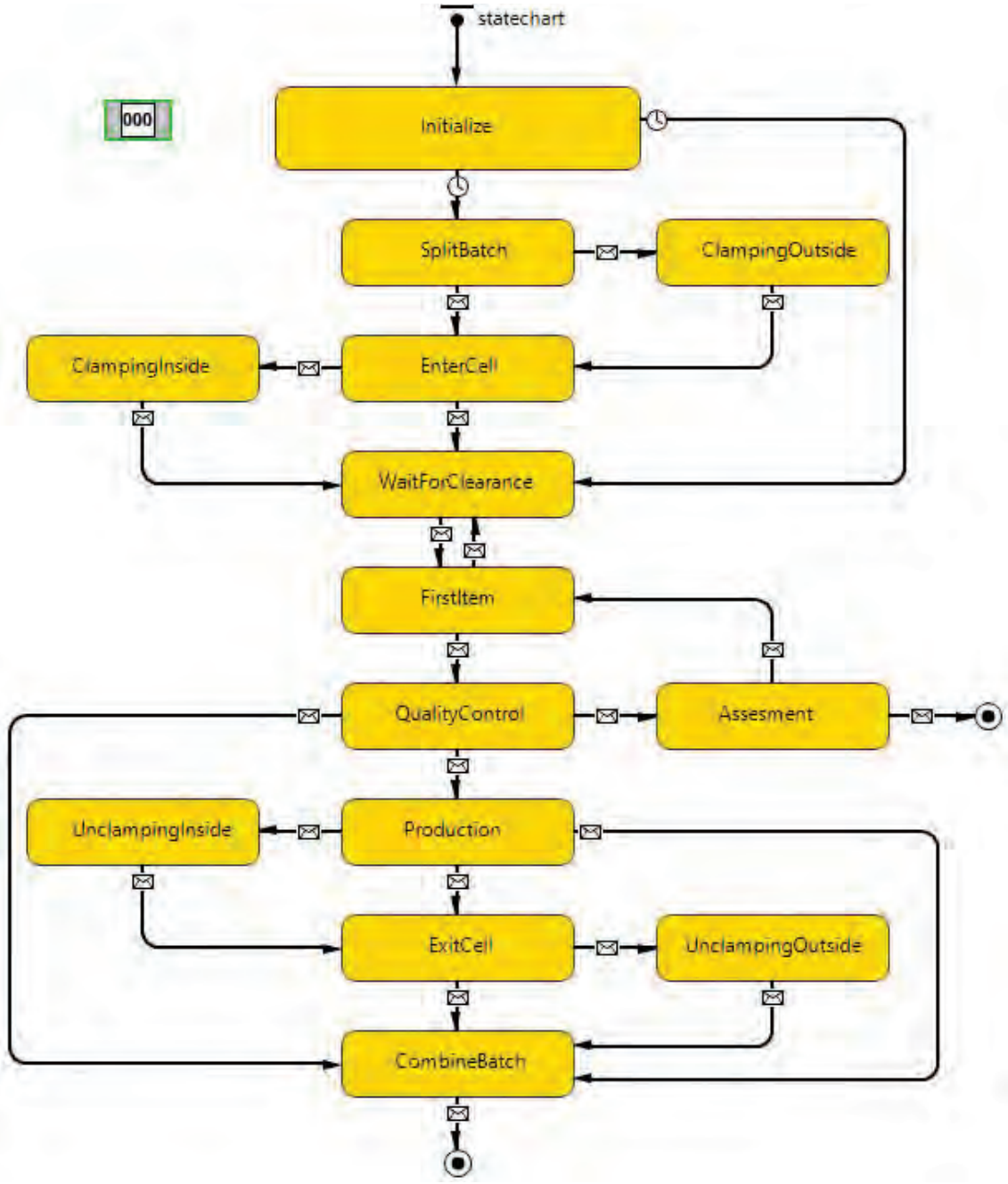

Figure 3.28: Statechart and presentation of the *wCE_Itembatch* agents.

the itembatch statechart, functions and events. Upon creation of the itembatch, the statechart enters the first state *Initialize*, used to delay further progression through the statechart by one millisecond to ensure the itembatch completely inherited its characteristics. This state also updates the workcell and its own statistics through the entry and exit actions. If the itembatch was not created by a suborder after a workorder is split during production, the statechart transitions from *Initialize* to *SplitBatch*.

Upon exiting the itembatch process flowchart, if the itembatch items are clamped by the operator, the transition from *SplitBatch* to *OutsideClamping* is triggered through the flowchart exit block action. If the itembatch items are clamped inside the cell, the transition from *SplitBatch* to *EnterCell* is triggered instead. The *Splitbatch* exit action executes the `split()` function and turns the itembatch presentation invisible in the workcell environment. The itembatch `split()` function creates the *wCE_Item* agents, assigns the relevant itembatch characteristics to each item, and determines the clamping and unclamping times for each item individually. Additionally, each item is assigned a unique batch number determining the order in which the items are milled. The items are entered into the appropriate process flowchart depending on the clamping location.

If the items of the itembatch are clamped outside, the state transition from *OutsideClamping* to *EnterCell* is triggered after all items are clamped and the pallets for mounting the clamped items onto pallets are reserved through the appropriate item process flowchart *Wait* block action. After all items are mounted on pallets and loaded into the cell inventory, the transition from *EnterCell* to *WaitForClearance* is triggered via message through the *wCE_Cell* item process flowchart.

If the items of the batch are clamped inside the cell, the state transition from *EnterCell* to *InsideClamping* is triggered after all items are loaded into the item exchange and the exchange is closed through the item process flowchart. After all items are clamped and entered into the cell inventory, the transition from *InsideClamping* to *WaitForClearance* is triggered via the message through the *wCE_Cell* item process flowchart as well.

The itembatch *WaitForClearance* entry state action is used to initiate the production clearance process for itembatches through lists and events part of *wCE_OutsideCell*, detailed in Section 3.8. When the itembatch is cleared for production, the state transition from *WaitForClearance* to *FirstItem* is triggered. The exit action of *WaitForClearance* adds itself and the first item to the appropriate lists used in the milling process also detailed in Section 3.8.

After the first item is successfully milled through the milling process and cleared to leave the cell through the process flowchart progression, the transition from *FirstItem* to *QualityControl* is triggered. If the milling process for this item is aborted due to a failed tool that cannot be immediately replaced, the transition back from *FirstItem* to *WaitForClearance* is triggered by the workorder `SplitCell()` function.

Upon entering the *QualityControl* state, if the itembatch consists of only one item, the linked toolset is released from the workorder through the `ReleaseToolset()` function, as there are no more items to mill. When the item quality inspection is successfully completed, the state transition from *QualityControl* to *CombineBatch* is triggered if the batch size equals one. If the batch size is larger, the state transition from *QualityInspection* to *Production* is triggered instead. If the item has failed the quality inspection, the transition between *QualityControl* and *Assessment* is triggered.

If the itembatch consists of only one item, the transition from *Assessment* to the final state for discarded itembatches is triggered through the item quality control process flowchart. The

itembatch is discarded and removed from the model as their are no more items to mill. If the itembatch consists of more items, the transition from *Assessment* to *FirstItem* is triggered after the assessment by the operator is completed. Through the transition action the last item of the batch is changed to the first item of the batch and added to the milling process. Since the item that failed the quality inspection is removed from the workorder and discarded, the batch size of the workorder, itembatch, and all items part of the itembatch is reduced by one. The milling process of the first item process starts again, but the quality inspection for this second first item is always successful.

After a successful quality inspection, the remaining items are added to the milling process by the `AddItemsToList()` function executed through the *Production* state entry action. If the milling process for one of the items is aborted, the workorder is split through the workorder `SplitCell()` function and the `SplitBatchWOCell(WCE_Workorder workorder)` function of the itembatch is executed with the new suborder as the input argument as well. This function creates the new itembatch for the new suborder and adjust the itembatch to only contain the completed items. If the only completed item is the first item that passed the quality inspection, the state transition from *Production* to *CombineBatch* is triggered. The new itembatch transitions from the first state *Initialize* to *WaitForClearance* after one millisecond.

If the itembatch consists of more than one item after the workorder split, or if all items of the itembatch are successfully milled and cleared to leave the cell through the process flowchart, the transition from *Production* to *InsideUnclamping* is triggered for items clamped inside. For items clamped outside the transition from *Production* to *LeaveCell* is triggered instead. Through the *Production* state exit action the toolset is released from the workorder.

For items clamped inside the cell, after items are unclamped by the clamping machine, loaded into the item exchange, and the exchange is opened, the transition from *InsideUnclamping* to *ExitCell* is triggered. After all items are retrieved from the item exchange and the complete batch are waiting to be combined in the item process flowchart, the transition from *ExitCell* to *CombineBatch* is triggered. Alternatively, for items clamped outside the cell, after all items are retrieved from the cell through the pallet exchange and separated from their pallets, the state transition from *ExitCell* to *OutsideUnclamping* is triggered. When all items of the itembatch are waiting to be combined after further unclamping, the state transition from *OutsideUnclamping* to *CombineBatch* is triggered.

Through the *CombineBatch* entry action, the `Combine()` function to combine the items into a batch again is executed and the itembatch presentation is made visible again in the workcell environment. The `Combine()` function releases the items of the batch from the *Wait* block in their respective process flowchart to exit the flowchart and removal from the workcell. The itembatch is entered into the appropriate process flowchart based on its clamping location. After progressing through the process flowchart the final state transition from *CombineBatch* to the final state is triggered on flowchart exit. Through the action of this state the itembatch is removed from the simulation.

3.7.3 Toolset Agents

Similar to the itembatch agents, the *wCE_Toolset* agents, each representing a set of tools as a singular entity, are used to track and control the progression of the toolsets throughout the workcell. In addition, they are used in process flowcharts when the tools are handled as a set, and are used to create the *wCE_Tool* agents the set consists off. The toolset agents are used to model both the standard toolset part of each milling machine toolbelt, and the special toolsets

required for the milling process per workorder.

The standard toolsets are created by the event *InitialStandardToolsets* executed at the start of a simulation run contained in *wCE_OutsideCell*, while the special toolsets are created by the function `SetupToolset(WCE_Workorder workorder)` part of the main agent instead. This function is executed through the workorder agent when a new special toolset is scheduled to arrive, with the workorder itself as the argument. Throughout this section the special toolsets are referred to simply as toolsets, while the standard toolsets will be explicitly named for clarity.

Each *wCE_toolset* agent contains a statechart, functions and an event to track and control its progression and create the *wCE_Tool* agents. When tools are processed throughout the workcell as a set, the toolset 2D/3D presentation, consisting of a gray container, a cylinder block and text, is visible in the model environment. The toolset agent statechart and presentation are shown in Figure 3.29.

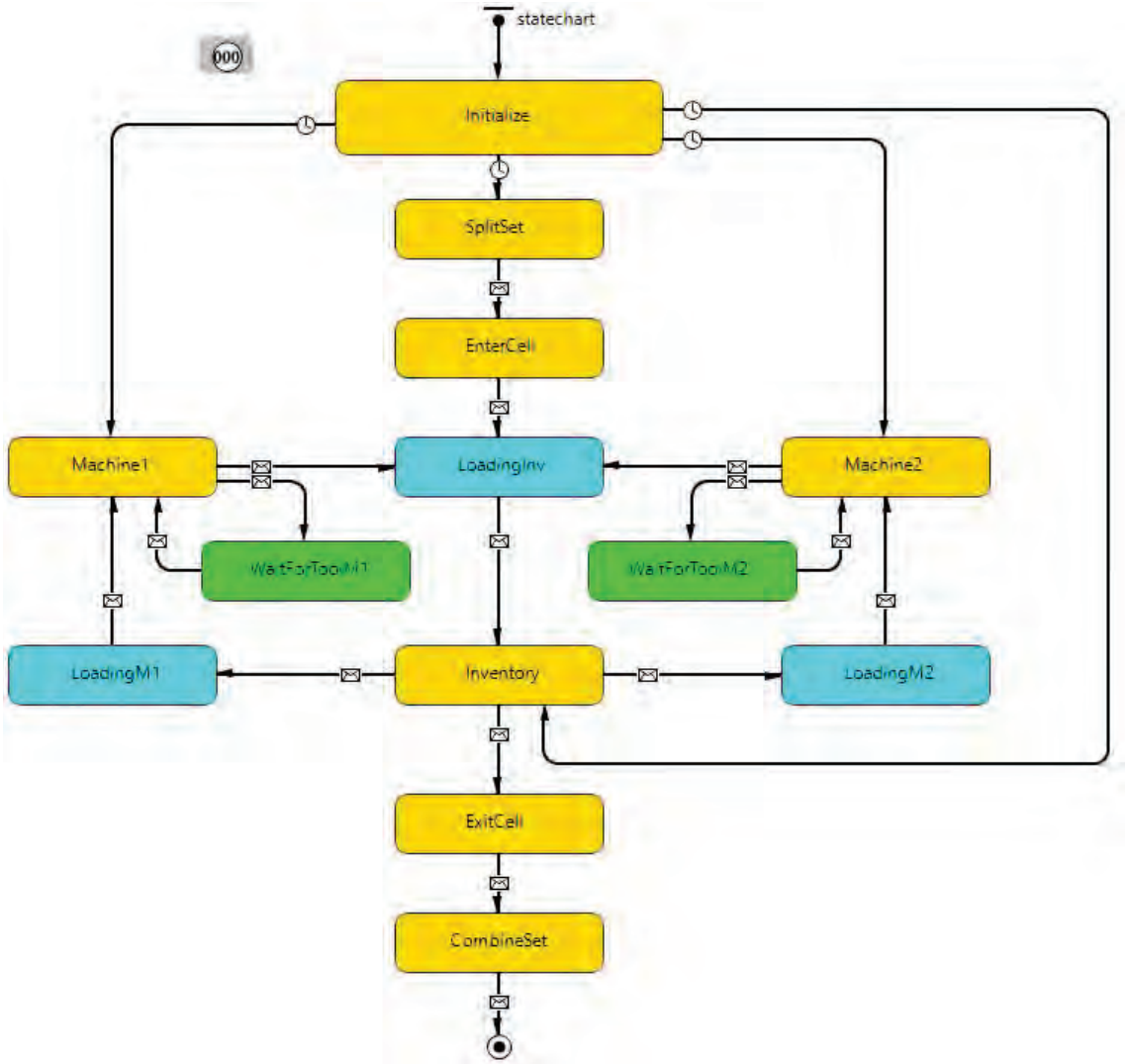

Figure 3.29: Statechart and presentation of the *wCE_Toolset* agents.

Through the function or event that adds toolsets into the simulation the toolset characteristics are assigned to the created toolset. The toolset characteristics are determined by the simulation options or inherited by the workorder that initiated the creation of the toolset. These character-

istics include defining the toolset as a standard or special toolset, the toolset PL number, the number of tools the set consists off, and the number of items the standard tools or special toolset are allowed to mill before replacement. The special toolsets are also reserved by the workorder that required them and inherit the machine number the workorder items are milled by.

Upon creation of a toolset, the toolset statechart enters the first state *Initialize*. After a one millisecond delay the appropriate state transition is triggered to start the toolset progression throughout the workcell model. The *Initialize* exit action is used to set the presentation text and cylinder block color based on the toolset characteristics. The statechart of the standard toolset for the first milling machine transitions from *Initialize* to *Machine1*, the statechart for the standard toolset for the second milling machine transitions from *Initialize* to *Machine2*, and the statechart of the special toolsets transitions from *Initialize* to *SplitSet*. The statechart transition from *Initialize* to *Inventory* is part of the model feature to fill the workcell inventory with toolsets at the simulation start not utilized in the developed model for this project.

The transition action to *SplitSet* enters the toolset into the toolset process flowchart. In the flowchart the toolset wait until they are required for milling and the cell has enough tool storage space available for the toolset. This process is controlled through the production clearance process. After the toolset is released to enter the cell and exits the process flowchart the state transition from *SplitSet* to *EnterCell* is triggered, executing the toolset `split()` function through the transition action.

The `split()` function creates the *wCE_Tool* agents and assigns the tool characteristics to each tool determined by its own characteristics and the simulation options. Standard tools are entered into the respective milling machine tool process flowchart, while the special tools are entered into the first tool process flowchart at *wCE_OutsideCell*. When the robot is seized to retrieve the first tool from the pallet exchange to load into the cell inventory, the state transition from *EnterCell* to *LoadInventory* is triggered.

Each tool stored in the cell inventory sends a message to statechart to trigger the state transition from *LoadInventory* to *Inventory* through the cell tool process flowchart *waitAtToolInventory* block entry action. The transition guard executes a function returning true if all required tools are waiting in the tool inventory, or false otherwise. Upon the *Inventory* state entry the milling machine function to start the milling process of the next item is executed if the toolset is linked to a workorder for the appropriate machine.

When the toolset waiting in the cell inventory is required for milling the next item, the state transition to start loading the toolset into the appropriate machine is triggered by a milling machine function. The transition from *Inventory* to *LoadingM1* is triggered by the first milling machine, while the transition from *Inventory* to *LoadingM2* is triggered by the second milling machine instead. Both state transitions release the tools of the toolset from *waitAtToolInventory* for further progression.

Similar to the transition from *LoadInventory* to *Inventory*, the relevant message to trigger the transition from *LoadingM1* to *Machine1* or from *LoadingM2* to *Machine2* is send to the statechart each time a tool from the toolset enters the milling machine tool process flowchart *waitAtMachine* block. A similar guard function for each transition, returning true if all required tools are at the machine tool inventory or false otherwise, to determine if the transition occurs is used as well. The transition to each respective machine state also executes the function to start loading the item for milling at the milling machine.

Both the standard toolset and the special toolset stay in the *Machine1* or *Machine2* state as long as all the tools of the toolsets are stored in the milling machine inventory. If a tool

failure occurs during milling, the failed tool is replaced and the transition from *Machine1* to *WaitForToolM1* or from *Machine2* to *WaitForToolM2* is triggered for the toolset the tool is part of. These transitions are also triggered if one or more standard tools are replaced after the milling process. The tools in question are released from the tool inventory process flowchart block for further progression.

The same message send each time a tool enters the *waitAtMachine* block is used to trigger and the state transition from either *WaitForToolM1* to *Machine1* or from *WaitForToolM2* to *Machine2*. The transitions are only allowed to occur if all tools are stored in the machine toolbelt again or if a failed tool cannot be replaced immediately. Since standard tools can always be replaced and are always part of the machine toolbelt inventory, the standard toolset statechart only transitions between *Machine1* and *WaitForToolM1* or *Machine2* and *WaitForToolM2* until the end of each simulation run.

When the milling process is successfully completed and the special toolset is not required for the next item on the milling machine list, or when the milling process is aborted after a failed tool cannot be replaced immediately, the toolset is removed from the milling machine and stored into the cell again. The transition from *Machine1* to *LoadInventory* or from *Machine2* to *Inventory* is triggered, and all tools are released from the milling machine *waitAtMachine* block for further progression. The statechart transitions from *LoadInventory* to *Inventory* after all tools are stored in the cell inventory again as described earlier.

In case the toolset is being stored in the cell after one of its tools cannot be replaced immediately, the transition occurs after all tools except the failed tool are stored. In the unlikely scenario the toolset consists of only one tool which is waiting for replacement, the transition occurs through the entry action of *LoadInventory* as no tools can trigger the transition are loaded into the cell inventory through the process flowchart. After the tool is replaced and waiting outside the cell through the tool process flowchart, the toolset can be used again for milling. When required by one of the machines, the toolset is loaded into the milling machine as described before, using the *LoadingM1* or *LoadingM2* entry action to release the replaced tool from its current process flowchart block to progress for manual loading into the machine *waitAtMachine* block.

When a toolset is selected to leave the workcell model, either after release from a workorder without any milling time left or to clear inventory space in the cell for reserved toolsets waiting for entry into the cell, the transition from *Inventory* to *ExitCell* is triggered through the functions used to determine if the toolset is to leave the cell. If a toolset is set to leave the cell while not in the state *Inventory*, the *Inventory* entry action triggers the transition instead as soon as the statechart enters the *Inventory* state.

The transition action from *Inventory* to *ExitCell* releases all tools in the cell *waitAtToolInventory* block. As soon as all tools have been retrieved from the cell and are waiting to be combined into a toolset again through the tool process flowcharts, the transition from *ExitCell* to *CombineSet* is triggered. The transition action executes the `combine()` function of the toolset. This function releases the tools from the final *Wait* block, after which the tools exit the flowchart and are removed from the simulation. The toolset is entered into the toolset process flowchart again to finalize removal from the workcell. After progressing through the flowchart, the transition from *CombineSet* to the final state is triggered upon exiting the process flowchart. The final state action removes the toolset from the simulation after the workcell statistics are updated.

3.7.4 Process Flowchart Agents

The remainder of the material agents, representing the items, tools, pallets and chip containers, are used throughout the process flowcharts and serve a more passive role. Their progression throughout the model is controlled through the flowchart block actions, the statecharts from the other material agents and some of the resource agents, as well as the functions and events from various agents.

Each of these agents contain a 2D/3D presentation to show their progression throughout the workcell environment and some functions to change their appearance when applicable. Their presentations are shown in Figure 3.30. Besides their presentation, the agents further contain the navigation bars and viewareas, parameters, variables, and functionalities to assign, store and export the agent characteristics, status, and data for statistical and analytical purposes common among all agents.

Figure 3.30: From left to right: The 2D agent presentation for *wCE_Item*, *wCE_Tool*, *wCE_Pallet* and *wCE_ChipContainer*.

The *wCE_Item* agents are created through the itembatch `split()` function, and entered into the appropriate process flowchart. The item presentation consists of a text on top of a rectangular box on top of a cylinder on top of a pallet object. The text displays the workorder ID number the item is part of. The rectangle represents the unclamped item, and has a different color depending on the clamping location and by which machine the item is milled. The cylinder represents the clamping part after the operator has clamped the item before being mounted on a pallet. Finally, the pallet represents the pallet the item is clamped on either mounted by the operator or directly clamped by the clamping machine. The clamping part is visible in the model environment after the item is clamped by the operator until unclamped from the item again, while the pallet is visible after mounted on the item by the operator until removed again.

