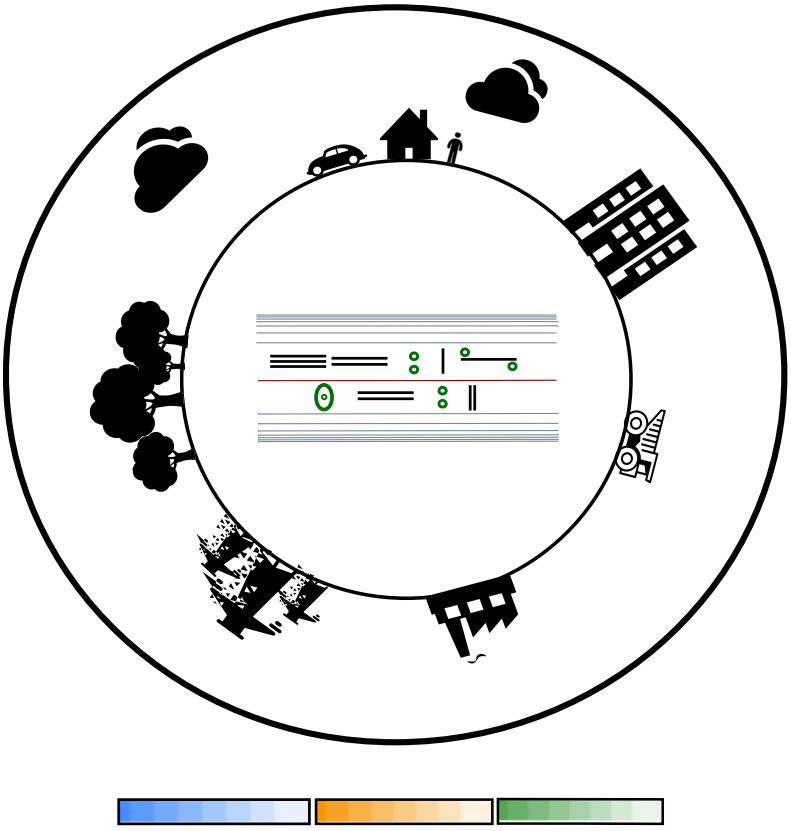
# DECODING SOCIO-TECHNICAL COMPLEXITY

Software development of an Agent-based Model-driven Integrated Environment



*Name: Sidney Niccolson*
*Topic: ICT, Agent-based Modeling and Simulation, Socio-Technical Systems*
*Date: February 6th, 2017*
*Field: Industrial Ecology*
*1st supervisor: Dr. Amineh Ghorbani (TU Delft)*
*2nd supervisor: Dr. Virginia Dignum (TU Delft)*

Universiteit Leiden

TU Delft

*Sidney Niccolson*
*Thesis Research Project*
*February 6th, 2017*
*s1650548 (Leiden) - 4452909 (Delft)*

# DECODING SOCIO-TECHNICAL COMPLEXITY

Software development of an Agent-based Model-driven Integrated Environment

Full Name : Sidney Iwan Niccolson
Mail : sidneyniccolson@gmail.com
University : Leiden University and Delft University Of Technology
Field : Industrial Ecology
Background : Bio-informatics
Thesis Project timespan : September 13th – February 6th
Student number : s1650548 (Leiden) - 4452909 (Delft)

Host institution : Delft University Of Technology
Name supervisors : Dr. Amineh Ghorbani and Dr. Virginia Dignum
Name of contact* : Dr. Amineh Ghorbani
Department : Engineering Systems and Services (Section: Energy and Industry)
Address : Faculty of Technology, Policy and Management, Building 31
Room : A3.200
City : Delft
Phone : +31 15 27 86703
Mail : A.Ghorbani@tudelft.nl

*Sidney Niccolson*
*Thesis Research Project*
*February 6th, 2017*
*s1650548 (Leiden) - 4452909 (Delft)*

