Stefan Kern BSc.

Spatial enabled Agent-based Modeling and Simulation of a Production Environment

MASTER'S THESIS

to achieve the university degree of

Master of Science

Master's degree programme: Geomatics Science

submitted to

Graz University of Technology

Supervisor

Ass.Prof. Dipl.-Ing (FH) Dr.techn. Johannes Scholz

Institute of Geodesy

Graz, September, 2018

EIDESSTATTLICHE ERKLÄRUNG

AFFIDAVIT

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis.

Datum / Date

Unterschrift / Signature

Abstract

Industry 4.0 and digitization are terms that have become increasingly important in recent years. These terms also include ideas and concepts to increase the efficiency of production plants. This thesis deals with Industry 4.0 in particular with the term Smart Factory, with the goal of being able to simulate a semiconductor production process virtually. This modeling is agent-based since this approach is suitable for implementing the idea of a smart factory, taking into account the conditions of semiconductor production. For this purpose, a modeling concept is presented which combines the aspects of Industry 4.0 and agent-based modeling. This approach is further implemented and analyzed in the form of a prototype. With this model, we try to create an efficient production environment in which the workers cover as short distances as possible and the available machines and interim storage facilities are used as efficiently as possible. On the one hand, the work shows how such an environment is modulated, but also how data can be obtained from it. By the captured data it is analyzed whether the chosen approach brings advantages for the production but also how the received information can be used to improve the production process.

Zusammenfassung

Industrie 4.0 und Digitalisierung sind Begriffe, die in den letzten Jahren immer mehr an Bedeutung gewonnen haben. Unter anderem beschäftigen sie sich mit Ideen und Konzepten, welche die Effizienz von Produktionsanlagen erhöhen. Diese Masterarbeit beschäftigt sich mit Industrie 4.0 - im Speziellen mit dem Konzept Smart Factory - um einen Halbleiterproduktionsprozess virtuell nachzubilden und simulieren zu können. Diese Modellierung erfolgt agentenbasiert, da sich dieser Ansatz dazu eignet, die Idee von einer Smart Factory unter Berücksichtigung der Gegebenheiten einer Halbleiterproduktion umzusetzen. Zu diesem Zweck wird ein Modellierungskonzept vorgestellt, welches die Aspekte von Industrie 4.0 und agentenbasierter Modellierung miteinander vereint. Dieser Ansatz wird in weiter Folge in Form eines Prototyps umgesetzt und analysiert. Mit diesem Prototyp wird versucht, eine effiziente Produktionsumgebung zu schaffen, in der die Arbeiter/innen möglichst kurze Strecken zurücklegen und die zur Verfügung stehenden Maschinen und Zwischenlager effizient genutzt werden können. Die Masterarbeit zeigt einerseits, wie eine solche Umgebung moduliert wird, aber auch, wie daraus Daten gewonnen werden können. Anhand der erhobenen Daten wird in weiter Folge analysiert, ob der gewählte Ansatz Vorteile für die Produktion bringt und wie die gewonnen Informationen dazu genutzt werden können, den Produktionsprozess zu verbessern.

Acknowledgements

I would like to take this opportunity to thank all those who, through their professional and personal support for the success of this thesis. First I would like to meet my supervisor Ass.Prof. Dipl.-Ing. (FH) Dr.techn. Johannes Scholz, who enabled me to work on this topic. I appreciate that he always took the time to answer my questions and help me with constructive suggestions. A big thank you goes to my family and friends who have always supported me and are therefore an important part of this work. Finally, I would like to thank my fellow students and the scientific staff of the Graz University of Technology for making these years of study a wonderful and unforgettable time.

Contents

List of Figures

List of Tables

List of Abbreviations

ABM	Agent-based modeling
ABMS	Agent-based modeling and simulation
ABS	Agent-based simulation
AI	Artificial intelligence
GIS	Geographical information system
IBM	Individual-based modeling
ODD	Overview, Design concepts, Details
OOP	Object oriented programming
UML	Unified Modeling Language
ID	Identifier
GIScience	Geographic information science
API	Application Programming Interface
CPS	Cyber-Physical Systems
IoT	Internet of Things
PGM	Portable Gray Map
GUI	Graphical user interface

Chapter 1

Introduction

1.1 Motivation

In modern industry, production time is an important cost factor. Therefore, it is important to optimize the production process. Additionally, there is a need for higher efficiency in manufacturing processes. In order to do so, modeling and simulation of the production environment is a possible option. This topic is part of the field Industry 4.0. The term defines the convergence of communication, information and industrial production (Hermann, Pentek, & Otto, 2016). In Industry 4.0 humans, production assets and products communicate and cooperate with each other to optimize the production process. These types of production environments are called "Smart Factories" ("Plattform Industrie 4.0," 2018). The arrival of the Internet of Things (IoT)in production marks the beginning of the fourth industrial revolution. Machines and production plants have intelligence and communicate with each other and with people. They are able to make autonomous decisions and control themselves and each other. This allows them to act flexibly from changes in production and to choose alternative ways to achieve their goals independently. In addition, all data is available in real time. Industry 4.0 brings, and intelligent factories bring great potential into production (Henning, Wolfgang, & Johannes, 2013).
A special form of production is semiconductor production. The semiconductor manufacturing is very flexible due to the following reasons (Scholz & Schabus, 2014a):

- Depending on the product, the overall processing time can vary from several days to a couple of weeks.
- For one product to be produced, several hundred production steps are necessary.
- For the same production step, several different tools can be used. These tools can also vary in processing time and quality.
- There are many production assets for the different degrees of completion.

These properties make it difficult to optimize and analyze such a production process. In this work, such a production process is modeled and simulated to obtain information on the process based on the generated results and to optimize it subsequently. There are several strategies to simulate an environment. In this thesis, Agent-based modeling and simulation (ABMS) will be used to simulate a semiconductor manufacturing environment. Due to the fact that ABMS is very flexible, it can handle flexible manufacturing assets.
In this thesis, a semiconductor production is modeled and simulated under the aspects of Industry 4.0 using Agent-based modeling (ABM). The aim is to clarify the following questions:

- Are semantic agents capable of representing and modeling a semiconductor manufacturing environment?
- Is there an advantage of semantically enriched base data for simulation?
- Is decision support possible - is it possible to analyze the simulation results and draw conclusions to support decision making?

The answers to the questions will be given in this thesis, which is structured into seven chapters. The first chapter in Introduction gives an About the objectives and approach of the work. The second chapter gives a theoretical introduction to the topic Industry 4.0 and ABM. After this chapter, the reader has the necessary basic knowledge to understand the following approach. In the beginning, the knowledge from theory is converted into the concept. No technical details are explained, but the functionality and the idea are brought closer to the model. In the next chapter four, the technical implementation of the model takes place. Here the specific function mode of the individual parts is dealt with and brought closer to the reader. Then follow in chapter five the description of the data used for the simulation. This is followed by the results with which the scientific questions are to be answered. Finally, the last chapter in which the work and the results are summarized once again. This is followed by the outlook, which describes further possibilities with the project, such as improvements and extensions.

1.2 Goals

The main goal of this thesis is the modeling and simulation of a semiconductor production environment. This model is based on the aspects of industry 4.0 and smart factory. The modeling is agent-based, which means, among other things, the control of the production process is decentralized. In this worker, three central questions are to be clarified.

- The first question is about the model and the process of modeling itself. It should clarify whether agents are able to represent a semiconductor production environment. This is a central question because it contains the complete thesis. The clarification of this question begins with the literature study and thus has a decisive influence on the concept of the model. Finally, this question is answered by validating the model with real data.
- The second question is about the model. The aim is to clarify whether it is advantageous if semantically enriched data is available for the simulation. These can be data about the intermediate storage workers, machinery, products or the environment. The influence of such additional information will be clarified in the context of this work.
- The last question is to clarify which data can be generated from the model and how it can be used. Is it possible to analyze the simulation results and draw conclusions to support decision making? The decision making concerns the planning of production. This should optimize the production process and thus also increase production. The actual optimization is not the goal of these workers. However, it should be shown that the model provides data to perform an optimization.

1.3 Approach

The following items will point out, how the answers to the scientific questions can be found. They will also be the approach to reach the desired results:

Basic knowledge for the concept. Here the goal is to collect the necessary basic knowledge. The focus is on the knowledge about industry 4.0, smart factory and ABM. This includes the literature but also the choice of a suitable software environment for the implementation. The most important literature for the work is mentioned in Section 1.4. Also, the components of the model are worked out at this point. The description is still conceptual and will be elaborated in the following steps.

Agents and functions. The components of the model are specified more precisely in this step. Agents are worked out with attributes and properties. It is important to determine what the individual agents do and how they interact with each other. The structure of the simulation environment is also defined in this step. An important point here is the definition of how the agents communicate with each other and exchange information. This ends in the description of the model concept and serves as a template for its implementation. The simulation scenarios are also defined here. There are different simulation scenarios with the variant in movement speed of assets, machinery failures, and different production assets.

Implementation.The designed model is implemented in the selected software. This goes hand in hand with the verification and calibration of the model. The process is interactive, each model of the model is tested and its functionality verified. This is to avoid model and programming errors. This process ends with the validation of the model with real-world data.

Analysis of the results. Analysis of the simulation results to verify their usability for decision support. This is achieved by different test runs of the simulation with different settings. Evaluate the advantage of semantically enriched data for simulation with the help the ABM and simulation developed in the thesis.

1.4 Literature

This section gives a first impression of the relevant literature for this project. First, some basic knowledge about agent-based modeling and simulation is needed. The papers from Macal and North (2010, 2009b) present the concept and technical principles of agent-based modeling and simulation. Another important literature on ABM's general statement is the release of Crooks and Heppenstall (2012), which covers all the necessary basics. The paper of Gilbert (2007), Bonabeau (2001) also provides the basics of ABM. These publications are important for the necessary theoretical foundations of ABM. In the work of Abdou, Hamill, and Gilbert (2012), ABM is considered from the point of view of a user, it is about designing and building an ABM. Another paper about the design of ABM is the work of Grimm and Railsback (2012), Grimm et al. (2010). It presents a protocol describing ABM. For an overview of possible software packages for creating an ABM, see the release of Crooks and Castle (2012) and De Smith, Goodchild, and Longley (2015). Also Gilbert and Bankes (2002) deals with this topic. Another important part of literature dealing with a focus on Agent-based Systems

for manufacturing Monostori, Kumara, and Váncza (2006), Negmeldin and Eltawil (2015) and Leitão (2009). This work is important for the conception of the model, which is created in the context of this thesis.
The second important part of the thesis and an essential part of the project is Industry 4.0. The article by Drath and Horch (2014) and Albach, Meffert, Pinkwart, and Ralf (2015) contains an overview of digitization and industry 4.0. Another basis for the work is provided by the paper of Hermann et al. (2016), in which the basic principles and the structure of an industry 4.0 scenario are presented. The basic structure of a Smart factory and an example are also presented. Another important term is IoT this topic is covered in the work of Mattern and Floerkemeier (2010). Another very important literature is the publication of Shrouf, Ordieres, and Miragliotta (2014). It explains the topic of industry 4.0 but above all the term smart factory.

Chapter 2

Theory

This chapter provides the theoretical overview of Industry 4.0 and Agent-based modeling (ABM). An introduction to the Industry 4.0 and Smart factory can be found in Section 2.1. An introduction and description of the basic features of ABM can be found in Section 2.2.This is followed 2.2.4 by an overview of the most important basic elements of this modeling approach. Section 2.2.8 shows a method for correctly describing an ABM. Section 2.2.9 provides an overview and comparison of the current ABM platforms. To conclude this chapter, ABM is considered in section 2.2.10 in the context of Geographic information science (GIScience) and a Geographical information system (GIS).

2.1 Industry 4.0

This chapter deals with the term industry 4.0, followed by an introduction to the basic topics covered in the thesis.

2.1.1 Definition

The term Industry 4.0 stands for the fourth industrial revolution. Figure 2.1 shows the development of the industry after the introduction of mechanical production plants using the water and steam power (first revolution), the introduction of mass production by means of electric energy (second revolution), the use of electronics and IT for automation (third revolution), The fourth revolution is characterized by networking and communicating systems using the latest internet technology (Roth, 2016). Figure **??** shows a framework for Industry 4.0 as defined by PwC (2016), but it can be seen that a variety of technologies are involved in building an Industry 4.0 environment. The most important basis is the data analysis. The data comes from different technologies, such as sensors, mobile devices and many more. The interaction takes place through the digitization of services, products, production and the entire value chain (PwC, 2016).

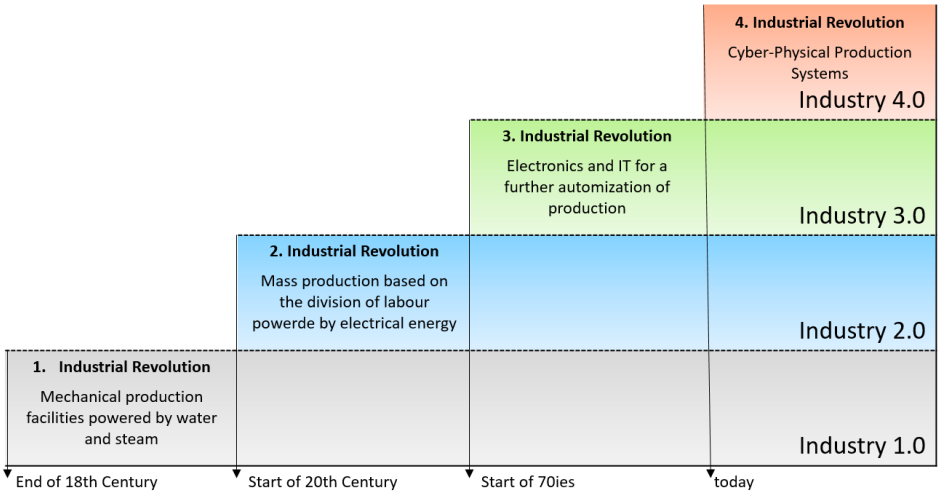


Figure 2.1: The four stages of industrial revolutions. (modified from (Spath, Gerlach, Hämmerle, Krause, & Schlund, 2013))

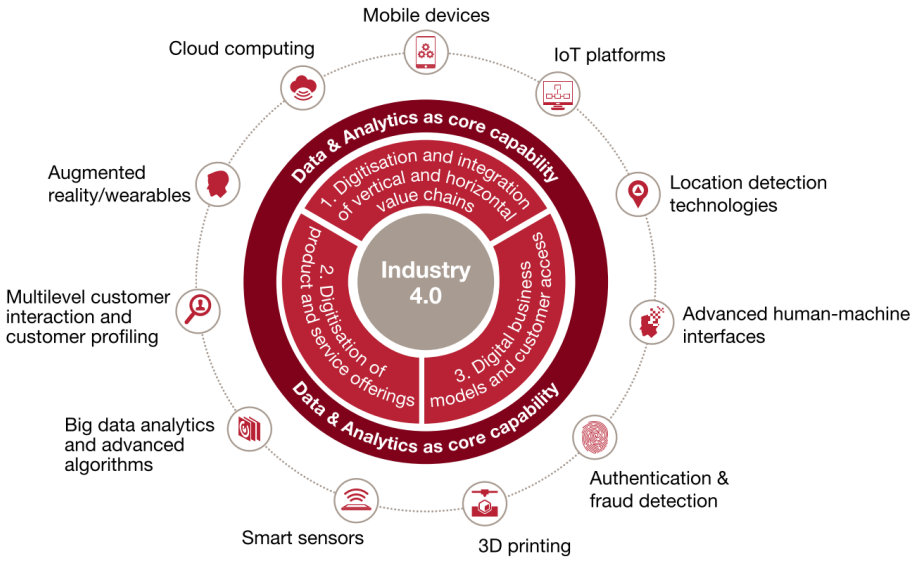


Figure 2.2: Industry 4.0 framework (PwC, 2016)

The fourth Industrial Revolution is characterized by communication between people, machines, and resources. Through this networked communication, one can speak of a paradigm shift from centrally controlled to decentralized controlled production processes. Intelligent products know their production history, their next production step and use this knowledge to steer through the production process by giving machines instructions and the transport

system the goal for the next work step (Albach et al., 2015). There have been many publications on the term Industry 4.0, in the work of Hermann et al. (2016), Internet of Things (IoT), Cyber-Physical Systems (CPS), Cloud Computing and Smart Factories have been defined as the most important components. These terms are treated below to better understand the idea behind the term Industry 4.0.

2.1.2 Internet of Things and Cyber-physical systems

The IoT is the key to the Fourth Industrial Revolution (Henning et al., 2013). There are several definitions for the term IoT, (van Kranenburg & Dodson, 2008) defines the IoT as a global, dynamic network structure with self-configuring capabilities, in which physical and virtual "things" have identities, physical attributes, and virtual personalities, and intelligent interfaces are integrated into an information network. That means, things use the internet to communicate and share information. These things exchange data and create opportunities for more direct integration of the physical world into computerized systems. This should lead to efficiency improvements, economic benefits, and reduced human intervention (Mattern & Floerkemeier, 2010). Figure 2.3 shows the parts of an object in IoT. These are objects that are equipped with sensors and actuators. The resulting system is able to collect, process, and store data to affect itself or the environment. This turns objects into smart objects and environments into smart environments. If one connects such an embedded system to each other or the Internet and makes its data and capabilities available as online services, the result is a digital revaluation of an object. By expanding with digital features, a physical object turns into a CPS. Such a CPS automatically collects data about their real environment and digital processes As the number of data increases; cloud computing can help. The data is then stored in a central location. This large amount of data can then be analyzed to gain new information (Albach et al., 2015). Thus, a CPS can communicate as components with mechanical and electronic parts that communicate over a data infrastructure, such as the internet. In a manufacturing environment, such cyber-physical systems may be, for example, intelligent machines, storage systems, and manufacturing facilities that can exchange information, initiate actions, and control each other. This results in an improvement of the complete industrial process of a company. This includes areas such as manufacturing, supply and lifecycle management, and materials management (Henning et al., 2013). In the Internet of Things, Data, and Services, each device can exchange information with any other device or person anywhere in the world.

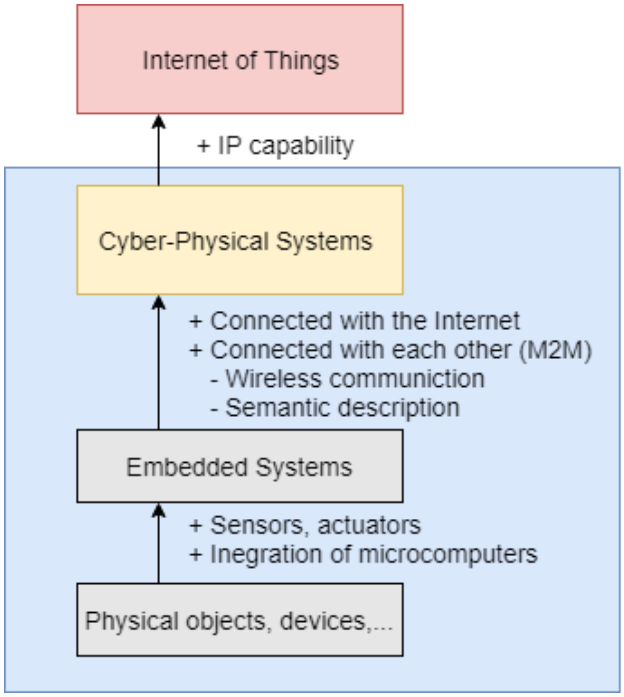


Figure 2.3: Schematic structure of an element in the IoT (modified from Albach, Meffert, Pinkwart, and Ralf (2015).

2.1.3 Smart factory

A term that is mentioned in connection with Industry 4.0 is the smart factory. Smart Factories are a key feature of Industry 4.0. They are created through the use of CPS and IoT technologies in the production of goods (Hermann et al., 2016). By using smart products that are identifiable, knowing their history, knowing where they are and where they need to go, there is a new approach to production. Figure 2.4 shows a concept for a smart factory in Industry 4.0.

All production processes in the company are networked and have their own intelligence. Machines communicate with each other but also with people and exchange data. Using IoT and CPS creates a dynamic production environment with real-time communication. Smart engineering means that the collected information is used in product development, production and sales. Logistic is self-organized to dynamically react to bottlenecks and failures. All collected data is analyzed to make the production process more efficient (Shrouf et al., 2014).

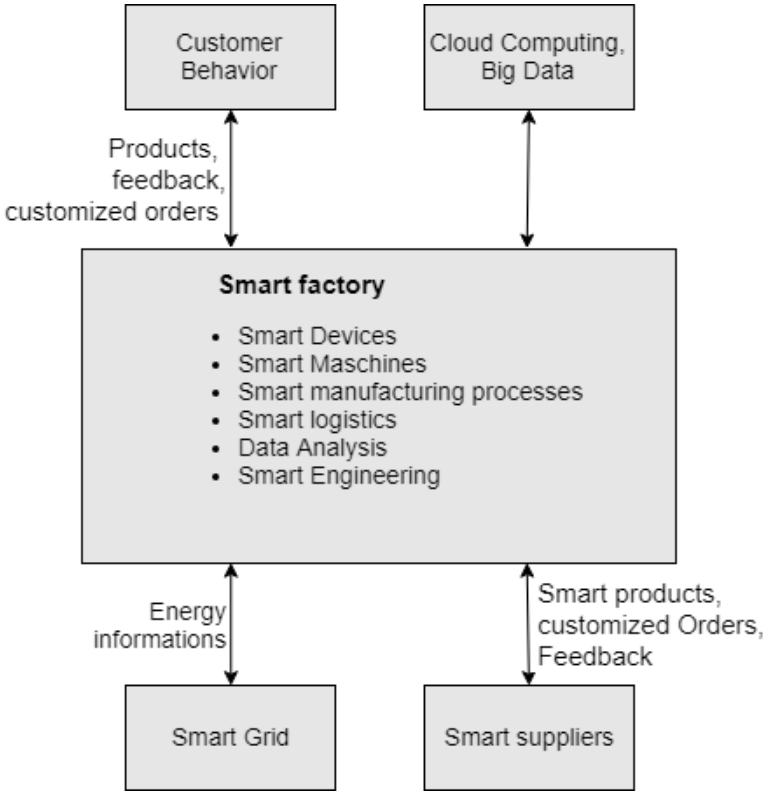


Figure 2.4: Reference architecture for a smart factory (modified from Shrouf, Ordieres, and Miragliotta (2014)

Exemplary application

The following example shows how an Industry 4.0 scenario could look like. Hermann et al. (2016) describes in his paper a possible application in Industry 4.0 as support for a transport system. The following state is assumed:

- Employees order transport requests by telephone
- Human-driven transportation vehicles.
- Transport planning is carried out centrally and is assigned manually to the vehicles.

From the description of the situation it can be seen that this system acts cumbersome. And adaptation to new situations is difficult. In an Industry 4.0 scenario, the same scenario would run as follows.

- By using a multi-agent system autonomously transport orders to be distributed.

- Self-optimization leads to efficient use of resources.
- Use the smart phone for transport orders
- Vehicles are Autonomous guided.
- Structure as a modular system.