Similar to items, the *wCE_Tool* agents are created through the toolset `split()` function, and entered into the appropriate process flowchart as well. The tool presentation consists of a text and cylinder. The text displays the toolset ID number its part of, and the cylinder represents the tool itself. The cylinder color depends on whether the tool is a standard or special tool and for which machine it is reserved to be used in the milling process, if any, and changes color accordingly.

The *wCE_Pallets* are created at each simulation run start by the *InitialPallets* event part of the main agent, and entered into the pallet process flowchart at *wCE_Cell*. The pallet presentation consists of a pallet object and a text showing the ID number of the item that is about to be clamped onto the pallet. The number of pallets added to the simulation depends on the item capacity of the cell, and can be changed through the simulation options. Since each item is clamped on a pallet before being stored in the cell inventory, the number of pallets in the workcell model are used to control the maximum cell item storage capacity, as items cannot enter the cell until pallets are available for clamping.

One *wCE_ChipContainer* is created at simulation start for each milling machine through the *InitialContainers* event part of *wCE_OutsideCell* as well. The filling of each container with chipped material and their replacement when full is controlled through functionalities of the

milling machines. The chip container presentation consists of a container object.

3.8 Model Functions and Decision Algorithms

As introduced and described throughout this chapter, the behavior of agents throughout the simulation model, as well as the progression of agents throughout the various process flowcharts, is often tracked and controlled through Java code defined in actions, functions and events. In this section the actions, functions and events that determine the workcell behavior for the workorder creation process, the pallet reservation process, the production clearance process, and the item milling process are described.

3.8.1 Workorder Creation Process

During each simulation run workorders agents are added to the workorder agent population through the events *InitialWorkorders* and *CreateWorkorder*, as well as the user controlled inject instant workorder button. If set in the simulation options, the *InitialWorkorders* event, scheduled for executing at the simulation start time, creates the specified number of instant workorders. Each time the inject instant workorder button is pushed by the user during simulation an instant workorder is added to the model as well. Instant workorders are workorders for which the itembatch and toolset arrive immediately at the workcell as well, regardless of the chosen itembatch and toolset arrival delay simulation options. Normal workorders are added to the model by the *CreateWorkorder* event, which is restarted and immediately executed through one of various other events or a schedule based on the chosen workorder arrival method. The inject workorder button restarts the *CreateWorkorder* event when pushed as well. The workorder arrival events, buttons, and schedule are all part of the main agent *wCE_OutsideCell*.

3.8.2 Pallet Reservation for Clamping Process

Before items can be clamped and stored in the cell, pallets have to be reserved for all items per workorder. When an itembatch for items clamped inside the cell enters the *waitForAvailablePallets* block in the itembatch process flowchart, or if the first item of an itembatch for items clamped outside the cell enters the *waitForAvailablePallets1* block in the item process flowchart, the workorder ID number is added to the *WOPalletList* collection The event *ReservePalletsTimer* is restarted as well. This event executes the function `ReservePallets()` after one millisecond each time its restarted.

If the *WOPalletList* collection contains one or more workorder ID numbers, the function `ReservePallets()` finds the itembatch with the first workorder ID number from the collection. If enough pallets are waiting in the *wCE_Cell* pallet process flowchart *waitPallet* block, The number of pallets equal to the itembatch batch size are reserved for clamping either outside or inside the cell and released from *waitPallet*. The workorder ID number is removed from *WOPalletList*, *ReservePalletsTimer* is restarted, and the function `ReservePallets()` is executed again. This cycle repeats until *WOPalletList* is empty or not enough pallets are available for the next workorder on the list. The *ReservePalletsTimer* event is restarted through the *waitPallet* on enter action each time an empty pallet enters this block as well.

Released pallets reserved for inside clamping enter the *waitPalletReadyCell* block. The on enter

action of this block checks if any itembatches are waiting in the itembatch process flowchart *waitForAvailablePallets* block and, if any are waiting, gets the first waiting itembatch. If the amount of pallets, including the just entered pallet, waiting in *waitPalletReadyCell* is equal or greater than the first waiting itembatch batch size, the itembatch is released from its block. The number of pallets equal to the batch size of the just released itembatch are released from *waitPalletReadyCell* as well.

All released pallets enter the next *Wait* block, *waitItemReadyCell*, while the released itembatch progresses through its process flowchart, splits and creates the items of the itembatch, and enters each item into the appropriate item process flowchart. All items of the itembatch are loaded into the item exchange, after which the first item retrieved for clamping progresses further until it enters the *waitForPallet* block in the *wCE_Cell* item process flowchart. The on enter action of this *Wait* block executes the function `wCE_OutsideCell.GetPalletCell(agent)` at the main agent, with the item entering the wait block as the input argument. This function finds the first pallet waiting in the *waitItemReadyCell* block, assigns the item ID number to the pallet, sets the text of the pallet presentation to the item ID number, and releases the pallet from the block for clamping. This process is repeated for all items loaded into the item exchange waiting for clamping.

Released pallets for outside clamping enter the *waitPalletReadyOut* block in the *wCE_Cell* pallet process flowchart instead. If any items are waiting in the top agent *waitForAvailablePallets1* block, the on enter action of this block finds all items of the same workorder waiting here. Then, if all items of the workorder are indeed waiting in this block, and the amount of pallets in *waitPalletReadyOut* is equal or greater than the batch size of the waiting item workorder, all items of the workorder and the same number of pallets are released from their respective blocks, and the *GetPalletOutTimer* event at *wCE_OutsideCell* is restarted. The released items enter the *waitForPallet* block, and the released pallets enter the *waitItemReadyOut* block.

Each time the *GetPalletOutTimer* event is restarted, the function `GetPalletOut()` is executed one millisecond later. First, this function checks if there are indeed one or more items and pallets waiting in *waitForPallet* and *waitItemReadyOut*, and the *GetPalletOutInProgress* boolean variable equals false. If true, the function gets the first waiting item and pallet, assigns the item ID number to the pallet, sets the text of the pallet presentation to the item ID number, releases the pallet from its block, and sets *GetPalletOutInProgress* to true.

After the pallet is retrieved from the pallet exchange and enters the *waitForPalletSplit* block in the *wCE_OutsideCell* pallet process flowchart, the on enter action of this block finds the item in the *waitForPallet* block with the matching ID number. If the item is indeed found, the item is released from its block for clamping and storage in the cell inventory afterwards. After the item is clamped to the pallet and retrieved from the pallet exchange by the robot, *GetPalletOutInProgress* is set to false and the function `GetPalletOut()` is executed to repeat the process for the next waiting item and pallet.

3.8.3 Itembatch and Toolset Production Clearance Process

Before the first item of a workorder can be milled, both the complete batch of items and special toolset need to be stored inside the cell inventory. After all items of the same batch are stored in the cell inventory, the itembatch statechart transitions into the state *WaitForClearance*. The entry action of this state enters the itembatch ID number into the *ItemBatchWaitListM1* collection and restarts the event *CheckClearanceM1* at the main agent if the items are milled by the first milling machine, or enters the itembatch ID number into the *ItemBatchWaitListM2*

collection and restarts the event *CheckClearanceM2* if the items are milled by the second machine instead. The event *CheckToolsetClearance* at *wCE_OutsideCell* is also restarted.

Each time *CheckClearanceM1* is restarted, it executes its action one millisecond later. The event action finds all itembatches through the itembatch ID numbers in the *ItemBatchWaitListM1* list and checks for each itembatch if the special toolset for the milling process is reserved. If reserved, the reserved toolset is found and the action checks if the reserved toolset is stored or in the process of being stored in the cell inventory, and not waiting for a replacement tool. If true, the itembatch is cleared to start the milling of its first item by delivering a message to the itembatch to trigger the statechart state transition from *WaitForClearance* to *FirstItem*. Through the *WaitForClearance* exit action the itembatch ID number is removed from *ItemBatchWaitListM1*.

The event *CheckClearanceM2* is identical to *CheckClearanceM1*, except for checking the availability of the special toolset for itembatches with the ID number part of the *ItemBatchWaitListM2* collection instead. Besides the itembatch statechart *WaitForClearance* entry action, both events are also restarted each time a toolset is cleared to enter the cell through the toolset process flowchart *waitForCell* block on exit action. The event for the appropriate milling machine is restarted when a replacement tool arrives at the cell through the item process flowchart on entering the *waitForManualLoad* block as well.

The event *CheckToolsetClearance* executes its action after one millisecond each time the event is restarted. If at least one toolset is waiting in the toolset process flowchart *waitForCell* block, and at least one itembatch is waiting for clearance for any milling machine, the action starts iterating over all toolsets waiting at the *waitForCell* block. If a toolset is reserved and the ID number of the linked itembatch is part of either the *ItemBatchWaitListM1* or *ItemBatchWaitListM2* collection, the action checks if the cell has enough storage space left for all tools of the toolset and the iteration ends. If enough storage space is available, the toolset is released from the *waitForCell* block and the number of tools the toolset contains is added to the current integer variable *NumberOfToolsInCellReserved* value. This variable is used to track the number of tools stored in the cell plus the number of tools cleared for entering the cell. If not enough storage space is available, the function `RemoveToolsets()` is executed instead, with the toolset size as its input argument. In addition to the itembatch statechart *WaitForClearance* entry action, the *CheckToolsetClearance* event is restarted through the itembatch `ReleaseToolset()` function, each time a toolset enters or exits the *waitForCell* block, and when a toolset enters the *wCE_OutsideCell* toolset process flowchart after being combined into a set again.

The function `RemoveToolsets()` creates a list of toolsets that are currently stored in the cell or cleared for entry into the cell, and are not currently reserved for milling by workorders or already cleared for replacement. If this list at least one toolset, the function iterates over the list until enough tool storage space is available based on the toolset size input argument. For each toolset iterated over its function `DoReplace()` is executed. This function is used to trigger its statechart transition from *Inventory* to *ExitCell* if the statechart is currently in the *Inventory* state.

3.8.4 Milling Process

The process of milling items in the workcell model is controlled through various functions, events, actions and collections part of the milling machine agents, as well as actions from other agents. Since both milling machine agents are instances of the same agent type, these functions, events and collections are identical for both machines. The functions, actions, events and collections of a each milling machine can be accessed from other agents through the respective machine

agent name, *wCE_MillingMachine1* and *wCE_MillingMachine2*, while the functions, actions and events part of each machine use the unique ID number to identify itself. The milling process described below describes the process for both machines.

In order to control the milling process, several collections part of the milling machine agent are used to track the items ready for milling, track the status of the standard toolset, and track the status of the special toolset required for milling the next item. The collection *ItemList* contains a list with item ID numbers ready to be milled on the milling machine. When one or more items are cleared for milling, the item ID numbers are added to this collection through the itembatch functions or statechart actions.

The collections *STAList* and *TAList* contain the tool ID numbers of the standard tools and special tools respectively currently stored in the milling machine inventory. The tool ID numbers are added to the appropriate list each time a tool agent enters the milling machine tool process flowchart *waitAtMachine* block, and are removed from the appropriate list each time a tool leaves the *waitAtMachine* block through the on enter and on exit action of the block.

The collections *SPLList* and *SPLWait* contain toolset ID numbers of standard toolsets, while the collections *PLList*, *PLWait* and *PLLoad* contain toolset ID numbers of special toolsets. The toolset ID numbers are added and removed from these collections through the toolset statechart transition actions. At the start of each simulation run, the just created standard toolsets are immediately loaded into the milling machines and the toolset ID number of the standard toolset is added to *SPLList*. When one or more standard tools are being replaced, the standard toolset ID is removed from *SPLList* and added to *SPLWait*. If all standard tools are stored in the machine inventory again, the toolset ID number is removed from *SPLWait* and added back to *SPLList*.

Similarly, when tools of a special toolset start being loaded into the milling machine, the toolset ID number is added to *PLLoad*. When all tools are stored in the machine toolbelt, the toolset ID number is removed from *PLLoad* and added to *PLList*. If during milling a tool of the special toolset part of the machine inventory breaks and is being replaced, the toolset ID is removed from *PLList* and added to *PLWait*. When the tool is replaced and stored back into the machine, or the milling process is aborted instead, the toolset ID number is removed from *PLWait* and added to *PLList* again. When the special toolset is removed from the milling machine, the toolset ID number is removed from *PLList*.

The milling machine functions `RequestToolset()` and `RequestItem()`, as well as the event *RequestToolsetTimer*, are used to initiate the process of storing tools from special toolsets from the cell into the machine and receiving the next item for milling. The *RequestToolsetTimer* event is restarted through the itembatch agents when one or more items are cleared for production and the *ItemList* collection is empty, meaning no items are currently ready and waiting for milling. This event is also restarted through another function of the milling machine itself. Each time the *RequestToolsetTimer* is restarted, the function `RequestToolset()` is executed one millisecond later. This function is also directly executed through the toolset agents statechart each time a toolset is stored in the cell inventory.

The function `RequestToolset()` finds the first item on *ItemList* if this collection is not empty and the collections *PLLoad*, *PLList* and *PLWait* are empty. If the special toolset required for milling this item is part of the model, reserved by the item workorder, not waiting on a tool replacement, and currently completely loaded into the cell inventory, the toolset is initiated for loading into the cell through the toolset statechart.

When all tools of a special toolset are stored in the machine inventory the function `RequestItem ()` is executed through the toolset statechart. This function is also executed through itembatches when adding new items to *ItemList* in case the toolset is already stored in the machine inventory. Through the cell item process flowchart milling machine *Release* block actions after items have been retrieved from the milling machine, the function is also executed.

When executed the function `RequestItem()` checks if the machine is not already currently in the process of milling an item and if the *ItemList*, *SPLList* and *PLList* collections all contain at least one ID number, meaning an item is ready for milling and both the tools of the standard toolset and a special toolset are all stored in the machine inventory. If true, the item and corresponding itembatch agent with the first item ID number in *ItemList* are found. If the item is currently waiting in the cell item process flowchart *waitAtInventory* block, and the special toolset ID number contained in *PLList* is indeed the ID number of the reserved toolset for the itembatch, the item is released from the block, the item ID number is removed from *ItemList*, and the milling machine is reserved for milling an item through the milling machine boolean variable *Reserved*.

When the item is retrieved from the cell and enters the milling machine item process flowchart *processItemM* block, the milling process is started. The on enter action of *processItemM* initiates the tool failure and repair functionalities of the milling machine. These functionalities are used to determine if a tool fails, pause production, initiate the tool replacement process, resume production if the tool can be replaced immediately, and abort production if not.

The functionalities determining tool failure and replacement initiation consists of the parameters *RepairNumber*, *RepairTimePerTool* and *RepairTime*, the collection *TARepairList*, the functions `GetTARepairList()` and `RepairTools()`, and the event *RepairToolEvent*. The on enter action of the *ProcessItemM* block executes the function `GetTARepairList()`, sets *RepairNumber* to one, determines the repair time per tool, assigns this time to *RepairTimePerTool*, and restarts the event *RepairToolEvent* with half of the *RepairTimePerTool* value as the input argument.

The function `GetTARepairList()` removes all tool ID numbers from *TARepairList* before adding the tool ID numbers from the collections *STAList* and *TAList*. The repair time per tool is determined by dividing the milling process time for the current item by the amount of tool ID numbers contained in *TARepairList*. When the event *RepairToolEvent* is restarted its code is executed after half the *RepairTimePerTool* time has expired. When executed, the event action gets the tool ID number at the position in the *TARepairList* collection at the value of *RepairNumber*, meaning the first ID number if *RepairNumber* is one, the second ID number if two, etcetera. The *wCE_Tool* agent corresponding with the tool ID number is found.

Next, using the function `randomTrue(wCE\_OutsideCell.ToolFailureChance)`, with the *ToolFailureChance* parameter at *wCE_OutsideCell* containing the chance value each tool can fail during the milling process set through the simulation options, the action determines if the current tool breaks or not. If a tool failure occurs, the function `RepairTools(tool)` is called with the tool agent as the input argument. In addition, the milling machine statechart transition from *ProcessingItem* to *RepairTool* is triggered, and the *processItemM* block is suspended, halting the milling process of the item. If the tool does not fail, and the tool ID number was not the last number contained in *TARepairList*, the event restarts itself with a timeout set to the *RepairTimePerTool* value. Regardless of tool failure occurring, at the end of the action Java code the *RepairNumber* is increased by one.

The function `RepairTools(tool)` sets the status variables *Repair* and *Manual* to true for the tool agent that is the input argument of the function. These variables are used to determine

if the tool leaves the milling machine inventory for repair and will be retrieved manually by the operator or not. The tool is then released from the tool process flowchart *waitAtMachine* block, and the toolset this tool is part of is located to trigger the appropriate toolset statechart transition.

If the failed tool is immediately replaced, after progressing through the tool process flowcharts the replaced tool enters the *waitAtMachine* block. The on enter action of this block resumes the item milling process by resuming the *processItemM* block. The milling machine statechart transition from *RepairTool* to *ProcessingItem* is also triggered. If the replaced tool was not the last tool in *TARepairList*, the *RepairToolEvent* event is restarted with *RepairTimePerTool* as the timeout value as input.

If the failed tool is not immediately replaced, the `AbortProduction()` function of the milling machine is triggered through the on enter action of the tool process flowchart *waitRepair1* block at *wCE_OutsideCell*. This function sets the boolean variable *ItemAborted* of the milling machine to true, triggers the state transition from *RepairTool* to *ProcessingItem*, and restarts the event *AbortProductionTimer* as well. After being restarted, this event sets the remaining time delay of the *processItemM* block for the item being milled to zero, and resumes the block as well, ending the item milling process immediately.

After milling is completed or aborted, the item agent in *processItemM* is released. Through the on exit action of this block, the milling machine state transition from *ProcessingItem* to *Waiting* is triggered, the functions `UpdateToolset()` and `UpdateStandardTools()` are executed, and the exiting items status boolean variables *QualityControl* and *Processed*, indicating if the item requires a quality inspection and was successfully milled, are updated accordingly.

The function `UpdateToolset()` finds the toolset agent currently present in the milling machine. If the milling process was not aborted, the toolset variable *ItemsLeft* value, used to determine how many items can be milled before the toolset requires replacement, is reduced by one. If there are no items waiting for milling, the toolset is removed from the milling machine through the toolset statechart. If the first item waiting for milling requires a different toolset, the toolset is also removed through the toolset statechart and the *RequestToolset* event is restarted as well.

Finally, the function `UpdateStandardTools()` is used to reduce the remaining items each standard tool can mill by one, and initiate the replacement process of any standard tools with zero remaining items left. The tool status variables for the tools requiring replacement are updated accordingly before being released from the *waitAtMachine* block. The appropriate standard toolset statechart transition is triggered as well, ensuring the next item cannot be milled before all standard tools are replaced and stored back in the milling machine inventory again.

3.9 Simulation Options

In order to run an AnyLogic simulation model an experiment is required. The standard experiment in AnyLogic is the simulation experiment, called *Simulation* in the workcell model. The model also contains parameter variation experiments which are used in the verification process and experiments described in the next chapters. For each experiment the top level or main agent is set, chosen from any agent type part of the model. In all experiments part of the workcell model the agent type *WCE_OutsideCell* is set as the top level agent. All parameters part of the top level agent are available in all experiments through the experiment properties.

For each experiment Java actions can also be defined, including on experiment setup, before simulation, and after simulation. The simulation start time or date for each run can be set in each experiment, as well as an optional end time or date. In the experiment window of the workcell *Simulation* experiment, functioning as the start screen of the simulation model when an experiment is executed from the AnyLogic development environment, text objects and control elements including edit boxes, check boxes, radio buttons and a slider are implemented. The control elements are used to review and change the simulation options before each simulation run after the Simulation experiment is started.

The values used by the control elements are variables added to the *Simulation* experiment. For each variable the initial value is defined, and where applicable, the parameters of the top agent type have a variable assigned to them. Upon the start of a simulation run, the parameter values are set to the value of the variable assigned in the *Simulation* experiment. By default, the *wCE_OutsideCell* agent parameters initial value is equal to assigned variable initial value in *Simulation*. Using the control elements before each simulation run, these values can be altered, assigning a different value to the parameters of the main agent.

The edit box control element value is linked to the appropriate variable part of *Simulation*, setting the value of the linked variable as the default value of the edit box. A minimum and maximum value for this box can be defined through the properties as well. The check box control element value is either true (checked) or false (unchecked), and can be linked to a boolean variable. The slider control element is used to define an integer or real number value between the minimum and maximum value. Again, this value can be linked to a variable and the minimum and maximum values can be assigned directly or through variables as well. Finally, the radio button control element contains a group of buttons of which only one can be selected at the time. An integer variable can be linked to the radio button, setting the variable value to the index value of the selected button starting at zero.

After starting the *Simulation* experiment of the workcell model, the starting window of the experiment is shown in the simulation environment. The simulation environment with the start window displayed is shown in Figure 3.31. The default simulation options are displayed in the experiment window and can be altered by the user before each simulation run is started. The simulation options can be divided into four categories: the general workcell model settings, the workorder arrival method, the workorder type, and the custom workorder characteristics. The simulation options for each category are described next. A complete overview of the simulation options, including the default values, can be found in the second section of AppendixB.

The simulation options under the general workorder model settings include a random number seed, the cell tool and item storage capacity, settings for the operator and robot, and several probabilities determining workorder characteristics and occurrences while processing workorders in the workcell. The random number seed can be specified to allow repeatable simulation runs. The minimum storage capacity for items in the cell is linked to the maximum batch size of workorders defined under the custom workorder characteristics. The minimum storage capacity of tools is similarly linked to the maximum number of tools per special toolset. These minimum capacity values ensure that at least one batch of items and toolset can be stored in the workcell when the workorder and toolset characteristics are determined by the custom workorder characteristics simulation options.