# Abstract

The need for life cycle system oriented tools has been stressed within the field of Industrial Ecology and has become gradually more important due to the increasingly complex environmental challenges that our society faces (Davis, Nikolic, & Dijkema, 2010; Halog & Manik, 2011). The present-day advancements in technology, mainly Information Communication Technology, allows for many opportunities to improve our understanding of complex systems. Agent-based Modeling and Simulation provides techniques to create a computer representation of a system, thereby enabling researchers to simulate complex systems. Thus, driving our comprehension of the anthropogenic influence on planet earth. More specifically, socio-technical systems are inherently complex and the environmental impact on the world is in part due to the interaction between individuals and technology, which has not been fully addressed. While Agent-based Modeling and Simulation allows for the analyses of socio-technical systems, stakeholder involvement has proven to be a barrier for modelers and scientist alike. This barrier comes forth out of the lack of programming experience needed to utilize computational tools. Especially in the field of Agent-based Modeling and Simulation the level of programming skills required to implement socio-technical models is relatively high. This study aims to develop a decision support tool that lowers the barrier for the construction of socio-technical agent-based models. The integrated environment provides means for model development, translation to simulations and intercommunication through sharable models. The cross-platform decision support tool that utilizes the MAIA (Modeling Agent systems based on Institutional Analysis) framework addresses the stakeholder involvement challenge inherent in ABMS. The computational methodology presented here uses a Model-driven Software Development approach to bring about the platform development.

*Sidney Niccolson*
*Thesis Research Project*
*February 6th, 2017*
*s1650548 (Leiden) - 4452909 (Delft)*

# Acknowledgements

First and foremost I would like to thank my first supervisor, Amineh Ghorbani for the valuable structured guidance and advise. Her work on the MAIA theoretical framework is very inspiring, MAIA presents an approach on how we can use computational tools to improve our understanding of complex systems. The personal meetings with her were motivating and essential for the development of the software. I learned a considerable deal with respect to Model Driven Software Development and automatic code generation that may prove valuable in my future career. Furthermore, I want to thank Virginia Dignum as a second supervisor for her feedback and the inspirational work on the OperA framework. I also would like to take this opportunity to thank Thorben Jensen for helping me at the start of this thesis study. His recommendations encouraged me, and pointed me in the right directions. Tai Sassen Liang was the only software developer that I had contact with from time to time and I want to give him an honorable mention. For understanding my long evenings from time to time at the computer, I'd like to thank my girlfriend Vhernadette. Moreover, special thanks to Michael for his support and the informal study group sessions we organized together. Finally an honorable mention goes to my fellow IE students and friends, for their warmth and openness which contributed to a strong cooperative study environment that I have never experienced before.

*Sidney Niccolson*
*Thesis Research Project*
*February 6th, 2017*
*s1650548 (Leiden) - 4452909 (Delft)*

# Table of Contents

# Abbreviations

ABMS: Agent-based Modeling and Simulation
MDSD: Model-driven Software Development
MDA: Model Driven Architecture
IE: Industrial Ecology
STS: Socio-technical Systems
MAIA: Modeling Agent systems based on Institutional Analysis
JET: Java Emitter Templates
EMF: Eclipse Modeling Framework
RCP: Eclipse Rich Client Platform
CAS: Complex Adaptive Systems
OOP: Object oriented programming
API: Application Interface
GUI: Graphical User Interface
OS: Operating System
JDT: Java Development Tools
XML: Extensible Markup Language
UML: Unified Modeling Language
ICT: Information, Communication Technology

*Sidney Niccolson*
*Thesis Research Project*
*February 6th, 2017*
*s1650548 (Leiden) - 4452909 (Delft)*

# 1 Introduction

Industrial Ecology (IE) emerged from the need to address the growing anthropogenic impact on the environment. It aims to reduce environmental burdens related to energy and material flows (Allenby & Graedel, 1993). Thereby IE takes a systems approach of assessing these flows. For instance assessing the total material life cycle of products, from virgin materials like ores, to finished materials, to discarded product, and to disposal (Heiskanen, 2002). One of the concepts of IE is that a product is part of an industrial system, in turn the industrial system itself is never isolated and is part of its surrounding systems and/or influenced by them (Chertow, 2000). Hence, the social, environmental, economical and technological aspects are of relevance for industrial systems. The interrelatedness of these aspects are represented by the notion of socio-technical systems (STS).