To improve the situation, a multi-agent system is used. Multi-agent systems are computer systems in which autonomous agents to achieve common goals. Each agent makes its own decisions. This is a decentralized control. Agent systems are discussed in detail in the following chapter 2.2. An important prerequisite for this scenario is the communication flow. Automated vehicles can easily be connected via communication technologies. However, humans need a user interface for communication. An important basis for the implementation of such a scenario is the ability of all actors to share and receive information. Such as the position and the order of each vehicle. Since each vehicle knows the other vehicles about others and also knows the open orders, it can independently carry out open work. Thus, a decentralized system is created.

2.2 Agent-based modeling and simulation

Agent-based modeling and simulation (ABMS) is an approach to simulate complex processes. It is a quite novel approach with increasing attention over the past decade. In publications ABMS is also known as Agent-based simulation (ABS), ABM or Individual-based modeling (IBM). ABMS is connected to fields like computer science, system dynamics, management, science, social science, Artificial intelligence (AI), robotics and many more. The most common use of ABMS is to model and simulate human behaviors with individual decision-making, to represent social interactions, group behavior, and collaboration (Macal & North, 2010, 2009b).
Agent-based modeling and simulation is a computational approach, to model and simulate dynamic processes with one or more autonomous agents. In such a model, agents are interacting among themselves and with a simulation environment. Agents can represent humans, vehicles, tools, geographical entities like an area or other abstract entities (Macal & North, 2013). ABMS is a bottom-up approach, this means that individual agent decisions and actions affect the resulting system (Crooks, Castle, & Batty, 2008). With this approach, it is possible to model the dynamic behaviors of multiple individual agents in an artificial world. Each agent has behavior rules. Agents may influence other agents and change the environment by their behavior. (Crooks & Heppenstall, 2012).

To model and simulate complex behaviors are just one part for ABMS another is to capture the emergence. Emergence is an important behavior which is not explicitly modeled. It appears because of agent interactions in the running model. This output is essential for further research of the model (Macal & North, 2013). ABMS simulations are depending on time. The simulation is continued until a termination condition, or a final condition is met. The simulation is often a process over a timeline, activity-based, steps over time or a discrete-event simulation (Macal & North, 2010, 2013).

2.2.1 Areas of Application

Agent-based models are used for a wide variety of topics and specialist work. However, all models have in common that self-acting objects are modeled. These objects all have their behavior and rules. Among other things, the models themselves can be different in the modeling of physical or social space (Crooks & Heppenstall, 2012). According to Macal and North (2008) and Balietti (2012), the application of ABM is based on various topics. Table 2.1 lists some of these areas. The list is not complete and shows only a cut of possible applications.

2.2.2 Advantages of agent-based modeling.

In literature, there are two different types of ABM, either with own advantages and disadvantages. De Smith et al. (2015) separates ABM in an exploratory and a predictive model. In the exploratory modeling approach, the ABM is used to understand the observations, phenomena, and processes. The focus is on a specific part of the system. This will create conditions to understand individual phenomena better. Important to mention, in this approach, it is not intended to simulate future conditions, but the focus is on the understanding of observation. The potential disadvantage of this approach is the lack of analytical methods for empirically evaluating ABM results. The second is the predictive modeling approach. Prediction models are used to extrapolate trends, predict future conditions and evaluate different simulation scenarios. These models are designed to mimic the real world or systems. Due to changes in the rules of behavior and/or initial states, effects are observed and evaluated.
In literature, there are three advantages of ABM over other modeling techniques (Bonabeau, 2001; De Smith et al., 2015).

Table 2.1: Areas of ABMS Application (Macal & North, 2008; Balietti, 2012).

Supercategory	**Application**
Biology	Population dynamics
	Animal group behavior
	Ecological networks
Crowds dynamics	Pedestrian movement
	Evacuation modeling
Infrastructure	Transport/traffic dynamics
	Hydrogen infrastructure
Economics	Artificial financial markets
	Trade networks
Business and Organizations	Manufacturing operations
	Supply chains
	Consumer markets
	Insurance industry
Society and Culture	Ancient civilizations
	Organizational networks
	Civil disobedience
	Social conflicts
	Migration

- **ABM captures emergent phenomena**
  It is not possible to model emergent phenomena directly. In ABMS the whole is more than the sum of all parts because every part has own behaviors and properties. Emergent is a result of this individual behavior and interaction with other individual system party and is an important part and result of an agent-based model. The whole model is a result of interaction form independent model parts through the so-called bottom-up approach (Bonabeau, 2001; North, 2014).

- **ABM provides a natural description of a system**
  ABM is usually the most natural approach to describe and simulate a system of discrete entities, because there is a correspondence between the real world and the agent-based model. Objects of the real world are described as ontologically appropriate agents in the form of program code. This property is called ontological correspondence With ABM objects will be observed instead of variables(Gilbert, 2007). The behavior of the subjects to be modeled and facts are usually complicated. Such a nonlinear discontinuous behavior is difficult to describe with conventional methods such as differential equation systems. The ABM approach makes it easier to describe complex structures. (Bonabeau, 2001), (De Smith et al., 2015). With ABM complex systems are modeled form bottom up in the form of individual objects with their own attributes, behavior, relations, and properties. The individuals with their activities are at the center of this approach, and this is a more natural way than describing system processes (Bonabeau, 2001). ABM is a natural way of describing the simulation environment. The environment again consists of entities that can represent the real world. Therefore, ABM is well-behaved to simulate people and their environment (De Smith et al., 2015), since the environment can also be implemented with physical barriers or resources, which can affect agent behavior (Gilbert, 2007). Furthermore, agents can have the ability to learn about other agents and their environment (Abdou et al., 2012). The ability to present moving agents makes behaviors easier to understand for the viewer. To model such complex ABM systems, Object oriented programming (OOP) is used, because it is are crucial to agent-based modeling. In OOP languages, objects with their own attributes and methods are used, like agents in ABM. Therefore, almost all ABS models are programmed in OOP languages, and ABM software toolkits are built on such a programming language (Abdou et al., 2012).

- **ABM is flexible**
  With ABM, it's easy to add more agents or change attributes. It's also easy to get the agents into a different environment. All of this can change the behavior of the system. As a result, small changes can change the behavior and complexity of the entire model. This feature of ABM makes this model approach very flexible (Bonabeau, 2001), (De Smith et al., 2015). Another possibility, which the model simply changes is to change the behavior of the agents or to experiment with the agents in groups. Thus completely new points of view on a topic can be realized (De Smith et al., 2015).

2.2.3 Limitation of agent-based models

ABM offers the possibility to model and simulate complex systems, but also has some limitations. An important point to note is the purpose of the model. The model is only as useful as the purpose for which it was created (De Smith et al., 2015). As with other modeling techniques, it is important to choose the right level of detail for the desired purpose. (Couclelis, 2000). Another difficulty is the modeling of systems themselves. So it is often difficult to implement the behavior, attributes, and interactions because this information is often difficult to quantify from the real world. Therefore, the modeling behavior is often difficult to calibrate and to verify (De Smith et al., 2015). With ABM complex systems and conditions can be modeled. As a result, the validation and verification of the model are similarly complex and requires a lot of time (Cooley & Solano, 2011). In addition, agent-based models can be harder to analyze, understand, and communicate than traditional analytic/mathematical models. Another factor is the computing power, the high computational effort of ABM remains a limitation in the modeling of large systems (Grimm, 1999) (De Smith et al., 2015). Another factor is the emergence. As mentioned, this is not modeled directly but arises from the system behavior of the model. The emergence and the resulting behavior is often difficult to understand and convey. That's because it's not based on a mathematical/analytic model. Agent-based models can be very sensitive to initial conditions and changes in the interaction rules. Therefore it is necessary to carry out several simulations runs with different initial data. To evaluate the robustness of the results (De Smith et al., 2015).

2.2.4 Structure of an agent-based model

A typical agent-based model consists of the following three elements (Macal & North, 2010):

- A set of agents with attributes and behaviors
- Agent relationships and rules, which define how other agents and the environment behave with the agents
- An environment with which the agents can interact

Another important part of an ABMS is a scheduler. The scheduler brings a time component into the simulation. It is responsible for the chronological succession in the simulation. The time counter of the simulation usually increases of one unit each step. In each step, the scheduler calls all agents

in a consistent or randomized order. The agents can also be called by intern rules to interact of discrete events (North, 2014).

2.2.5 Agents

There is no precise definition in the literature for the term "agent". The exact definition of an agent does not go far beyond the property of autonomy. Therefore, some authors refer to any form of one autonomously acting component as agents (Macal & North, 2010). An agent can be any actor who can influence himself, other agents, or the environment (North, 2014). Most agents are defined by their properties. So sign (Gilbert, 2007), (Bonabeau, 2001) Agents as actors with the following behaviors, they can share information and make actions and decisions dependent on this information. In addition, agents can reproduce, adapt and learn their abilities (Gilbert, 2007), (Bonabeau, 2001). Another definition comes from Wooldridge and Jennings (1995). He define the agent by using a list of properties. Much of these properties are found in agents in a variety of ABM models. This approach was picked up and extended by De Smith et al. (2015), Crooks and Heppenstall (2012), and Macal and North (2009b). These defined properties are explained in the following list.

- **Autonomy:** Agents are autonomous units, meaning they do not need centralized control. They can independently process information and exchange this information with other agents. With this information, they can make independent decisions. You can communicate and interact with other agents. Their behavior can influence the simulation and other agents.
- **Heterogeneity:** This means that the agent represents a single-object with its own properties and attributes. Thus, for example, no average agent must be defined. Each agent is a separate autonomous individual. However, there may be groups of agents, but these groups arise due to the bottom-up approach through the union of similar autonomous individuals.
- **Active:** Agents are active and can influence the simulation. Activity is a key factor in ABM and an agent can have the following characteristics
  - **Pro-active/goal-directed:** Agents have goals that they pursue. The agent tries to reach this goal or goals with his behavior.
  - **Reactive/perceptive:** An agent can be designed to perceive its environment. In this context, it is also possible that an agent

has prior knowledge of its environment. This may be reflected in the form of boundaries, obstacles and destination, or in the perception of other agents. This knowledge can be implemented in the form of a map. This allows the agent to avoid obstacles and find goals. Furthermore, this knowledge can also reflect the perception of other agents.

- **Bounded rationality:** Agents can have a kind of "limited" rationality, which is a consequence of their heterogeneity. By doing so, the agents are able to make independently adaptive decisions about their goals based on their attributes and goals. This form of ABM model is often found in the context of social sciences related to rational-choice paradigm.
- **Interactive/communicative:** Agents have the ability to share information with other agents in their neighborhood.
- **Mobility:** Agents can move the space (environment), in with they are situated, of a model. But agents can also be stationary. It is thus possible to model any form of moving objects or dynamic behaviors.
- **Adaptation/learning:** Agents can be defined to have memory over old states. As a result, it is possible that the previous state affects the current state of the agent. This gives the agent the ability to learn. Agents can also be equipped with the ability to develop their own functionality.

Not all of these properties apply to each agent. It is, therefore, possible for an agent to have only a few of these listed properties. The importance of the individual properties also varies depending on the application. There can also be different agents within a model, which differ in their characteristics. The agent can represent any kind of entity. (Crooks & Heppenstall, 2012; Macal & North, 2009a). Figure 2.5 shows an illustration of an agent. It contains the basic elements of an agent, which are attributes and methods. Attributes can be fixed static or dynamic, which means that they change during the simulation. Through behavioral rules, the agent interacts with other agents or with the environment, which in turn can affect attributes as well as behavior.

2.2.6 Agent behavior and relationships

ABM is concerned with the modeling of agent relations and interactions and the modeling of agent behavior. This means that normally only local information is available. There is no central control unit which collects and passes on the information of all agents. (Macal & North, 2010). Each agent has rules, behaviors, and thus the ability to influence other agents and their environment. This is shown in Figure 2.5. The rules are typically built on literature, expert knowledge, data analysis, or numerical work, and the rules generated from them are the basis for the agent's behavior. On the one hand, the rule can apply to a specific agent as well as to a group of agents (Crooks & Heppenstall, 2012). Figure 2.6 shows a schematic interaction between agents. Conditional statements (if-then-else) and limits are used as the simplest implementation of a set of rules.

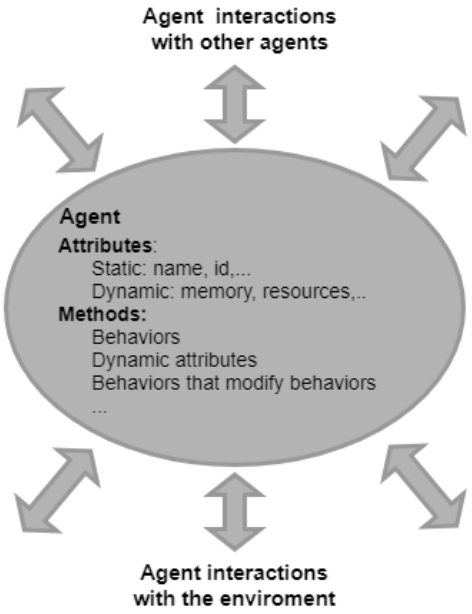


Figure 2.5: Illustration of an agent (modified from (Macal & North, 2010))

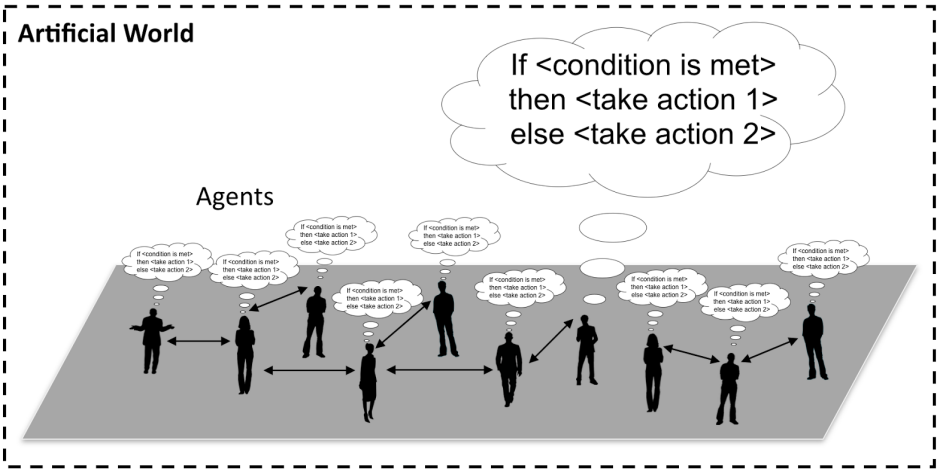
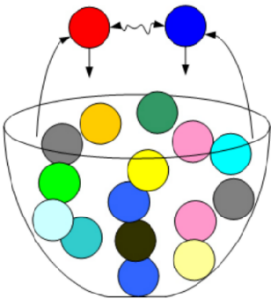


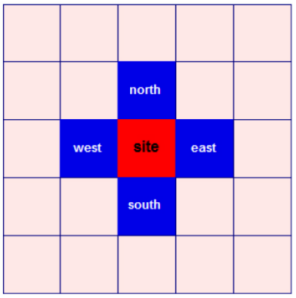
Figure 2.6: Example of interaction between agents. Each agent uses defined rules to interact (Heppenstall & Crooks, 2016).

The complexity of an agent's behavioral rules varies and depends on how much information is available. An example of a complex behavior is when the agent changes behavior due to past events (Macal & North, 2009b). As mentioned in the example, agents can also make decisions based on memories. Such learning processes are based on rules that are adjusted, based on previous results, but they can also be used for complicated algorithms (De Smith et al., 2015). Each behavior is based on a set of rules, how a certain behavior is triggered differently. The behavior can be triggered by a specific action or done on a schedule (Crooks & Heppenstall, 2012). Every agent has a set of rules based on his behavior and reacts to his environment. However, it still has to be described how information is exchanged. The relationship or spatial relationship between agents is called topology. It describes how information is exchanged and how actions can be triggered. Macal and North (2009b) describes the most common typologies for an ABM, the different types are shown in Figure 2.7.

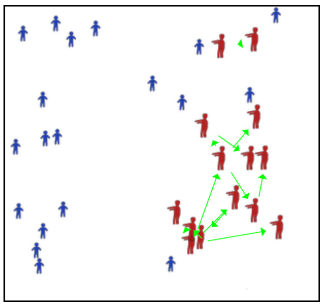
In the "soup" or aspatial model (see Figure 2.7(a)), there the agents have no position, and the model has no spatial reference. Agents are usually randomly selected to interact, and after an interaction, the agents disappear into the soup again. The Cellular Automata topology, shown in Figure 2.7(b), is a grid-based relationship. The agents move from grid cell to grid cell and examine their grid cell neighborhood. Generally, no more than one agent occupying a cell at a time. The neighborhood is represented here by adjacent grid cells using Von Neumann neighborhood, Moore neighborhood or their offshoots. The usage of a Cartesian coordinate system is shown in 2.7(c), the agents move in two, three, or more-dimensional space. Neigh-



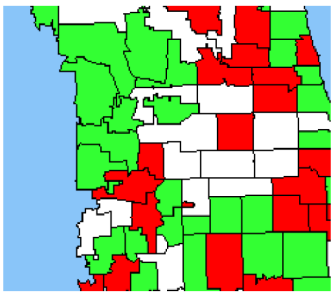
(a) "Soup" Model, agents without spatial referencel



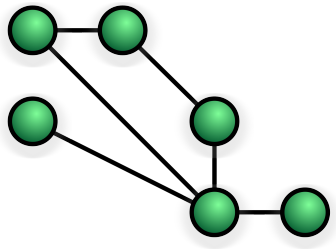
(b) Cellular Automata with grid cell environment using the Von Neumann neighborhood as an example.



(c) Euclidean topology in a 2-, 3- or more-dimensional Euclidean space.



(d) GIS-topology with a geo-referenced environment allows spatial relationships .



(e) Network topology: The nodes represent the agents and the edges represent the relationships. The nodes can exist with or without spatial reference.

Figure 2.7: Basic forms of environment and topology in ABM (modified from Macal and North (2009b))

borhood relations here refer to the Euclidean distance between the agents. In the Geographic Information System GIS topology, agents move over a geo-spatial landscape 2.7(d). Relationships can be described via the topological operators like touch, inside, disjoint, etc.. Networks make it possible to define the environment of an agent more generally and sometimes more precisely. In a network topology 2.7(e), networks can be static or dynamic. For static networks, links and nods are predefined and will not change in the model. For dynamic networks, links and nods can arise and disappear according to the mechanisms contained in the model. Regardless of which topology is used in an agent-based model, for connecting the agents, the key idea is local interaction and local information transfer between the agents (Macal & North, 2009b).

2.2.7 Environment

Gilbert (2007) defines the environment of an agent as the virtual world in which he acts. Crooks and Heppenstall (2012) and Macal and North (2010) describe the environment as a space that forms the basis for an agent to act or interact with other agents. The environment can be designed much like an agent, with the difference, that it does not require the ability to interact with the environment (Abdou et al., 2012). Depending on the model, agents can change the environment through their behavior (O'Sullivan, Millington, Perry, & Wainwright, 2012). The environment can, in the simplest case, be used to provide the agent with information about his neighborhood. However, as in any GIS, extensive geographic information may be provided to the agents. The environment can restrict their actions through their design actions. Thus, depending on the position of the agents, or other rules could be used. For example, speed restrictions or hints may appear (Macal & North, 2010). In general, the environment is a geographic space with physical features. Such models are considered spatially explicit. But there are also models in which an abstract environment is used, such as modeling a knowledge space (Gilbert, 2007).

Figure 2.7 shows different types of an ABM environment: An Environment without spatial reference is shown in 2.7(a), this environment is mostly used for abstract models. The grid-based environment 2.7(b), defines the position of agents in a grid. In Euclidean space 2.7(c), the agents act in a two or more dimensional coordinate coordinate system. Geo-referenced data (2.7(d)) may also be used for the environment. It is also possible to link ABM with a GIS. Another possibility is to create the environment as a network (2.7(e)). It will differentiate between networks with and without coordinately known nodes. The choice of suitable topology for the environment is dependent on the model. The topology is important, and thus the environment defines how agents interact locally and exchange information. The different types can also be combined as desired (Macal & North, 2009b).

2.2.8 Design an agent-based model

Agent-based models usually consist of different agents. Each agent has its attributes, rules, and behavior in the environment. Due to this fact the description and communication of an ABM is difficult. A detailed description of an ABM is usually incomplete. Therefore, Grimm and Railsback (2012) have developed a standardized format for describing ABM, the so-called Overview, Design concepts, Details (ODD) protocol. The ODD protocol attempts to create a generic format and structure to describe and document

an ABM. The aim of the ODD protocol is to fully describe an ABM as well as to simplify the writing and reading of a model description (Grimm et al., 2010). In ODD, an ABM is displayed hierarchically. First, an overview of model structure and processes. Then follows a checklist in which the modeler shows his design decisions. Due to this, the design should become better understandable. Details and processes are described at the end of the ODD protocol (Grimm & Railsback, 2012). Table 2.2 shows the structure of the ODD protocol.
As mentioned above the first part gives an overview of the basic model structure. Each model is based on a question, a problem or a hypothesis. Therefore, the first point of the ODD protocol is a summary of the model's goal. It is important not to describe how it works, but what it is used for. This is followed by the description of entities, state variables, and scales. This includes all agents with their attributes and main variables as well as the environment. The scale provides information about the spatial and temporal resolution of the model. In the process overview, the basic processes are described. This takes the form of keywords such as "work", "wait", "transport". The detailed description of this process will follow later in detail. Here only the expiry of the model is described. The process of the model is controlled by the scheduler as described in chapter 2.2.4. The order, in which the scheduler calls the agents, performs the process and updates the variables, is described in this section (Grimm et al., 2010). The second part of the ODD protocol is the design concepts. It will use a checklist to understand why the ABM was designed in this way. The design concept is defined by the following ten points (Grimm & Railsback, 2012):

- **Emergence**: Which results of the model are caused by the adaptive behavior of agents. Are these results based on rules and are therefore predictable?
- **Adaptation**: Can the agent adjust his behavior over and how are these actions triggered? Is this behavior influenced by your own action or by the environment?
- **Objectives**: Does the agent have a goal which he wants to reach? If an agent has one goal, how to determine whether the objective is achieved?
- **Learning**: Does the agent change his behavior based on experience and how does it work?
- **Prediction**: Can the agent predict future scenarios to make decisions? Which rules or models are available for this?
- **Sensing**: What information can agents perceive and collect? Will this information be considered for their adaptive behavior and how?

Table 2.2: The elements of the ODD protocol (Grimm & Railsback, 2012)

ODD	ODD element	Questions to be answered
Overview	1. Purpose/goal	What is the goal of the model?
	2. Entities, variables and scale	Which entities are in the model?
		Which attributes have these objects?
		What spatial and temporal resolution is the model?
	3. Process overview and scheduling	Which objects do what and in which order?
		When are variables updated?
		How is time modeled - as steps, continuously, discrete events?
Design concepts	4. Design concepts	see list design concepts
Detail	5. Initialization	What is the initial state of the model?
	6. Input data	What external data is the model using and how can you change the course of the model?
	7. Submodels	What are the parameters of the partial models, and are there reference values?
		What are the submodels that represent the processes in detail?

- **Interaction**: How do agents interact with each other and with the environment?
- **Stochasticity**: Are there random or partially random processes? How do these processes affect the system?
- **Collectives**: Are there aggregations, such as group formation, for agents. Do these groups cause emergence or are they triggered?
- **Observation**: What data must be recorded during the simulation to understand the results so that it can be analyzed?