If checked, the operator and robot priorities are used to prioritize certain tasks through the operator and robot process flowchart *Seize* blocks. The operator shift schedule is set to always on shift by default. The probability values are set by real numbers between zero and one. These probability fractions are used to determine if a workorder is processed by the first or second

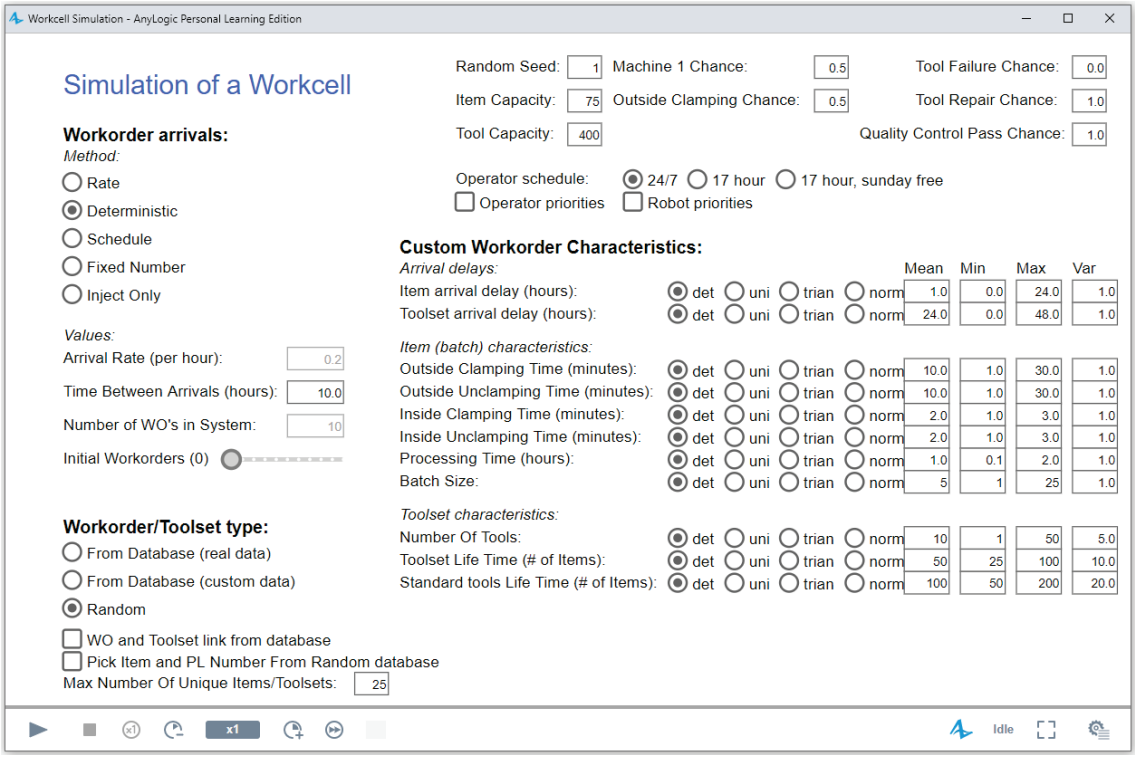

Figure 3.31: Simulation experiment environment showing the workcell model simulation options window, with the simulation run control buttons at the bottom.

milling machine, as well as clamped by the operator outside the cell or the clamping machine within. Also, the probabilities are used to determine if a tool failure occurs during milling for each tool in the milling machine toolbelt, to determine if a failed special tool can be immediately replaced at the TSC, and to determine if the first milled item of the workorder passes the quality inspection at QC.

The workcell simulation model allows for one workorder arrival method per simulation run in addition to the user injecting workorders into the simulation via the injection buttons part of the workcell model environment. Workorders arrive at the workcell deterministically every ten hours by default, where the time between arrivals can be altered by the user. Workorders can also arrive according to an adjustable exponentially distributed rate per simulation run set to 0.2 per hour by default.

If chosen before a simulation run, workorders can arrive according to a weekly schedule part of *wCE_OutsideCell*. Alternatively, workorders can arrive at the workcell immediately until a predefined number of workorders, including suborders, are present at the workcell, with a new workorder arriving as soon as the predefined number of workorders are not present at the workcell anymore. This predefined, or fixed, number of workorders can be defined before each simulation run, and is set to ten by default.

The simulation can also run without any workorders arriving except through the injection buttons. Finally, unless the fixed number of workorders at the workcell option is chosen, up to ten initial workorders can arrive at the cell at the start of the simulation. The itembatch and toolset for the initial workorders arrive immediately at the workcell as well.

Through the workorder type simulation options the workorder is set to determine its characteristics partially from databases or completely based on the custom workorder characteristics simulation options. There are two workorder database options available, real workorder and toolset characteristics from databases based on values from KMWE workorders and toolset compositions, or custom workorder and toolset characteristics from custom databases created by the workcell model developer. Through the random workorder type option the workorder and toolset characteristics are completely determined according to the workorder characteristics simulation settings.

For all workorder types the machine number and clamping location are determined by the probability settings described earlier. For the real type workorders the remaining characteristics are determined by retrieving the values from various databases included in the model. For custom database type workorders, the clamping and unclamping times are also determined using the custom workorder characteristics simulation options, with the remaining characteristic values retrieved from various databases as well. An overview of all databases included in the model can be found in the first section of Appendix B.

In addition, for both custom and random type workorders, the workorder item types and special toolsets can be linked. Linked item types and toolsets use the same special toolset for each item of the same item type. Item types and toolsets are linked through the item type IT numbers and toolset PL numbers defined in the model databases. If item types and toolsets are not linked, and there numbers not retrieved from a database, the random type workorders assign a unique IT and PL number per workorder, meaning toolsets are only used for one workorder.

The custom workorder characteristics retrieved based on the simulation option parameters include the item and toolset arrival delay, clamping times, item milling process time, the workorder batch size, the special toolset lifetime and number of tools per set, and the life time of the standard tools. All these workorder and toolset characteristic values can be determined deterministically or by using a uniform, triangular or normal distribution function. The values for each characteristic include a mean, minimum, maximum and variation value. These values are used for the chosen distribution function accordingly.

3.10 Performance Measurements and Statistics

During simulation runs, data about the workcell performance is collected throughout the model to calculate, display and export the workcell performance measurements and statistics. This data is collected throughout the model agents during runtime and stored into parameters or variables. The collected data is used to calculate the workcell performance through several functions. Text files are setup to export the workcell performance data, as well as data collected for most resource and material agents upon agent destruction during or at the end of each simulation run. An overview of the collected data is given next.

Through the appropriate process flowchart block and material agent statechart actions, the number of workorder, itembatch, item, toolset and tool arrivals, departures and currently in the workcell are tracked during simulation runtime. Additionally, arrivals, departures and currently present in the workcell are tracked for these agents as well. Upon arrival and departure of each of these agents in the system or in the cell, a function for that particular agent type is executed to update the average number, also called the work in progress or WIP level, of the agent type agents in the workcell system or inside the cell. The WIP functions for the items and tools inside the cell calculate their respective cell storage capacity utilization as well.

Using the tracked number of arrivals and departures, the event *UpdateStatistics* calculates the arrival rate and throughput of workorders, itembatches, items, toolsets and tools in the system and inside the cell every hour in simulation time. This event also executes the average WIP functions and calculates utilities for the workcell resources. Each resource agent tracks the duration its respective statechart is in each state. These duration times are used by the *UpdateStatistics* event to calculate the fraction of time each milling machine is milling items, the clamping machine is either clamping or unclamping, the robot is handling items, tools and pallets, and the operator is working. Additionally, the fraction of time working while the operator is on shift and off shift are calculated separately as well.

In addition to the workcell performance data, each workorder, itembatch, item, toolset, and tool agent tracks their individual progression through the workcell as well. The arrival or start times of many processes are collected and used to record the duration of these individual processes throughout their progression through the model. These duration measurements include the time in system, time in cell, time waiting for operator shift, time in cell inventory, time milling, and many more.

When each agent leaves the workcell and is removed from the model, a function for the respective agent type is executed to calculate the average duration, also called the lead time, of the agents of the same type in the workcell model and inside the cell specifically. The average durations for all the other process durations of the agents are calculated as well.

During the simulation, several viewareas can be navigated to in order to view a selection of the current performance measurements, an overview of most performance data, and stack charts showing the average process duration times for the workorders, itembatches, items, toolsets and tools. Upon these agents removal from the model, their individual characteristic and status values, as well as their collected process duration times, are exported to the appropriate text file through print functions. Upon simulation end, the statechart state duration times of the resource agents and all general and average workcell performance data are exported to text files as well.

With all aspects of the developed workcell simulation model described in this chapter, the verification process of the model is described in the next chapter.

Chapter 4

Model Verification and Performance

In the previous chapter, the developed workcell simulation model was described after introducing the AnyLogic software package used to develop the model, and detailing the modeling choices and assumptions made during the development of the model. In this chapter, the verification process for the developed workcell simulation model is described. This verification process is used to verify if the workcell model runs, functions and performs as expected and intended. Besides the verification process, any problems, unexpected or unintended behavior, or model limitations of the developed model discovered during the verification process are documented. The basic workcell model performance determined during the verification process is included in this chapter as well.

The validation process and further development of the workcell model as an accurate representation of the KMWE workcells is beyond the scope of this graduation project. Given the workcell model size and complexity, thoroughly verifying every detail of the developed model is beyond the scope of this graduation project as well. Instead, the verification process of the model focusses on three parts. First, the application of the simulation settings in the workcell model and the creation of workorders, itembatches, items, toolsets and tools with correct characteristics is verified. Second, the model behavior and functionalities to process single workorders are verified and the basic workcell performance is determined. Third, the workcell behavior and functionalities when processing multiple workorders simultaneously is verified, and the workcell performance is for two cases is determined as well.

Before describing the verification process and results of these three parts in the remainder of this chapter, an overview of the verification method used in the process is given. While the verification process results described in this report applies to the developed model described in this report, parts of the verification process where used during the development of the model, and the model was further developed based on testing and verification results as well. During the development of the model the functionalities of the workcell and behavior of the model agents were mainly tested or verified through observing the agent presentations in the workcell environment, the process flowchart block contents, the values of parameters and variables, and statechart state and transitions during simulation runs of the *Simulation* experiment. The export data of the workcell performance and agent statistics was used for verification during the later phases of the development as well.

In general, the methodology used to verify the model starts with specifying what part or function of the model to verify. After specifying what to verify, the simulation input and output are specified. The simulation input consist of determining the simulation experiment type, the

experiment settings and the simulation options. The *Simulation* experiment is used to verify the model behavior and functionalities through observations during simulation, or the export data of a single simulation run after simulation. The parameter variation experiment *Verification* is used to verify the model functionalities over multiple simulation runs through data collected from all runs and exported afterwards. Which components of the workcell model to observe and when to observe during simulation runs, or the parts of the export data to review after simulation, are specified as the simulation output. The expected or intended output results are determined as well.

Finally, the chosen simulation experiment is executed with the specified settings and simulation options as input before the simulation runs. If specified, observations of the simulation model components during simulation runs at the appropriate simulation times are compared with the expected results. Similarly, the specified export data is reviewed and compared with the expected output results after simulation by importing the text files containing the exported data into Matlab or Excel. Matlab is mostly used as the imported data can be sorted, filtered and analyzed through scripts and functions.

During the verification process, most unintended or unexpected behavior discovered when comparing the simulation output with the expected output was analyzed and solved during the end of the model development. For the developed model described in this report, the results of the verification process, as well as the workcell model performance are described next.

4.1 Application of Simulation Options

The first part of the model verification process is focused on the correct application of the simulation options throughout the workcell model. All simulation options related to the workorder arrival/creation methods and values are applied correctly throughout the model. Workorder arrivals at the workcell according to an arrival rate where verified using the *Verification* experiment by running the simulation multiple times varying the random number seed for multiple arrival rate values, and exporting the average arrival rate of each run. The mean and standard deviation of the average arrival rate for all runs with the same arrival rate value where calculated and compared with the set arrival rate value for these runs. All other arrival rate methods where verified using the *Simulation* experiment. The arrival times of the workorders exported after each Simulation run where used to compare with the expected arrival times of workorders.

The workorder and toolset characteristics are derived from either one of the databases or the custom workorder and toolset characteristics simulation options based on the chosen type of workorders and toolsets. The workorder and toolset characteristics are all correctly derived from the databases or simulation options based on the chosen type. The toolset and item type numbers are also correctly linked per workorder when chosen.

All the workorder and toolset characteristics simulation options are correctly applied when determining the workorder and toolset characteristics after creation in the workcell model during simulation. The application for each workorder and toolset characteristic option was verified for each distribution method; deterministic, uniform, triangular, and normal. The build in distribution functions in the AnyLogic software package, including exponential rates, where not independently verified themselves. Since the itembatch, item and tool characteristics are inherited by their respective agents from the workorder and toolset characteristics, their characteristics where also verified successfully as well. The remaining simulation options are also correctly applied throughout the model, including the operator and robot priorities and the

operator shift schedules.

4.2 Model Functionalities and Performance for Single Workorders

The second part of the verification process is focused on verifying the various model functionalities used to process single workorders. For each workorder in the workcell model, the individual item agents created by the itembatch agents progress throughout the model environment and process flowcharts, while the itembatch agents track the progression status of the batch of items through its statechart. Similarly, the special toolset used to mill the items track its progression status while the tools progress throughout the model environment to mill items. After each simulation run using the *Simulation* experiment, the characteristics and progression durations for workorders, itembatches, items, toolsets, and tools are exported for use in the verification process.

Excluding the delayed replacement of failed tools during milling, the associated item and tool agents progress throughout the workcell according to a deterministic path with predictable handling and processing times when processing a single workorder. These handling and processing times are based on the workorder and toolset characteristics, the quality inspection result of the first milled item, and the number of failed tools. These characteristics and occurrences can be determined by setting deterministic simulation options, or through the exported data afterwards.

By calculating the progression durations for itembatches and comparing them with the exported simulation data, the functionalities of the workcell and the total processing time, or lead time, for each itembatch and its workorder are determined and verified for most of the workcell functionalities. The itembatch progression durations are focused upon because the associated itembatch for each workorder tracks the progression states and duration of the milling process of the batch of items for the workorder. These durations are dependent on the handling and progression times of the associated items and tools from the linked toolset as well.

The calculated progression durations and lead time of an itembatch can be considered the minimum processing times of a workorder, because they do not take waiting times until the workcell resources are available due to other workorders into account. The minimum progression durations are determined by considering the handling times for items, tools and pallets by the workcell resources as constant, and the processing times as deterministic per workorder. The processing times are the item clamping, unclamping and milling time. As the workcell resources can only handle or process one agent simultaneously, the minimum progression durations are also dependent on the batch size of the workorder and the reserved toolset size.

Furthermore, the progression durations depend on the clamping location for the items of each workorder. Additional functionalities of the model effect the durations as well, including failing the quality inspection of the first item, the toolset arriving at the workcell after the itembatch, the toolset already available in the cell, removing other toolsets from the cell before the toolset can be stored, tool failure during milling, and failed tools that cannot be immediately replaced.

Starting with the milling and clamping location, the workcell functionalities verification and determined minimum itembatch lead times are described next. An overview of the formulas used to calculate the minimum progression durations for itembatches for most functionalities can be found in Appendix C.

4.2.1 Clamping and Milling Location

The most basic progression route through the model for an itembatch in the workcell depends on the clamping and milling location. As both milling machines are identical and have identical handling times for exchanging and storing tools and items, the itembatch lead times should be exactly the same for each machine if all other variables are equal. This was confirmed for the model during the verification process, in addition to confirming the correct functionality of each machine when milling items from a single workorder.

The clamping location does affect the itembatch lead times. Items clamped and unclamped by the operator, or outside the cell, consists of a two part process, first clamping and then mounting on pallet, and use the pallet exchange to enter and exit the cell. Items clamped by the clamping machine inside the cell us the item exchange instead. An overview of the itembatch lead times for workorders with various batch sizes and toolset sizes can be found in Table 4.1.

Batch size	**Itembatch lead times (hours)**					
Clamping location:	**Outside the cell**			**Inside the cell**		
Toolset size:	**1**	**5**	**50**	**1**	**5**	**50**
1	5.82	5.91	6.91	5.32	5.41	6.41
2	7.45	7.59	9.15	6.43	6.58	8.14
3	9.06	9.20	10.76	7.55	7.69	9.25
4	10.64	10.78	12.34	8.66	8.80	10.36
10	22.11	20.25	21.81	15.32	15.47	17.03
25	43.77	43.91	45.48	31.99	32.13	33.69

Table 4.1: Minimum itembatch lead times for workorders with various batch and linked toolset sizes, and both clamping locations.

The lead times shown in Table 4.1 are retrieved from the export data from simulation runs and compared with the calculated lead times. A few inconsistencies where discovered and investigated during the verification process. The itembatch lead time for items clamped inside the cell is five seconds longer when unclamping the first item after milling, and unclamping the remaining items after milling as well, if the toolset size is greater than three. Upon investigation it is determined that this is caused by the robot being unavailable due to retrieving a tool from the milling machine.

For items clamped outside the cell the itembatch lead time is two minutes longer if the batch size is greater than two. It is determined that this is caused by the operator loading the pallet of the second unclamped item part of the remaining items milled after a passed inspection before moving the combined itembatch to the general storage area outside the cell for removal from the workcell model.

Besides these inconsistencies, the itembatch lead times and progression durations from the simulation match the calculated times. This confirms the functionalities and expected behavior of the model in processing a single workorder for both clamping locations.

4.2.2 Quality Inspection Fail

When the first item of a batch fails the quality inspection, the item is discarded, the operator performs a quality assessment, and the last item of the remaining batch of items is assigned as

the new first item of the workorder. This item can never fail quality inspection. For workorders with a batch size greater than one this effectively means that the itembatch lead time is the time present in the workcell until the quality assessment is completed plus the progress durations of a workorder from milling the first item until removal from the workcell for a workorder with an itembatch size reduced by one compared to its starting size.

In Table 4.2 the difference between the itembatch lead times for otherwise identical workorders that passed the first quality inspection, shown in Table 4.1, and workorders that failed the inspection. When comparing the simulated lead times of itembatches with a failed quality inspection with the calculated times similar inconsistencies where found for workorders that passed inspection. For workorders with items clamped inside, an extra five second duration was found when unclamping the second item before its quality inspection if the toolset size is greater than three. For workorders with items clamped outside, the itembatch lead time is two seconds longer if the original batch size is greater than four.

Batch size	**Itembatch lead times (hours)**					
Clamping location:	**Outside the cell**			**Inside the cell**		
Toolset size:	**1**	**5**	**50**	**1**	**5**	**50**
1	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01
2	8.17	8.17	8.17	8.17	8.17	8.17
3	8.18	8.23	8.80	8.18	8.23	8.79
4	8.22	8.27	8.83	8.18	8.23	8.79
10	8.22	8.27	8.83	8.18	8.23	8.79
25	8.22	8.27	8.83	8.18	8.23	8.79

Table 4.2: Difference between itembatch lead times for workorders that passed and failed the first quality inspection.

Besides these inconsistencies, the simulated item batch lead times and lead time differences between workorders that pass or fail the first quality inspection match the calculated expected lead time, confirming the behavior of the workcell model and the quality inspection process in particular as expected and intended.

4.2.3 Itembatch and Toolset Arrival Delay

For each workorder the itembatch and linked toolset can arrive at the workcell after a delay. Since the lead time of a workorder is equal to its itembatch lead time plus the itembatch arrival delay the itembatch arrival delay alone does not affect the itembatch lead time itself. However, the toolset arrival delay can increase the itembatch lead time if it is greater than the itembatch arrival delay plus the duration of clamping and storing the itembatch items in the cell.

The Duration of clamping and storing the items of a single itembatch depends on the batch size and clamping location. Table 4.3 contains this duration for several batch sizes and both clamping locations. If the toolset arrival delay for a particular workorder is greater than the itembatch arrival delay and the duration until stored into the cell, the itembatch lead time is increased by the toolset arrival delay minus the itembatch arrival delay and duration combined.

During the verification process it was confirmed that the itembatch and toolset arrival delay are applied correctly since the resulting increased itembatch lead times from simulations match the predetermined lead time increases based on the difference between the itembatch delay plus the duration until stored and toolset delay.

Batch size	**Itembatch duration until stored in cell (hours)**	
Clamping location:	**Outside the cell**	**Inside the cell**
1	0.34	0.06
2	0.67	0.11
3	1.00	0.16
4	1.33	0.21
10	3.31	0.52
25	8.27	1.29

Table 4.3: Time between the itembatch arrival at the workcell and its items clamped and stored in the cell inventory for several batch sizes and clamping location.

4.2.4 Linked Toolsets and Item Types

Upon arrival at the workcell, workorders check if a toolset with a matching PL number is available in the workcell before initiating the creation of a new toolset. If an available toolset is reserved, the itembatch lead time is reduced since the toolset is already stored in the cell inventory. Alternatively, if the cell tool inventory has to remove one or more toolsets before the new toolset for a workorder can be stored, the itembatch lead time is increased for each toolset that is removed. Table 4.4 contains the calculated lead time reduction and increase for several toolset sizes.

Toolset size	**Toolset already stored in cell duration reduction (hours)**	**Toolset removal from cell duration (hours)**
1	-0.02	0.02
5	-0.06	0.07
10	-0.11	0.13
50	-0.49	0.63

Table 4.4: Itembatch lead time reduction if toolset is already stored in cell and lead time increase per toolset removed from cell before loading linked toolset for several toolset sizes.

By running a simulation where only one workorder is processed simultaneously and each workorder requires a toolset with the same PL number, the resulting itembatch lead time reduction matches the determined lead time reduction if the linked toolset was already available in the cell. Similarly, by running a simulation with a low tool inventory capacity, the resulting itembatch lead time increase matches the determined increase if a toolset is removed from the cell inventory before loading the linked toolset.

If an available toolset in the cell for a workorder can only mill part of the items the workorder is split. The current workorder batch size is adjusted to the number of items the available toolset can mill, and a new workorder is created with a batch size equal to the remaining items. The arrival delay for the itembatch and new toolset for the new workorder are increased by 24 hours.