From an IE perspective socio-technical systems are the physical embodiment of interrelations between man and technology, where analysis of such a system takes into account the contribution of different social and technical systems (Borrás & Edler, 2014). These social and technical systems are interdependent in the sense that together they may serve a common function (e.g. STS around transportation), yet actors within the system have different goals (e.g. manufactures, decision makers, construction workers). Technical artefacts (e.g. cars) are dependent on the local infrastructure and arise from complex structures of supply chains that produce only parts of the product (e.g. car tires or electronics) (Guide & Wassenhove, 2003). Qualitative and quantitative research assists in gaining understanding of socio-technical systems, however the complex nature of socio-technical systems creates uncertainties on many levels. For example to reach sustainability on a systems level in transportation, collaborative efforts from various stakeholders with different perspectives is needed. These actors need to be steered towards a common goal which should include resource conservation, carbon emissions reduction, improvements in production efficiency and economic viability (Bichraoui, Guillaume, & Halog, 2013). It is evident that it is uncertain who should play what roles, what technologies are most suitable, what infrastructure is needed, what social behaviour needs to be changed and what incentives are needed. According to Boons & Baas (1997), "Industrial Ecology demands the coordination of activities of economic actors as well as governmental agencies". To elaborate further on the complexity of STS several theoretical models have been established, such as the macro-micro-macro model (Amineh Ghorbani, 2013). This model explains that the system in which an individual is embedded influences the individual's behaviour and actions. In turn this results in emergent patterns of interaction and outcomes for the system as a whole. In the STS around the transportation case, we have many individuals that make decisions on whether to buy or lease a car, or to take public transportation. From a broad perspective one may argue that each decision affects the use and flow of energy and materials in society (Axtell, Andrews, & Small, 2001). The change in energy and material flows, its cause and effects due to technology and interaction of actors in a socio-technical system cannot be easily predicted. In the field of IE solutions are not clearly visible, for example what policies are needed to reach sustainable levels of transportation when there are rebound effects and changes in technology, such as self-driving cars? As mentioned by Herring & Roy increased energy efficiency leads to lower costs of energy services and in turn effects consumer behaviour (Herring & Roy, 2007). Thus rebound effects are typically evident in socio-technical systems (Arvesen, Bright, & Hertwich, 2011). For

more information on IE and its relations to sustainability concepts and STS, please refer to the glossary 'An introduction to IE'.
In order to understand complex phenomena occurring within socio-technical systems, an abstract representation (e.g. a conceptual model) is useful to arrive at better insights of the system. A conceptual model is a high-level description, meaning it is human-understandable, yet not interpretable by a computer. Moreover, a conceptual model can be translated to a computational model which can be simulated with different sets of inputs, in which emergent patterns of interactions can be assessed under changing conditions. Agent-based Modeling and Simulation (ABMS) provides ways to model and simulate complex socio-technical systems (Amineh Ghorbani, 2013), wherein individuals with their personal values and norms are key components that shape such a model. ABMS can be used to achieve a better understanding of socio-technical systems through the development of building blocks that together describe the socio-technical system.
It should be mentioned that ABMS has limitations, for instance the computer science expertise that is needed to develop an agent-based model and simulate it properly is not always apparent (Amineh Ghorbani, 2013). This makes it difficult to translate social-technical system models (conceptual) into a final simulation model for analyses. A lack of familiarity with computational tools creates a barrier to involve relevant actors early in model development. The need for system-oriented tools to support policy and regulation has been stressed by several scientist active in the field of IE, such as Chris Davis et al. on the potential of Information, Communication Technology (ICT) (Davis et al., 2010) and Anthony Halog et al. research in the direction of integrated Life Cycle Assessment (LCA) (Halog & Manik, 2011). ABMS has been applied in various studies in the field of IE (Kraines & Wallace, 2006) (Axtell et al., 2001), however the limitations of ABMS has not been readily addressed. There are guidelines for actively involving stakeholders (participatory ABMS) in the simulation process, but defining methods for stakeholder involvements at all stages with current ABMS tools is rather difficult. Hence the key question remains on how to bring ABMS within reach of stakeholders with limited programming experience. In turn what software packages would need to be developed that lower the barrier of modeling and simulation? These scientific challenges will be explored within this research study, more specifically regarding the application of software packages.