Describing a model does not always require all of the listed concepts. Parts that are not needed for the description can be omitted. However, concepts such as: *emergence*, *interaction*, *randomness*, and *perception* are emerging in every model (Grimm & Railsback, 2012). The last part of the ODD protocol contains a detailed description of the model. It first describes the initial status of the model. As mentioned in the description of ABM, small changes in the initial state can have an impact on the simulation results. Therefore, it is important to define the initial state. The next item is the description of external data sources which will be used during the simulation or initialization, as well as a description of how the data will be exchanged. In the last part, the section "Process overview and scheduling" describes in detail which processes exist and how they are tested. The detailed description of the ODD protocol can be found in the paper Grimm et al. (2010) or Grimm and Railsback (2012).

2.2.9 Software and Toolkits

There are a lot of toolkits related to ABM ("Wikipedia- Comparison of agent-based modeling software," 2018). Nikolai and Madey (2009) give their work an overview of the existing toolkits and organize them according to various criteria. Multi-platform support (Microsoft Windows, Mac OSX, and Linux) is common, and much of it uses the Java and C++ programming languages to implement the models. The other toolkits use proprietary logo dialects as well as visual programming languages, similar to, e.g., Unified Modeling Language (UML) diagrams. Therefore, all toolkits require a basic knowledge of programming. Mainly object-oriented programming languages are used OOP. Thus, the required level of modularity for implementing agent-based simulation can also be achieved (Crooks & Castle, 2012), (Nikolai & Madey, 2009). There are free tools as well as commercially distributed solutions. The intended purpose can range from multifunction tools to specific applications (Nikolai & Madey, 2009). The most notable representatives are: Repast, Swarm, NetLogo, AnyLogic and MASON (Macal & North, 2009b), (De Smith et al., 2015), (Crooks & Heppenstall, 2012). Table 2.3 compares the five toolkits. These are discussed in the following sections.

Table 2.3: Five ABM toolkits in comparison (De Smith, Goodchild, & Longley, 2015; Crooks & Heppenstall, 2012)

Name	Repast	Swarm	MASON	NetLogo	AnyLogic®
Website	https://repast.github.io/index.html	http://www.swarm.org	https://cs.gmu.edu/~eclab/projects/mason/	https://ccl.northwestern.edu/netlogo/	http://www.anylogic.com/
Developers	University of Chicago, Department of Social Science Research Computing, USA	Santa Fe Institute/SWARM Development Group, USA	Laboratory and Center for Social Complexity, George Mason University, USA	Centre for Connected Learning and Computer-Based Modelling, Northwestern University, USA	XJ Technologies, Russia
Date of inception	2000	1996	2003	1999	2000
Language	JAVA (Repast Simphony),C++ (Repast HPC), Phyton	Objective-C, JAVA	Java	Proprietary scripting	Java, proprietary scripting
Tutorial with source code	yes	yes	yes	yes yes	
Charting, graphing, statistics	yes (integrated & external Java libraries)	yes (R- and S-Plus statistics packages)	yes	yes	yes (integrated & external Java libraries)
Programming experience	Strong	Strong	Strong	Basic	Moderate
GIS functionality	yes (ArcGIS, OpenMap, Java Topology Suite...)	Yes (e.g. Kenge GIS library for Raster	GeoMason-Extension, third party GeoTools	NetLogo-Gis Extension	support for shapefiles
Manual, mailing list	yes	yes	yes	yes	yes

2.2.10 ABMS in GIScience

Agent-based systems are used for experimentation and exploration of ideas. ABM is strongly influenced by the developments in programming, data processing, and interface design and therefore gains in importance. (De Smith et al., 2015). The use of ABM in GIScience for spatial sciences is a relatively new methodology, although it has long been on the research agency. The spatial sciences, with focus on the representation of space, are an important part of the geographic research. In spatial science, spatial properties of phenomena are usually measured, analyzed and presented (Torrens, 2010). For processing and displaying spatial data, GIS systems are a useful medium, but such systems are not well-behaved to handle dynamic modeling. With the help of ABM, however, it is possible to model dynamic processes, which is why the GIS and ABM are linked (Crooks & Castle, 2012). The problem with GIS and dynamic processes is time. Often, the data is not continuously available for the desired period, so the missing period must be modeled. In this case, ABM can be of help, because unlike GIS it is designed to model dynamic processes (De Smith et al., 2015; Crooks & Castle, 2012). ABM is used for a variety of geographic applications, but space, spatial thinking has emerged as the central focus of ABM (Torrens, 2010).

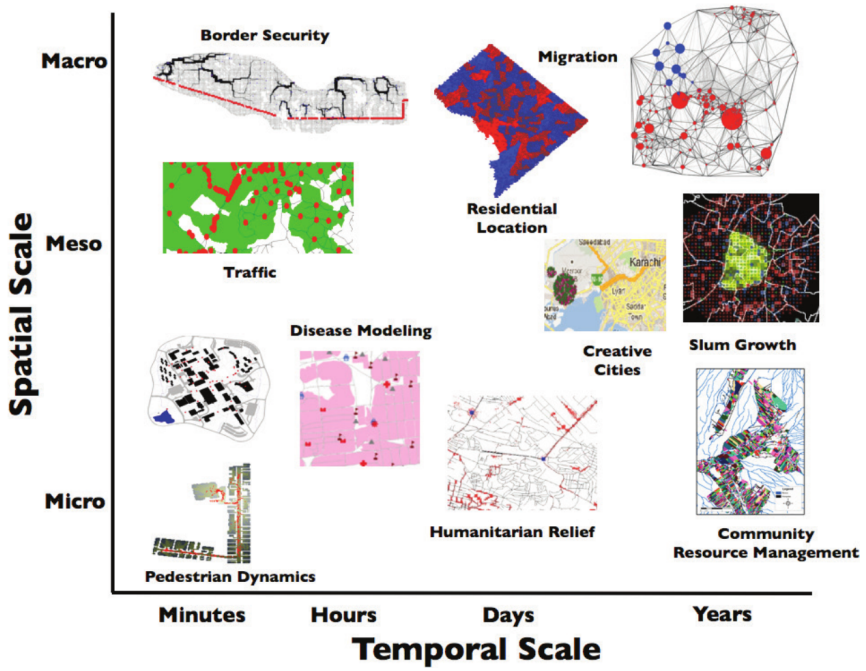
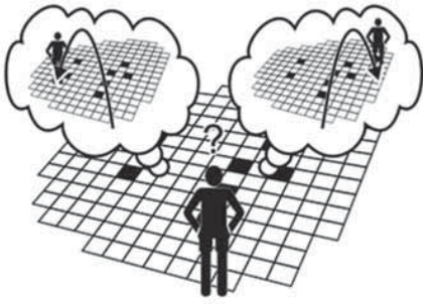


Figure 2.8: ABM applications with different spatial and temporal scale (Crooks, 2009)

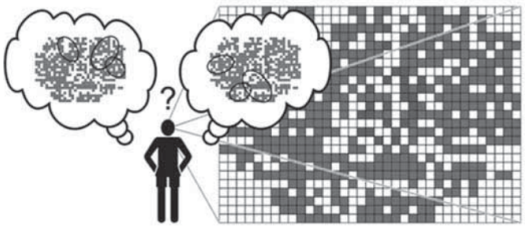
(a) Pedestrian/mobile agents



(b) Residential agents

(c) Hunter-gatherer agents

(d) Farmer/land-use change agents



(e) Property developer agents

Figure 2.9: Schematic representation of common ABM problems (O'Sullivan, Millington, Perry, & Wainwright, 2012)

Figure 2.8 shows applications of ABM with different spatial and temporal characteristics. Applications range from small-scale events such as evacuations to large-scale simulations of traffic or migration. Also, the time scale can vary from a few hours to years (resource management)) (Helbing, 2012). In the work of O'Sullivan et al. (2012), agent types are distinguished according to the type of spatial problem. The different spatial issues are shown in Figure 2.9 and the agent types are described below. It is a general overview of possible problems and thus only a part of the possible spatial problems.

Pedestrian/mobile agent (figure 2.9(a)): This mobile agent can reach a specific destination. To do this, he interacts with his environment and with other agents. For example, the environment may be represented in the form of a road network or a building geometry. The position and the resultant environment of the agent is the most important reason for the demerit for his next step.

Residential agent (figure 2.9(b)): Also this model is about the movement of the agent. The agent is looking for a new location, which, he prefers to the current one, due to his rules. However, the movement is not continuous as in the case of the pedestrian-agent, but it simply moves to the new position. In this model, the nature of the environment does not affect the movement, but other agents influence the decision.

Hunter-gatherer agent (figure 2.9(c)): In this model they combined two previous types. The agent is trying to tap resources based on information from the environment. Whether the agent moves on, is dependent on how many resources are available. The movement is the same as with the pedestrian agent. The target search as in the residential agent. In this model, the actions of the agent have a direct impact on the environment.

Farmer agent (figure 2.9(d)): This agent type has an impact on its environment. He interacts with his environment and can, therefore, change his relationship with the spatial environment. This relationship change takes place in which he or she manages resources, such as harvesting, selling or expanding land, with the aim of gaining many resources. It is unlikely that the agent moves.

Property developer agent (figure 2.9(e)): As with the former farmer agent, this agent gives the environment and other spatial objects a voice. However, this agent can evaluate complex spatial relationships to make decisions. This form of an agent is used, for example, to simulate the real estate market or to simulate urban growth.

Chapter 3

Approach

In order to answer the scientific questions, which have been defined in section 1.1, an experiment is done. The experiment is realized by an agent-based model. The property and capabilities of the model are examined in a test environment. The chosen approach for the model is explained in this chapter. The approach is inspired by the ideas of industry 4.0 and smart factories, which were presented in section 2.1. Furthermore, the ideas of an agent-based manufacturing control described in the work of Leitão (2009) are included in the approach. Since the model simulates a semi-conductor production, for the creations, the conception of the model, as well as the test scenarios, are inspected by the work of Scholz and Schabus (2014b, 2015).

3.1 Basic knowledge for the concept

The first step is to set the framework for the model to implement. In the process, basic assumptions are made for the model from the initial situation and the task. We know that a semiconductor production is to be simulated. From the work of Scholz and Schabus (2014b), the following basic structures are extracted.

- On a product, many work steps are carried out on different machines. The machines for the work steps can be distributed in the factory.
- The structure of the production environment can change depending on the product. The construction is mostly in the form of corridors. The production halls/areas are separated by the airlock.
- There are machines, interim storage facilities, corridors, and airlock.
- The products are brought from worker to machine

From this information, the following functions for the model can be derived.:

- The structure of the environment is in the form of a factory hall with aisles and machines as well as intermediate storage. From this, it can

be deduced that there are machines and intermediate storage. These things can be considered as important elements.

- Workers transport the products. Mobile agents, which can independently navigate through the factory to handle transport orders.
- The simulation environment should be changeable.
- Products arrive and leave via airlocks in the individual production sections. There are an input and an output in each area. Such an area can be simulated independently.

3.2 Agents and functions

From the elaborated properties of the model components such as agents and basic functions can now be determined. This is done under the point of view of industry 4.0 and Agent-based modeling (ABM). The purpose of the individual elements is now described and serves as a starting point for the implementation of the model. The collected basic components of the model are now shown.

- Products: Products are clearly identifiable, know their own position and their work steps.
- Workers: These must be able to move through the production hall. They transport the products. The navigation of the workers takes place independently. They receive the transport orders from other agents or functions.
- Machines/Tools: The tools process the product and perform a specific work step. You have a fixed position in the factory. Machines know how much work they have and share this information. Besides, they can be broken.
- Stocks: Intermediate storage serves as an intermediate step if there is no free machine. They have a certain capacity, which can be queried.
- Simulation environment: A room in which the model works. This brings the spatial component into the simulation. The structure of the factory with corridors and machines and stocks is determined as a result of this.

An important part is the form of process or manufacturing control. This is an essential part of the model and will be covered in the next section.

3.3 Manufacturing control

The most important element in the thesis is the implementation of the model from the point of view of Industry 4.0. This means, considering the extracted basic elements of the model, which all elements communicate with each other and the control in the factory is decentralized (Shrouf et al., 2014). The core element in a production process is the manufacturing control. The manufacturing control checks the processes in the factory. The progress of the product is monitored while it moves through the various steps in the factory. Many decisions have to be made, such as: which product is produced on which machine, when will it be released for onward transport and in which order should the products be made? Traditionally, such processes are hierarchically and centrally controlled (Leitão, 2009). With the number of required work steps as well as the number of interactions between departing components, the complexity of scheduling increases. And the system is becoming slower to adapt to new situations (Cantamessa, 1997). In work, Leitão (2009) provides a manufacturing control based on agents. This approach is based on the fact that all components of manufacturing are defined as agents. This means they work autonomously, are intelligent and work together. Thus, a product which is executed as an agent knows its past, knows which work step is next and can independently request further transport. These properties are also used in the description of an industry 4.0 scenario (Albach et al., 2015). This example shows the difference between hierarchical systems, which are listed in table 3.1. Figure 3.1 shows a simple example of how production control could be done with the help of agents. The part-agent asks the available machine agents who free resources to perform the drilling step on the part. Upon this request, it receives the following answers from the machines:

- Machine #1 replies: I am out of order, I cannot execute this operation.
- Machine #2 replies: I am overloaded.
- Machine #3 replies: I have free capacity for the request.

The job is carried out by machine #3, and the transport from the current position of the part to the machine is done using a transport agent.

Table 3.1: Comparison of traditional and agent-based approach (Leitão, 2009)

traditional	**agent-based**
Centralized solution for every process	Decentralized solution with cooperation between agents on more than one control function
Static architecture	Flexible, dynamic architecture
Top-down approach	Bottom-up approach
Communication with one intelligence in the top levels	Communication with many intelligence distributed through the control levels
Efficiency through the specialization	Efficiency through the flexibility
Slow reaction to disturbances	Fast reaction to disturbances
Effective with large quantity and small variation of products	Effective with small - large quantities and medium to high variation of products

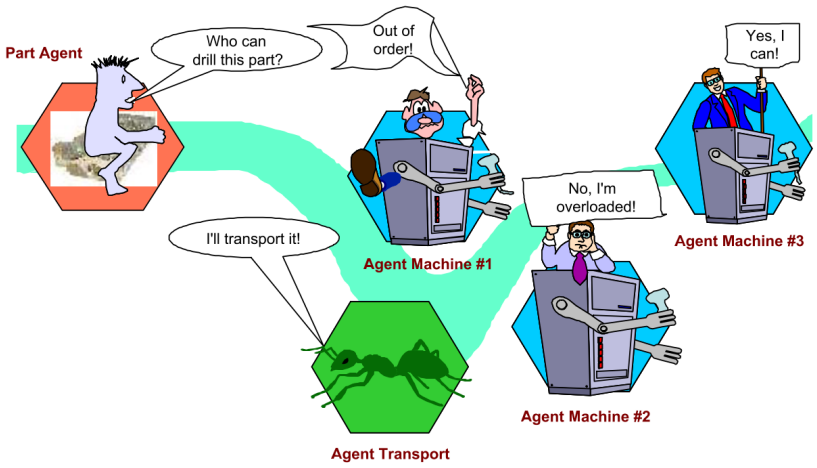


Figure 3.1: Agents in manufacturing control (Leitão, 2009)

Based on this example and the features described in section 3.2, the following manufacturing controls have been created for the model. There is a

central administration of all information of the model. This function does not schedule but acts as an intermediary for requests from agents. In other words, this function includes methods to find the right next machine for a product. Figure 3.2 shows the process control as an interface between the actors. Therefore, agents do not communicate directly with each other but can query all the information they need. The process planning is carried out by the agents themselves. A central information point makes it easier to keep an overview of the processes.

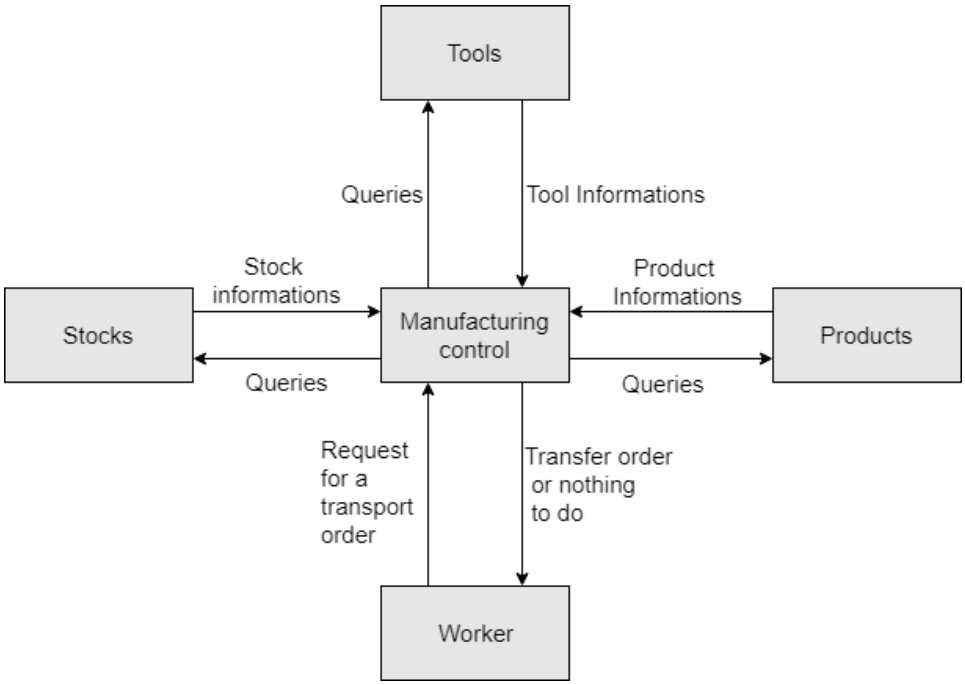


Figure 3.2: Process control overview

3.4 Model concept

Now that all components of the model have been determined, an overview of the planned simulation process follows. The flow of the model is shown in Figure 3.3. Here is the simulation flow from a worker's point of view. The following process repeats until all products are finished and applies to all worker agents.

- The worker has no order, and he sends an inquiry to manufacturing control if there is a product to transport. The manufacturing control queries all products, select a suitable order and sends it to the worker.

- The worker autonomously navigates.
- The worker has arrived at the destination and picks up the product.
- Since the product knows its next step, it is looking for a suitable machine for this job. In addition, a request is sent to the manufacturing control, which searches all matching machines and sends a response.
- The worker transports the product to the selected machine.
- The product will leave at its destination. The worker is thus free again for new orders, and the cycle starts again

The fact that the worker and not the product occupies the active part of the model has the following reason. If the product wants to be transported on its own, the question must be clarified: At what time will the transport request be made. When the product is ready earlier, if so how much sooner? This problem does not occur when the worker is looking for suitable products. Thus, always the product, which is selected, for transportation which is available next can be selected. This is to ensure that the worker is a short plug unladen.

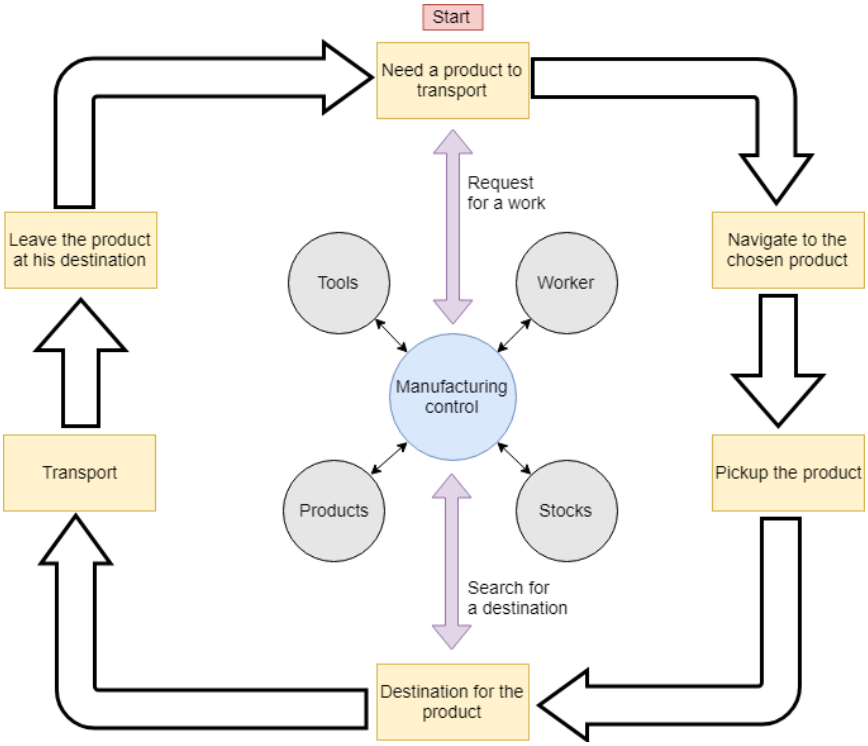


Figure 3.3: Schematic model process

3.5 Implementation and testing

The implementation of the model takes place in a software/toolkit for ABM. Section 2.2.9 shows an overview of some software options. The selection of the appropriate toolkit was made according to the following criteria.

- It should be freely available.
- There must be documentation about the functionality.
- All functions for the model must be given, or it must be possible to implement functions or to use them from external software libraries.
- There should be a tutorial and sample programs.
- It should run both Windows or Linux system.

The choice of software fell on Repast Simphony [1], since all conditions are met here.

In order to answer the scientific questions, a testing environment has to be created. The following data are required for this:

- Environment layout: where are paths and machines located.
- Wafer types with the associated production steps.
- Average production time at the machines.
- Comparison results: Average production time traveled distance per type, transport density.

For this purpose, an attempt is made to obtain real data from the industry. If this does not work your own test data will be created and the functions of the model shown.

[1] Repast Simphony https://repast.github.io/

Chapter 4

Modeling and implementation

This chapter deals with the description of the model used in this thesis. The model is built according to the approach outlined in Chapter 3. First, Section 4.1 describes the structure of the model. Followed by a detailed description of the agents and functions used in Section 4.2. Section 4.3 is a brief description of the key points for implementation in Repast Simphony After that, in Section 4.4, a description of the most important processes in the simulation comes in the form of process overview and process planning. Finally, in sections 4.5 to 4.9 an insight into the implemented processes is given.

4.1 Structure of the model

The model is the basis for the implementation of a prototype application and simulates production processes of products, in particular, the semiconductor production, or productions, in which people transport goods between different production assets or stocks. The model attempts to simulate such a production process, in order to be able to infer a real production chain. The focus, of the model, is placed on the transportation of goods between the production elements. The structure of the model is based on the work of Scholz and Schabus (2014b). In this work, an ontology for indoor semiconductor production is described. As a simplification, however, only a completed production area is considered, doors and air locks are not modeled. This would only be an extension of the environment but would not have any influence on the basic functionality. The products come and leave the model on a grid cell that can be considered as a link to other production areas. The model consists of the following elements:

- **Production hall:** The production hall is the spatial environment of the simulation. These form the framework for the simulation.
- **Tools (production unit):** The machines have a fixed position in the production hall. They carry out work steps to manufacture a product.
- **Workers:** The workers carry (transport) products through the pro-

duction hall. They must, therefore, be able to navigate through the spatial environment.