The itembatch lead time for the original adjusted workorder should be equal to an itembatch lead time with identical characteristics, but reduced by the toolset already stored duration. The itembatch lead time for the added workorder should be identical equal to an itembatch lead time with identical characteristics as well. The new workorder lead time should be increased by 24 hours compared to an otherwise identical workorder due to the added itembatch and toolset

delay.

By simulating single workorders consecutively with identical PL numbers and toolsets with a life time slightly higher than the workorder batch size, both the split workorder functionality and the workorder and itembatch lead times where verified for various batch sizes and toolset sizes.

4.2.5 Tool Failure and Replacement

During milling, each standard tool of the milling machine and the required tools from the special toolset have a chance to fail based on the tool failure chance set in the simulation options. If the operator is immediately available, the time to replace a failed tool is determined as 0.44 hours, or 26 minutes and 20 seconds. Replacing standard tools in between milling after they have milled their maximum allowed number of items takes the same amount of time. During the verification process the correct functionality of the tool failure and standard tool replacement process and duration where confirmed.

Failed special tools can have a chance not to be replaced immediately as well based on the tool repair chance set via the simulation options. When a failed tool cannot be replaced immediately, the production is aborted and the item stored back in the cell inventory. If the item is the first item of a batch, or the second first item after a failed quality inspection, the itembatch has to wait until the tool is replaced to start the first item milling process again for the aborted item.

The delayed replacement tool is available at the workcell exactly 24 hours after determining it could not be immediately replaced. The itembatch lead time is increased by 24 hours plus the duration of loading the toolset and item into the milling machine and the milling time until aborted.

While the exact itembatch lead time increase due to the first item milling process being aborted could be determined for at least workorders with a batch size and toolset size of one, simulation results were analyzed to confirm the correct functionality of delayed tool replacements instead.

Similarly, simulation results were analyzed to verify the correct functionality of splitting a workorder if a broken tool cannot be immediately replaced for items milled after the first item has passed quality inspection. In this case, the original workorder and itembatch are adjusted to only contain successfully milled items, with a new workorder and itembatch added to the model containing the unmilled items.

The original itembatch lead time is equal to the lead time up until the moment the failed tool cannot be replaced plus the time it takes to remove and unclamp any milled items stored in the cell, if any, and combining the items into a batch before removal from the model. The new itembatch lead time is equal to the 24 hour replacement delay plus the lead time of an identical itembatch starting at milling the first item until removal from the model.

Besides the functions above, the effects of other simulation options, including the operator shift schedules and the item and tool storage capacity in the cell where verified for single workorders as well. The minimum item and tool storage capacity for single workorders where determined as the maximum batch size and toolset size as well. In the next section, the verification process of the model functionalities and workcell performance for processing multiple workorders are described.

4.3 Model Functionalities and Performance for Multiple Workorders

After verifying the correct application of the simulation options and the functionalities of the workcell model when processing single workorders, the model functionalities when processing multiple orders simultaneously where verified. This process consisted of running numerous simulation runs with an increasing number of workorders being processed by the workcell to determine if the workcell functionalities continue working with predicted behavior through observations during simulation and analysis of exported simulation data.

Starting with only the base functionalities required to process workorders as the foundation, the additional functionalities already verified for single workorders where tested individually first and in combination with each other afterwards. Discovered limitations or other problems with the model functionalities present in the current model are reported this section.

In addition to verifying continued functionality, the average workorder throughput, lead time, work in progress, and the item throughput are included in this section to show the workcell performance for the individual functionalities. Based on the simulation settings also used as the basis for the workcell experiments in the next chapter, the performance for two cases are shown and described for the verified functionalities.

Since the verification process mostly used the fixed number of workorders in the workcell option to determine the workorder arrivals during simulation, the simulation results for both cases are based on workorders arriving at the workcell to ensure 10 workorders are simultaneously processed in the workcell. In the first case, workorders have a deterministic batch size and toolset size of 10, and an item milling time of 1 hour. In the second case, workorders have a randomly assigned batch size between 1 and 25, and a toolset size between 1 and 50 according to a discrete uniform distribution. The item milling time is determined per workorder according to a normal distribution with a mean of 1 hour, a standard deviation of 1, a minimum value of 0.01 hour, and a maximum value of 4 hours. The clamping times, toolset and standard tool life time, itembatch and toolset arrival delay are all constant.

Due to the variability and randomness of almost all simulation runs for both cases, 100 runs are performed per unique simulation setting to determine the average performance and the 95 percent, or the double standard deviation 2σ, range of performance based on these 100 runs.

4.3.1 Milling and Clamping Location

Without any additional functionalities active, items of each workorder are either clamped and unclamped outside the cell or inside the cell, and milled on the first or second milling machine. By default, a workorder has a 50 percent chance to be clamped outside the cell, and a 50 percent chance to be milled by the first milling machine. These leads to up to four different progression routes throughout the model for workorders present at the workcell simultaneously. During verification, the continued functionality for multiple workorders being processed with both clamping methods and both machines was confirmed. The measured workorder performance for both cases outlined earlier is used as a base for comparison throughout this section as well.

While the workcell performance for single workorders is identical per machine regardless of the fraction of workorders milled by the first machine, the performance is affected by this fraction when multiple workorders are processed simultaneously. A significant increase in performance is

expected the more the workorders are processed by both machines in equal proportion, with the lowest and identical performance for workorders processed exclusively by either the first or the second machine.

Table 4.5 shows the average workorder performance between workorders only milled on machine 1, only on machine 2, or on both machines with a 50 percent chance for either per workorder for both the deterministic and variable workorders cases. As expected, the performance significantly increases when workorders have a 50 percent chance for processing on either machine compared to one machine exclusively for both cases. For the Deterministic workorder case, the workorder and item throughput is increased by approximately 65 percent, and the workorder lead time is decreased by around 60 percent. For the variable workorder case, the throughput is increased by approximately 54 percent, and the lead time is reduced by around 66 percent. The performance for both cases on either the first or second machine is almost identical as well, only affected by the variability in clamping location for the deterministic workorder case, and the variability of the workorder characteristics as well in the variable workorder case.

Milling Location	**WO throughput (per hour ($\pm 2\sigma$))**	**WO lead time (hour ($\pm 2\sigma$))**	**WO in system (# ($\pm 2\sigma$))**	**Item throughput (per hour ($\pm 2\sigma$))**
Deterministic WO:				
Both	0.139 ±0.004	71.14 ±1.918	10.00 ±0.000	1.394 ±0.038
Machine 1	0.084 ±0.000	117.9 ±0.024	10.00 ±0.000	0.838 ±0.000
Machine 2	0.084 ±0.000	117.9 ±0.026	10.00 ±0.000	0.838 ±0.000
Variable WO:				
Both	0.077 ±0.008	128.7 ±13.01	10.00 ±0.000	0.998 ±0.074
Machine 1	0.050 ±0.006	195.0 ±21.70	10.00 ±0.000	0.650 ±0.052
Machine 2	0.051 ±0.006	193.8 ±21.15	10.00 ±0.000	0.653 ±0.047

Table 4.5: Average workorder performance for items milled on milling machine 1, milling machine 2 or both.

Similar to the fraction of workorders milled on each milling machine, the workcell performance is effected by the fraction of workorders clamped outside or inside the cell as well. This performance difference in favor of clamping inside the cell is expected as the minimum itembatch lead time for workorders clamped outside is longer than workorders clamped inside.

Table 4.6 shows the average workorder performance between workorders only clamped inside the cell, only clamped outside the cell, or in both locations with a 50 percent chance for either per workorder for both the deterministic and variable workorders cases. As expected, the average workorder and item throughput are slightly higher, and the average workorder lead time shorter in both cases for workorders exclusively clamped inside the cell compared to workorders clamped exclusively outside the cell, and the performance for workorders with an equal chance for both clamping methods in between. The average base performance with both clamping methods is closer to the inside clamping performance. This is expected due to the clamping machine and operator being able to clamp and unclamp items simultaneously.

4.3.2 Quality Inspection Result

After verifying the basic workcell functionality, the first item quality inspection failure functionality was successfully verified for multiple workorders. However, it should be noted that in

Clamping location	**WO throughput (per hour ($\pm 2\sigma$))**	**WO lead time (hour ($\pm 2\sigma$))**	**WO in system (# ($\pm 2\sigma$))**	**Item throughput (per hour ($\pm 2\sigma$))**
Deterministic WO:				
Both	0.139 ±0.004	71.14 ±1.918	10.00 ±0.000	1.394 ±0.038
Inside	0.144 ±0.005	68.74 ±2.236	10.00 ±0.000	1.443 ±0.047
Outside	0.123 ±0.002	80.56 ±1.311	10.00 ±0.000	1.229 ±0.021
Variable WO:				
Both	0.077 ±0.008	128.7 ±13.01	10.00 ±0.000	0.998 ±0.074
Inside	0.081 ±0.008	121.9 ±11.53	10.00 ±0.000	1.051 ±0.083
Outside	0.068 ±0.007	144.0 ±13.89	10.00 ±0.000	0.888 ±0.050

Table 4.6: Average workorder performance for items clamped inside the cell, outside the cell, or both.

the simulation performance calculations no distinction is made between discarded items, and workorders if containing only one item, and successfully milled items and completed workorders.

Due to an assessment and a second quality inspection for workorders with the first item failing inspection, a decrease in performance is expected when the quality inspection pass chance decreases. Table 4.7 shows the performance for the deterministic and variable workorder cases for simulation runs with a quality inspection pass chance of 100, 0 and 95 percent chance.

As shown in Table 4.7, in both cases the workorder and item throughput is slightly lower, and the workorder lead time slightly longer, when items have a 5 percent chance to fail quality inspection compared to none. As expected, the performance is significantly lower in case all first items fail inspection.

Quality inspection pass chance	**WO throughput (per hour ($\pm 2\sigma$))**	**WO lead time (hour ($\pm 2\sigma$))**	**WO in system (# ($\pm 2\sigma$))**	**Item throughput (per hour ($\pm 2\sigma$))**
Deterministic WO:				
100%	0.139 ±0.004	71.14 ±1.918	10.00 ±0.000	1.394 ±0.038
0%	0.104 ±0.002	94.70 ±1.964	10.00 ±0.000	1.045 ±0.022
95%	0.138 ±0.003	71.97 ±1.787	10.00 ±0.000	1.378 ±0.034
Variable WO:				
100%	0.077 ±0.008	128.7 ±13.01	10.00 ±0.000	0.998 ±0.074
0%	0.065 ±0.005	151.6 ±12.01	10.00 ±0.000	0.845 ±0.052
95%	0.076 ±0.008	129.8 ±13.24	10.00 ±0.000	0.988 ±0.076

Table 4.7: Average workorder performance for first item passing quality inspection with a 100, 95, or 0 percent chance.

4.3.3 Tool Failure and Replacement

The tool failure during milling and delayed replacement functionalities where both verified to continue working under all tested simulation settings as well. However, it should be noted that in the simulation performance data calculations workorders split into suborders count as

normal workorders, affecting the calculated average number of workorders and itembatches in the workcell environment and cell itself. In particular, each workorder split is expected to increase the average workorder and itembatch throughput, increase the work in progress level, and decrease the lead time, due to adding an additional workorder and altering the existing workorders batch size.

Each tool failure and delayed tool replacement is expected to decrease the workcell performance. In Table 4.8 the workcell performance for both cases for several scenarios can be found. First, a 100 percent tool failure chance leads to a very significant decrease in performance as expected. The average workorder and item throughput are decreased by over 96 percent in both cases, and the workorder lead time is increased by almost 2000 percent for the deterministic workorder case and over 1500 percent for the variable workorder case.

Tool failure chance (tool replacement chance)	**WO throughput (per hour ($\pm 2\sigma$))**	**WO lead time (hour ($\pm 2\sigma$))**	**WO in system (# ($\pm 2\sigma$))**	**Item throughput (per hour ($\pm 2\sigma$))**
Deterministic WO:				
0% (100%)	0.139 ±0.004	71.14 ±1.918	10.00 ±0.000	1.394 ±0.038
100% (100%)	0.005 ±0.000	1474 ±64.79	10.00 ±0.000	0.053 ±0.002
100% (99%)	0.009 ±0.001	998.1 ±153.6	10.01 ±0.007	0.048 ±0.003
0.1% (95%)	0.138 ±0.004	71.88 ±1.799	10.00 ±0.001	1.373 ±0.036
Variable WO:				
0% (100%)	0.077 ±0.008	128.7 ±13.01	10.00 ±0.000	0.998 ±0.074
100% (100%)	0.003 ±0.001	2114 ±497.5	10.00 ±0.000	0.036 ±0.008
100% (99%)	0.008 ±0.002	991.0 ±233.4	10.01 ±0.011	0.032 ±0.008
0.1% (95%)	0.076 ±0.009	129.1 ±14.68	10.00 ±0.002	0.978 ±0.085

Table 4.8: Average workorder performance for a tool failure chance of 100 or 0 percent per tool, and a special tool immediate replacement chance of 99 percent with a 100 percent tool failure chance or a chance of 95 percent with a 0.1 percent tool failure chance.

Interestingly, increasing the tool replacement delay chance to 1 percent for each failed special tool increases the workorder throughput, and decreases the workorder lead time as well. This increase in workorder performance is explained by counting each split workorder as two unique separate workorders as mentioned earlier. The average workorder wip level is also very slightly increased as well, as each split workorder increases the workorders in the workcell by one above the fixed number of 10 workorders set as the workorder arrival method. As expected, the item throughput does further decrease after increasing the tool replacement delay chance by one percent.

Compared to no tool failures and delayed replacements, a tool failure chance of 0.1 percent with a tool replacement delay of 5 percent lowers the workorder performance insignificantly, but does lower the item throughput more significantly. It is assumed that for these settings the increase in the number of workorders due to the workorder splits almost completely offsets the expected workorder performance decrease due to the added small chance for tool failure and delayed replacement.

4.3.4 Linked Toolsets and Cell Capacity

Running simulations for multiple workorders with a limited amount of item type and toolset numbers, and additionally linking each item type number to a unique toolset number is verified as functioning. For unique item type and toolset numbers per workorder, the minimum item and tool inventory capacity in the cell where determined as the maximum batch size and toolset size for multiple workorder simulation runs.

However, during the verification process it was discovered that simulations with a limited pool of item type and toolset numbers in combination with a low item and tool storage capacity in the cell can cause a deadlock. This deadlock is caused when the special toolsets reserved for workorders with its items already stored in the cell cannot enter the cell because all tools stored in the cell belong to toolsets reserved by workorders for which the items are not stored in the cell yet. In the current model no functionality or code is available to temporarily remove reserved toolsets from the workcell for workorders not stored in the cell yet to free storage space for the toolsets required for milling the workorders already stored in the cell. Through further testing the minimum item and tool storage capacity in case toolsets stored in the workcell can be reserved by new workorders where determined as at least three times the maximum batch size and toolset size to prevent deadlock during simulation.

Similar to the delayed tool failure replacement workorders can be split into two suborders if available toolsets in the workcell can only partially mill the batch of items, with similar additional effects on the workorder and itembatch performance. The measured performance results comparing workorders with unique item type and toolset numbers with a pool of 1 linked set of numbers and a pool of 25 linked sets of numbers for both cases are shown in Table 4.9.

Item and toolset type connection	**WO throughput (per hour ($\pm 2\sigma$))**	**WO lead time (hour ($\pm 2\sigma$))**	**WO in system (# ($\pm 2\sigma$))**	**Item throughput (per hour ($\pm 2\sigma$))**
Deterministic WO:				
Unique	0.139 ±0.004	71.14 ±1.918	10.00 ±0.000	1.394 ±0.038
1 linked set	0.144 ±0.005	68.84 ±2.123	10.00 ±0.001	1.440 ±0.046
25 linked sets	0.144 ±0.004	69.07 ±1.996	10.00 ±0.000	1.435 ±0.042
Variable WO:				
Unique	0.077 ±0.008	128.7 ±13.01	10.00 ±0.000	0.998 ±0.074
1 linked set	0.097 ±0.011	104.3 ±11.42	10.22 ±0.044	1.023 ±0.089
25 linked sets	0.078 ±0.008	126.5 ±12.79	10.01 ±0.012	1.004 ±0.080

Table 4.9: Average workorder performance for simulation with unique item types and toolset types, 1 linked item type and toolset type, or 25 linked item types and toolset types.

In the deterministic case, the workorder and item throughput are slightly increased when using a set of linked item type and toolset numbers compared to unique numbers per workorder. The difference between one and 25 sets of numbers seems insignificant. For the case with variable workorders the difference in performance between 25 sets of linked numbers and unique numbers is insignificant, indicating it makes little to no measurable difference. The increase in performance for 1 set of unique numbers compared to the other two scenarios does seem significant, even for the item throughput. As indicated by the increased workorder wip level many workorders are split at the start to reserve toolsets for milling part of the batch of items. This is expected as the batch size and toolset size are variable between workorders and any free toolset can be

reserved for milling for each new workorder in case of 1 set of numbers. It is unclear however how much the increased number of workorders effect the measured performance exactly.

Table 4.10 shows the workorder and item performance for both cases to compare the performance between the default item and tool storage capacity of the cell with the minimum capacity and three times the minimum capacity based on the maximum batch size and toolset size for each case. Compared to the default capacity, the measured performance for simulations with deterministic workorders is decreased by over 60 percent with the minimum capacity and around 25 percent with three times the minimum capacity. For the variable workorders the difference in performance between the default capacity and three times the minimum capacity seems negligible, while the performance is decreased by around 50 percent for the minimum capacity. This indicates the item capacity might significantly impact performance, as the item capacity is identical for the default and three times minimum capacity scenarios. The influence of the cell item and tool inventory capacity is further explored in the next chapter.

Item capacity, tool capacity	**WO throughput (per hour ($\pm 2\sigma$))**	**WO lead time (hour ($\pm 2\sigma$))**	**WO in system (# ($\pm 2\sigma$))**	**Item throughput (per hour ($\pm 2\sigma$))**
Deterministic WO:				
75, 400	0.139 ±0.004	71.14 ±1.918	10.00 ±0.000	1.394 ±0.038
10, 10	0.052 ±0.001	189.7 ±2.840	10.00 ±0.001	0.515 ±0.008
30, 30	0.105 ±0.003	94.59 ±2.522	10.00 ±0.000	1.045 ±0.028
Variable WO:				
75, 400	0.077 ±0.008	128.7 ±13.01	10.00 ±0.000	0.998 ±0.074
25, 50	0.039 ±0.004	252.0 ±27.17	10.00 ±0.000	0.501 ±0.033
75, 150	0.076 ±0.007	129.8 ±11.18	10.00 ±0.000	0.989 ±0.075

Table 4.10: Average workorder performance for simulation with default item and cell storage capacity, minimum capacity, or 3 times minimum capacity.

4.3.5 Operator Shift

The final workcell functionality that was successfully verified as working as intended are the operator shift schedules. The schedules influence on the performance for both cases are shown in Table 4.11. In this table the workcell performance for operators being always working at the workcell, or on shift, with a 17 hour day schedule and a 17 hour day schedule with Sunday free. While the performance decreases for both cases between the three scenarios based on the decreasing hours the operator is available per week, the performance decrease is larger for the variable workorder case.

4.3.6 General Performance

With all the workcell functionalities verified and their individual impact on performance shown for the two workorder cases, the base functionality performance is compared with the combined performance when including a quality inspection failure chance of 5 percent, a tool failure chance of 0.1 percent, and a delayed tool replacement chance of 5 percent. This setup of workcell functionalities is also used in the experiments described in the next chapter.

Operator schedule	**WO throughput (per hour ($\pm 2\sigma$))**	**WO lead time (hour ($\pm 2\sigma$))**	**WO in system (# ($\pm 2\sigma$))**	**Item throughput (per hour ($\pm 2\sigma$))**
Deterministic WO:				
27/7	0.139 ±0.004	71.14 ±1.918	10.00 ±0.000	1.394 ±0.038
17 hour day	0.132 ±0.003	74.76 ±1.681	10.00 ±0.001	1.325 ±0.030
27 hour day, Sunday free	0.117 ±0.003	84.17 ±2.467	10.00 ±0.000	1.170 ±0.035
Variable WO:				
24/7	0.077 ±0.008	128.7 ±13.01	10.00 ±0.000	0.998 ±0.074
27 hour day	0.054 ±0.007	183.0 ±21.60	10.00 ±0.000	0.694 ±0.071
27 hour day, Sunday free	0.050 ±0.006	192.8 ±23.65	10.00 ±0.000	0.654 ±0.059

Table 4.11: Average workorder performance for simulation with the operator always on shift, 17 hour per day shift, or 17 hour per day shift and Sunday free.

As shown in Table 4.12 a very small decrease in the workorder throughput and similar increase in the workorder lead time is measured compared to no quality inspection and tool failure for both cases. A slightly larger decrease for the item throughput is measured for both cases as well, indicating the workorder performance is similarly effected by workorder splits caused by delayed tool replacements compared to the workorder performance with only a chance for tool failures and delayed replacement described earlier.

Additionally, the workcell performance is measured for a simulation including a scenario with all functionalities active together. In addition to the quality inspection, tool failure and delayed replacement chance, this scenario includes 25 sets of linked item and toolset type numbers, the 17 hour day shift with Sunday free for the operator, and an item and tool capacity set to three times the minimum capacity for each case. The measured workorder and item performance are included in Table 4.12 as well.

Functionalities	**WO throughput (per hour ($\pm 2\sigma$))**	**WO lead time (hour ($\pm 2\sigma$))**	**WO in system (# ($\pm 2\sigma$))**	**Item throughput (per hour ($\pm 2\sigma$))**
Deterministic WO:				
Base	0.139 ±0.004	71.14 ±1.918	10.00 ±0.000	1.394 ±0.038
QC and tool failure	0.136 ±0.003	72.89 ±1.844	10.00 ±0.001	1.354 ±0.034
All	0.071 ±0.003	137.5 ±5.569	10.00 ±0.001	0.709 ±0.029
Variable WO:				
Base	0.077 ±0.008	128.7 ±13.01	10.00 ±0.000	0.998 ±0.074
QC and tool failure	0.076 ±0.007	129.7 ±12.18	10.00 ±0.002	0.970 ±0.066
All	0.044 ±0.005	221.5 ±27.34	10.00 ±0.008	0.557 ±0.050

Table 4.12: Average workorder performance for simulation with no additional functionalities, quality inspection failure and tool failure and delayed replacement, or all functionalities.