- **Stocks:** Stocks are optional for the production process. They serve as a stopover during transport between machines and have a fixed position in the environment.
- **Manufacturing control:** The Manufacturing control is the dispatcher for the workers. It assigns orders to the workers.

How these elements are related and how they are implemented in the model is explained in the following sections. The implementation of the ABM is done in the software Repast Simphony ("Repast - The Repast Suite," 2018).

4.2 Agents, attributes and other important parts

The description of this section will be related to the basic elements of an ABM from Section 2.2.4.The described model uses the following agents, relations, and the environment. Another important part of this section is the conception of the product and the central process control.

- **Agents:** The main actors are workers, machinery, and stocks. Above all, the workers, as the only mobile agents. They have great importance to the model. Each worker can carry a production item. The machines are supplied by the workers and can each perform a specific work step on the product. The stocks are another part of the model and these as intermediate points to keep the production flow going. The decision, which product is to be transported, is made by a separate manufacturing control.
- **Relations**:
  - A worker transports a production item, to and from machines and stocks;
  - The machines process the product when a worker supplies the machine;
  - The manufacturing controllers regulate the sequence of the orders for the workers;
  - Stock can store product and are filled and emptied by the workers;

- **Environment**: The environment is the area in which the simulation takes place. Through the environment comes a spatial reference in the simulation. A raster map defines the position of simulation elements such as walls, paths, stocks, and machines. The worker agents are moving into this environment to transport products to the machinery and storage facilities. The input of the layout of the environment is based on a raster map. As a result, all objects are assigned raster cells, and the agents interact at the grid level. In addition, there is a graph model with nodes and edges. This graph model is used for navigating the workers.

For a better overview, the agent classes and their most important attributes, functions and relationships are shown in Figure 4.1 in the form of a Unified Modeling Language (UML) class diagram. Table 4.1 shows these agent classes and other important classes in list form. Each class is listed with its basic purpose. Furthermore, the minimum number of the respective agents is given, and a distinction can be made between mobile and static classes.

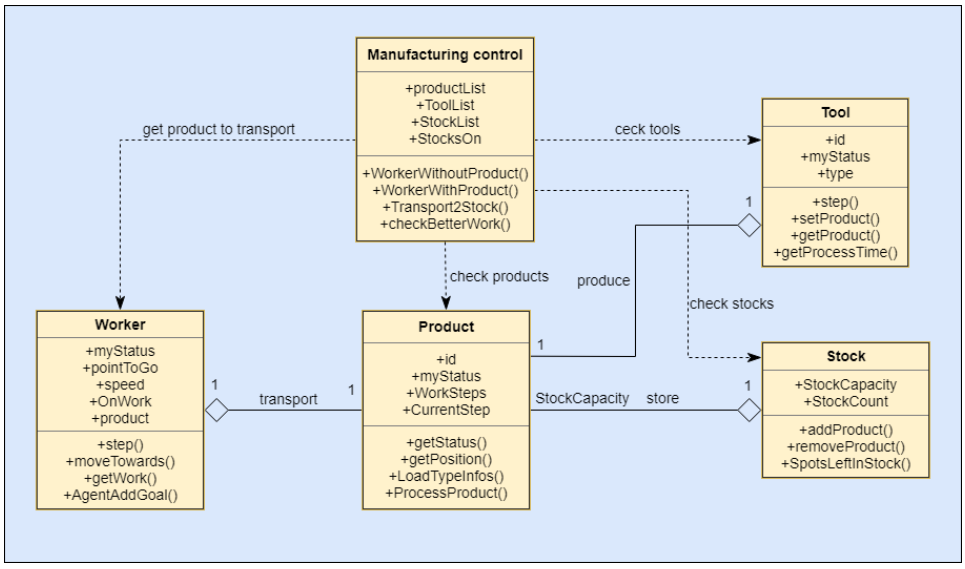


Figure 4.1: Class diagram of the model with the most important functions and attributes of each agent class.

Table 4.1: Description of the agent classes and their task

Classes	Task	Position	Count
Worker	transport the product	dynamic	minimum one
Tool	produce one work step on the product	static	ate least one per work step for the product
Stock	stocking of products between work steps	static	zero to arbitrary
Products	goods which are processed	dynamic (transport by a worker)	one to arbitrary
Manu. control	distribution of work	no position	one

The Spatial Scale is related to the speed of the agents. Depending on the size of the cells, the workers move faster or slower, which influences how fast machines are free, or how long they have been filled. This value must be calibrated with the model. The temporal resolution of the model has a turn-based implementation. A worker moves one step per round with a maximum speed of one cell.
A detailed description of the processes can be found in section 4.4. The following subsections describe the modeled agent classes and important object classes in detail.

4.2.1 Worker

The workers are essential for the simulation. They represent virtual factory workers and transport the products to be produced through the factory. Every worker can carry one product. To pick up a new product, the old one must first be stored in stock, or delivered to a machine. The transport takes place between machines and intermediate storage. The motion model, for the simulation, is described in detail in Section 4.7.1. The workers receive an order from the *manufacturing control* class to transport a product. The transport is based on minimal costs and is, therefore, the shortest route. Whenever a worker is without a job, manufacturing control assigns a new job. The actual ”intelligence”, in which order the products are transported is outsourced to the *manufacturing control*. The workers themselves only

execute orders and do not make their own decisions. The description of the class manufacturing control, as well as the scheduling of the workers, follow in Section 4.2.6. In figure 4.1 you can see the most important attributes and functions. Figure 4.2 shows the functionality of the agency class Worker. The illustrated process is performed once per simulation round. It first checks whether the worker currently has an active transport request, this is done by using the attributes *On Work*.

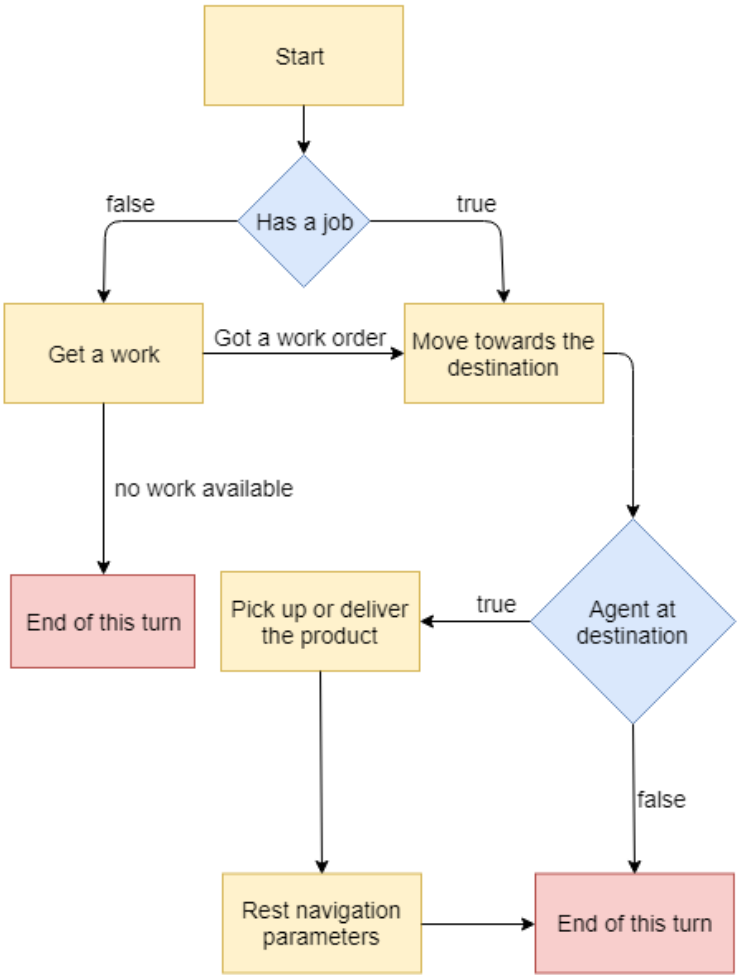


Figure 4.2: Schematic overview of the agent class worker

If no order has been assigned yet, a product is being searched for transport. This is done via the manufacturing control. If no work is available, the worker waits for a simulation step. If a product is available for transport, it is assigned to the worker. The position and destination of the goods to

be transported can be read from the assigned product. If the worker has an order, he moves along way points to his destination. The way points are expressed by the variable *pointToGo*. If the transport target is not reached with this step, this simulation step ends. The goal is reached when the distance to the target is less than the movement speed of the agent. The movement speed differs depending on whether the agent is loaded or not. The speed of how fast a worker is allowed to move can be inquired about the product to be transported. When the target point is reached, the *addGoal* function is called. Depending on the order, a product is picked up or delivered. Finally, the navigation parameters are reset so that the worker is ready for a new job in the next round.

4.2.2 Tools

The machines are the second important part of the model because the products are produced on them. Machines have a fixed location in the factory. Each Tool has a type, depending on the type of machine other work steps can be performed. The production time, the time how long a product is processed depends on the type of machine. The machine type is determined when creating the production environment - this is explained in detail in section 4.5.1 The machines can break while processing a product. This feature is optional and must be activated. Because of a failure, the production time of the machine is increased by a random value. However, this machine is longer occupied. These random events may change the simulation result. Figure 4.3 shows the functionality of the agency class Tools. The illustrated process is performed once per simulation round. First, it checks if the machine is working on a product. If there is no product in the machine, the function is ended. If, on the other hand, a work step is being executed, it is checked whether machine failures are activated in the settings. With the break down enabled, failures are simulated by a random number generator. The size of the random pool, as well as the impact area, can be set in the simulation settings. In case of failure, the production time is increased. As a result, the machine takes longer to work. In addition, it is no longer considered for job scheduling during the time of the failure. Furthermore, each round is carried out one working tick. Once all work steps have been completed on this machine, the product can be picked up from the machine again. A machine can take different statuses. This status is important for the order assignment of the process control function. The states of the variable *myStatus* are listed in the following table 4.2. In addition to the description, the color is also indicated with which the status is displayed in the graphical simulation environment. The effects of these states are dealt with in detail in section process control 4.2.6.

Table 4.2: Description of the attribute *myStatus* of the class tools.

Status	**Description**	**Color**
Free	The machine is free and ready for a product	Green
Work	The machine is currently producing a product	Orange
Wait	A worker is on his way to the machine. During this period, the machine is not available for other jobs	Yellow
Wait for exchange	A product is in the machine. A worker comes up with a new product for the machine and takes the finished one with him.	Cyan
Break Down	The machine is broken. It is not available for new orders	Gray
Finished	The product in the machine is ready and must be picked up	Red

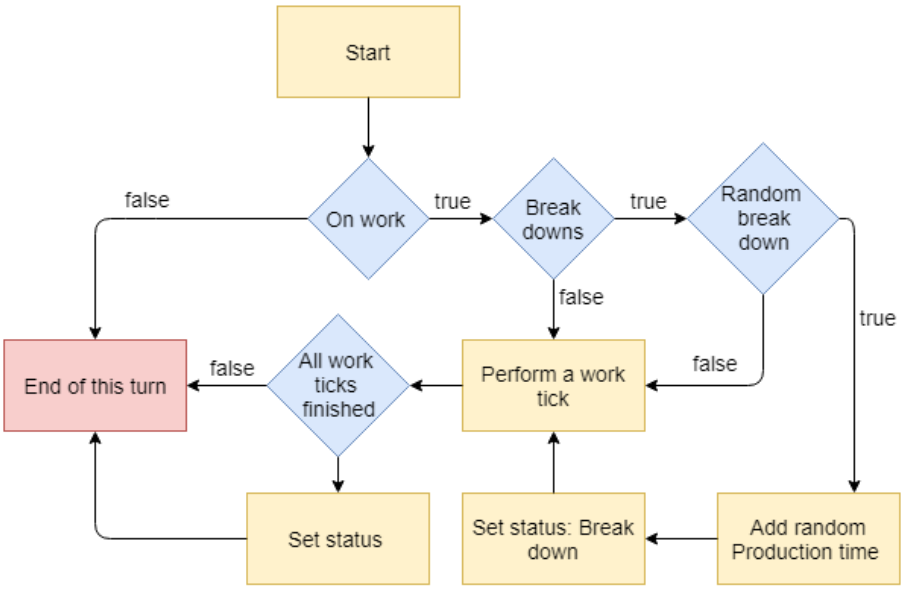


Figure 4.3: Schematic overview of the agent class Tool

4.2.3 Products

The products are produced in the simulation. They are not real agents but are defined as object classes. That means they are always in conjunction with an agent of the class Worker, a Machine or a Stock. Each product has its own unique Identifier (ID). This ID can be used to access the product and query its status and position. The position is updated at the end of transport and is therefore always a machine or a warehouse. Based on the work of Scholz and Schabus (2014b), each product has a type and list of production processes that it needs to run. The type of product determines the required work steps and the transport speed. As a simplification, only one product type is used in this model. The status variable (*myStatus*) is essential for the model. It filters the products for the process flow and provides information for the transport by the workers. In the following tables 4.3, 4.5, 4.4 the various states and their meaning are shown. The statuses are ordered by its use case. Table 4.3 shows conditions which are important for the scheduling of the transports. Only products with one of these status values will be considered when assigning new orders because these products are currently not assigned to a worker. The statuses listed in Table 4.5 refer to the case where the products are picked up by a worker. The different cases are needed for the right actions to be taken. For example, a machine is free again after picking up, or stock has again more storage space. Table 4.4 deals with statuses for the transport of the products. Again, the distinction is made depending on the action which is set at the destination. For example, after transport to a machine, it must be set to busy, or an existing product in the machine must be removed. Setting the correct status is very important for the simulation process. If this is implemented incorrectly, it can lead to erroneous behavior of the entire simulation.

Table 4.3: Description of the attribute *myStatus* of the class product. Status with position information.

Status for the process control	
Status	Description
Free	This is the initialization status. The product is at the starting point of the simulation.
atTool	The product is currently on a machine.
atStock	The product is in an interim storage.
finished	The product is finished and is at the end point. It is no longer included in the simulation.

Table 4.4: Description of the attribute *myStatus* of the class product. Status with transport information with loaded worker.

Status for transport, the worker is loaded	
Status	Description
Transport2Endpoint	All work steps are finished, the product is transported to the end point.
Transport2Tool	The product is brought to a machine for the next step.
Transport2Stock	The product is taken to the next stock.
WaitForExchange	The transport of the product takes place to a machine, which is still a product. The finished product from the machine is removed and transported
Wait4Transport	There are no machines available, so there is no transport in this simulation round.

Table 4.5: Description of the attribute *myStatus* of the class product. Status for transport information with unloaded worker.

Status for transportation, the worker is unloaded	
Status	Description
WaitForPickUpAtStartPoint	The product will be picked up at the starting point.
WaitForPickUpAtTool	The product is picked up by a machine.
WaitForPickUpAtStock	The product is picked up from stock.

4.2.4 Stocks

The intermediate storage is an optional part in the production chain. They serve to reduce the waiting times of the workers on an occupied machine and to create more fluid flow. Each machine is assigned a stock. This is the closest intermediate storage to the respective machine. Since the intermediate storage facilities are not necessary for production, these can be switched on or off at the simulation start. The most important attributes and properties

are the storage capacity and the number of available places. This information is needed for scheduling the production process. If a warehouse is full, the worker must wait for a free storage area or a free machine.

4.2.5 Environment

As already mentioned, this is the simulation of a production hall. Therefore, the environment in which the agents interact reflects such a hall. For the simulation, the structure was simplified, so it is a self-contained production area. There are no doors or other obstacles. Only the starting point and the end point in which the production goods enter or leave the hall can be seen as a link with other production areas. The environment consists of two parts, the visible and the invisible part. The visible part consists of the following parts:

- Corridors: In this area, the agents can move.
- Walls: Barrier can not be transgressed.
- Tools: Position of the machines, the type differentiation is done using various integer values see table 4.8.
- Stocks: Position of the intermediate storage.
- Start point: At this point, the products to be produced lie on. The production chain starts at this point.
- End point: When all the steps have been taken on a product, they are brought to the end point and are considered finished.

The nodes for the navigation of the working are not visible. These are located in front of machines and warehouses as well as at the intersections of the paths in the hall.

4.2.6 Manufacturing control

The most important part of the model is the manufacturing control. In this function, all information about products, machines, and stocks converge. This function is always called when a worker needs a task. Therefore, the actual intelligence of the model is created in this function. Figure 4.4 shows the schematic sequence of the function. The function is divided into two parts. It is distinguished by whether the worker is free and has no product with him, or whether he is looking for a loaded product for a suitable

machine. These two cases are described in this subsection. In addition, the subfunction transport to stock is described in detail, since some things have to be considered with this.

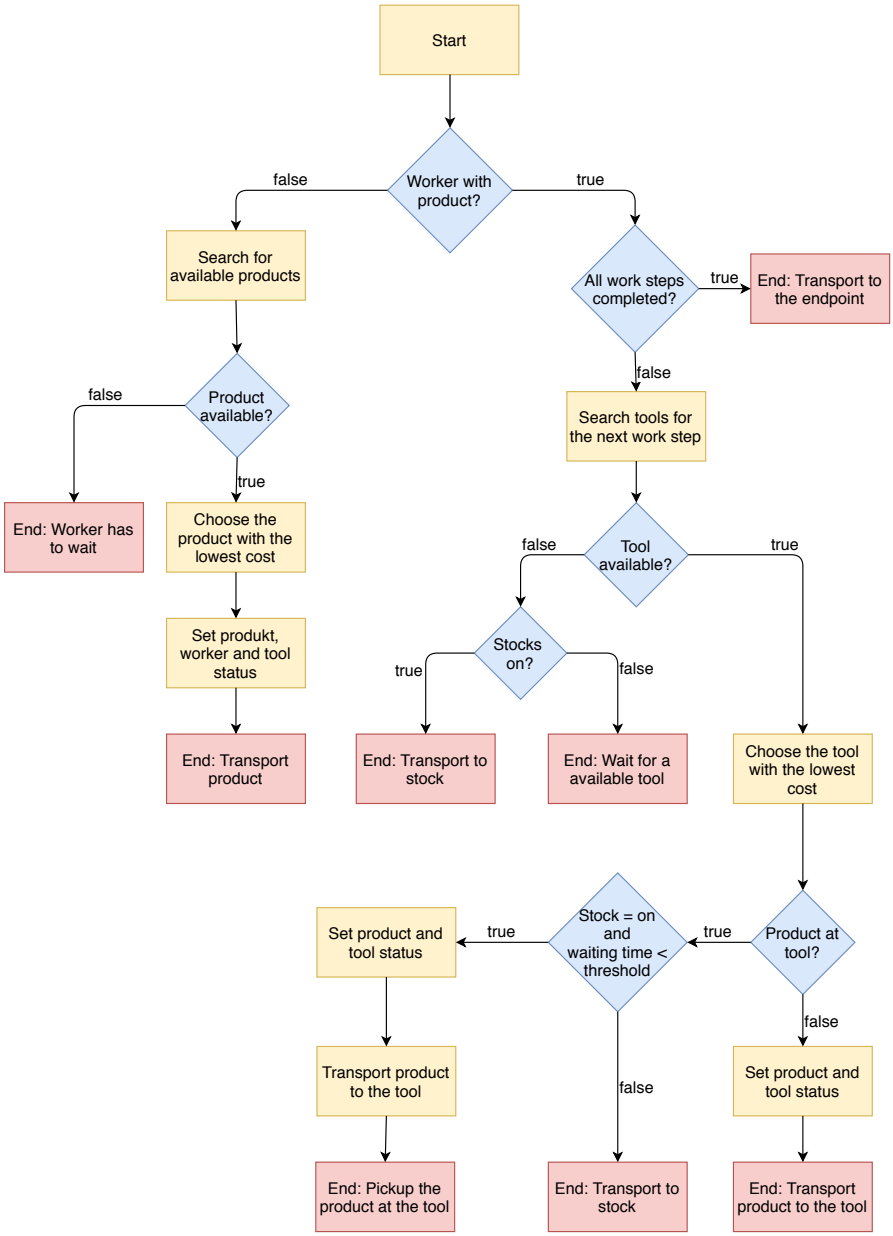


Figure 4.4: Flowchart of the class Process Control, when calling the *getwork* function

Worker without product

If the worker is not loaded, a free product is searched for transport. First, it checks if there are currently any products that need to be transported now or in the near future. Products with the following status are included in the selection.

- **Free:** The product is at the starting point of the simulation.
- **at tool:** The product is currently on a machine, and it is not distinguished whether the production process is already completed or not.
- **at stock:** The product is in interim storage.

If no product is available for transport, the worker waits for a simulation round. An **accessibility value** is calculated for all eligible products. This value is composed of the **distance** to the product, the remaining **production time** at the machine and a **weighting**.

$$\text{accessibility value} = \frac{\text{distance}}{\text{speed}} + \text{remaining production time} + \text{weighting}$$

The distance is calculated by a node edge structure using the Dijkstra's algorithm. Since the accessibility value is given in simulation ticks, the distance must still be divided by the speed of the workers. If the product is being processed on a machine, remaining working time is added to accessibility. Also, a weighting value can be added. This additional value can be used to control the simulation behavior. The next product is selected via the accessibility value, the product with the lowest value is selected for the transport. This means that the sum of travel time and waiting time is the smallest for this product. In order to influence this selection, the weighting comes into play. This is determined at simulation start and applies to the entire run. Depending on the place where the free product is located, a factor can be applied. For example, products that are in finished machines can be treated preferentially. An overview of the weighting options can be found in Table 4.6. By varying these values, priorities can be set in the simulation. The unit of weighting, as well as the accessibility, are simulation ticks.

Once the product to be transported has been selected, it is given to the worker and is excluded until the end of transport for other orders. This is done by setting a status variable. An overview of the respective status variables can be found in Table 4.5. The worker is now assigned the product, and it can complete the job and pick up the product at its location. A special rule when selecting a product applies when the product is in a temporary storage facility. Since it may happen that for several workers the closest

Table 4.6: Overview of the weighting options of the simulation.

Weighting	Description
At tool	The product is being processed on a machine.
At stock	The product is stored in an interim storage facility.
At finished tool	The product is ready to pick up in a machine.
At the start point	The product is at the starting point of the simulation and is therefore not yet in production.

product is in stock, only so many workers are allowed to wait at the stock for products to be in it. If this happens, another product is sought which is not in this intermediate storage. This is to prevent too many workers from waiting in the same place.

Worker with product

The second task of Process Control is the search for a transport destination for a just taken product. The first step is to check whether all work steps have been completed on the product. If this is the case, the transport to the endpoint takes place. As soon as the product is at the end point, it is no longer relevant for the further simulation and removed from the simulation. If the product is not ready yet, it is necessary to search for a suitable machine for the next step. Since there is normally more than one suitable machine, one has to be selected. This is done as in the product search via an **accessibility value**. Only machines are considered which currently no other worker selected. This means the following machine status is taken into account. The overview of all machine pauses can be found in Table 4.2.

- **Free:** The machine has no work at the moment.
- **Work:** A product is currently being processed on the machine.
- **Finished:** The machine is finished, but there is still a product in the machine.