For both cases, the performance is significantly lower compared to the base and previous scenario. This is an expected results as both the cell inventory capacity and the operator shift showed a decreased performance greater than the increased performance for 25 sets of linked item and toolset type numbers.

Chapter 5

Experimentation

After the verification process and the shown impact on the workcell performance from various workcell functionalities for two cases, this chapter further explores the influence on the workcell performance through simulation experiments. Before describing and showing the three simulation experiment scenarios and the results, the basic simulation options settings that form the foundation of the experiments are described.

As introduced in Chapter 3, the simulation options can be divided into four groups, the general simulation settings, the workorder arrival method settings, the workorder and toolset type settings, and the custom workorder and toolset characteristics settings. The workorder arrival method settings are set per experiment scenario and will be described in each experiment section. The workorder and toolset type settings are set to random, using the custom workorder toolset characteristics settings to determine the characteristics for each workorder and toolset during the experiment simulations. Each workorder has a unique item type and toolset number by default.

The general simulation settings are mostly constant throughout the experiments. Each workorder has a 50 percent chance of being processed by the first milling machine, and a 50 percent chance of being clamped and unclamped outside. The quality inspection pass chance is 95 percent, the tool failure chance 0.1 percent, and the delayed replacement chance for a failed special tool is set to 5 percent. By default, the operator is always on shift, and the item and tool capacity of the cell inventory is set to the default values of 75 items and 400 tools in the first two experiment scenarios. No operator and robot priorities are used throughout the experiment simulations.

Most custom workorder and toolset characteristics settings are kept constant at the default settings of the workcell model. Each standard tool has a life time of milling 100 items, and each special toolset a life time of 50 items. The outside clamping and unclamping time is constant at 10 minutes per item, and the inside clamping and unclamping time is constant at 2 minutes per item. The itembatch arrival delay is constant at 1 hour after the workorder arrival, and the toolset delay is constant at 24 hours.

In the first and third experiment scenario, the two workorder characteristics cases also used for the performance overviews in the verification process in the previous chapter are used. The deterministic workorder case has a constant item milling time of 1 hour, and a constant batch size and special toolset size of 10. The variable workorder case has a item milling time set per workorder according to a normal distribution with a mean of 1 hour, a standard deviation of 1, a minimum value of 0.01 hour, and a maximum value of 4 hours. Each workorder batch size is set randomly according to a discrete uniform distribution between 1 and 25 items, and each

toolset size is set randomly according to a discrete uniform distribution between 1 and 50 tools. In the second experiment scenario the item milling time is constant for all workorders at 1 hour.

In all experiment scenarios the workcell model is simulated using the *Experiments* parameter variation experiment to execute multiple simulation runs based on setting a range of values for the parameters of the main agent *wCE_Outsidecell*, Similar to the *Verification* parameter variation experiment used throughout the verification process. By varying the random number seed of the simulation options multiple runs with identical simulation options are performed. In the experiments 100 runs per unique simulation setting are executed to determine the mean workcell performance and variation in the workcell performance for each unique simulation setting. Unless mentioned otherwise, the simulation time of each run is set to 26 weeks.

In the experiments, the workcell performance is primarily investigated through the average workorder and item throughput of the workcell, the average workorder lead time and WIP level, and the average processing utilization of both milling machines measured per simulation run. For each of these performance measurements the mean and coefficient of variation are determined from the results of the 100 runs per unique simulation setting. The coefficient of variation, also known as the relative standard deviation, is calculated by dividing the standard deviation by the mean and is expressed as an percentage throughout this chapter.

The first experiment scenario is focused on determining the maximum workcell performance for the deterministic and variable workorder cases for multiple workorder arrival methods. The variability of the workcell performance results between simulation runs and the general expected relations between the performance measurements are investigated as well. The second scenario revolves around investigating the influence of the workorder batch size and toolset size on the workcell performance with a constant deterministic item arrival rate at the workcell. The third scenario investigates the influence of the item and tool capacity of the cell inventory on the performance for both the deterministic and variable workorder cases with workorders arriving at the workcell to maintain a fixed number of workorders in the workcell system.

5.1 Workorder Arrival Method

The first workcell model experiment scenario is centered around determining the maximum workcell performance using the deterministic and variable workorder cases with the default general workcell simulation settings introduced earlier in this chapter for three workorder arrival methods. The maximum workcell performance is defined as the maximum average workorder and item throughput with a minimal average workorder lead time in a stable workcell system, Both the workorder and item throughput are considered since workorders can vary in batch size and suborders increase the amount of workorders processed by the workcell without increasing the number of items processed.

The workcell system is considered stable when the average workorder throughput is equal to the average workorder arrival rate over the duration of each simulation run. In practice the average workorder throughput will always be smaller than the average arrival rate due the remaining workorders in the workcell at the end of each simulation and no workorders are already being processed at the start of each simulation as well. When required, extra simulation runs at half the default simulation time per run are performed to confirm the workcell stability by comparing the average workorder wip level and lead time over the full and half simulation times.

Besides determining the maximum performance the first experiment results are also used to

investigate some general expected relations between the considered performance measurements. The average item throughput should be equal to the average workorder throughput times the average workorder batch size if corrected for suborders. Since suborders are counted as new workorders in the performance calculations the average item throughput is expected to be a slightly less, but proportionally constant as long as the fraction of suborders is constant as well.

The average machine utilization is expected to be proportional to the item throughput for both machines and the average utilization of both machines as well. While the fraction of milling time for the first machine could be different per simulation run due to the randomness of assigning a milling machine per workorders randomly, it is expected that the average milling machine processing utilization is equal over 100 otherwise identical simulation runs.

Finally, the amount of variability in the workcell performance results per run and between runs is expected to negatively impact performance for increasing variability. During these experiments and the following experiments the relative variability is calculated for the performance measurements to investigate if increasing variability results in lower performance. At the end of this section the causes of variability are discussed.

The three arrival methods are based on a fixed number of workorders in the system, an exponential arrival rate, and a deterministic arrival rate defined by the time between arrivals. Next, the experiment setup, the expected simulation results and the simulation results analysis are described for each arrival method.

5.1.1 Fixed Number of Workorders

The first arrival method for which the maximum workorder performance is determined is workorders arriving at the workcell to maintain a fixed number of workorders in the workcell system. By controlling the workorder arrival rate through the workorder departure rate, or throughput, at a constant workorder WIP level in the system, the workcell system will remain stable throughout each simulation run.

In order to find the maximum workcell performance for both the deterministic and variable workorder case, the workcell performance data is exported for each simulation run for simulations with an range of the fixed amount of workorders in the workcell. Using the Experiments parameter variation the fixed amount of workorders in the system values are set from 1 to 20 with a step of 1 for both cases. As mentioned before, for each fixed number of workorder value for each case 100 simulation runs are executed with a unique random number seed per run.

For both cases, the average workorder throughput is expected to increase with an increased amount of workorders in the system until reaching a maximum. For higher numbers of workorders in the system the workorder lead time is expected to increase further due to the relation between the average throughput, wip level and lead time. Since the average batch size is higher for the variable workorder case the maximum workorder throughput for this case is expected to be lower and reached at a lower number of fixed workorders in the workcell compared to the deterministic workorder case. Due to the expected increased variability and a higher average toolset size the item throughput and milling machine utilities are expected to be lower for the variable case as well.

Based on the simulation results for the deterministic workorder the maximum average workorder throughput is determined at 0.139 per hour with a coefficient of variation, or a relative variability, of 1.5 percent, at a number of 12 fixed workorders in the system. Figure 5.1 shows the mean of

the average workorder throughput with the area of values around the mean within one standard deviation over the range of the fixed number of workorders. The figure shows the workorder throughput remaining level for increasing numbers of fixed workorders in the system after 12.

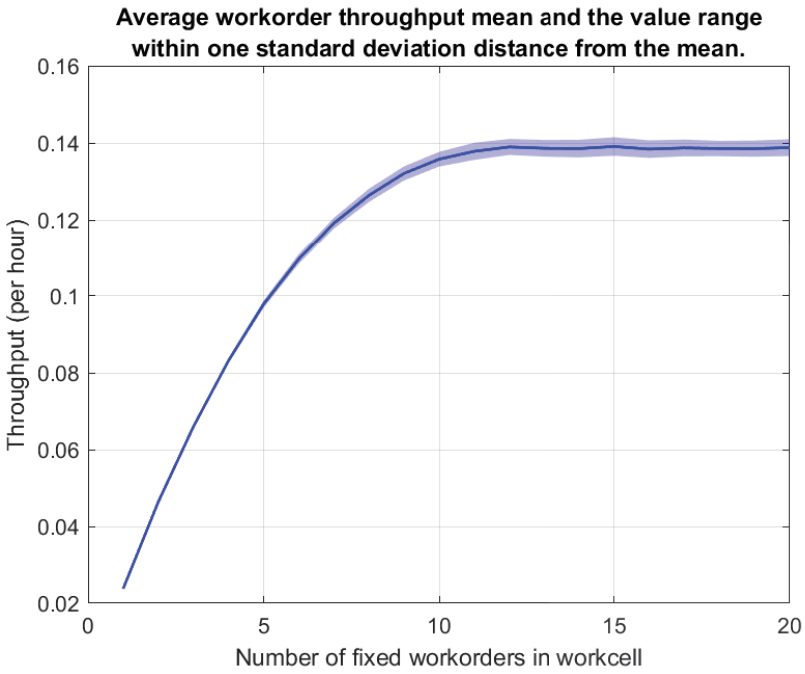

Figure 5.1: The average workorder throughput mean and range of values within one standard deviation from the mean over a range of fixed amount of workorders in cell for the deterministic workorder case.

As expected, the average workorder lead time increases linear with the increasing fixed amount of workorders in the system from 12. Over the range of fixed numbers of workorders in the system, the item throughput is approximately 9.96 times higher than the workorder throughput. Therefore it can be concluded that on average the relative amount of suborders remains level over the range of the fixed numbers of workorders in the workcell. This is expected behavior as both the batch size per workorder and the number of tools per special toolset remains equal throughout the simulation runs for the deterministic workorder case.

The simulation results also show the expected relation between the item throughput and the milling machine utilization for both machines over the complete range of fixed numbers of workorders in the system. Therefore, the maximum workcell performance is determined at a fixed number of workorders in the system of 12. Table 5.1 shows an overview of the measured performance at this fixed number for the deterministic workorder case.

Based on the variable workorder case simulation results the maximum average workorder throughput is determined at 0.076 per hour with a coefficient of variation of 4.4 percent at a number of 8 fixed workorders in the system. Similar to the deterministic case, Figure 5.1 shows the mean of the average workorder throughput with the area of values around the mean within one standard deviation over the range of the fixed number of workorders. The figure shows the workorder throughput remaining level for increasing numbers of fixed workorders in the system after 8.

As expected, the measured performance results show the same relations between each other as for the deterministic workorder case, but with a higher relative variability and lower performance.

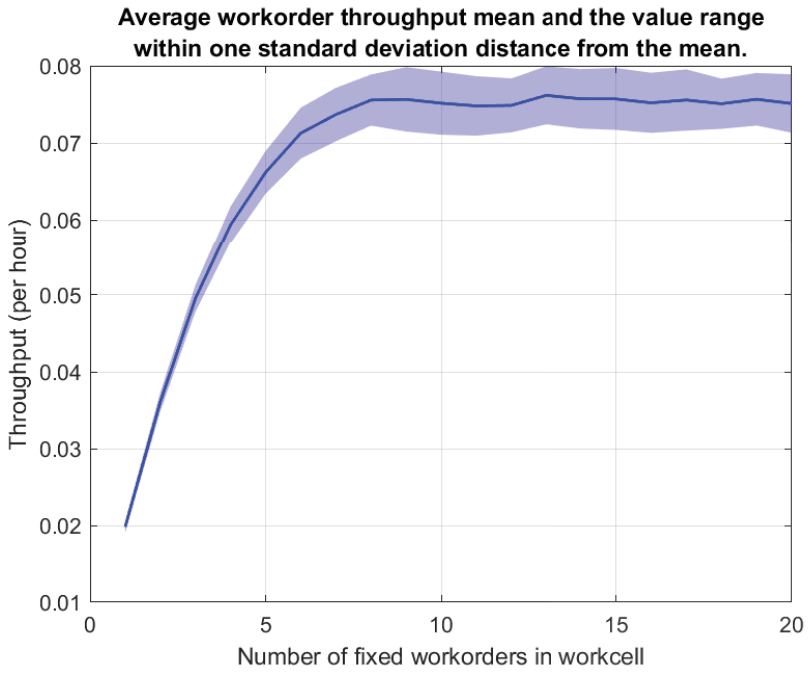

Figure 5.2: The average workorder throughput mean and range of values within one standard deviation from the mean over a range of fixed amount of workorders in cell for the variable workorder case.

The average item throughput level over the range of fixed number of workorders in the workcell at approximately 12.8 times the average workorder throughput. An overview of the measured workorder performance for this case at the fixed number of workorders in the system of 8 can also be found in Table 5.1.

Workorder case:	**Deterministic**		**Variable**	
Number of fixed WO's in workcell:	**12**		**8**	
Performance Measurements	**Mean**	**CV**	**Mean**	**CV**
WO Throughput (per hour)	0.139	1.492 %	0.076	4.386 %
WO lead time (hours)	85.48	1.504 %	104.7	4.300 %
WO WIP level	12.00	0.006 %	8.002	0.014 %
Item throughput (per hour)	1.384	1.476 %	0.964	3.344 %
Milling machine 1 utilization	0.689	4.271 %	0.623	7.382 %
Milling machine 2 utilization	0.699	3.677 %	0.611	8.054 %
Average milling machine utilization	0.694	1.482 %	0.617	2.760 %

Table 5.1: Overview of the mean and coefficient of variation of the performance measurements for both workorder cases at the optimum number of fixed workorders in the workcell.

5.1.2 Deterministic Workorder Arrivals

The second arrival method for which the maximum workorder performance is determined is workorders arriving at the workcell at deterministic times with a constant time interval between arrivals. For this arrival method the workorder throughput is not linked with the arrival

rate, meaning the maximum workorder performance is determined by the maximum workorder throughput for which the workcell system is stable.

For this simulation experiment the time interval between workorder arrivals is varied between a range of 6.1 and 10 hours with a step of 0.3 for the deterministic workorder case. Since the workorder arrival rate is the inverse of the time interval between arrivals this equivalent to a workorder arrival rate ranged between approximately 0.1 and 0.16 workorders per hour. For the varied workorder case the time between workorder arrivals is varied between 8 and 20 hours with a step of 1, equivalent to a workorder arrival rate ranged between 0.05 and 0.125 hours.

While the maximum workorder throughput could be similar to the maximum workorder throughput with the fixed number of workorders in the system arrival method for both cases, it is expected that the maximum throughput at a stable workorder system is lower. The relation between the workorder throughput, item throughput and both milling machine utilities is expected to be similar to the fixed number of workorders method due to using the same workorder characteristics cases.

The simulation performance results for the deterministic workorder case show a maximum workorder throughput of approximately 0.137 workorders per hour. The average workorder throughput over the range of simulated arrival rate, the inverse of the inter arrival time range, is shown in Figure 5.3. However, since at reaching the maximum throughput the workorder arrival rate is increasingly higher than the throughput, indicating the workcell system is not stable before reaching the maximum throughput.

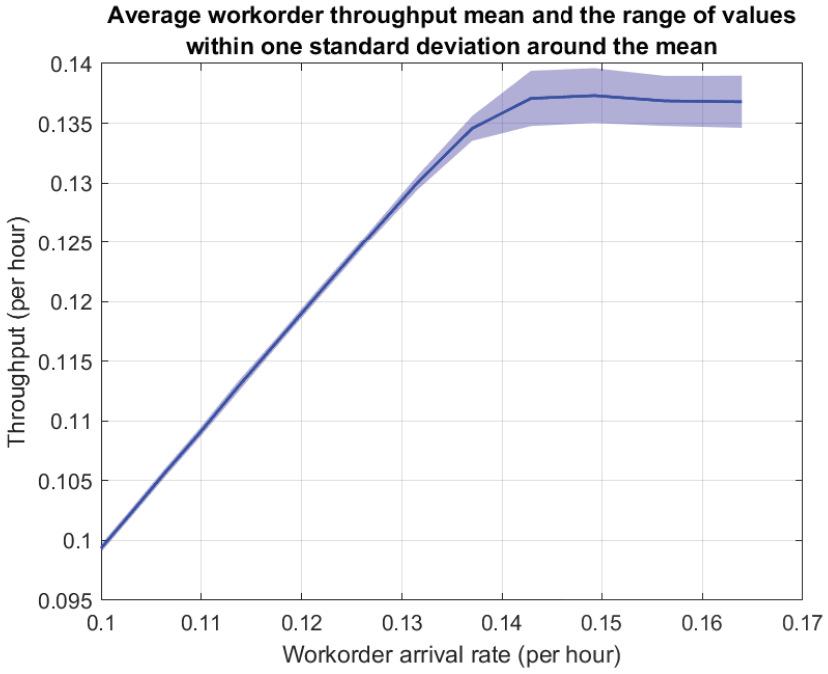

Figure 5.3: The average workorder throughput mean and range of values within one standard deviation from the mean over a range of deterministic workorder arrival rates for the deterministic workorder case.

As mentioned earlier, due to the manor of calculating the average throughput for each simulation run the measured average throughput is always lower than the measured average arrival rate. From Figure 5.3 it is hard to determine from which point the throughput starts increasingly dropping in value compared to the arrival rate. Determining the relative difference between the

measured average arrival rate and throughput rate could give a better insight in the maximum stable arrival rate. This relative difference over the range of measured deterministic arrival rate values is shown in Figure 5.4.

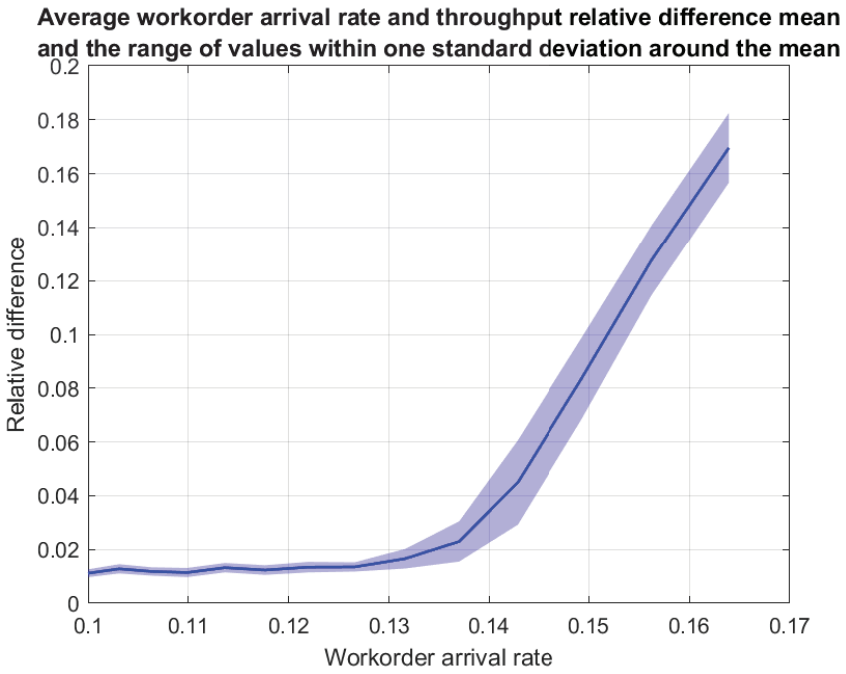

Figure 5.4: The measured average workorder arrival rate and throughput relative difference mean and range of values within one standard deviation from the mean over a range of deterministic workorder arrival rates for the deterministic workorder case.

As shown in Figure 5.4, the relative difference between the arrival rate and throughput starts to increase consistently around a deterministic arrival rate of approximately 0.1275. The closed simulated workorder inter arrival time is 7.9 hours. Therefore it is assumed that the system simulation is stable for inter arrival times of at least 7.9 hours. Running the same experiment simulations at half the simulation length per run further strengthen this assumption as the average workorder WIP and lead time start to diverge from each other for inter arrival times lower than 7.9 hours.

At the inter arrival time of 7.9 hours the mean average workorder throughput over 100 runs is 0.125 hours, the mean average workorder lead time is 64.37 hours, and mean average WIP level is 8.13. Table 5.2 provides an overview of these and the other performance measurements and their relative variance for workorders arriving with the inter arrival time of 7.9 hours.

As expected, the relation between the average workorder and item throughput, as well as the relation between the average item throughput and the average milling machine utilities is similar compared to the relations for the fixed number of workorder arrival method simulation results. While the maximum average workorder throughput for both arrival methods for the deterministic case are similar, the average workorder throughput, and therefore the average item throughput and milling machine utilities as well, are lower for the deterministic workorder arrival method in a stable workorder system.

In the variable workorder case, the maximum average workorder throughput is estimated at approximately 0.075 per hour, as shown in Figure 5.5. This maximum workorder throughput is similar the measured throughput for the fixed number of workorders arrival method as well.

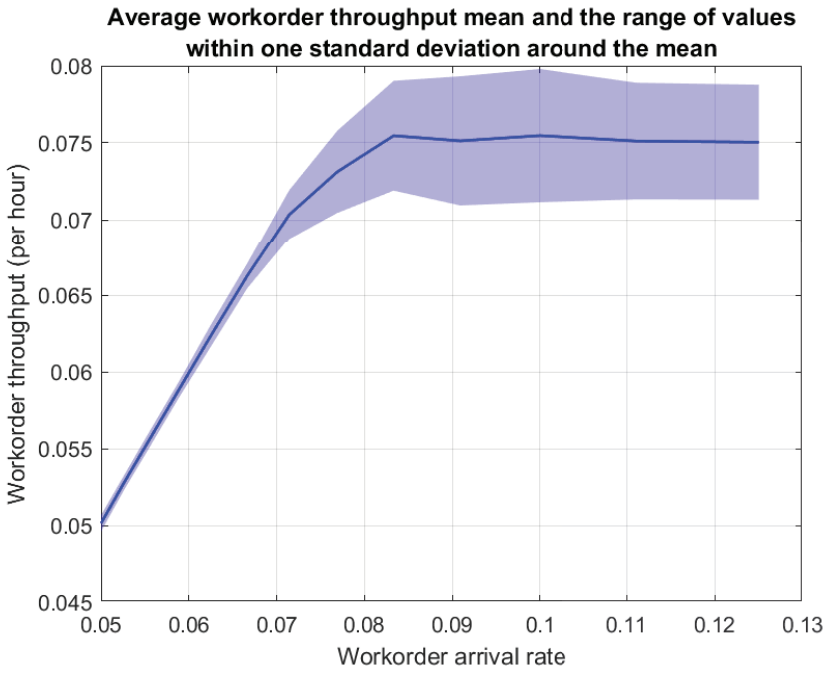

Figure 5.5: The average workorder throughput mean and range of values within one standard deviation from the mean over a range of deterministic workorder arrival rates for the variable workorder case.

However, the workcell system appears becoming unstable before reaching the maximum workorder throughput. In similar fashion to the deterministic workorder case, the minimum inter arrival time for which the system is stable is estimated by looking ant the relative difference between the measured average workorder arrival rate and throughput shown in Figure 5.6. From this relative difference it the maximum arrival rate is estimated at approximately 0.06 workorders per hour. The closed simulated inter arrival time corresponding with this arrival rate is 17 hours, equivalent to an arrival rate of approximately 0.059. Comparing the average workorder WIP level and lead time with results from simulations at half the duration further indicate that the system becomes unstable for inter arrival times lower than 17 hours.

At the workorder inter arrival time of 17 hours the mean average workorder throughput is 0.059 workorders per hour, the mean average workorder lead time is 71.58 hours, and mean average workorder WIP level is 4.25. Overview of all considered performance measurements and their relative variance can also be found in Table 5.2. As expected and in concert with the results for the deterministic workorder case, the relations between the workorder and item throughput and the item throughput and milling machine utilities are similar to the relations found for the previous workorder arrival method. The workcell performance is also lower at the minimum estimated stable inter arrival time compared to the maximum performance for the fixed number of workorders in the system arrival method.

5.1.3 Exponential Workorder Arrival Rate

The third arrival method for which the maximum workorder performance is determined is workorders arriving at the workcell according to an exponential rate. Due to the random nature of workorders arriving at the workcell according to an exponential arrival rate extra variability in

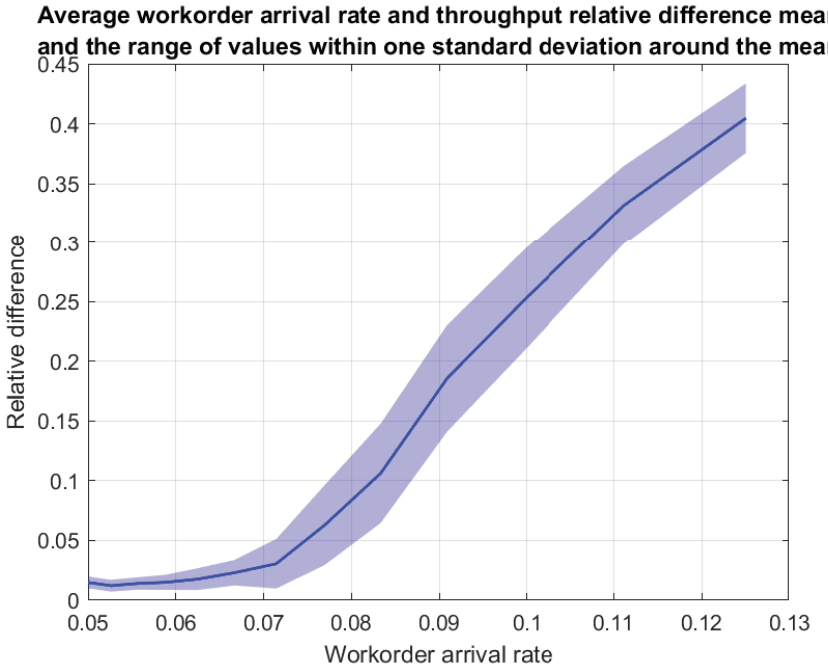

Figure 5.6: The measured average workorder arrival rate and throughput relative difference mean and range of values within one standard deviation from the mean over a range of deterministic workorder arrival rates for the variable workorder case.

the workcell process per simulation run is expected. Therefore, a lower workcell performance is expected compared to deterministic arrivals for a stable system, with more relative variability in the measured results as well. No other differences between this arrival method and the previous two are expected.

For this simulation experiment the workorder arrival rate value is varied over a range between 0.1 and 0.16 workorders per hour with a step of 0.005 for the deterministic workorder case. For the variable workorder case the workorder arrival rate value is varied between 0.04 and 0.1 workorders per hour with a step of 0.005.

The simulation performance results for the deterministic workorder case show a maximum average workorder throughput of approximately 0.137 workorders per hour, as shown in Figure 5.7. This is a similar result as the maximum throughput for the other two arrival methods.

The maximum exponential arrival rate value for which the workcell system is stable is determined using the same methodology applied for the deterministic workorder arrivals earlier. Using the relative difference between the measured workorder arrival rate and throughput show in Figure 5.8 the workcell system appears stable up to an arrival rate of approximately 0.12 workorders per hour. Comparing the measured average workorder lead time and WIP levels with simulations results from runs at half duration further indicate system instability for higher workorder arrival rates.

At the exponential workorder arrival rate value of 0.12, the mean average workorder throughput is 0.119 workorders per hour, the mean average workorder lead time is 76.03 hours, and mean average workorder WIP level is 9.16. An overview of the workcell performance measurement and relative variability at the arrival rate of 0.12 for the deterministic workorder case can be found in Table 5.3.

Workorder case:	**Deterministic**		**Variable**	
WO inter arrival time (hours):	**7.9**		**17**	
Performance Measurements	**Mean**	**CV**	**Mean**	**CV**
WO Throughput (per hour)	0.125	0.325 %	0.059	0.893 %
WO lead time (hours)	64.37	4.478 %	71.60	10.81 %
WO WIP level	8.128	4.478 %	4.251	10.75 %
Item throughput (per hour)	1.249	0.165 %	0.757	3.563 %
Milling machine 1 utilization	0.624	4.395 %	0.490	9.185 %
Milling machine 2 utilization	0.628	4.402 %	0.480	10.51 %
Average milling machine utilization	0.626	0.163 %	0.485	5.757 %

Table 5.2: Overview of the mean and coefficient of variation of the performance measurements for both workorder cases at approximately the lowest time interval between workorder arrivals in the workcell.

The maximum workorder arrival rate is lower than the equivalent maximum deterministic arrival rate for a stable workorder system. Therefore, the overal workorder performance is also lower in comparison as well. As expected, the relative variability in the measured performance results are significantly higher as well. Further analysis of the simulation results show no difference in relations between the performance measurements compared to the other arrival methods.

Finally, the simulation performance data for the variable workorder case show the expected results. Figure 5.9 shows a maximum average workorder throughput of 0.075 per hour, similar to the maximum throughput of the other arrival methods. The maximum exponential arrival rate value is estimated at 0.055 per hour based on the relative difference between the measured average workorder arrival rate and throughput as shown in Figure 5.10 and strengthened through the comparison of the average workorder lead times and WIP levels with the results from simulations with half duration.

For the variable workorder case from simulations with an exponential workorder arrival rate value of 0.055, the mean average workorder throughput is 0.055, the mean average workorder lead time is 82.18 hours, and the mean average workorder WIP level is 4.56. An overview of the performance measurements and relative variance can also be found in Table 5.3. As expected, the overall performance is lower compared to the deterministic arrival method at a lower estimated stable arrival rate and with increased relative variability.

5.1.4 Variability Analysis

Based on the results of this first experiment, the amount of variability in the measured performance results do appear to negatively impact the workcell performance for increased variability. While a more thorough investigation into the effect for all sources of variability in the model are beyond the scope of this graduation project, some analysis into the variability based on the experiment results and additional simulations is performed.

For each simulation all workcell functionality outcomes and workorder characteristics not deterministically assigned are sources for variability within each simulation run and between identical simulation runs with different random number seeds. In the default simulation settings for the experiments there are five sources of variability. These sources are the milling and clamping location per workorder, the first item quality inspection fail chance, the tool failure

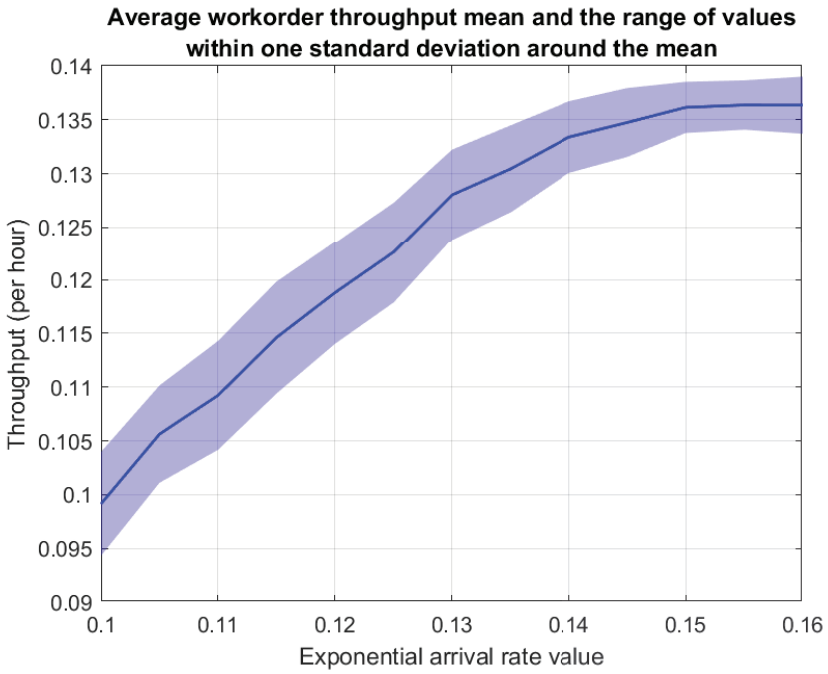

Figure 5.7: The average workorder throughput mean and range of values within one standard deviation from the mean over a range of exponential workorder arrival rate values for the deterministic workorder case.

chance during milling per tool, and the delayed replacement for broken special tools chance.

In addition, the variable workorder case adds three additional sources of variability to the system, the batch size and item milling time per workorder, and the toolset size per toolset. The results from the first experiments show an increased relative variance for the performance measurement compared to the relative variability for the deterministic workorder case.

Analysing some of the experiment simulation result data between runs with identical simulation settings did not find a clear relation between the workcell performance measurements and the fraction of occurrences per run for any of these variability sources. For example, over 100 identical runs, higher proportions of the number of items milled by either machine over the other does not relate to a higher or lower average performance per run.

Preliminary analysis in the variability of these fractions or proportions during a single simulation run show a very high variety in the fraction of items currently in the system for both the milling and clamping location. Although both are assigned randomly with a 50 percent chance per location for each workorder, the current fraction of items in the system for either milling location or clamping location can be as high as 100 percent.

Given the performance results from the verification process in the previous section, with a difference in performance between using one milling machine or both machines of over 50 percent for both the deterministic and variable workorder cases, it is assumed that the workcell performance during a single simulation are temporarily negatively impacted when the current milling location fraction leans to almost one location exclusively for the current items present in the workcell. Therefore, further investigation into the effects of variability on the workcell performance over the duration of a single simulation run is recommended, as well as further development of the simulation model for more deterministic workorder arrival methods or

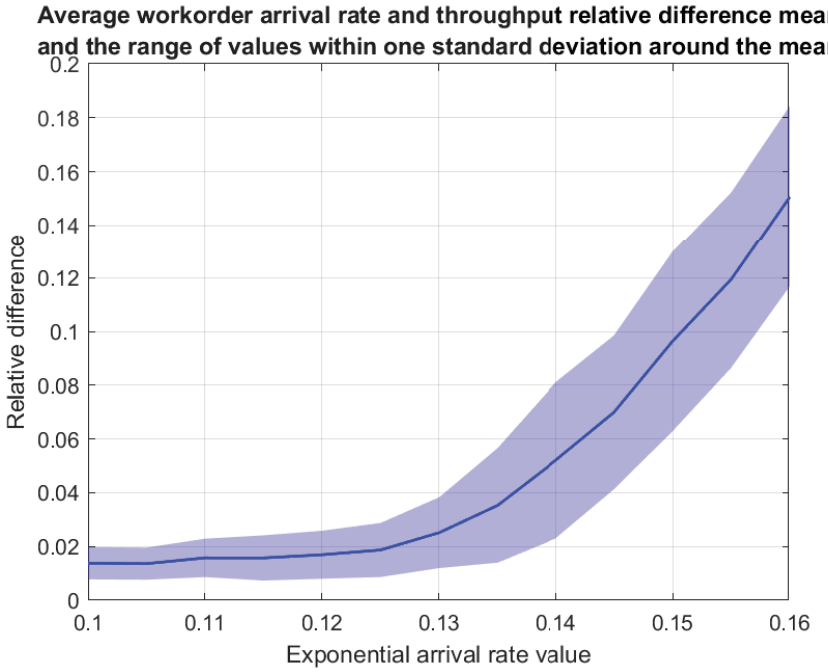

Figure 5.8: The measured average workorder arrival rate and throughput relative difference mean and range of values within one standard deviation from the mean over a range of exponential workorder arrival rate values for the deterministic workorder case.

scheduling based on the workorder milling and clamping location.

5.2 Itembatch and Toolset Size

The second workcell model experiment scenario is focused on investigating the effects on the workcell performance for workorders with different batch sizes and toolsets with different amount of tools per set. For this scenario three cases are considered. The first case investigates the performance measurements for workorders with an increasing batch size and a fixed toolset size per toolset. The second case investigates the effect on the workcell performance for increasing toolset sizes with a constant batch size per workorder. In the third case, the effect on the workcell performance is investigated for workorders with both increasing batch sizes and toolset sizes combined.

The default experiment simulation settings where used for all cases with a constant milling time of 1 hour per item for all workorders. In all cases workorders arrive deterministically at the workcell to ensure an average item arrival rate of one per hour. The item arrival rate is kept constant to test the effects of varying batch and toolset sizes under a constant average item load in the workcell throughout the experiment simulation runs.

5.2.1 Variable Batch Size

For the first case in this scenario the influence on the performance measurements for different workorder batch sizes is investigated. The Experiment parameter variation simulation settings

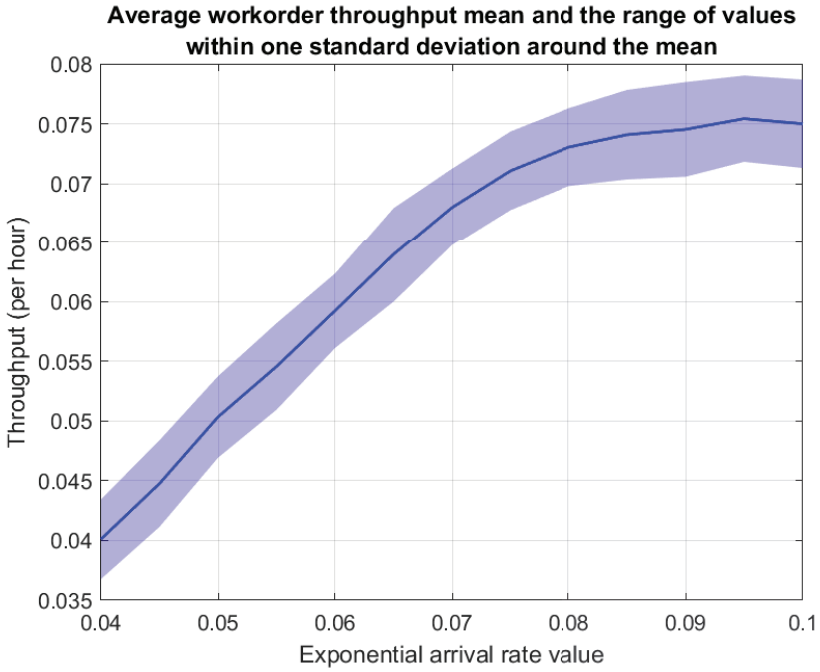

Figure 5.9: The average workorder throughput mean and range of values within one standard deviation from the mean over a range of exponential workorder arrival rate values for the variable workorder case.

are set to vary the workorder batch size between 1 and 25 with a step of 1. The workorder inter arrival time is set equal to the batch size. This ensures a constant item arrival rate at the workcell for each simulation run. Per unique setting 100 runs are performed with a unique random number seed per run. The toolset size is set to 5 for each toolset.

Based on the preliminary investigation into the variability and suspected effect on the workcell performance per simulation run, workorders with a smaller batch size are expected to have an increased performance with less relative variability. This behavior is expected because the milling and clamping are determined per workorder and not per item. This means that for smaller batch sizes a smaller fraction of items is assigned to one milling and clamping location per workorder, causing less variability in the fraction of current items in the system assigned to one machine or clamping location over the other.

Due to the workorder arrival rate decreasing for workorders with higher batch sizes in order to maintain a constant item arrival rate, the amount of workorder is lower for higher batch sizes. Less workorders also means less toolsets, leading to loading less toolsets into the cell and retrieving less toolsets from the cell. This should impact the workcell performance positively, but it is expected that this impact is very small, and will not compensate the expected decrease in performance for higher batch sizes due to extra variability per simulation run.

The simulation results show a small but significant decrease of the average item throughput for higher workorder batch sizes as expected. The relative variability of the average item throughput increases for higher batch sizes, indicating the expected increase in variability for higher batch sizes is valid. Figure 5.11 shows the average item throughput decrease over the range of the workorder batch sizes, and the increased range of values around the average item throughput mean due to the increase in variability as well.

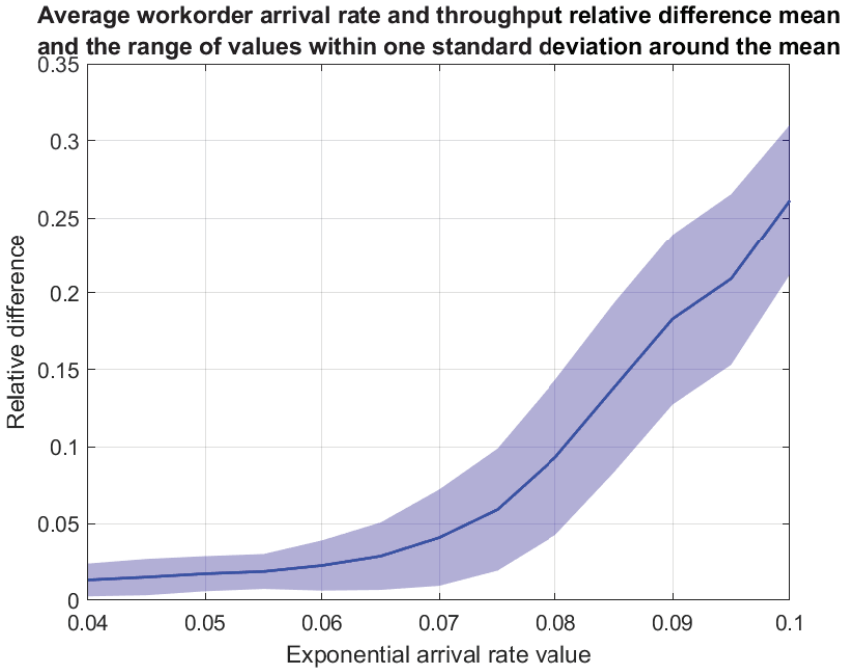

Figure 5.10: The measured average workorder arrival rate and throughput relative difference mean and range of values within one standard deviation from the mean over a range of exponential workorder arrival rate values for the variable workorder case.

The relation between the item throughput and milling machine utilities is similar to the relation from the previous experiment, with a level proportion between them over the range of simulated batch sizes. The relative variance increases for the average milling machine utilities for larger batch sizes as well. Figure 5.12 shows the average utilization from both milling machines combined. Table 5.4 provides an overview of the performance measurements and relative variation for the simulation runs with a batch size of 1 and 25. The overview shows an significant increase in the coefficient of variation for all performance measurements.

5.2.2 Variable Toolset Size

In the second case for this experiment scenario the toolset size is varied instead. The toolset size is varied between 2 and 50 with step of 2. The workorder batch size is set to 10 items for all workorders, and the workorder inter arrival time is set to 10, which is equivalent to an item arrival rate of 1.

Varying the toolset size with all other settings remaining equal is expected to only have a minor impact on the workcell performance due to the relative short extra handling times for additional tools per toolset. However, an increased number of tools per toolset does lead to an increased chance of tool failure occurring during milling per item, and consequently an increased chance for delayed toolset replacement occurring. Besides a minor negative impact on the average performance, it is expected to lead to a larger proportion of suborders per workorders on average. Therefore, it is expected to see a minor decrease in the item throughput for larger toolset sizes with an even smaller or no decrease in the workorder throughput.

As shown in Figure 5.13 the mean of the average item throughput very slightly decreases for

Workorder case:	**Deterministic**		**Variable**	
Exponential WO arrival rate value (per hour):	**0.12**		**0.055**	
Performance Measurements	**Mean**	**CV**	**Mean**	**CV**
WO Throughput (per hour)	0.119	4.010 %	0.055	6.640 %
WO lead time (hours)	76.03	18.49 %	82.18	16.61 %
WO WIP level	9.164	21.59 %	4.560	20.71 %
Item throughput (per hour)	1.183	3.994 %	0.694	7.502 %
Milling machine 1 utilization	0.591	6.426 %	0.447	12.81 %
Milling machine 2 utilization	0.595	4.951 %	0.452	11.65 %
Average milling machine utilization	0.593	3.967 %	0.450	8.520 %

Table 5.3: Overview of the mean and coefficient of variation of the performance measurements for both workorder cases at approximately the highest workorder arrival rate values.