The **accessibility value** is composed of the travel time to the machine and the remaining production time. The smallest value is used to select the next machine. If no machine is available, this may occur if all machines

are currently being used by other workers, the transport will take place in an intermediate store. The implementation of the interim storage facilities will be explained in detail in the following subsection. If the stocks are full or deactivated in the simulation, the worker waits until a machine becomes free. If a machine has been selected as the destination for transport, it will be locked for other jobs during the transport process. It will not be released to other workers until the product has arrived at the machine. As a result, there are no double transports, and the scheduling is simplified. The model offers the possibility to exchange a finished product with one still to be produced on a machine. If there is such an exchange, the worker is waiting for the machine to finish the production. Then the new part is delivered to the machine and the old transported away. The machine will not be available to other workers during the waiting time. To prevent a machine from being removed from the scheduling process for too long, the maximum waiting time can be controlled with the so-called *for exchange threshold*. This value is entered during the initialization of the simulation. And is stated in simulation sticks. If the waiting time for an exchange is greater than the specified threshold, the transport takes place in an interim storage facility. Therefore, this option is only available if the intermediate alerts are activated in the simulation.

Transport to stock

Transport to a temporary storage facility is a function in which a few cases are differentiated. Therefore, it is now considered in detail. Figure 4.5 shows the function schematically. Depending on where the product is located, other destinations are available for transport. There are three possible starting situations:

- The product is at the starting point: This is the simplest case, the product is transported to the warehouse for the next step. If the interim storage is full, the worker must wait. Since products for different work steps are stored in the stocks, only half of the storage space may be filled with products from the starting point. This should prevent the stock from being continuously occupied for future work steps.
- The product is on a machine: The product is taken to the intermediate storage of the machine from which it was taken. When this is full, it will be taken to the stock for the next step. Should there be no free space there, too, the worker is waiting.
- The product is in stock: If the product is in a warehouse, it is first checked if it is the intermediate storage for the next step. If this is not

the case, the transport takes place in the next intermediate storage. The second option is the product is already in stock for the next step. To keep the times short in which the workers wait for free machines, they are looking for a better job. This function is explained in the following subsection.

Transport between two interim storage facilities is also possible. This is intended to shorten the transport routes and increase the dynamics of the system. Of course, this functionality is only used if the intermediate storage is activated in the simulation.

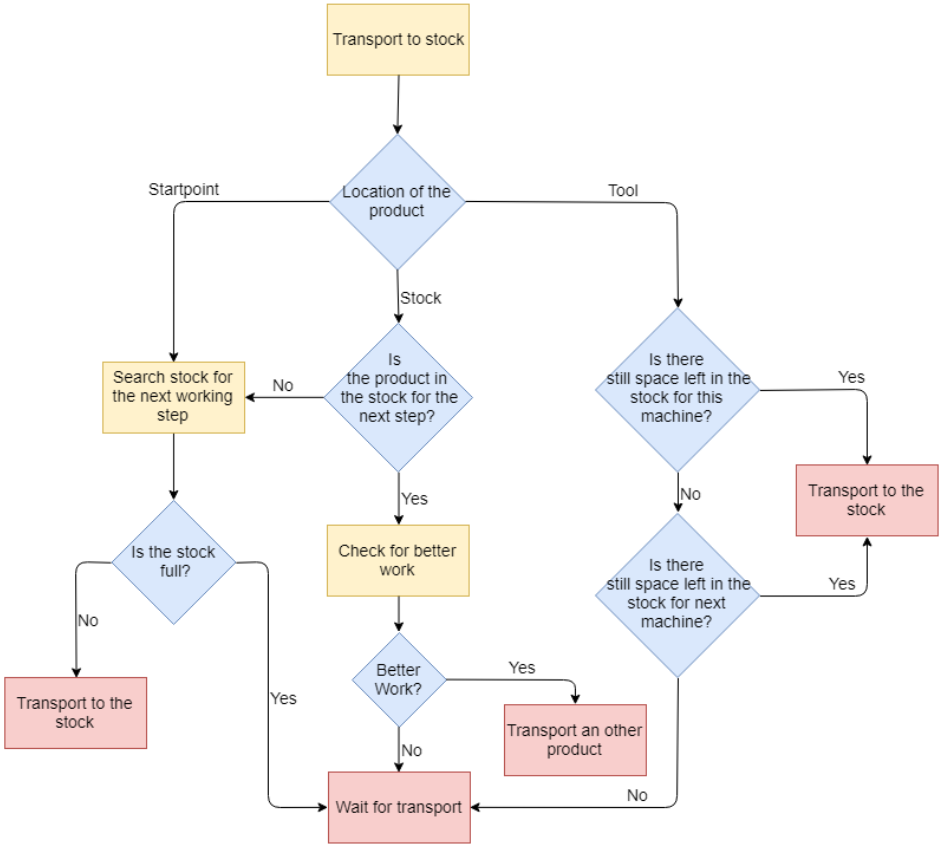


Figure 4.5: Process overview of the function Transport to stock

Find better work

The idea behind this function is to keep the waiting times at the intermediate storage facilities for a free machine small. Without this feature, the worker waits as long at stock until the next machine is free. As to be seen in Figure 4.5, this function is called if the product to be transported is in the interim storage area for the next work step but no machine is available. The function works in the same way as the search for a free product 4.2.6. With the difference that products which need the occupied machines as the next work step are excluded in the search. If the availability value of the new product is less than the waiting time at the machines, this is selected as the new transport order. The other product remains in the stock

4.3 Implementation in Repast Simphony

The model is implemented with Repast Simphony version 2.5[1]. Since Repast Simphony uses Java as the programming language, you also need the Java Runtime Environment. Recommended for Repast Simphony 2.5 of the latest version of Java 8. Here, Java SE 8[2] is used. The development environment is Eclipse oxygen 1a[3], as it is also installed with Repast Simphony on Windows. The introduction of Collier and North (2013) explains both the basics of Eclipse and Repast Simpohny and was used as the basis for this modeling. Repast Simphony 2.5 provides the basic features for agent-based modeling. This includes a programming library as well as a graphical user interface. This user interface can be expanded with diagrams and setting options.
According to the documentation of the Application Programming Interface (API) "Repast Simphony 2.5 API" (2018) the central feature in Repast Simphony is the *Context* class. In this context, all agents and parts of the simulation are loaded. Therefore, the context represents the entire model. The *Context* can also be divided into different areas by means of *SubContext*. The *Context* acts as a container for agents and model elements. For your spatial connection to come about, every context can be define one or more *"projections"*. This *"Projection"* creates a space in which the agents can act. The space can be in the form of a grid, Euclidean space, physical space, networks or geographic space (Geographical information system (GIS) data)
The functionality of the individual objects has already been described in section 4.2.1 to 4.2.5. These must now be implemented as object-oriented Java classes. How the functional processes of the whole model look and work are described in the following chapter 4.4.

[1] Repast Simphony 2.5: https://repast.github.io/download.html
[2] Java SE 8 download https://www.java.com/en/
[3] Eclipse https://www.eclipse.org/

4.4 Process overview

The simulation consists of a cyclic sequence of different processes. A cycle represents a period in which a worker takes a step. At normal speed, this means that the worker travels a distance from a grid cell. Repast refers to such a time step as so-called *"tick"*. All-time units in the model are given in the form of a multiple of these *"tick"*.

Table 4.7: Tabular process overview of the model.

#	**Process**	**Member**	**Description**
A	Initialization	Worker, Tools, Products, Environment, Stocks	Create the agents and the simulation environment from the database
0	Manufacturing control	Worker, Products, Tools, Stocks	A transfer order for a worker is searched. The worker may either have to pick up a product somewhere or transport a taken-up product to its destination.
1	Transport	Worker, Environment, Products	Transport of the products through the simulation environment
2	Manufacture	Tools, Products	The machines perform one work step on the product, the number of work steps per machine is specified in *ticks*.
3	Model-Output	Worker, Tools, Products, Environment, Stocks	Extracting observations and metrics from the model. In the form of key figures and a raster map.

Table 4.7 describes the processes involved in the implementation of the model with the classes participating in the process. The initialization process uses the input data to create the model state in the form of the object instances

for the time $t = 0$ or *"tick"* 0. After initialization, the simulation process begins. The processes $1-3$ are executed per time step and agent. The process 0 (process control) is only executed if a worker does not have a transport request. The order in which the agents are addressed in a simulation round is random and varies in each time step.

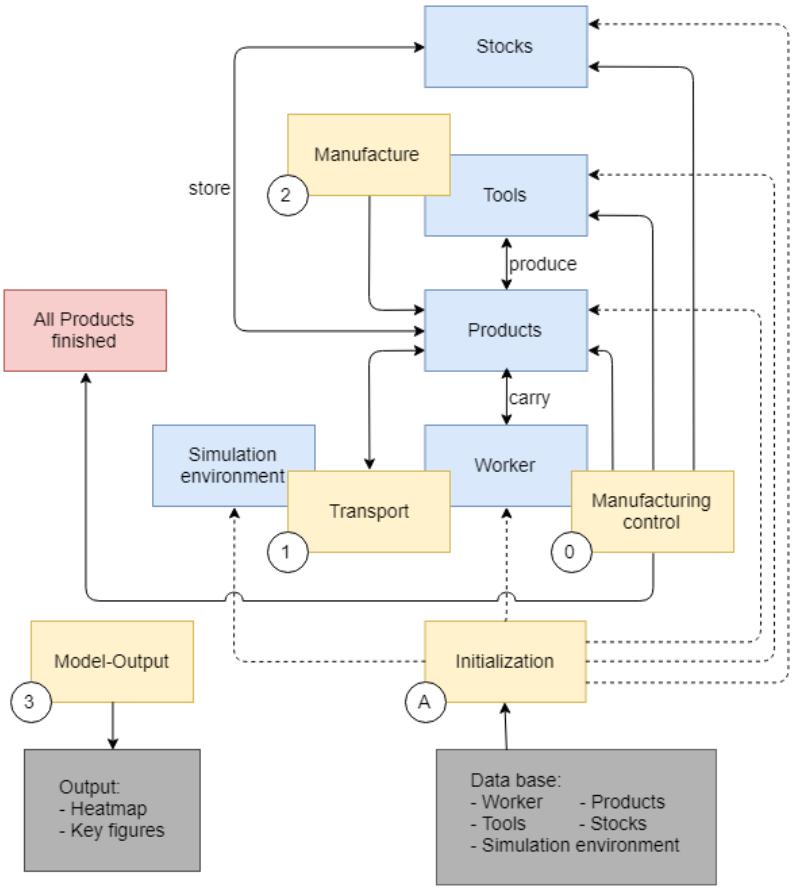


Figure 4.6: The schematic flow diagram of the model. Shown are the processes (yellow), the classes / agents (blue) and the data base / output (gray).

Figure 4.6 shows this cyclic sequence again schematically in the form of a flow chart. The simulation starts with the initialization (A). Then the workers start to transport products. For this purpose, the function process control (0) is called for each job. The processes Transport (1) and Manufacture (2) are executed permanently. Also, data is collected during the entire simulation process. This takes place in the process model output (3). The termination condition of the simulation (in red) occurs when all products have been finished. This condition is controlled by the process control.

The sequence of classes and agents has already been described in Chapter 4.2. The procedures described in this chapter have been implemented in Repast Simphony. In the following sections 4.5-4.9, the implementation of the processes listed here will be described in detail again.

4.5 Initialization

During initialization, all input files are read in and added to the ”context” of the simulation. The data will either be read from external files. The structure of these files is described in subsection 4.5.1. The initialization is done in a predetermined order so that all data is available at the right time. The dependencies of the input data are shown in Figure 4.7. The figure shows why the order of data entry is important and how it depends on it.

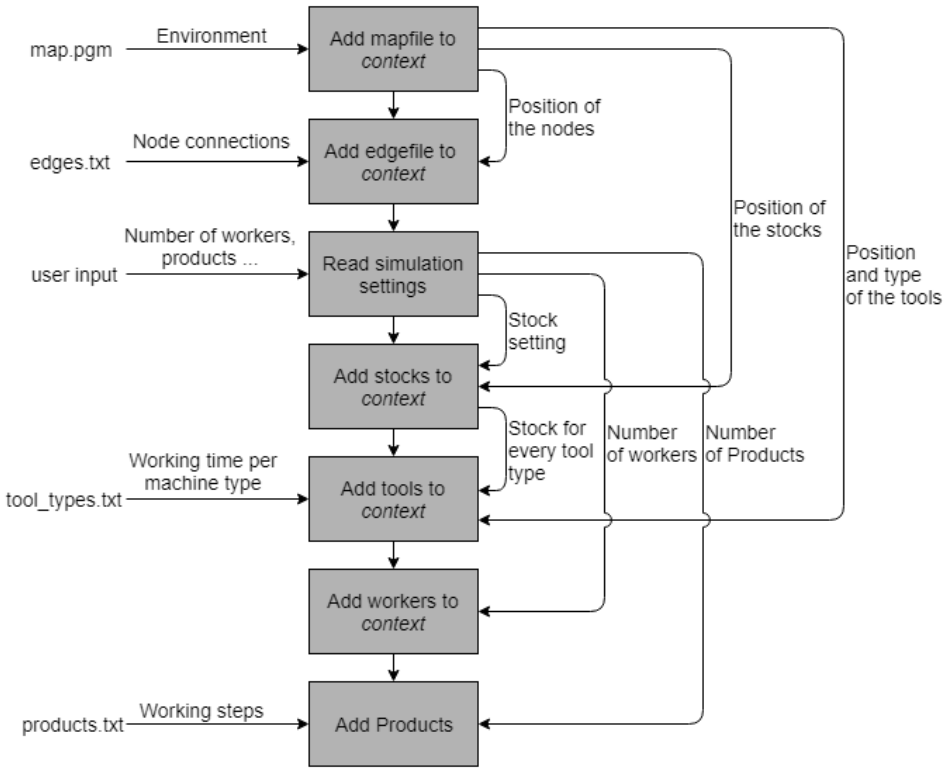


Figure 4.7: Schematic sequence of data initialization. The input data must be processed in order.

4.5.1 Date base

The structure of the individual input files and the setting options are explained in this subsection.

Input of the environment

The simulation environment is created as Portable Gray Map (PGM)file and loaded into the simulation. It is a raster map with integer values. The construction of this file shown in listing 4.1. Important is the header of the file. The width and height as well as the largest occurring number are specified. Various integer values are used to realize the individual conditions in the hall. An overview can be found in table 4.8.

Listing 4.1: map.pgm example.

```
P2
24 7
102
1   1   1   1    1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   0   0  30    0   0   0   0   0  31   0   0   0  31   0   0   0  32   1   1   7   1
1   0   0   0    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   7   1
1   2   0   0  100   0   0   0   0 101   0   0   0   0   0 102   0   0   3   1   7   1
1   0   0   0    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   7   1
1   0   0   0   25   0   0   0   0   0   0   0   0   0   0   0   0  25   0   1   7   1
1   1   1   1    1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
```

Table 4.8: Structure and description of the input grid.

Integer	Description
0	Walk-in areas as well as: corridors and open spaces
1	Walls and not accessible obstacles
2	Start point
3	End point
7	Style for rooms
25	Stocks
30-99	Tools: each number represents one machine type and can occur multiple times
100-250	Nodes for navigation

Definition of the edge structure

Since the position of the nodes is defined via the map, it must still be determined which of these nodes are interconnected by edges. This is done via a simple edge list. In listing 4.2 you can see the composition of the file. Each line in the file has the following structure: *note from, note too, directed, weighting.* If no weight is specified, the Euclidean distance between the nodes is used as the weight. With the input of true or false, it can be determined whether it is a directed or undirected graph.

Listing 4.2: edges.txt example.

```
0 , 1,true
1 , 2,true
2 , 3,true
3 , 4,true
4 , 5,true
5 , 6,true
3,  6, false, 50
```

Machine types

The position and type of tools in the simulation environment are determined by the input of the raster map. Also, an indication of the production time is required for each machine. This is the time it takes the machine to work on the product. The information is given in *ticks*. The structure is very simple, so only the machine type and the working time must be listed. In listing 4.3 you can see the composition of the file

Listing 4.3: Tool_types.txt example.

```
0,300
1,150
2,200
3,100
4,800
5,200
6,400
7,380
```

Working steps

As mentioned in section 4.2.3, a list of the work steps to be carried out is required for the products to be produced. Each product knows which work step has to be performed next via this list. Therefore, the structure of the file is again list form. In the example from Listening 4.4, seven steps are performed. The first value is the work step and the second value is the machine type at which this step is executed.

Listing 4.4: Worksteps.txt example.

```
0 , 10
1 , 0
2 , 1
3 , 8
4 , 1
5 , 13
6 , 12
```

User input

Through the input files, the simulation scenario is defined. The definition of the environment and the products have been made. Via settings with the help of the Graphical user interface (GUI) it is now possible to fine-tune the simulation. What options are there and how they affect is explained in this subsection. The list of settings is shown in Table 4.9. The presentation of the settings is done in categories. The first part is the basic settings. This is followed by the weightings which have already been explained in section 4.2.6. The simulation behavior can be influenced by the weights, as products are preferred in certain situations. It should be noted that the weighting must be specified negative so that a product in a particular situation is treated preferentially. Finally, there are settings for the optional machine failures.

Table 4.9: Description of the user input

Setting	Description
Number of products	It determines how many products are present in the simulation for production.
Number of workers	This is the number of workers in the simulation
Stock	Activate interim storage in the simulation
Stock capacity	Maximum number of products in the intermediate storage
Weight finished	Weighting when the product is finished on a machine
Weight at tool	Weighting when the product is being processed on a machine
Weight start point	Weighting when the product is at the starting point and thus not yet in the production chain
Weight stock	Weighting when the product is in an intermediate storage facility.
Wait exchange threshold	This setting only takes effect when the intermediate bearings are active. The value indicates from which remaining waiting time a worker with a product is waiting for a machine which is currently working.
Data export	Exports of simulation results
Break down	Switch on machine failures
Random pool	He probability with which it comes to a failure. When the machine is working, a number is rolled out of the pool in each *tick*. If this number is zero, the machine is broken.
Down Time	The duration of the machine failure in *ticks*
Down time random multiplicator	The duration can be multiplied by a random value.

There are also optional settings for testing the model behavior. These settings are described in Table 4.10.

Table 4.10: Description of the user input for model testing

Setting	Description
Random Mode	The product to be transported and the required machine are selected randomly. No information about distance, production time and weights is used.
Without process time	For the selection of the products and tools only the distance and weights are used.
Without spatial Information Product	The distance to the products is neglected for the selection.
Without spatial Information Tools	The distance to the tools is neglected for the selection.

4.6 Manufacturing control

The mode of operation of the manufacturing control has already been explained in detail in Chapter 4.2.6. This function is called by the workers before each transport. The manufacturing control has internal methods to retrieve relevant information of all objects and agents used in the model. For this purpose, there are lists about the different agent classes. The function itself is executed as a singleton because it means there is an instance of this class. This ensures that all agents have the same level of information when polling. The various object classes of the simulation such as workers, products, tools, and stocks are managed in lists. There is only one instance of these lists. These lists can be searched for managing the objects. It is possible to search specifically for an object ID, even for a specific object status. This implies that very simply all free machines can be called a certain type or queried. These methods for this preselection are defined directly in the list file type. This simplifies the search for the right product or a machine for transport. The further function flow is implemented in Java/ Repast Simphony as shown in Figure 4.6.

4.7 Transport

The transport of products through the simulation environment is an integral part of the model. The functionality of the product and destination selection for transport has already been described in Sections 4.6. This section explains the implementation of the motion model and the basics needed for it. The transport itself is carried out by the workers. They are the only moving agent of the model. An important prerequisite for the simulation is that the workers can independently navigate from their location to a destination. During the movement, the worker should choose the shortest route to the destination. In addition, workers are able to avoid other workers and obstacles. Only one worker may be on each grid cell of the simulation environment. An exception to this rule only applies to start and finish points and to interim storage. Because the movement model is kept as simple as possible.

4.7.1 Movement model

The implemented movement model consists of three parts:

- **A Graph Model with nodes and edges:** The graph structure is used for path finding by the environment.
- **A raster map:** The raster map defines objects in the room like walls, machines, floors. The collision query is also done at the grid level
- **Cartesian coordinate system:** The movement itself takes place in a Cartesian coordinate system. This is done by direction and a distance from the starting point to the destination point.

To enable the independent navigation of a work from its point of view to any target point, a graph model is used. This model consists of nodes and edges. Edges connect two nodes with each other. A distinction is made between directional and non-directional edges. On directional edges, you can only move in a defined direction. To find the shortest route to a destination in the graph network, the Dijkstra algorithm is used. In doing so, we calculate the shortest path to all other nodes from a starting node. The calculation is made by edge weights, which correspond to the duration or the ability to move on this edge. The model first looks for the next node from the current viewpoint as well as the destination node. As a result of the Dijkstra algorithm one obtains a node list. This node list will be used as waypoints for navigation. This node list allows the worker to move from

waypoint to waypoint on a straight line until they reach their destination. The list of the waypoint is the basis for the movement model. Figure 4.8 shows the navigation procedure. The worker in blue starts from his point of view and wants to navigate to the machine which is highlighted in cyan. First, the closest waypoint to the worker's point of view is determined. from this point the route to the destination point is calculated. This results in the waypoints A to E. The worker goes off the waypoints in sequence and comes to the closest point to the machine. From point E, the worker only has to move a short distance to reach his destination. It is important to mention that certainly, the worker does not move directly on the graph. The waypoints only indicate the direction to turn at intersections. This freedom in movement makes it possible for the worker to back away and avoid other workers.

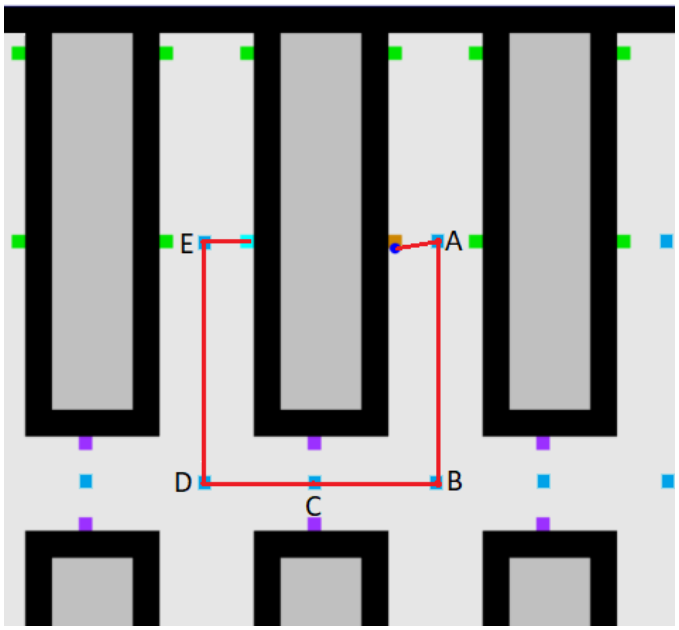


Figure 4.8: Example of navigation using waypoints

The graph network provides property for finding a path. The movement itself, however, takes place in a Cartesian coordinate system. It has the advantage that the worker can move in any direction and at any speed. The agent moves towards a waypoint at its defined speed. As already mentioned, work can end up dodging other agents. The calculation of when it is necessary to make an evasive movement is at the grid level. By importing the environment, walls are already defined in a grid. In addition, the current position of the worker is also placed in this grid. Now both the workers and all the butts are drawn on a raster map. Whenever a worker moves, he or she

checks out whether another object is entering the future location. Should this be the case, the worker tries to avoid this object. Figure 4.9 shows how the dodging of objects works. The workers tried to reach the target point B from its position A. He tries to reach the goal on the shortest way, a straight line. Before each step, it checks whether the targets grid cell is accessible. If there is an obstacle on the target cell, all neighborhood cells (Moore neighborhood) are checked. Each cell is checked for accessibility, and the distance to the destination B is calculated. The new direction of movement is in the direction of the grid cell which is walkable, and the distance to the target point B is the lowest. With this concept, simple obstacles can be avoided.

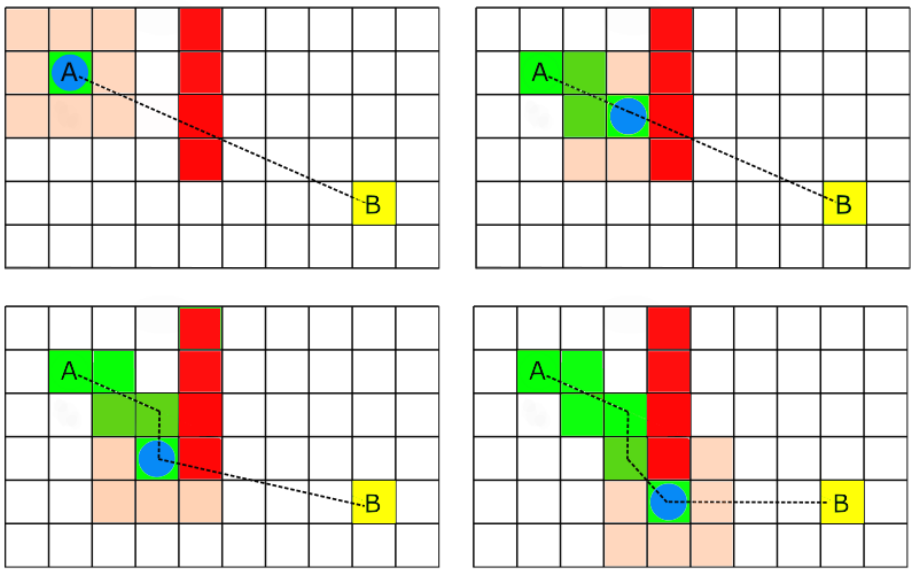


Figure 4.9: Concept for avoiding obstacles

4.8 Manufacture

The manufacturing process on the machines runs permanently. The time in the simulation is measured in ticks, as well as the production time on the machines. If a machine starts working on a product, the start time is saved. In listening 4.5 is a section of implemented code to see. It shows how the duration of the work step is determined. The variable ”Production Period” is defined in advance over the machine type. If a machine failure occurs as described in section 4.2.2, the start time is increased by the duration of the failure. As a result, the machine is occupied longer.

Listing 4.5: Calculation of the working time

```
Worktick =  RepastEssentials.GetTickCount()-Starttime;
Process Time Left = Production Period- Worktick;
if (ProcessTimeLeft == 0){
Product is finished
}
```

If a product is edited on a tool, the remaining working time can be queried. This is important for the selection of the transport time. Important for the machines is their status. An overview of the different states can be seen in Table 4.2. With the help of this state the tools to communicate with other agents or with the manufacturing control.

4.9 Model output

It is important for the evaluation of the model to generate and export observational variables. For this purpose, characteristics are recorded by the products, tools, and workers. Important for comparing different results is the acquisition of the simulation time and the simulation settings. The generation of the result data, as well as their statement, will be dealt with in this section.

4.9.1 Characteristics of the model

Most of the data collected are about production, products, tools, and workers. The information is collected during the simulation and output at the end. The information is written both in the console of the Java environment and a file. A section of such an output can be seen in listening 4.6. It shows how long the pure production time for a product is. As well as information as long as the product was the different production stations.

Listing 4.6: Extract from the data output about the products

```
Data Products
Production time all steps: 1350
Nr  ID  Start Finished  Dist     WaitTo  WaitTr Work  Trans   Stock
1   9   1     3196      472      144    128      1350  1574     0
2   16  1     3967      512      419    474      1350  1706     18
3   11  80    4442      491      755    247      1350  1708     303
```

The description of the characteristic values can be found in the following list.

- **Finished at time:** Time when the product is finished. At this time, the product has arrived at the end point and is therefore no longer part of the production process.
- **Distance:** Distance traveled during the manufacturing process.
- **Time at start point:** Time how long the product is at the starting point. From this point on, the production process begins.
- **Time wait at tool:** The time how long the product waits for transport on a machine. During this time, no worker has the job to transport the product.
- **Waiting time:** Waiting time is the product being picked up by a worker. The product is assigned to a worker. This value is the waiting time until the product is picked up by the worker. This includes all transport waiting times regardless of whether the product is picked up from the starting point of a machine or an intermediate storage facility.
- **Work time:** Time how long the product has been processed on machines. This value is normally equal to the sum of the required individual steps. However, this value can be increased by machine failures.
- **Transport time:** Time how long a product is on the way. This time it is transported by a worker to the different destinations.
- **Time in Stock:** This is the period of time how long the product is in a storage facility.

Also data about the workers are collected. Listening 4.7 shows an excerpt from the output file. In it, you can see the data of the workers. Thereby, three characteristic values are written down which are explained below.

Listing 4.7: Extract from the data output about the worker

```
Data Worker
ID      Dist loaded     Dist empty      No work
1       2058            543             138
2       1904            702             210
3       2078            1048            774
```

- **Distance without wafer:** This value is the distance the worker has traveled without product. It is the distance that has to be covered to pick up a product.

- **Distance with wafer:** This distance the worker has loaded a product.
- **Time without Work:** During this period, the worker does not have to do anything. There are no products available for transport.

In the last part of the output, information about the tools are collected. Listening 4.8 shows the data about the machines, the information of two machines can be seen. The information is similar to the Tool Status presented in Table 4.2.

Listing 4.8: Extract from the data output about the tools

```
Data Tools
ID Type Time  Products  Free  Work  Waiting Finished Break Break time
1  0    300   8         6261  2400  1227    1197     0      0
6  0    300   12        5756  3600  1308    421      0      0
2  1    150   11        7889  1650  431     1115     0      0
7  1    150   9         186   1350  556     493      1      4500
```

- **Production time:** The defined production time of the machine for a product.
- **Number of products:** The number of products processed on this tool.
- **Time free:** During this time, the machine had no work and is not being supplied.
- **Work time:** So much time has worked the machine.
- **Waiting time:** That's how long the machine has to wait for a planned product.
- **Time finished:** The machine is ready, and the product is ready for further transport.
- **Breakdowns:** This is the number of machine failures.
- **Breakdown time:** During this time, the machine was out of service.

4.9.2 Heatmap

Another important output is a heatmap. This heatmap shows where products are moving, or in other words, where workers are transporting products. This will show the moving density of the products. As a result, it can be determined in which areas a large product movement is. As a result, it can be ascertained about in which areas are how often how many products are.

Figure 4.10 shows an example of a heatmap. This output is generated in Repast Simphony and displayed in the GUI. The different colors represent the movement density. The color scale ranges from white to green and yellow to red. Wherein white means zero density and red density of 1016. This means at this point have 1016 products passed. This value is preset and may need to be adjusted. Also, the heatmap is still written as an ASCII raster in a file and can thus be processed with other programs.

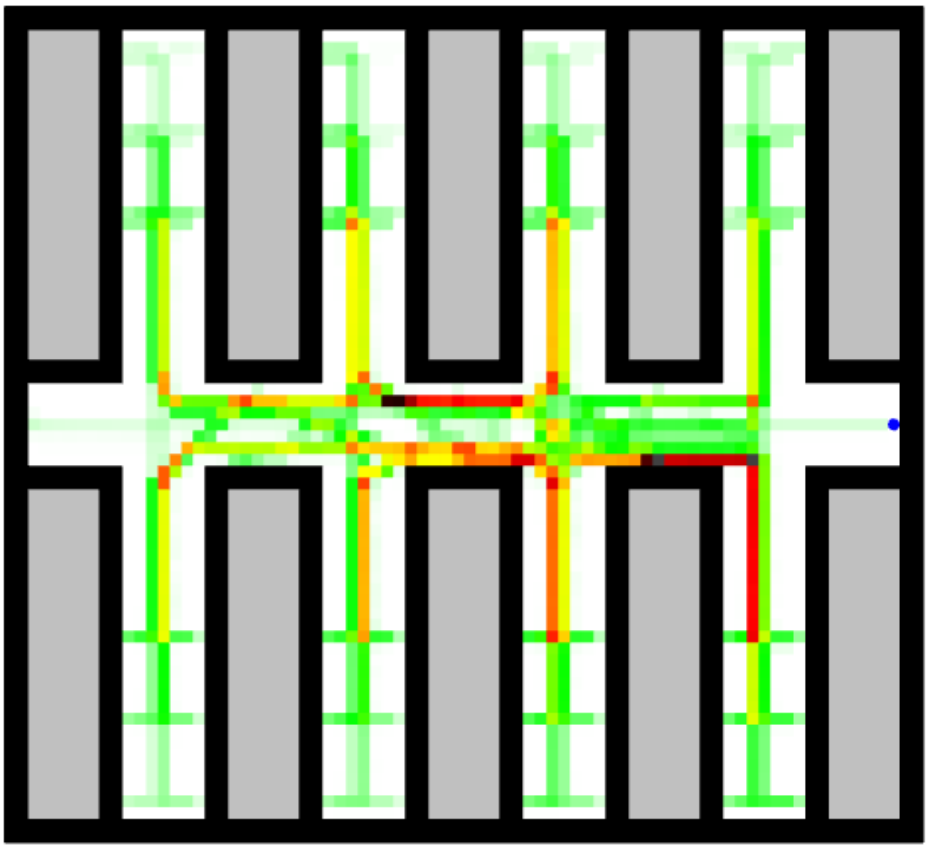


Figure 4.10: Example of heatmap

4.9.3 Time series

With Repast Simphony it is possible to create a time series during the simulation. This can be done in the GUI. The following time series has been created for the model. Or performance reasons, the time series are not displayed live, but are first created in post-processing. For further processing, the data can be exported as a *csv* file. With the help of all exported data, the results can be better visualized and analyzed with external programs.

- Number of products in the stock in the course of the simulation.
- Number of free machines in the simulation process.
- Number of finished tools during the simulation process
- Number of broken machines

Chapter 5

Generation of test data

This chapter deals with the generation of test data. With the help of the results from these data, the question of this thesis is answered. The chapter begins with the description of the data in Section 5.1. Subsequently, the validation and calibration of the model and the data.

5.1 Test data

At the beginning of the project, it was planned to obtain test data from industry, but this was not possible. Therefore, to test the own model data is generated. The model requires the following data:

- **A layout of the production environment:** This layout contains the complete structure of the environment. On the one hand walkable areas such as paths, corridors, and open spaces. But also not based objects such as walls or obstacles. Another important factor in setting up the production environment is the position of the machines and intermediate storage. For the navigation of the workers also a graph network is needed.
- **Definition of the products:** For the products, a list of necessary work steps and transport speed are defined.
- **Definition of the machines:**The position and the type are determined by the environment layout. For this, the production time for a work step on the machines is still needed.

It can be seen from the paper of Scholz and Schabus (2014b) that the construction of semiconductor production is in the form of corridors in which the machines are located. Based on this information, a test layout is accompanied. The created layout is shown in Figure 5.1. In the picture, the walls are black and impassable areas are gray. Tools are green, with four tools of the same type in each row. This results in 16 different machine types. The six squares in purple in the middle of the layouts are intermediate storage.

The interim storage is therefore placed at the main corridor in front of the aisles with the tools. The point in magenta is the starting point, from here begins the production chain of the products. At the blue square on the right side is the end point, here the finished products are delivered. The blue circle on the green square is the starting point of the works. At this point, the workers are placed at the initialization of the simulation. The generated map of the production environment has a size of 78 x 71 pixels. With a maximum of one pixel per step and a step length of one meter, the factory floor is 78 x 71 meters.

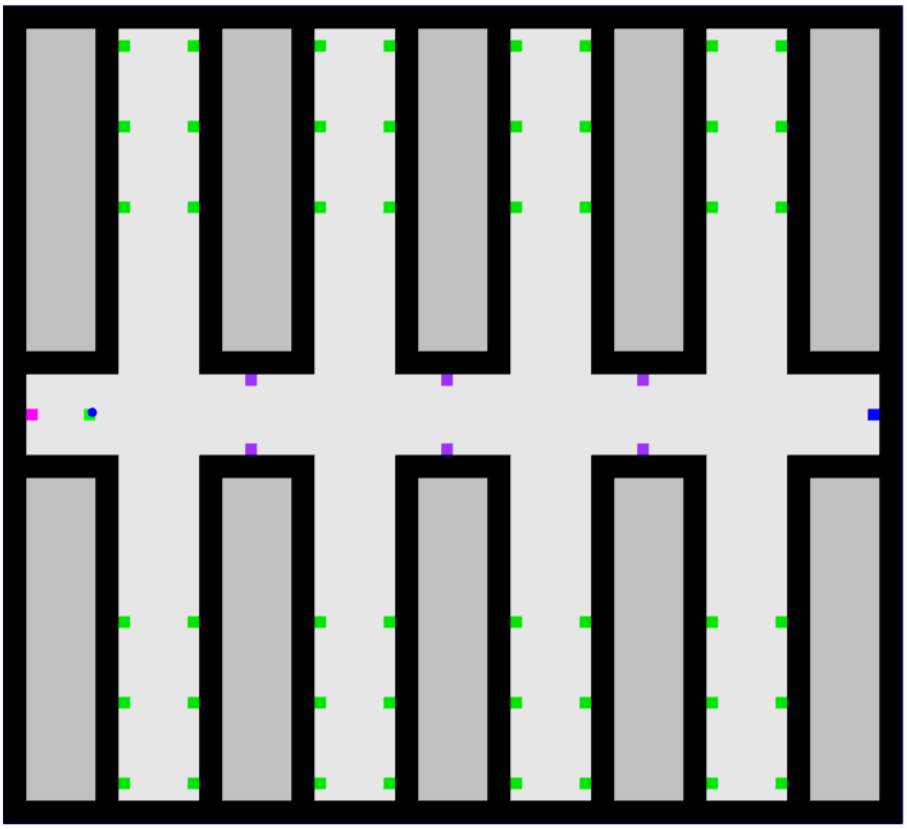


Figure 5.1: Test environment for the model

The list of the work steps for the products were created randomly. For a product, 100 work steps must be performed. For this purpose, the types of machines defined in the card are randomly arranged one after the other. The duration of one step on the machines was also randomly selected.

5.2 Verification, calibration and validation

The verification, calibration, and validation of the created ABM is an important part of the modeling process (Crooks & Heppenstall, 2012). This is clear when looking at Figure 5.2. Creating the model is an iterative process. The concept and the implementation are adjusted until the model results are satisfactory. In the literature. The basic process for the evaluation of ABM consisting of **verification**, **calibration** and **validation** has established itself (Crooks, Heppenstall, & Malleson, 2018). There are several approaches for validation in the literature.

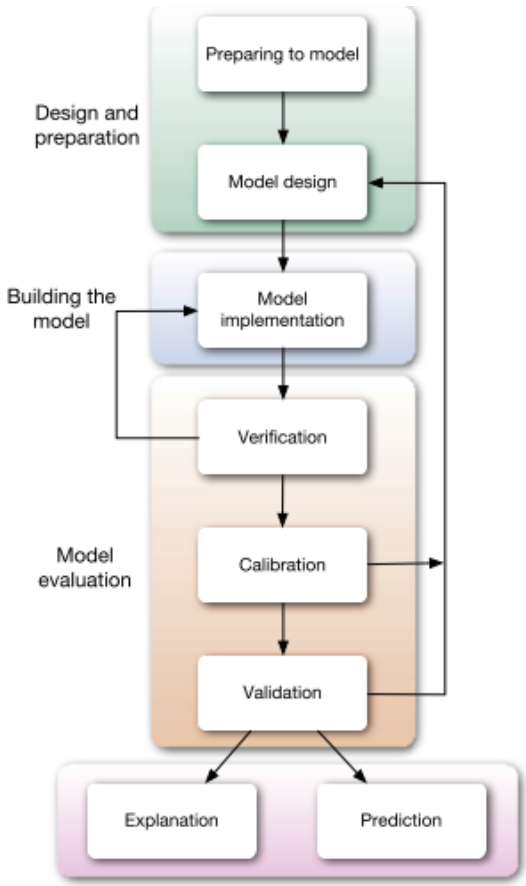


Figure 5.2: Presentation of the modeling process (Crooks, Heppenstall, & Malleson, 2018)

Verification

Verification is the process that ensures that the implemented model corresponds to the model design or that the model has been correctly programmed (Crooks & Heppenstall, 2012). Checking whether the model behaves as expected is called internal validation. The internal validation is more difficult to perform because it must be evaluated why the model comes to a result. However, it is often difficult to say whether the unexpected result is a programming error or a logical error in the model. One way to overload the system is to bring the model in extreme situations where the results can be predicted more easily (De Smith et al., 2015). An additional difficulty is that the simulations are mostly based on random numbers. Thus, repeated simulation runs can lead to different results. Therefore, the model will be tested component by component to check the function of each function. The verification of the model can take more time than the actual modeling and programming (De Smith et al., 2015; Crooks et al., 2018). The validation of the model created in this project was traced back to a simple test environment in which the actions of the agents can be easily tracked and controlled. The test environment for verifying the model is shown in Figure 5.3. The environment has five different machine types and three intermediate storage. At each machine type, a work step is performed. Also, a bottleneck is integrated into the layout. The two machines on the right-hand side have a much longer working time than the rest, followed by a single but fast machine. Due to this structure, the workflow is heavily influenced, and the intermediate storage facilities must be used.

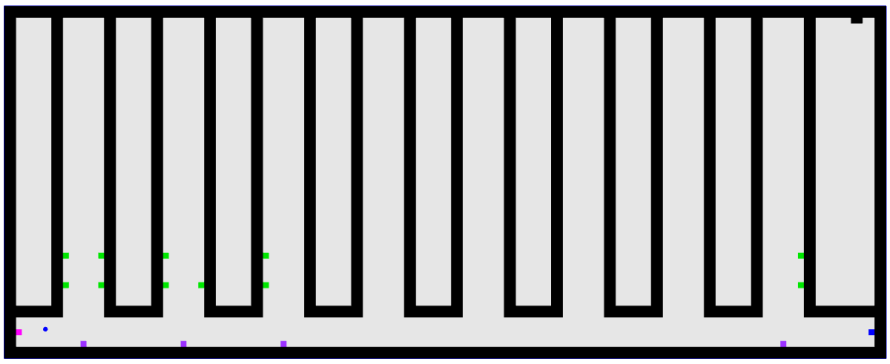


Figure 5.3: Test environment for model verification

Table 5.1 shows some settings for testing the system. In this case, the weightings are tested. In all test runs In all runs, 20 products are manufactured with five workers. There is room for five products in the interim storage

facilities. For the comparison, the characteristic data of the products are summed up from the simulation and compared with each other. Depending on the simulation settings, these results should change. Test number one is the default setting and serves as a reference value. All weights are set to zero and the waiting time at the tools is set to 100 ticks. In the second column of the table, products produced on machines are given priority. As expected, this reduces the waiting time for finished products. As the next test, products which are at the starting point are preferably transported. This is also reflected in the results. In column number 4 are products which are in a warehouse should be treated with higher priority. In the next test, products that are currently being processed in machines are given preferential treatment. Since the workers are already waiting at the machines until the products are ready, waiting time on finished machines is reduced to zero. But also the waiting time for transport is reduced. At the last attempt, the waiting time for the product exchange is set to zero. This does not result in any extreme values. However, the overall speed is lower than with a higher waiting value.

Table 5.1: Settings of runs with extreme values

Test Nr.:	1	2	3	4	5	6
Weight Start point:	0	0	-500	0	0	0
Weight finished:	0	-500	0	0	0	0
Weight tool:	0	0	0	0	-500	0
Weight stock:	0	0	0	-500	0	0
Exchange threshold:	100	100	100	100	100	0
Start point [ticks]	25788	12488	**9312**	32449	86991	15716
Wait tool [ticks]	16805	**7542**	17951	16416	**0**	17513
Wait transport [ticks]	6932	9343	10448	8338	**3578**	10668
Work time [ticks]	27000	27000	27000	27000	27000	27000
Transport time [ticks]	34093	32085	31859	34377	33033	31437
Stock time [ticks]	21131	41859	35551	**15017**	9034	33980
Distance [LE]	10055	10042	10083	10087	9855	10095
Simulation ticks [ticks]	8934	8960	8974	9790	13878	9565

This is one way to verify the entire model. Other functions, such as the motion model or the correct sequence of work steps, were tested during the implementation of the individual program parts.

Calibration

During calibration, the model parameters are set so that they correspond to the real system to be depicted. This process requires data on which the model is based (De Smith et al., 2015). The model adjusted using the calibration observed data. This process is necessary because the theoretical assumptions for the general structure of the model are not precise enough and therefore still need to be adjusted (Crooks et al., 2018). The calibration process is usually iterative and is usually repeated until the results of the model correspond to the real collected data. If the result of the model cannot be adapted to the real values, it is necessary to redesign and program aspects of the model. Thus the calibration can be seen as verification of the model. How a model corresponds to the real world depends on the purpose of the model (De Smith et al., 2015). Calibration is not yet a validation, but it can be seen as a fine-tuning in which the optimal model parameters are found (Crooks & Heppenstall, 2012). Typically, there are two approaches to calibration, either quantitative or qualitative. Quantitative calibration calculates the difference between the simulation output and the real data. In qualitative calibration, the model results are evaluated using human intuition. This method is used in spatial models with GIS and spatial data visualization (Crooks et al., 2018). Since no real data are available for the model created in this work, the calibration is performed only on the plausibility of the results.

Validation

The validation of a model concludes the evaluation of the model. The goal of validation is to prove that the model is sufficiently accurate. The traditional approach for validation is based on the calibration approach (Guerini & Moneta, 2017). Since there is a risk that the model will be adapted to the data during calibration, validation is performed with a new or different data set. This condition is achieved when the model can reliably replicate real conditions without further calibration. Validation is carried out according to the same qualitative and quantitative methods as calibration (Crooks et al., 2018). In the work of Guerini and Moneta (2017), however, this validation method is seen only as a first step. Despite the use of different data for cal-

ibration and validation, the model may be adapted to the data. Therefore, this process is improved to prove that the modeled data generation mechanism corresponds to the real data generation mechanism. In the approach of Guerini and Moneta (2017), the causal structures are estimated by the model or the real world and then compared with each other. This allows the ABM to be compared with empirical data as well as different models. Brown, Page, Riolo, Zellner, and Rand (2005) presents a further approach for the validation and spatial in particular land use models. Measurements of the spatial similarity of model results and reference data provide information about the accuracy and variability of the model.

Chapter 6

Results

In this chapter, the results of the experiment and the answers to the scientific question that are asked in the introduction will be given. The questions of the work are answered on the basis of the results from the model. For this purpose, the simulation is executed in different settings, and the results are displayed. The following subjects are covered in this chapter.