Batch size:	**1**		**25**	
Performance Measurements	**Mean**	**CV**	**Mean**	**CV**
WO Throughput (per hour)	0.993	0.021 %	0.040	0.763 %
WO lead time (hours)	31.67	0.531 %	79.95	5.512 %
WO WIP level	31.56	0.529 %	3.200	5.586 %
Item throughput (per hour)	0.993	0.021 %	0.984	0.504 %
Milling machine 1 utilization	0.498	1.713 %	0.491	8.131 %
Milling machine 2 utilization	0.497	1.719 %	0.500	7.936 %
Average milling machine utilization	0.497	0.092 %	0.496	0.465 %

Table 5.4: Overview of the mean and coefficient of variation of the performance measurements for workorders with a batch size of 1 and 25.

increasing toolset sizes, with the mean of the average item throughput only falling outside the standard deviation range of values for the average item throughput at a toolset size of two for toolset sizes of 48 and 50. Therefore it can be concluded that the effect on the item throughput for an increasing toolset size is almost insignificant and lower than the expected minor decrease. Similar to the variable batch size case and the previous experiment, the milling machine utilities remain proportionally level to the item throughput over the range of toolset sizes.

Interestingly though, the average workorder throughput actually increases for higher toolset sizes as shown in Figure 5.14. This indicates that the expected proportionally growing number of suborders for higher toolset sizes does not only negate the slight item throughput decrease but gives the impression that higher toolset sizes might actually improve the workorder throughput. However, the slight item throughput decrease does indicate a decrease in the overal workcell performance.

Table 5.5 provides an overview of the performance measurements and their respective relative variation from the simulation results for a toolset size of 2 and 50. Besides the small difference in performance, the differences in the coefficient of variation are also small. This suggest the toolset size has minimal effect at best on the overall workcell performance and variation in the performance results.

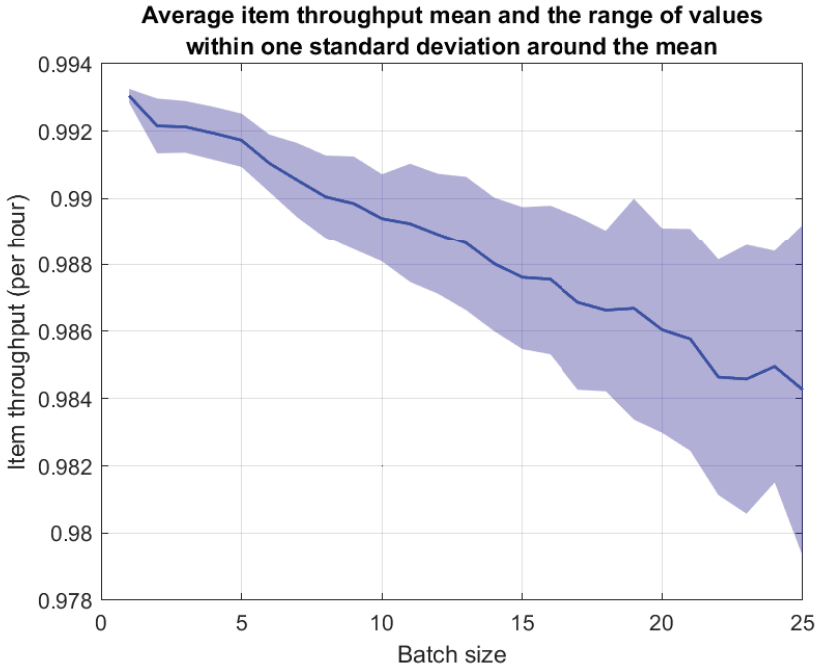

Figure 5.11: The average item throughput mean and range of values within one standard deviation from the mean over a range of workorder batch sizes.

5.2.3 Variable Batch and Toolset Size

In the last case for this experiment, the workorder batch size and toolset size are both increased simultaneously, combining the effects on performance from the first two cases. Similar to the first case, the batch size is varied between 1 and 25 with a step of 1, and the workorder inter arrival time is equal to the batch size. The toolset sizes is equal to two times the batch size, varying the toolset size between 2 and 50 with a step of 2.

For this case the workcell performance is expected to be similar to the performance of the first case, with possibly a noticeable slightly larger decrease in performance and increase in the relative variability for higher batch sizes due to the limited effect of the increasing toolset size measured in the second case. Otherwise no noticeable differences in performance or relations

Toolset size:	**2**		**50**	
Performance Measurements	**Mean**	**CV**	**Mean**	**CV**
WO Throughput (per hour)	0.099	0.206 %	0.101	0.719 %
WO lead time (hours)	50.80	1.219 %	58.50	2.348 %
WO WIP level	5.061	1.195 %	5.946	2.329 %
Item throughput (per hour)	0.989	0.156 %	0.988	0.195 %
Milling machine 1 utilization	0.494	4.885 %	0.491	4.996 %
Milling machine 2 utilization	0.498	4.875 %	0.501	4.836 %
Average milling machine utilization	0.496	0.151 %	0.496	0.176 %

Table 5.5: Overview of the mean and coefficient of variation of the performance measurements for workorders with a toolset size of 2 and 50.

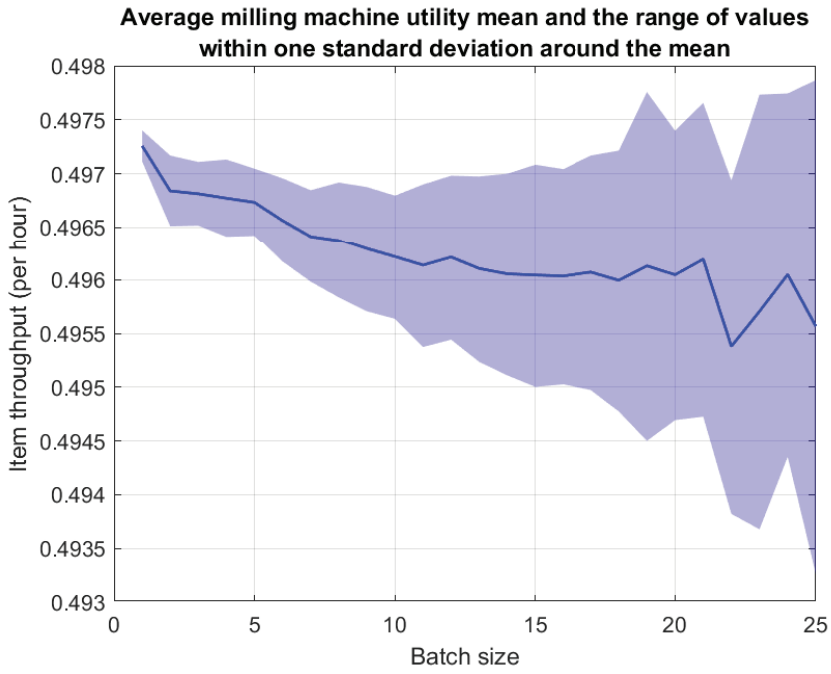

Figure 5.12: The average milling process utilization for both machines mean and range of values within one standard deviation from the mean over a range of workorder batch sizes.

between the performance measurements are expected.

Figure 5.15 does show a similar decrease in the average item throughput for increased batch sizes compared to the first case shown in Figure 5.11 earlier. To determine if a noticeable larger decrease in performance occurs, the performance measurements and the relative variance for workorders with a batch size of 25 are compared for the first and this case. Effectively, this compares the difference in performance for a toolset size of 5 with a toolset size of 50. An overview of both performance measurements can be found in Table 5.6.

The additional decrease for the average item throughput is of the same order as the decrease measured in the second case. The average milling machine utilization remains proportionally level with the item throughput, causing a similar decrease in performance as the item throughput. However, due to the relative larger variability for the milling machine utilities, this effect appears insignificant. This suggest that the effects of varying batch size and toolset sizes on the item and milling machine performance are additional without any other interactions.

Looking at the performance difference for workorders, the average lead time and WIP level are significantly greater when toolsets have a size of 50 compared to 5. The relative variation is also significantly greater for the large toolset size. This indicates that an increased proportion of suborders impacts the workorder performance and relative variation in performance significantly more than the overall workcell performance is effected by the increased toolset size.

5.3 Item and Tool Inventory Capacity

The final experiment scenario is centered around investigating the influence of both the item and tool storage capacity in the cell. The effect of the cell storage capacity on the workcell

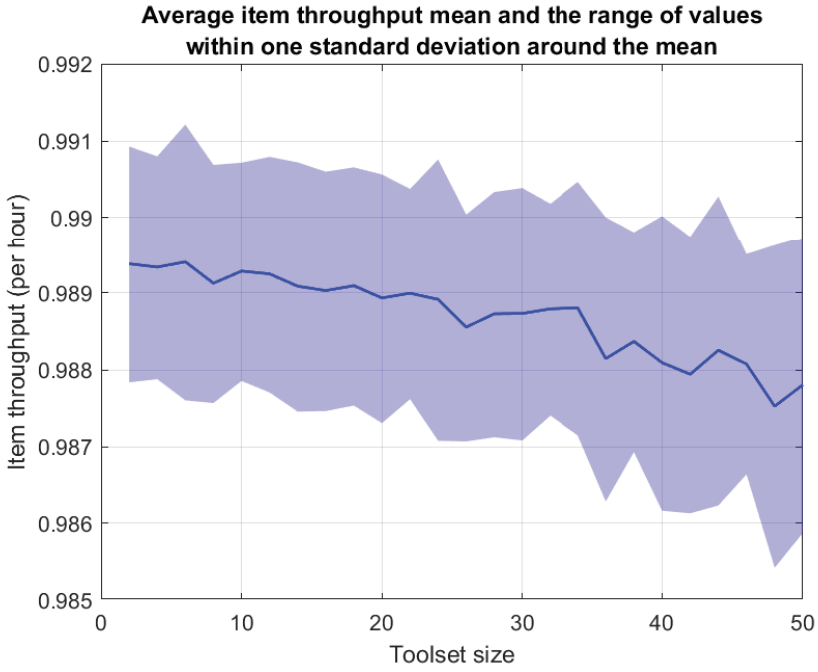

Figure 5.13: The average item throughput mean and range of values within one standard deviation from the mean over a range of toolset sizes.

performance is investigated for first the deterministic workorder case, and the variable workorder case afterwards. In this scenario workorders arrive according to a fixed number of 15 workorders in the workcell. This amount of workorders is greater than the optimal number of workorders determined in the first experiment for both cases, allowing investigating if the cell inventory capacity was a limiting factor on the maximum performance measured earlier.

Throughout the experiment simulations, the item and tool capacity are varied. All other simulation settings are kept at the default experiment settings described at the start of this chapter. The workcell performance results obtained during the verification process already indicated the item storage capacity could have a large effect on the workcell performance. Since each workorder requires a unique special toolset in these experiment and both have to be present in the workcell, it is expected that the tool storage capacity could have an equally large effect

Toolset size:	**5**		**50**	
Performance Measurements	**Mean**	**CV**	**Mean**	**CV**
WO Throughput (per hour)	0.040	0.763 %	0.041	1.681 %
WO lead time (hours)	79.95	5.512 %	103.7	20.70 %
WO WIP level	3.200	5.586 %	4.352	20.93 %
Item throughput (per hour)	0.984	0.504 %	0.976	1.009 %
Milling machine 1 utilization	0.491	8.131 %	0.488	7.918 %
Milling machine 2 utilization	0.500	7.936 %	0.497	7.537 %
Average milling machine utilization	0.496	0.465 %	0.493	0.987 %

Table 5.6: Overview of the mean and coefficient of variation of the performance measurements for workorders with a batch size 25 and a toolset size of 5 and 50.

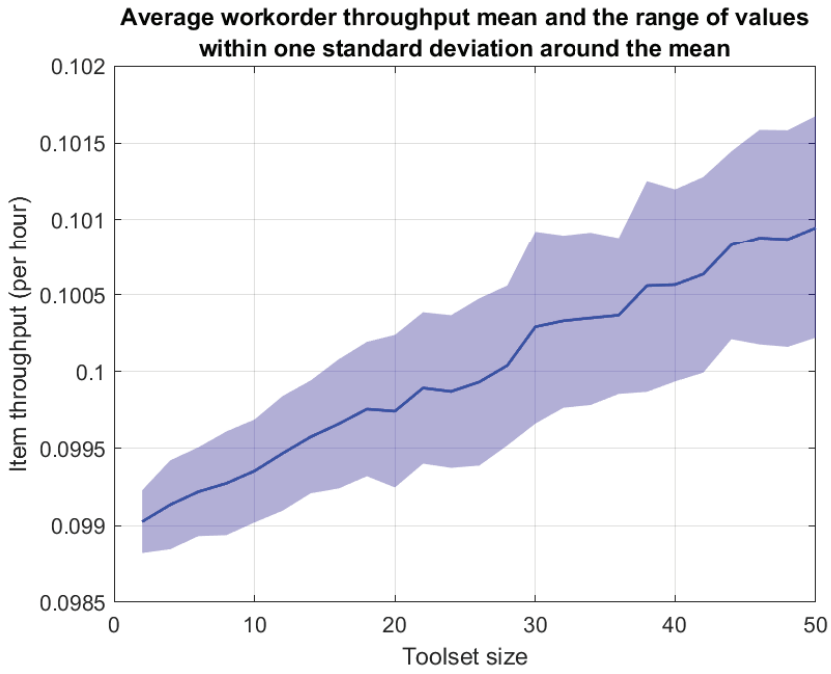

Figure 5.14: The average workorder throughput mean and range of values within one standard deviation from the mean over a range of workorder batch sizes.

on the workcell performance as well.

The simulation setup is similar for both cases. For each case three different experiment variations are performed using a different *Experiments* parameter variation setup. In the first experiment variation both the item and tool inventory capacity are varied between 1 and 15 times the respective maximum batch and toolset size. In the second variation the tool capacity is varied between 1 and 15 times the maximum toolset size with a fixed item capacity of 3 times the maximum batch size. For the last experiment variation, the item capacity is varied between 1 and 15 times the maximum batch size with a fixed tool capacity of 3 times the maximum toolset size.

The performance relations between the performance measurements are expected to be similar as in the previous experiments, and the first experiment with the fixed number of workorders in the workcell arrival method in particular for both cases. Therefore, only notable differences are mentioned for the results of this last experiment.

5.3.1 Deterministic Workorder Case

For the deterministic workorder case the maximum batch size and toolset size are both 10. This leads to varying the item and/or tool storage capacity between 10 and 150 with a step of 10, as well as a fixed item and tool capacity of 30, in the appropriate experiment variation setup. In Figure 5.16 the average workorder throughput over the range of the relative item and/or tool capacity for each experiment variation is shown.

As can be seen in Figure 5.16 both the item and tool capacity contribute almost equally to the average workorder throughput, meaning a relative increased capacity for both is required to

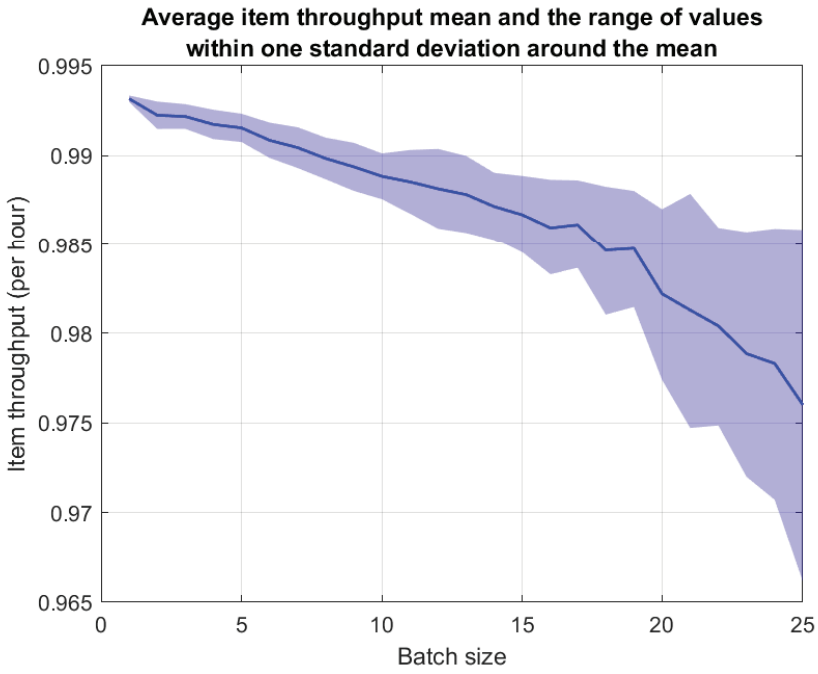

Figure 5.15: The average item throughput mean and range of values within one standard deviation from the mean over a range of workorder batch and toolset sizes.

further increase the workorder throughput. Analysis of the other performance measurements show similar results due to the similar relations between them as expected. The relative variation of the performance measurements are comparable to the first experiment results as well.

In case one of the two capacities is relatively lower compared to the other, the item capacity is slightly more punishing on the workcell performance than the tool capacity. It is assumed that this is caused due to items requiring more time to load and retrieve from the cell, especially those clamped and unclamped inside the cell by the clamping machine, compared to tools. This would negatively impact the workcell performance more in case the item capacity is the limiting factor. Further experimentation is required to confirm this assumption.

Due to the higher relative item capacity of 8 times and higher, the average workorder throughput is able to reach a higher maximum at 15 workorders in the workcell compared to the first experiment. In this experiment the maximum average workorder throughput is approximately 0.146 workorders per hour compared to 0.139 workorders per hour. This maximum is reached at an item and tool capacity of approximately 12 times the minimum capacity compared to the default 7.5 times item capacity and 40 times tool capacity for the deterministic workorder case.

5.3.2 Variable Workorder Case

For the variable workorder case the maximum batch size is 25, and the maximum toolset size is 50. This leads to varying the item storage capacity between 25 and 375 with a step of 25, varying the tool capacity between 50 and 750 with a step of 50, as well as a fixed item and tool capacity of 75 and 150 respectively, in the appropriate experiment variation setup. In Figure 5.17 the average workorder throughput over the range of the relative item and/or tool capacity from each experiment variation is shown.

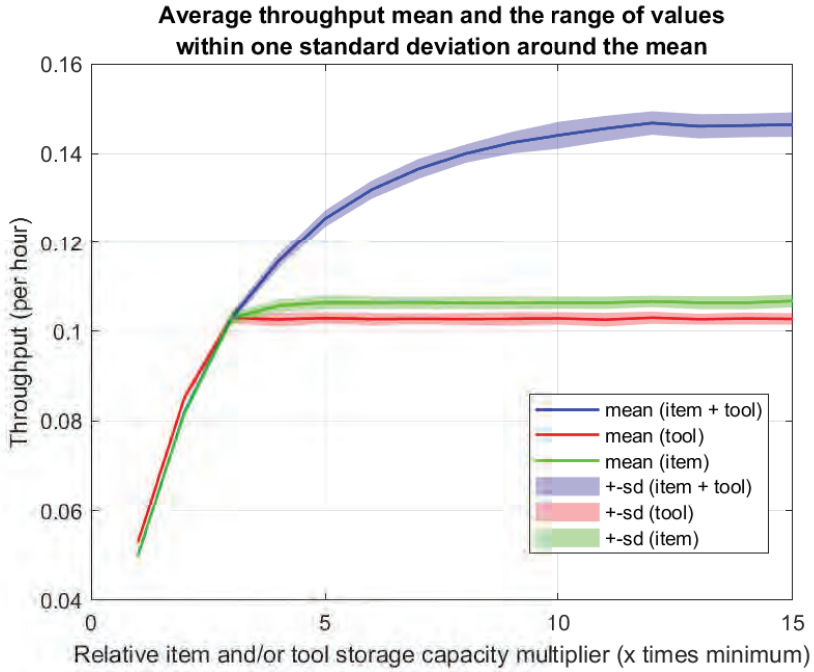

Figure 5.16: The average workorder throughput mean and range of values within one standard deviation from the mean over the range of the relative item and tool storage capacity for the varied item, varied tool, and varied both experiment scenarios for the deterministic workorder case.

As can be seen in Figure 5.17 both the item and tool capacity contribute almost equally to the average workorder throughput, meaning similar to the deterministic workorder case, a relative increased capacity for both is required to further increase the workorder throughput. Analysis of the other performance measurements show similar results due to the similar relations between them as expected. The relative variation of the performance measurements are comparable to the first experiment results as well.

Also similar to the deterministic case, the item capacity is more punishing in case one of the relative storage capacities is the limiting factor for the same assumed reasons. However, due to the increased relative variability, the mean average workorder throughput for both the varied tool and varied item capacity case do fall within the range of values within one standard deviation to its respective mean.

For the variable workorder case the average workorder throughput is also able to reach a higher maximum at 15 workorders in the workcell compared to the first experiment. In this experiment the maximum average workorder throughput is approximately 0.089 workorders per hour compared to 0.076 workorders per hour. This maximum is reached at an item and tool capacity of approximately 8 times the minimum capacity compared to the default 3 times item capacity and 8 times tool capacity. The increased variability and higher average batch and toolset size compared to the deterministic case are likely the cause for encountering other limiting factors on the workcell performance when the relative item and tool capacity are increased. Further experimentation and investigation is recommended to gain a better inside into the limiting factors on the workcell performance.