- General scenario: The data obtained and the analysis possibilities of this data are shown here.
- Effect of additional information on simulation: This section explains how storage and spatial information affect the simulation.
- Improvement of the simulation scenario based on the results: This section discusses how to improve the simulation result from the simulation data.

6.1 General Scenario

In this section, the question is to be clarified whether ABM is able to reproduce a semiconductor production. Therefore, settings corresponding to those of a semiconductor production are selected for this simulation. As stated in the paper of Scholz and Schabus (2014a), several hundred work steps on different machines can be required to finish a product. The environment described in Section 5.1 is used for the simulation. The simulation settings are listed in Table 6.1. On the basis of this setup, the data obtained and thus possible analyses of the model are shown.

Table 6.1: Simulation settings

Worker:	10
Products:	100
Stock capacity	15
Weight start point:	0
Weight finished:	0
Weight tool:	0
Weight stock:	0
Exchange threshold:	100
Breakdown time :	1500
Breakdown multiplier:	4
Breakdown random pool:	15.000

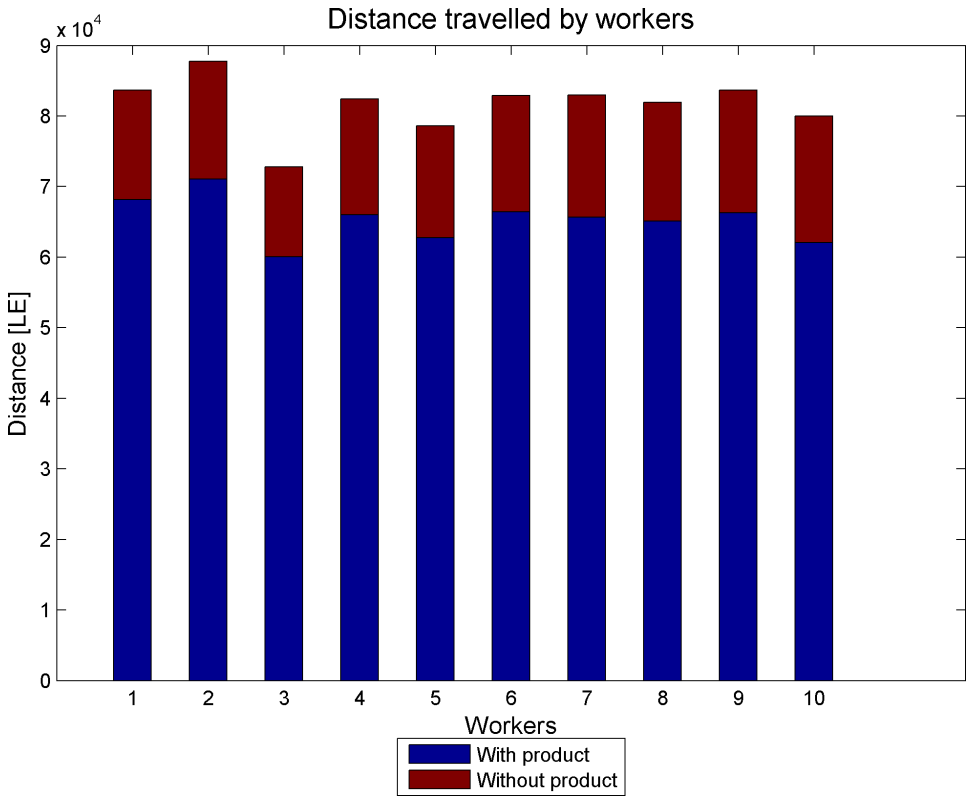


Figure 6.1: Distance travelled by workers

Workers transport products through the production facility. Figure 6.1 shows the distance traveled by the workers. A distinction is made between working with and without a product. It can be seen that the workers cover about eight times the distance loaded. Thus it can be said that the workers do not move a large part of the route empty. This would be in a real factory in the cost factor. Figure 6.2 shows where the workers moved with the products. It is to be recognized that above all in the main middle corridor as well as at the beginning of the machine aisles an increased movement takes place. This can be traced back to the fact that on the one hand, the front area has to be passed to reach the machines behind it, but also due to the search for the nearest machine. What can also be seen from this display is that there is very little movement in front of some tools. From this, it can be concluded that these machines work less compared to the other machines.

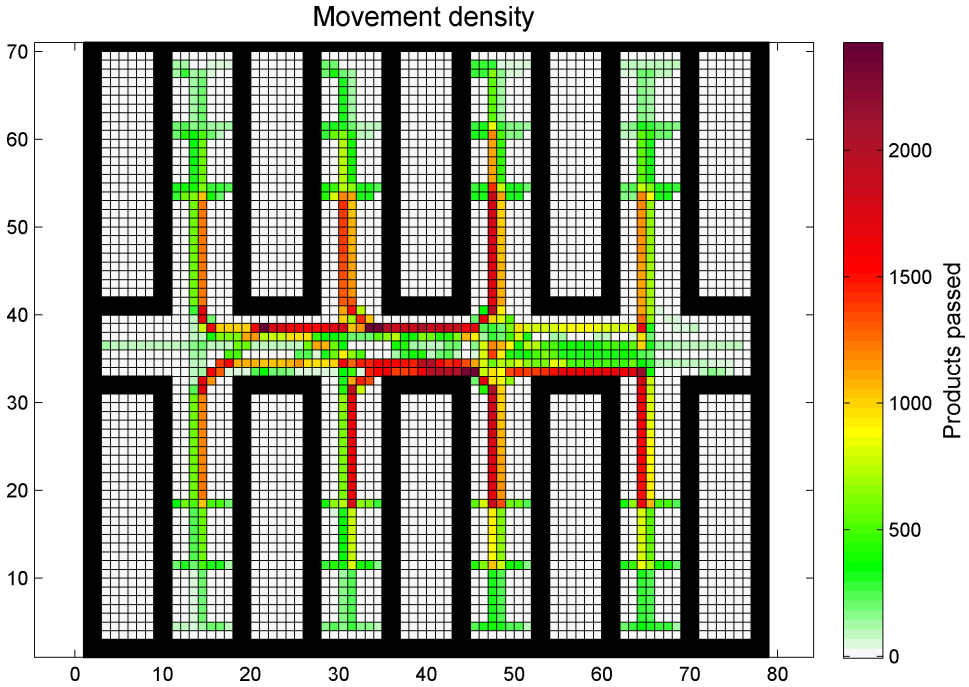


Figure 6.2: Density of product movement

In the course of the simulation the machines have different states, these are: *free*, *working*, *waiting*, *finished* and *broken*. How long, in relation to the total simulation time, the tools are in the different states are shown in Figure 6.3. The tools are grouped by type. The less a machine has worked, the longer the green bar. The waiting time (yellow bar) depends on how far the worker has to travel to the machine. The waiting time for a finished machine to be selected as an order for a worker is shown in red. The blue bar is the working time and depends on how many products are manufactured on this

machine.

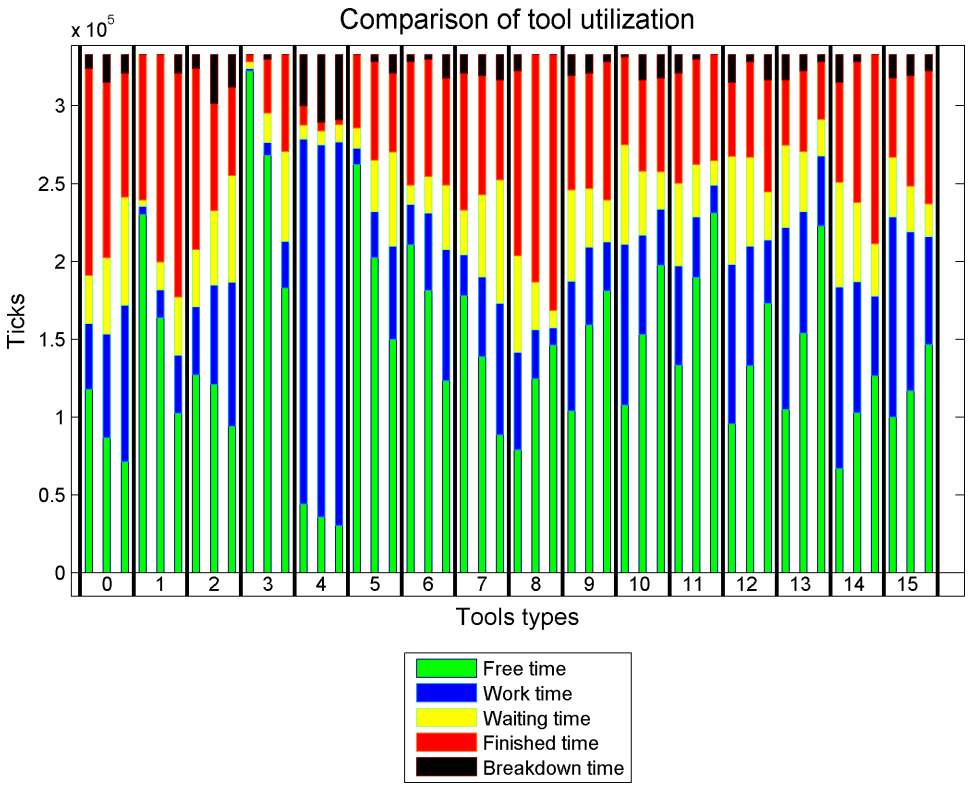


Figure 6.3: Status of the tools during simulation

The time course of products in the warehouse and some tool states can be seen in Figure 6.4. The course of the simulation can be deduced from the time series. Thus, like the blue plot to see the bearings after the first half of the simulation only almost no longer needed. The number of free machines is roughly constant in the middle part of the simulation run. During this period, the simulation runs on all machines. This can also be compared with Figure 6.1. This lists the production time of the products. Each blue line indicates the time the product has been removed from the starting point until it reaches the end point. It is to be recognized that towards the middle of the course the first products are finished and thus the production runs on all machines. As expected, the number of free machines increases again towards the end of the simulation. The third plot shows the finished tools. This value also remains approximately the same, except for simulation start and end. The last figure shows when machines are broken.

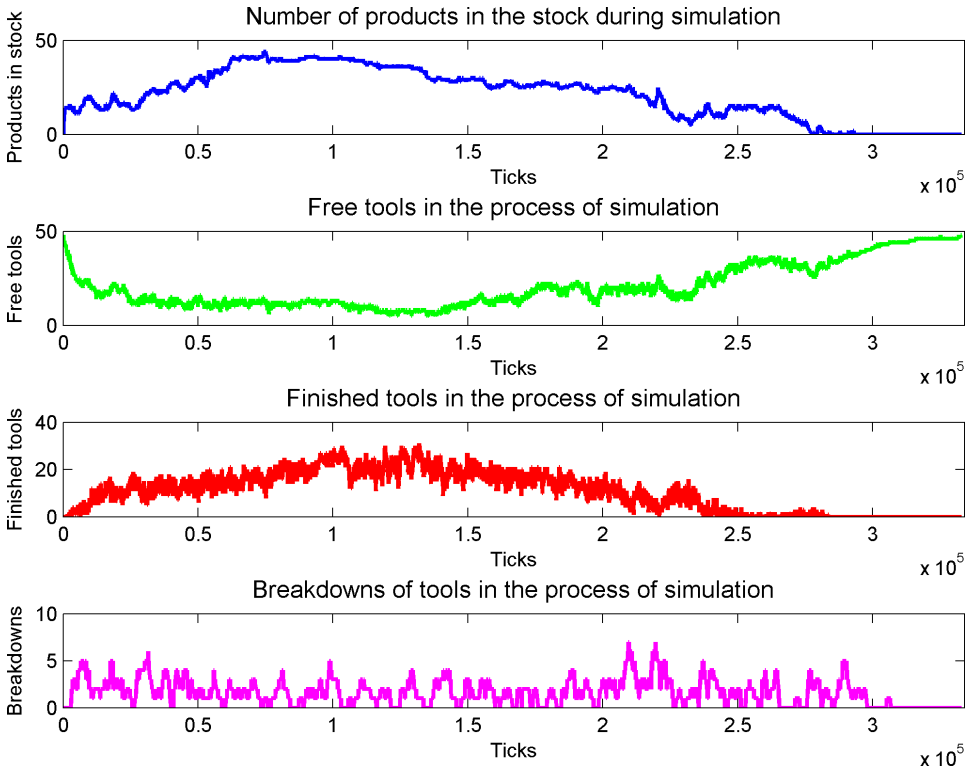


Figure 6.4: Time series of machines and stocks

The time how long a product is in production can be seen in Figure 6.1. The products are sorted by production entry when they leave the starting point. It is to be recognized that from this start time it cannot be concluded at the end of production. So products which later go into production can be finished quite earlier.

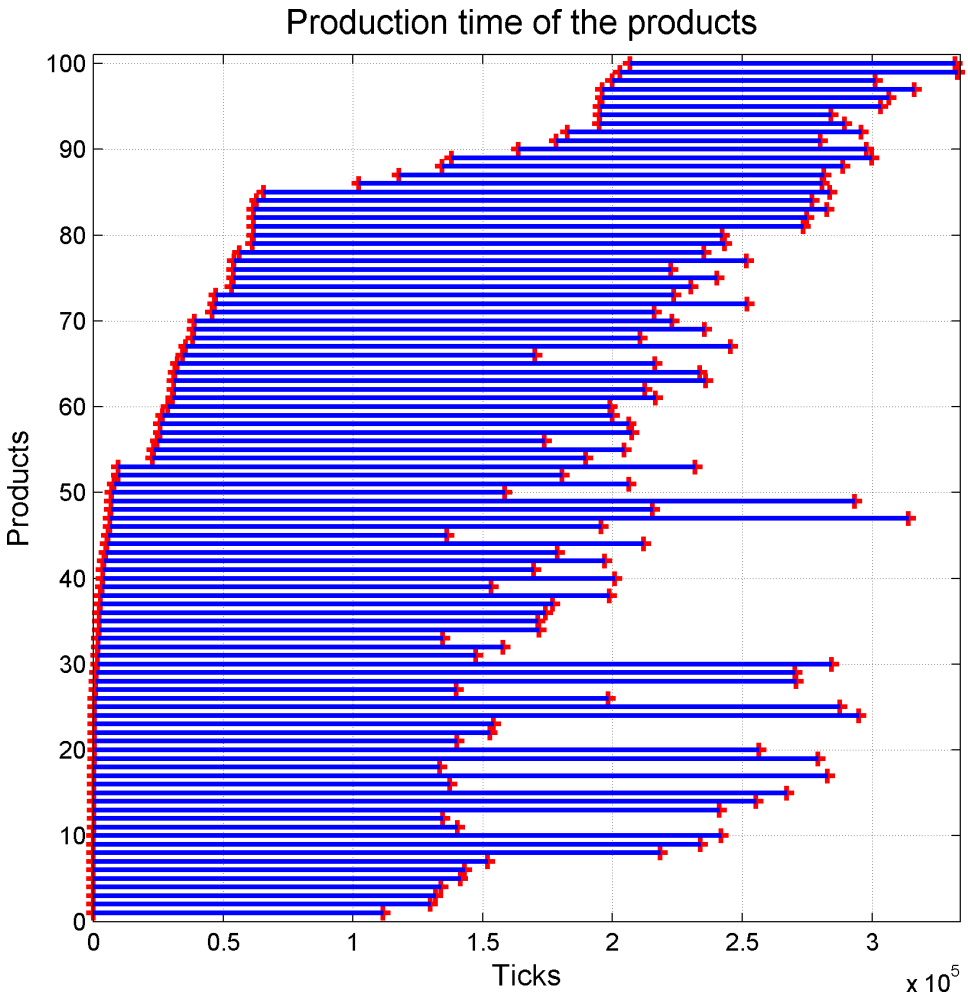


Figure 6.5: Production time of the products

In the last figure 6.6, the products are treated in more detail. The order of the products is the same as in previous plot 6.5. However, here the line spaces are shown, how long are the products in the different production stages. The working time is the same for all products and was therefore not shown in the plot. The transport time is approximately the same for all products. This is plausible, as all products require the same work steps on adjacent machines. The largest proportion is the waiting time, which is the time in which the product is finished on a machine without until it is assigned for further transport by a worker. This value decreases with a higher number of workers. The waiting time for transport is the time from the assignment for transport to the arrival of the worker at the product. This value depends on the distance between worker and product. The last two values are the time in a stock and the time caused by machine failures.

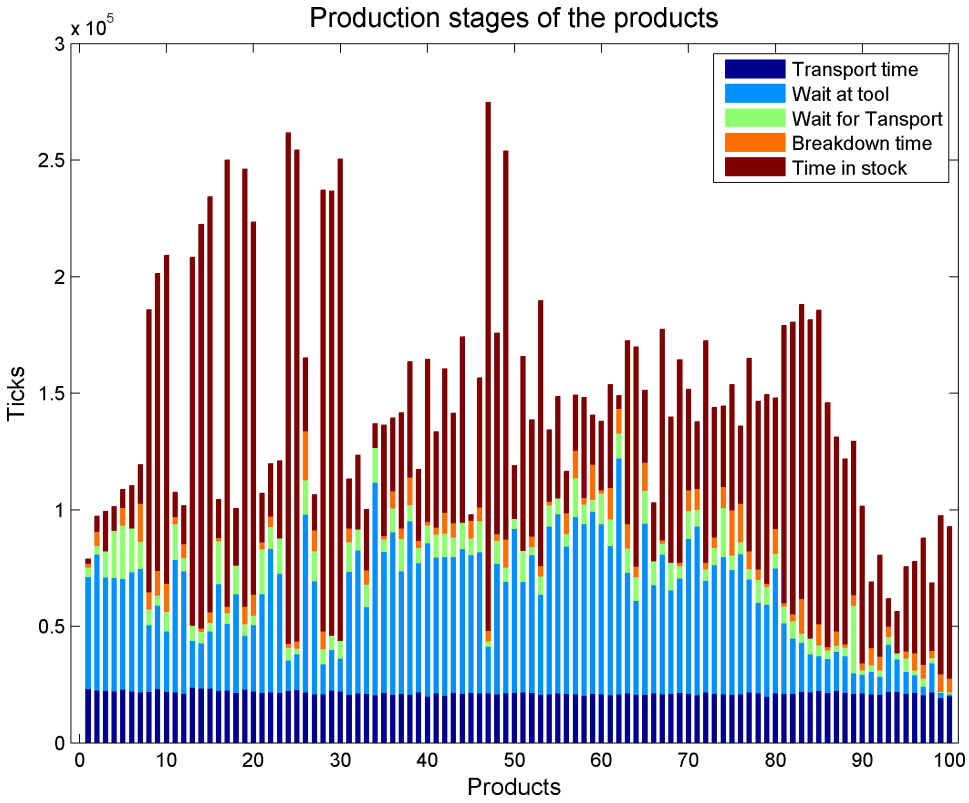


Figure 6.6: Status of products during simulation

6.2 Effect of additional information on simulation

In the model, the simulated production process is accelerated using distance information and production time data on the machines. The following results are intended to clarify how this data affects the simulation and whether it has an advantage. To be able to compare the results of the individual tests better, the random machine failures are deactivated. 50 products are manufactured with the same settings as shown in Table 6.1. The simulation is made with the following settings.

- **Normal:** During this run, workers, have spatial information about products and tools at their disposal. Also, the running times of the machines can also be queried and included in the planning. Interim storage facilities are also available. This represents the developed optimal case.
- **Without stocks:** The same information is available to the workers as in the *normal* case, but the stocks must not be used.

- **Spatial information products:** The workers, have no spatial information about the machines during this run. However, the interim stocks can be used again.
- **Spatial information tools:** Now the workers lack spatial information about the products. The first best product is always selected for transport, regardless of the distance.
- **Without spatial information:** No spatial data are available for this test, neither for the machines nor the products item **Random:** The products are selected randomly from all available products. Machines are selected normally.

The results of the experiments are shown in Table 6.2 and Figure 6.7. This shows the total runtime of the simulation and the difference between the runtime and the runtime with the *normal* settings. The results are sorted in ascending order. Since the model is also subject to a certain coincidence, the values were averaged from three runs. The time required depends on the information available to the model. The model is designed to retrieve data from all actors. If this data is not available or only partially available, the system can no longer run optimally.

Table 6.2: Simulation settings

Setting	**Normal**	**No stocks**	**Spatial info. products**	**Spatial info. tools**	**No spatial info.**	**Random**
Runtime	156030	164388	165270	171104	178541	181722
Runtime differenz	0	8358	9240	15074	22511	25692
Increase in duration		5,4%	5,9%	9,7%	14,5%	16,5 %

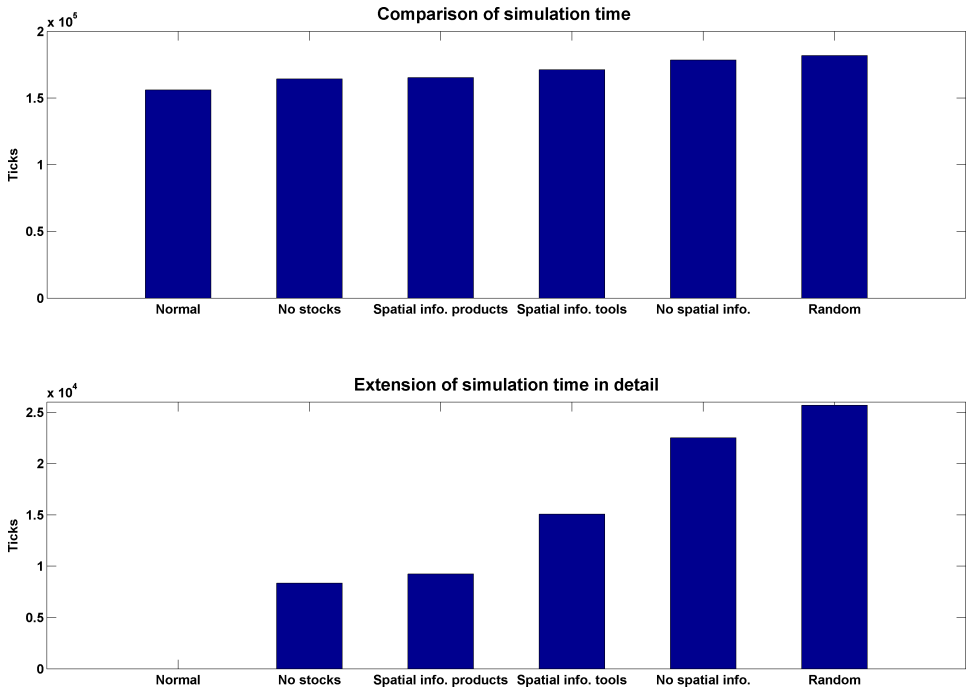
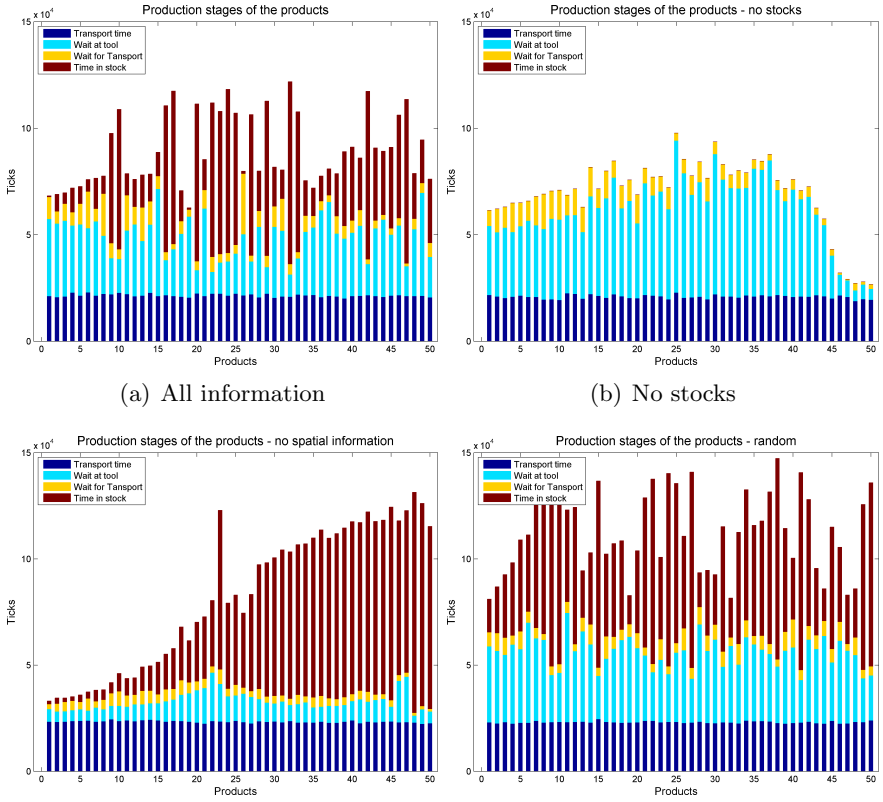


Figure 6.7: Comparison of runtime of simulation scenarios

The effect of the missing information on the simulation is now explained. Normal results are used as a reference for the comparison. These are compared with the settings *no stock*, *without spatial information* and *random*. Figure 6.8 shows the generated product data. These show how long a product has been in a specific production stage. It is to be recognized how missing model structures or data affect each other. In the upper left corner (6.8(a)) you can see the fastest variant. The bearings are activated, the workers can request distance information and use this data to cover the shortest possible distances. Compared to the test without bearing (6.8(b)), the products are longer in the production cycle, but it must be considered that products in the stocks are in this cycle but do not occupy workers or machines. Due to the stocks, the waiting time (light blue) is less. However, it cannot be generally claimed that short waiting times are equivalent to a shortened production time. This is shown in Figure 6.8(c). Compared to the other experiments, the waiting time is the shortest here. As the course of the products shows, the order in which the products are finished corresponds approximately to the order in which they start production. This is because the distance from the worker to the product is not taken into account when selecting the products. The only criterion for the selection is the production time on the machines. This usually makes the first product in the internal list of products the next best. If the products are selected randomly for transport (6.8(d)), the waiting times but also the storage times and thus the

total production time increase.

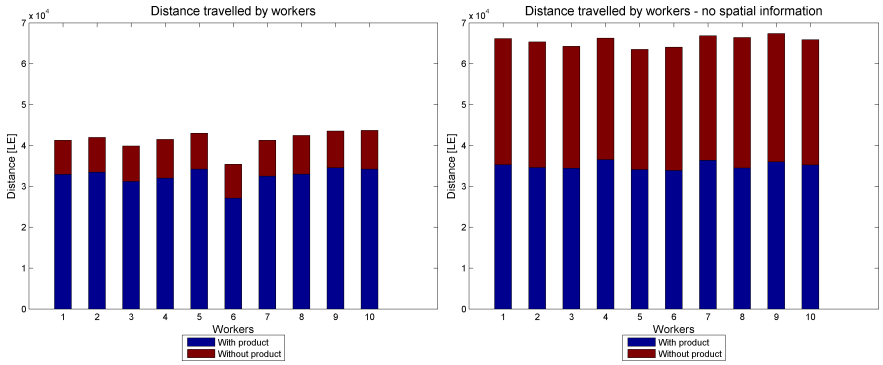


(a) All information (b) No stocks

(c) No spatial Information (d) Random

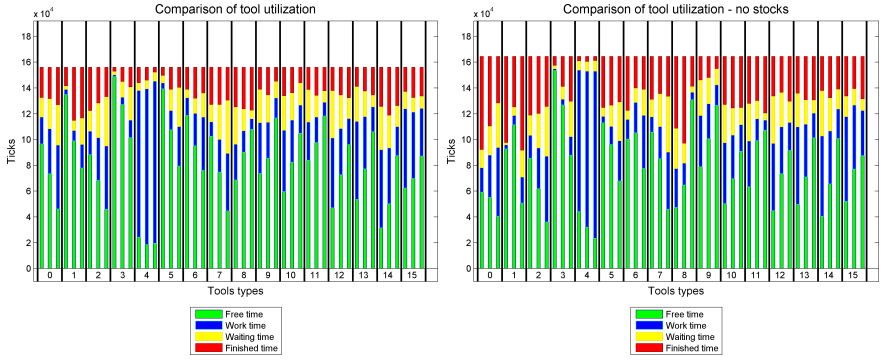
Figure 6.8: Comparison of product data from four test runs

The influence of distance on the choice of the next product and transport targets becomes clear when looking at Figure 6.9(b). This plot shows the distance traveled by the workers. It is to be recognized that without distance information to the destinations, the workers are on their way empty about half of their distance. Thus it can be said that workers can use distance information in their decision to choose their routes more effectively. On the basis of the machine data, it is difficult to make any statements about the workload. The machine data can be seen in Figure 6.10. The most striking differences to the reference data (6.10(a)) can be seen in Figures 6.10(c) and 6.10(d). When evaluating without spatial information, you can see that the machines are free for a longer time and that the finished time is shorter. This is due to the fact that the products are produced in one sequence in this test setup. In the picture in the lower right corner, it can be seen that by random selection of targets, all machines of the same type have approximately the same load. And no product is given preferential treatment.

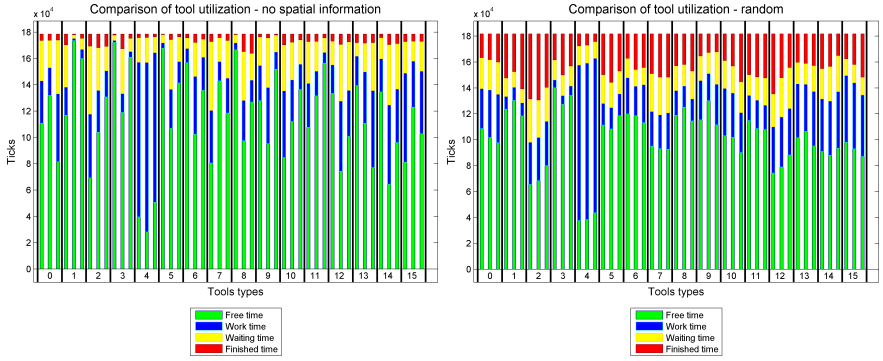


(a) normal (b) No spatial information

Figure 6.9: Comparison of worker behavior



(a) All information (b) No stocks



(c) No spatial Information (d) Random

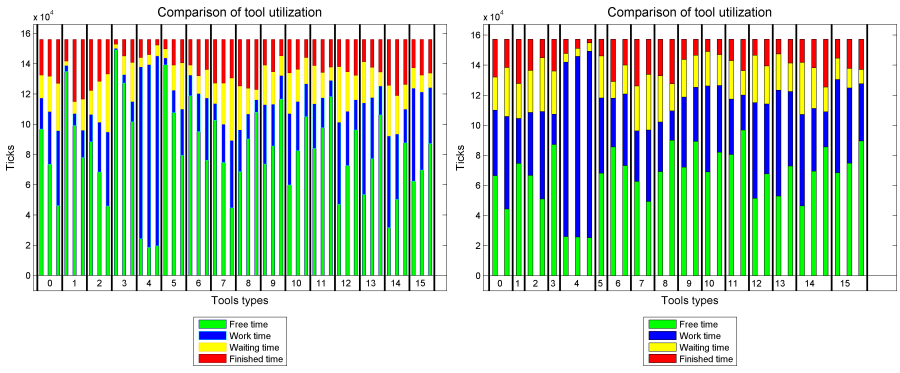
Figure 6.10: Comparison of tool data from four test runs

6.3 Improvement of the simulation scenario based on the results

The model provides a lot of data about the production process. An important aspect is to use this data to improve the overall process to save costs and resources. For optimization, the initial data from the simulation are analyzed. The aim is to improve simulation time with fewer tools. The number of workers is assumed to be constant. For this purpose, the data on products and tools are analyzed. In Figure 6.11(a), the initial situation is to be desired, whereby it is to be recognized that some tools are not used a large part of the time. Tools that are free for most of the time are removed, and the simulation is then called up again. This is an iterative process as the effect on the other machines is not exactly predictable. The first step in this example is to remove a type 1 and three machines. Table 6.3 shows the effects of these iterative processes. The machine is constantly being reduced. The basis for this is the simulation data.

Table 6.3: Comparison of number of machines and simulation time

Number of tools	48	46	43	38	34	32	30
Runtime	156030	152989	153346	152058	154716	157188	166477



(a) Utilization of the tools as a starting point for optimization (48 tools)

(b) Utilization of the tools after the optimization (32 tools)

Figure 6.11: Comparison of tool data

The reason for free machines can be on the one hand because the machine works fast and therefore fewer machines are needed, but on the other hand also because the machine is not needed so often in the working process.

Furthermore, this type of machine can only be required at the beginning or end of the production chain. The absence of such a machine then affects the operating time. In the course of the experiment, the number of tools is reduced from 48 to 32, while the runtime remains approximately the same. Figure 6.11(b) shows the utilization of the tools after the reduction. It is to be recognized that the working time has increased and the idle time is reduced. The model provides other data to optimize the system. For example, the number of workers can be varied, or settings can be made using the weightings. Another possibility would be to position the machine differently in the factory hall. Due to the object based construction of the model other data can also be extracted from the model, so every machine, every warehouse, every worker or every product can be observed individually. Thus the ABM offers enough data to optimize a production process.

Chapter 7

Summary and outlook

In this chapter, a summary of this thesis and an outlook for possible extensions or tasks according to this thesis and the resulting Agent-based modeling and simulation (ABMS) is given.

7.1 Summary

The goal of the thesis was to create an agent-based model of a semiconductor production to simulate a production process. The approaches from industry 4.0 and Smart factory were to be combined with the properties of ABM. To achieve this goal, the necessary literature on this topic was first prepared and presented. The similarities between the idea of a smart factory and an ABM have converged. Based on the theory, a concept for modeling a semiconductor production environment using ABM was developed. In the concept shown, the production process takes place without a central control unit. This decentralized approach is an essential part of this work, as it weeps away the concepts of ABM and Industry 4.0. The decisions of which product is to be transported and where it is to be taken are adapted to the respective situation. With the aim of keeping the distance and waiting times short. In the chosen approach, all actors involved are executed as agents or as objects, allowing each actor to act independently on the basis of defined rules. Workers can query machines and products and make decisions based on a set of rules. Thus, empty distances can be reduced. In addition, machine data and information about the stocks can also be retrieved. Products also know where they are and what their next step is, as this corresponds to the idea of a smart factory. The created model has all important components of production, workers, machines, warehouse, and a production environment. The structure makes it easy to add additional rules or tests other scenarios. Subsequently, it was shown that the complicated model can simulate a production process. The model is equipped with many adjustment possibilities, which should allow versatile use. It was shown that the model is designed in such a way that the production process runs as effectively as possible. With the help of the generated data from the simulation, it is possible to optimize the initial situation. Unnecessary

production components can be identified from the data. Furthermore, by analyzing the movement of the workers and the machines, the production ballast can be further optimized. The ABM model based on object-oriented programming offers the possibility to extend the model as desired and to improve the model or to generate other output data.

7.2 Outlook

The development of the approach chosen in this thesis is by no means complete. It is primarily a prototype, improvements and extensions are therefore possible and also necessary. The most important point here is the reference data from practice. The model created lacks calibration and validation. Validation is a central point in the modeling of a system. Without it, it can be shown that the model does not work, however, whether it corresponds to the real system. Therefore, comparative data is the most important point for further improvements to the model. Based on the findings of a correct validation, the model can be further improved. With the help of this improved model, production processes from the real world can then be simulated and improved. In further consequence, the production process could be further improved. In the prototypes implanted in this thesis, the machines perform a specific work step, which takes a specified amount of time. This implementation could also be refined. To better adapt the amount of work to the individual machines to the real conditions, depending on the production process. Another extension would be the implementation of different products. The products can differ not only between different work steps but also by different transport speeds. Also, the implemented movement model could be further improved so that it corresponds to that of a real man. Another approach is to see the agents who give the property into the future. This could take the form of workers communicating with each other, exchanging information about when they arrive at their destination and what they have planned next. The production environment could also be expanded to cover a larger area of production. Doors could be integrated with air locks and staircases. Different areas with different movement speeds or rules of behavior would also be conceivable. However, the most important extension in the sense of Geographic information science (GIScience) would be a spatial optimization, based on the validated model. To finally improve the real production process with the help of the model or the simulation. This allows the process to be simulated and optimized even before the start of production. With such an approach, problems in the production process can be detected at an early stage before a real product is produced.

Bibliography

Abdou, M., Hamill, L., & Gilbert, N. (2012). Designing and building an agent-based model. *Agent-Based Models of Geographical Systems*, 141–165.

Albach, H., Meffert, H., Pinkwart, A., & Ralf, R. (2015). Management of permanent change. *Management of Permanent Change*, 1–240. doi:10.1007/978-3-658-05014-6. arXiv: arXiv:1011.1669v3

Balietti, S. (2012). *Social Self-Organization*. doi:10.1007/978-3-642-24004-1. eprint: arXiv:1011.1669v3

Bonabeau, E. (2001). Agent-based modeling: methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, *99*(suppl. 3), 7280–7287. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/12011407

Brown, D. G., Page, S., Riolo, R., Zellner, M., & Rand, W. (2005). Path dependence and the validation of agent-based spatial models of land use. *International Journal of Geographical Information Science*, *19*(2), 153–174. doi:10.1080/13658810410001713399

Cantamessa, M. (1997). Agent-balsed modeling and management of manufacturing systems. *Computers in Industry*, *34*(97), 173–186. doi:10.1016/S0166-3615(97)00053-5

Collier, N. & North, M. (2013). Repast java getting started, 1–36. Retrieved from http://repast.sourceforge.net/docs/RepastJavaGettingStarted.pdf

Cooley, P. & Solano, E. (2011). Agent-Based Model (ABM) Validation Considerations. *SIMUL 2011, The Third International Conference on Advances in System Simulation*, (100), 134–139.

Couclelis, H. (2000). Chapter 2 : Modeling frameworks , paradigms , and approaches, 1–15.

Crooks, A. (2009). Agent-Based Modeling and Geographical Information Systems. *A Practical Primer on Geostatistics*, 348. Retrieved from http://pubs.usgs.gov/of/2009/1103/

Crooks, A. & Castle, C. (2012, January). The integration of agent-based modelling and geographical information for geospatial simulation, 219–251.

Crooks, A., Castle, C., & Batty, M. (2008). Key challenges in agent-based modelling for geo-spatial simulation. *Computers, Environment and Urban Systems*, *32*(6), 417–430.

Crooks, A. & Heppenstall, A. (2012). Introduction to Agent-Based Modelling. *Agent-Based Models of Geographical Systems*, 85–96.

Crooks, A., Heppenstall, A., & Malleson, N. (2018). Agent-Based Modeling. *Comprehensive Geographic Information Systems*, (December), 218–243. doi:10.1016/B978-0-12-409548-9.09704-9

De Smith, J. M., Goodchild, M. F., & Longley, P. (2015). Geospatial analysis: a comprehensive guide to principles, techniques and software tools - 5th edition. Retrieved December 31, 2017, from http://www.spatialanalysisonline.com/HTML/

Drath, R. & Horch, A. (2014). Industrie 4.0: hit or hype? [industry forum]. *IEEE Industrial Electronics Magazine*, *8*, 56–58.

Gilbert, N. (2007). Agent-Based Models, 1–20.

Gilbert, N. & Bankes, S. (2002). Platforms and methods for agent-based modeling. *Proceedings of the National Academy of Sciences*, *99*(Supplement 3), 7197–7198. doi:10.1073/pnas.072079499

Grimm, V. (1999). Ten years of individual-based modelling in ecology: what have we learned and what could we learn in the future? *Ecological Modelling*, *115*(2), 129–148. doi:https://doi.org/10.1016/S0304-3800(98)00188-4

Grimm, V., Berger, U., DeAngelis, D. L., Polhill, J. G., Giske, J., & Railsback, S. F. (2010). The ODD protocol: A review and first update. *Ecological Modelling*, *221*(23), 2760–2768. doi:10.1016/j.ecolmodel.2010.08.019

Grimm, V. & Railsback, S. S. (2012). Designing, Formulating, and Communicating Agent-Based Models. *Agent-Based Models of Geographical Systems*, 361–377.

Guerini, M. & Moneta, A. (2017). A method for agent-based models validation. *Journal of Economic Dynamics and Control*, *82*, 125–141. doi:10.1016/j.jedc.2017.06.001

Helbing, D. (2012). Agent-based modeling. *Social self-organization*, 25–70.

Henning, K., Wolfgang, W., & Johannes, H. (2013). Recommendations for implementing the strategic initiative INDUSTRIE 4.0. *Final report of the Industrie 4.0 WG*, (April), 82. doi:10.13140/RG.2.1.1205.8966

Heppenstall, A. & Crooks, A. (2016). Agent-based modeling in geographical systems -AccessScience from McGraw-Hill Education Agent-based modeling in geographical systems, 1–9. doi:10.1036/1097-8542

Hermann, M., Pentek, T., & Otto, B. (2016). Design principles for industrie 4.0 scenarios. *Proceedings of the Annual Hawaii International Conference on System Sciences*, *2016-March*, 3928–3937. doi:10.1109/HICSS.2016.488. arXiv: arXiv:1011.1669v3

Leitão, P. (2009). Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence*, *22*(7), 979–991. doi:10.1016/j.engappai.2008.09.005. arXiv: arXiv:1406.0223v1

Macal, C. M. & North, M. J. [Michael J.]. (2008). Agent-Based Modeling and Simulation: AMBS Examples. *2008 Winter Simulation Conference (WSC)*, 101–112. doi:10.1109/WSC.2008.4736060

Macal, C. M. & North, M. J. [Michael J.]. (2009a). Agent-based modeling and simulation. In *Winter simulation conference* (pp. 86–98). WSC '09. Austin, Texas: Winter Simulation Conference.

Macal, C. M. & North, M. J. [Michael J.]. (2009b). Agent-based modeling and simulation, 86–98.

Macal, C. M. & North, M. J. [Michael J.]. (2010). Tutorial on agent-based modelling and simulation, 151–162. Retrieved from https://link.springer.com/article/10.1057%2Fjos.2010.3

Macal, C. M. & North, M. J. [Michael J.]. (2013). Introductory tutorial: agent-based modeling and simulation, 362–376.

Mattern, F. & Floerkemeier, C. (2010). From the internet of computers to the internet of things. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6462 LNCS*, 242–259. doi:10.1007/978-3-642-17226-7_15. eprint: 9780201398298

Monostori, L., Kumara, S., & Váncza, J. (2006). Agent-Based Systems for Manufacturing. *CIRP Annals - Manufacturing Technology*, *55*(2), 697–720. doi:10.1016/j.cirp.2006.10.004

Negmeldin, M. A. & Eltawil, A. (2015). Agent Based Modeling in Factory Planning and Process Control. *Industrial Engineering and Engineering Management (IEEM), 2015 IEEE International Conference on*, 1810–1814.

Nikolai, C. & Madey, G. (2009). Tools of the trade: a survey of various agent based modeling platforms. *Journal of Artificial Societies and Social Simulation*, *12*(2), 2. Retrieved from http://jasss.soc.surrey.ac.uk/12/2/2.html

North, M. J. [Michael J]. (2014). A theoretical formalism for analyzing agent-based models. *Complex Adaptive Systems Modeling*, *2*(1), 3. Retrieved from http://casmodeling.springeropen.com/articles/10.1186/2194-3206-2-3

Plattform Industrie 4.0. (2018). Retrieved January 6, 2018, from http://www.plattform-i40.de/I40/Navigation/DE/Home/home.html

Repast Simphony 2.5 API. (2018). Retrieved May 15, 2018, from https://repast.github.io/docs/api/repast_simphony/index.html

Repast - The Repast Suite. (2018). Retrieved January 21, 2018, from https://repast.github.io/

Wikipedia- Comparison of agent-based modeling software. (2018). Retrieved January 21, 2018, from https://en.wikipedia.org/wiki/Comparison_of_agent-based_modeling_software

O'Sullivan, D., Millington, J., Perry, G., & Wainwright, J. (2012). Agent-based models – because they're worth it? In A. J. Heppenstall, A. T.

Crooks, L. M. See, & M. Batty (Eds.), *Agent-based models of geographical systems* (pp. 109–123). Dordrecht: Springer Netherlands. doi:10.1007/978-90-481-8927-4_6

PwC. (2016). Industry 4 . 0 : Building the digital enterprise, 1–36. doi:10.1080/01969722.2015.1007734

Roth, A. (2016). *Einführung und umsetzung von industrie 4.0*. Springer Gabler.

Scholz, J. & Schabus, S. (2014a). An Indoor Navigation Ontology for Production Assets. *Proceedings of the 8th International Conference, GIScience 2014, Vienna, Austria, September 24-26, 2014*. 204–220. doi:10.1007/978-3-319-11593-1_14

Scholz, J. & Schabus, S. (2014b). An Indoor Navigation Ontology for Production Assets. *Proceedings of the 8th International Conference, GIScience 2014, Vienna, Austria, September 24-26, 2014*. 204–220. doi:10.1007/978-3-319-11593-1_14

Scholz, J. & Schabus, S. (2015). Geographic Information Science and Technology as Key Approach to unveil the Potential of Industry 4.0. *in Proceedings of ICINCO 2015 Vol.2, 12th International Conference on Informatics in Control, Automation and Robotic*, (2008), 463–470.

Shrouf, F., Ordieres, J., & Miragliotta, G. (2014). Smart factories in Industry 4.0: A review of the concept and of energy management approached in production based on the Internet of Things paradigm. *IEEE International Conference on Industrial Engineering and Engineering Management*, *2015-January*, 697–701. doi:10.1109/IEEM.2014.7058728

Spath, D., Gerlach, S., Hämmerle, M., Krause, T., & Schlund, S. (2013). Industrie 4.0 – Produktionsarbeit der Zukunft *. *Werkstattstechnik*, *103*, 130–134. doi:10 . 1007 / 978 - 3 - 658 - 04682 - 8. eprint: 9809069v1 (arXiv:gr-qc)

Torrens, P. M. (2010). Agent-based Models and the Spatial Sciences. *Geography Compass*, *4*(5), 428–448. doi:10.1111/j.1749-8198.2009.00311.x

van Kranenburg, R. & Dodson, S. (2008). *The internet of things: a critique of ambient technology and the all-seeing network of rfid*. Network notebooks. Institute of Network Cultures. Retrieved from https://books.google.at/books?id=PilgkgEACAAJ

Wooldridge, M. & Jennings, N. R. (1995). Intelligent agents: theory and practice. *Knowledge Engineering Review*, *10*, 115–152.