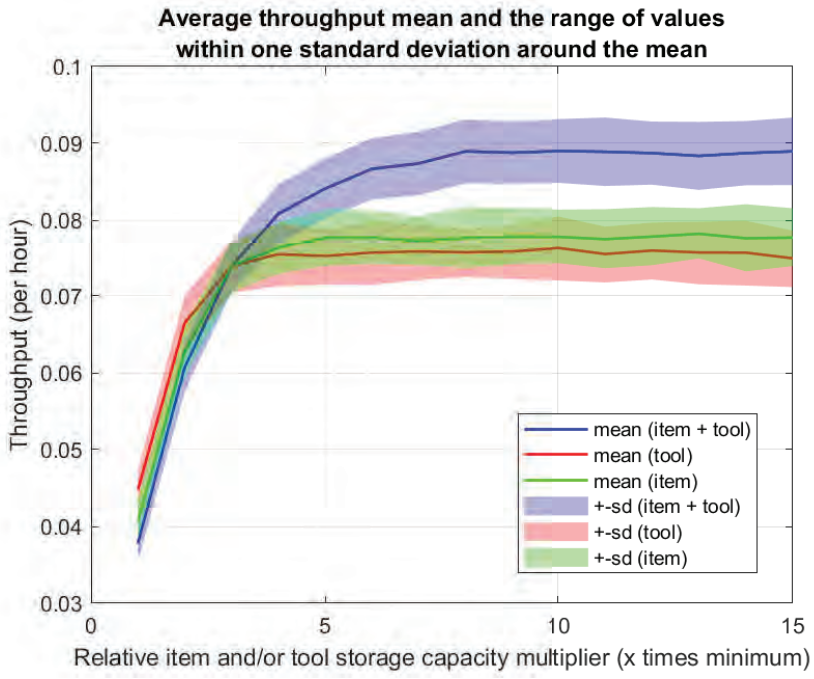

Figure 5.17: The average workorder throughput mean and range of values within one standard deviation from the mean over the range of the relative item and tool storage capacity for the varied item, varied tool, and varied both experiment scenarios for the variable workorder case.

Chapter 6

Conclusion and Recommendations

The objective of this graduation project was to develop a simulation model that functions as a digital twin to the workcells at KMWE. In order to develop this simulation model, a detailed understanding of the workcells at KMWE was gained first. Next, AnyLogic was chosen as the software package to develop the workcell model. Among other considerations, AnyLogic was chosen due to its capabilities to integrate separately developed models into a larger model.

Next, the model was developed by starting with the most basic functionalities of the workcell model and building it out from there. During most of the development process, the model was developed with the integration of the model into a larger, factory wide, model in mind. However, during the model development process, model integration into a larger factory wide model was deemed infeasible by the AML project. During development some of the modeling choices where influenced by the expected integration. This contributed to the most notable difference between the workcell model and the KMWE workcells. This difference is the lack of any workorder scheduling and production planning, as this was expected to be part of the larger integrated model. Some other notable features missing from the model are any form of maintenance or workcell component failures and repairs that would disrupt the workcell process. Most other functionalities of the KMWE workcells are incorporated in the model.

The verification process of the developed simulation model functionalities confirmed the workcell model behavior and basic performance working as intended and expected. Based on the verification results it is recommended to add the functionality to temporarily retrieve toolsets reserved for workorders not loaded into the cell from the cell to prevent a potential simulation deadlock from occurring. This deadlock can only occur if toolsets in the cell inventory can be reserved by new workorders, and both the item and tool inventory capacity are lower than three times the maximum workorder batch size and toolset size.

Performing several simulation experiments with the workcell model showed that the relative item and tool capacity in the cell are a major limiting factor on the workcell performance as expected. In addition, under a constant item load, the workorder batch size does slightly negatively impact the workcell performance for increasing batch sizes. A common factor discovered during the experiments is that an increased relative variability in the workcell performance measurements has a significant negative impact on the workcell performance. Further investigation and simulation runs indicate that a large source of variability with a negative impact on the performance is the random assignments of the milling machine and clamping location per workorder on workorder arrival.

Based on the simulation experiment results it is recommended to further develop the workcell model to include workorder scheduling and planning for the workcell based on a workload per machine. This would more accurately resemble the workorder arrivals and load distribution at the KMWE workcells, and is expected to reduce the relative variability in the workcell performance due to random milling machine assignment. Expanding the workcell model to include loading items and tools into the cell based on production schedules per milling machine is recommended as well. Another recommendation for further development is calculating the workcell performance measurements over one or more predetermined time intervals within one simulation run in order to gain more insight into the workcell performance over time per run.

Finally, starting the validation process and further developing the model until validated was beyond the scope of this graduation project. Therefore, the developed model cannot be confirmed as an accurate representation, or a digital twin, of the workcells at KMWE Precision components. It is highly recommended to further develop the workcell model until validated for further use within the AML project, starting with adding workorder scheduling and planning. Validating the workcell model would allow for meaningful insights into the effects on the workcell performance at KMWE based on the measured effects with the workcell model through simulation experiments.

Bibliography

[1] AnyLogic, "Multimethod Modeling." https://www.anylogic.com/use-of-simulation/multimethod-modeling/. (accessed: 26-5-2020).

[2] AnyLogic, "Discrete Event Modeling." https://www.anylogic.com/use-of-simulation/discrete-event-simulation/. (accessed: 26-5-2020).

[3] AnyLogic, "Agent Based Simulation Modelling." https://www.anylogic.com/use-of-simulation/agent-based-modeling/. (accessed: 26-5-2020).

[4] AnyLogic, "System Dynamics." https://www.anylogic.com/use-of-simulation/system-dynamics/. (accessed: 26-5-2020).

Appendices

Appendix A

Workcell Process Flow

On the next page the complete process flow for a workcell at KMWE Precision Components is shown.

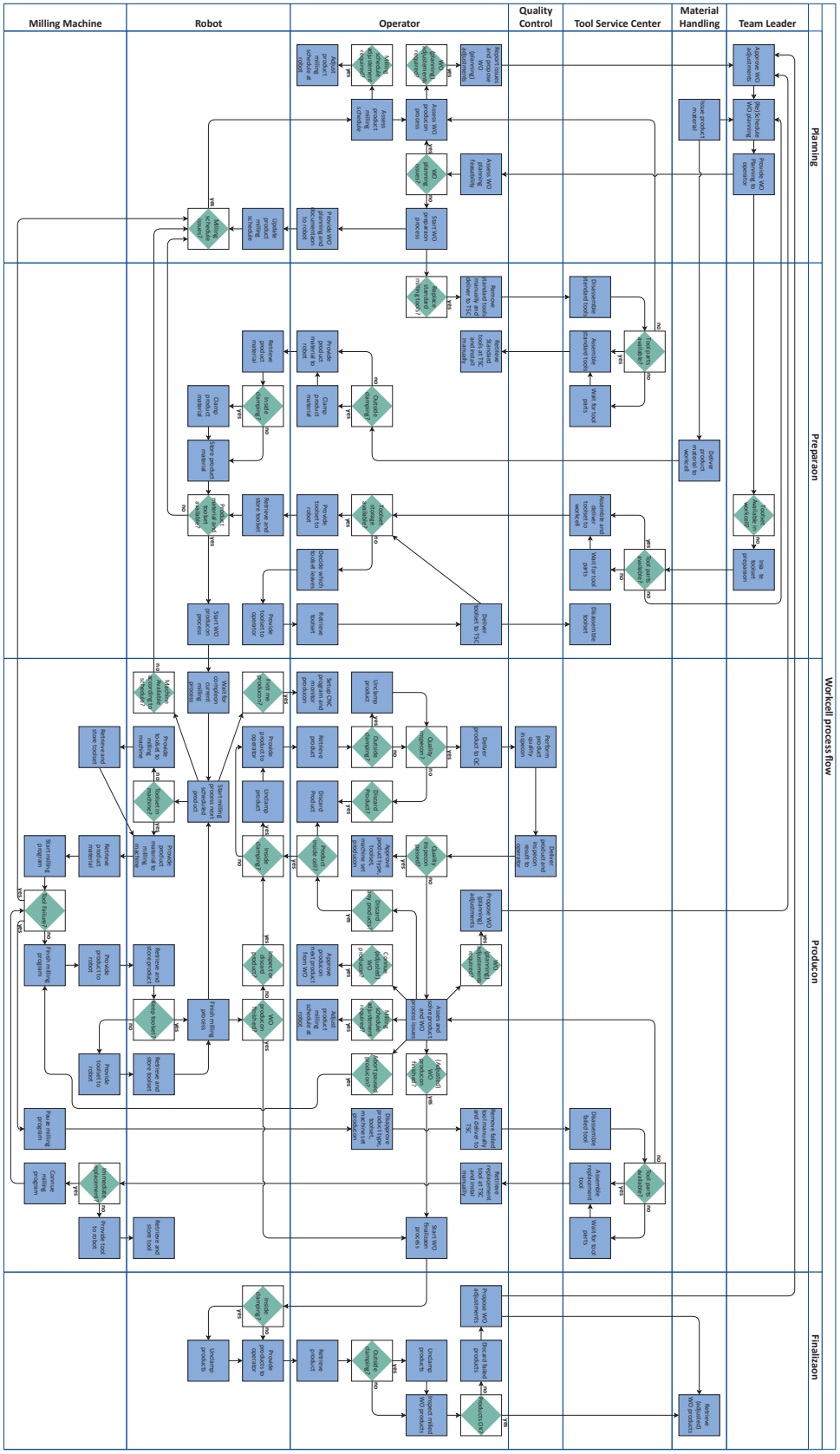

Figure A.1: Complete workcell process flow.

Appendix B

Model Database and Simulation Options

A full overview of the databases included in the workcell model and the simulation options available through the *Simulation* experiment setup window before each simulation run are given in this appendix.

B.1 Model Database Overview

For each database in the simulation model a short description, the column names and data type, and the number of row entries is given.

cell_data (48 entries):
This database contains many of the operator, robot and milling machine handling times used in the process flowcharts based on estimates for the real workcells at KMWE. The tool replacement time at TSC and item quality inspection time at QC are included as well. Also included is the percentage per minute the chip containers are filled at each milling machine while milling. An overview of the database columns, column values data type and description can be found in the table below:

column name	**data type**	**description**
name	String	name of the handling time
value_1	double	value of the handling time, mean value for normal distribution
value_2	double	value of the handling time, standard deviation value for normal distribution
value_3	double	value of the handling time, minimum value for normal distribution
value_4	double	value of the handling time, maximum value for normal distribution
comment	String	handling timeunit, use of distribution, other clarification

pls_data (60 entries):
This database contains the toolset PL numbers in combination with each tool TA number used

to create the standard toolsets and tools for each machine. Each standard toolset consists of 30 tools. An overview of the database columns, column values data type and description can be found in the table below:

column name	data type	description
pl_number	String	standard toolset PL number
ta_number	String	tool TA number contained in standard toolset with corresponding PL number

real_item_data (1073 entries):
This database is imported from KMWE data containing item numbers, milling times and batch sizes for workorders the real database workorder types option is chosen. An overview of the database columns, column values data type and description can be found in the table below:

column name	data type	description
item	String	item type IT number
wc	String	workcell number, not used in model
mach_su	double	first time milling setup time, not used in model
mach_run_hr	double	milling time per item in hours
su_qty	int	number of items milled per machine setup, used as batch size in model, set to 25 if higher

real_pl_data (732 entries):
This database contains a selection of toolset PL numbers from KMWE. Used for real database type toolsets in model. Consists of only one column:

column name	data type	description
pl_number	String	toolset PL number

real_ta_data (14264 entries):
This database contains tool TA numbers linked to PL numbers also contained *real_pl_data*. Used to determine number of tools for each linked toolset for real database type toolsets in model. An overview of the database columns, column values data type and description can be found in the table below:

column name	data type	description
pl_number	String	toolset PL number
ta_number	String	tool TA number part of toolset with linked PL number

custom_item_data (50 entries):
This database contains the item milling processing time, the minimum and maximum batch size, and the required toolset PL number for each defined item type. Used by custom database workorder types. An overview of the database columns, column values data type and description can be found in the table below:

column name	data type	description
item_number	String	item type IT number
process_time	double	milling process time for item type
size_min	int	minimum batch size for workorders for item type
size_max	int	maximum batch size for workorders for item type
pl_number	String	toolset PL number required for milling this item type items

custom_pl_data (25 entries):
This database contains a selection of custom toolset PL numbers also used in *custom_item_data*. Used for custom database type toolsets in model and can be linked to the item types defined in custom_item_data as well. It consists of only one column:

column name	data type	description
pl_number	String	toolset PL number

custom_ta_data (325 entries):
This database contains tool TA numbers linked to PL numbers also contained *custom_pl_data*. Used to determine the number of tools for each linked toolset for custom database type toolsets in model. An overview of the database columns, column values data type and description can be found in the table below:

column name	data type	description
pl_number	String	toolset PL number
ta_number	String	tool TA number part of toolset with linked PL number

random_item_pl_data (25 entries):
This database contains a list of 25 item type IT numbers and PL numbers that can be used and linked to each other for the random workorder and toolset type. An overview of the database columns, column values data type and description can be found in the table below:

column name	data type	description
item_number	String	item type IT number
pl_number	String	toolset PL number

operator_priorities (20 entries):
This database contains the priorities for seizing the operator for the different handling processes in the process flowcharts. An overview of the database columns, column values data type and description can be found in the table below:

column name	data type	description
name	String	handling process name
value	int	handling process priority

robot_priorities (16 entries):
This database contains the priorities for seizing the robot for the different handling processes in the process flowcharts. An overview of the database columns, column values data type and description can be found in the table below:

column name	data type	description
name	String	handling process name
value	int	handling process priority

B.2 Simulation Options Overview

For each simulation option a descriptive name, the Java data type of the option value, and the default value are stated. Where applicable, the value range for each option is included, using inf if no minimum or maximum value is set. Other option or database variables can be used as

minimum or maximum values as well, using their names when applicable.

The general simulation options are stated in the table below:

option name	**value type (range)**	**default value**
Random seed number	int (1, inf)	1
Item capacity	int (maximum itembatch size, inf)	75
Tool capacity	int (maximum toolset size, inf)	400
Machine 1 chance	double (0, 1)	0.5
Outside clamping chance	double (0, 1)	0.5
Tool failure chance	double (0, 1)	0
Tool repair chance 1 chance	double (0, 1)	1
Quality inspection pass chance	double (0, 1)	1
Operator schedule	int choice (24/7 (0), 17 hour day (1). 17 hour day sunday free (2))	24/7 (0)
Operator priorities	boolean	false
Robot priorities	boolean)	false

The workorder arrival method options are stated in the table below:

option name	**value type (range)**	**default value**
Workorder arrival method	int choice (arrival rate (0), deterministic (1), schedule (2), fixed number (3), Inject only (4))	deterministic (1)
Arrival rate (per hour)	double (0, inf)	0.2
Time between arrivals (per hour)	double (0, inf)	10
Workorders in system	int (0, inf)	10
Initial Workorders	int (0, 10)	0

The workorder and toolset type options are stated in the table below:

option name	**value type (range)**	**default value**
Workorder/toolset type	int choice (real database (0), custom database (1), random (2))	random (2)
Workorder and toolset link	boolean	false
IT and PL numbers from random_item_pl_data database	boolean	false
Max IT /PL numbers from random_item_pl_data	int (1, # of entries in *random_item_pl_data* database)	# of entries random_item_pl_data database

The option names and value range for all custom workorder characteristics are stated in the table below:

option name	**value range**
Method	deterministic (0), uniform (1), triangular (2), normal (3)
Mean	Min, Max
Min	0, inf
Max	Mean, inf
Var	0, inf

The value data type and default value of the Mean, Min, Max and Var options for each characteristic are stated in the table below:

characteristic name	**value type**	**default values (Method, Mean, Min, Max, Var)**
Itembatch arrival delay (hours)	double	deterministic (0), 1, 0, 24, 1
Toolset arrival delay (hours)	double	deterministic (0), 24, 0, 48, 1
Toolset lifetime (# of items)	int	deterministic (0), 50, 25, 100, 10
Standard toolset lifetime (# of items)	int	deterministic (0), 100, 50, 200, 10
Outside clamping time (minutes)	double	deterministic (0), 10, 1, 30, 1
Outside unclamping time (minutes)	double	deterministic (0), 10, 1, 30, 1
Inside clamping time (minutes)	double	deterministic (0), 2, 1, 3, 1
Inside unclamping time (minutes)	double	deterministic (0), 2, 1, 3, 1
Item processing time (hours)	double	deterministic (0), 1, 0.1, 20, 1
Batch size	int	deterministic (0), 5, 1, 25, 1
Toolset size	int	deterministic (0), 10, 1, 50, 5

Appendix C

Minimum Itembatch Lead Time Formulas

The table below the next page contains the formulas to calculate the itembatch status durations in minutes to determine the minimum itembatch lead times based on the workorder batch size and linked special toolset size, as well as the item clamping times and milling times constant per workorder. The status duration formulas take the clamping location and first item quality inspection into account. Therefore, the durations are equal to zero if the condition stated above the relevant table block is not true. To determine the lead time for a particular itembatch the calculations have to be done in order as the batch size is changed when the quality inspection fails.

Itembatch progression states (bold), condition (italic), and breakdown per state	**Itembatch State duration (minutes)**	**Constant numbers explanation**
if outside clamping:		
Split:	**0.3333**	Operator: move itembatch to clamping area (0.3333).
OutsideClamping	**outsideClampingTime * batchsize**	Operator: clamp item. (outsideClampingTime).
EnterCell	**9.8333 * batchsize**	
get pallet (per item)	2.25	Robot: pallet from inventory to pallet exchange (0.25). Operator: retrieve pallet from cell (2).
mount item on pallet	5.3333	Operator: move item to pallet exchange (0.3333), mount item on pallet (5).
load item into cell	2.25	Operator: load item into pallet exchange (2). Robot: item from pallet exchange to inventory (0.25).

if inside clamping:

Split:	**0.3333**	Operator: move itembatch to item exchange (0.3333).
EnterCell	**0.3333 * batchsize**	Operator: load item into item exchange (0.3333).
InsideClamping	**(0.75 + insideClampingTime) * batchsize**	Robot: item to clamping machine (0.25), item to cell inventory (0.25). Item wait for pallet at clamping machine (0.25).

FirstItem:	**1 + (1.3333 * toolsetSize) + millingTime**	
toolset split	0.3333	Operator: move toolset to exchange (0.3333).
load tools into cell	(0.5833 * toolsetSize) + 0.1666	Operator: load tool into cell (0.5). Robot: retrieve tool from pallet exchange (0.0833), store tool into inventory while operator loads next tool (0), store last tool into inventory (0.1666).
load tools into machine	0.75 x toolsetSize	Robot: move tool from inventory to milling machine (0.25). Milling machine: load tool into machine (0.5).
mill item	0.5 + millingTime	Robot: move item from inventory to milling machine (0.25). Milling machine: mill item (millingTime). Robot: move item from milling machine to inventory (0.25).

if outside clamping:

QualityControl:	**256.5833 + outsideUnclampingTime**	
item exit cell	2.25	Robot: move item from inventory to pallet exchange. Operator: retrieve item from pallet exchange.
remove item from pallet	2.3333	Operator: remove item from pallet (2), move item to clamping area (0.3333).
item unclamping	2 + outsideUnclampingTime	Wait for operator loading pallet into exchange (2). Operator: unclamp item (outsideUnclampingTime).
item quality inspection	250	Operator: move item to QC (10). Item quality inspection (240).

if inside clamping:

QualityControl:	**250.8333 + insideUnclampingTime**	
item unclamping	0.5 + insideUnclampingTime	Robot: move item from inventory to clamping machine (0.25). Clamping machine: unclamp item (insideUnclampingTime). Robot: move item from clamping machine to item exchance (0.25).
item exit cell	0.3333	Operator: retrieve item from item exchange.
item quality inspection	250	Operator: move item to QC (10). Item quality inspection (240).

if QC fail:	*batchsize = batchsize - 1*	adjust batch size for failed item

if QC fail AND batchsize > 0:

Assessment:	**240**	Operator: quality assessment time
FirstItem2:	**0.5 + (0.75 * toolsetSize) + millingTime**	
load tools into machine	0.75 x toolsetSize	Robot: move tool from inventory to milling machine (0.25). Milling machine: load tool into machine (0.5).
mill item	0.5 + millingTime	Robot: move item from inventory to milling machine (0.25). Milling machine: mill item (millingTime). Robot: move item from milling machine to inventory (0.25).

if QC fail AND batchsize > 0 AND outside clamping:

QualityControl2:	**256.5833 + outsideUnclampingTime**	Same as QualityControl when outside clamping.

if QC fail AND batchsize > 0 AND inside clamping:

QualityControl2:	**250.8333 + insideUnclampingTime**	Same as QualityControl when inside clamping.

if batchsize > 1

Production:	**(0.75 * toolsetSize) + ((0.5 + millingTime) * (batchsize - 1))**	
load toolset into machine	0.75 * toolsetSize	Robot: move tool from inventory to milling machine (0.25). Milling machine: load tool into machine (0.5).
mill remaining items	(0.5 + millingTime) * (batchsize - 1)	Robot: load item into machine (0.25), load item into cell (0.25).

if batchsize > 1 AND inside clamping:

InsideUnclamping:	**((0.5 + insideUnclamping Time) * (batchsize - 1)) + ((0.25) x (batchsize - 2))**	Robot: move item from inventory to clamping machine (0.25). Clamping machine: unclamp item (insideUnclampingTime). Robot: move item to item exchange (0.25). Next item wait for robot to move pallet from clamping machine to inventory (0.25).
ExitCell:	**0.3333 * (batchsize - 1)**	Operator: retrieve item from item exchange (0.3333).

if batchsize > 1 AND outside clamping:

ExitCell:	**0.25 + (4.3333 X (batchsize - 1))**	Robot: move first item from inventory to pallet exchange (0.25), move rest of items from inventory to pallet exchange while operator removes previous item from pallet (0). Operator: retrieve item from pallet exchange (2), remove item from pallet (2), move item to clamping area (0.3333).
OutsideUnclamping:	**2 + (outsideUnclampingTime x (batchsize - 1))**	Wait for operator loading pallet into exchange (2). Operator: unclamp item (outsideUnclampingTime).

if batchsize > 0

Split:	**0.3333**	Operator: move itembatch to local storage area (0.3333).