## 1.1 Thesis goal and scope

This thesis project is aimed to develop a decision support tool (platform) based on ABMS, thereby providing incentives for stakeholder to participate more actively in the process of modeling and simulation of STS. Two development features for such an platform are relevant to address: (1) the usefulness and (2) usability respectively (Amineh Ghorbani, 2013). (1) Usefulness refers to the content of the tool, whether the tool serves its aim and is as comprehensive as possible. Thus taken into account all complex dimensions of socio-technical systems in a realistic manner, such as social, institutional and physical aspects.
Different applications were developed that addresses the usefulness by putting relevant STS concepts into a model (conceptual modeling) (Le Page, Becu, Bommel, & Bousquet, 2012). For instance INGENIAS is a software platform that includes social concepts for conceptual model development (García-Magariño, Gómez-Sanz, & Fuentes-Fernández, 2009). This software tool is not aimed specific for simulations, but can be used as a stepping stone to develop simulations through a model-driven approach. easyABMS was developed to focus more on the methodological aspects of simulation development (Garro & Russo, 2010). easyABMS

uses for actual simulations the REPAST platform. CORMAS is a generic agent-based simulation platform that was developed to address specifically natural resource issues (Le Page et al., 2012). Moreover, The GAMA platform is a tool dedicated to complex environmental models, while integrating Geographical Information System (GIS) data (Drogoul et al., 2013). It uses a high-level modeling language, which is embedded in a generic tool. Other tools address the whole cycle of ABMS in an abstract manner such that various models and simulations can be created such as NETLOGO, however a good level of programming experience is typically needed. OperettA is a prototype tool for the design, analysis and development of multi-agent organizations developed by Dr Virginia Dignum and colleagues at the TU Delft (Aldewereld et al., 2016). The OperA framework on which OperettA is based makes a distinction between individuals & organizations, autonomy and collective behavior, social and selfish behavior. In which organizations refer to the notion that they exists to fulfill a common goal. OperA is a framework for the specification of organizational structures, while allowing inclusion of actors that act according to their own demands and capabilities in the modeling process. Lastly Dr. Amineh Ghorbani together with Virginia Dignum and colleagues developed a conceptual framework called MAIA (Modeling Agent systems based on Institutional Analysis) that is more aimed at active stakeholder involvement with regards to modeling and simulations of various socio-technical systems. The MAIA framework includes a meta-model that describes dimensions of an STS from a high level, thus addressing the usefulness by creating a comprehensive abstract representation of such a system. It covers concepts such as sociability, roles, organizations (A. Ghorbani, Dijkema, Bots, Alderwereld, & Dignum, 2014). MAIA can be used for policy-social oriented research and focuses on large-scale projects in which various stakeholders are involved. An important feature that separates MAIA from the other tools described is that MAIA is aimed to be understandable for stakeholders working in a multidisciplinary setting and tries to specify concrete steps that are needed to go from model towards simulation. OperettA is quite similar to MAIA in respect to the fact that both model aspects of STS making an explicit distinction between organizations and individuals. The MAIA framework is used as a basis for the development of the decision support tool in this thesis project.
Next to the usefulness, (2) the usability is an important concept in this thesis. The usability is reflected in the so called practicality/ease-of-use of the tool, considering all the steps needed from model to simulation. More specifically, providing a tool for the translation of a high-level conceptual language to a low-level computational language. This project mostly addresses the usability providing the basis for rapid development of STS-oriented ABMS, while implementing the MAIA meta-model. Additionally a Model Driven Software Development (MDSD) approach is taken that utilizes meta-models for software development.

This thesis study will therefore address the following research question:

- How can a platform for rapid development of Agent-based Models and Simulations that utilizes Model Driven Software Development be realized, using a high-level language?

In order to assess how such a platform can be achieved, the viability needs to be examined. The viability is in this respect determined by the technical feasibility, which will be addressed through these subquestions:

- What software modules are needed to develop the platform?
- How can these modules be interlinked, to bring about a workflow-based application?
- What steps are needed to translate socio-technical conceptual models into simulations?

## 1.2 Thesis outline

The Materials & Methods sections (chapter 2) addresses what components are used to develop the decision support tool, thereby explaining the concepts of ABMS, MAIA and MDSD, as well as several software modules. Chapter 3 (Model-driven software development for ABM) goes into detail on how these components interact and what methodology has been developed for the realization of the platform. Chapter 4 (AMIE - Agent-based Model-driven Integrated Environment) addresses the developed final product, that is the software architecture and functionality. This is proceeded by a case study (Chapter 5) that illustrates a set of scenarios to demonstrate conceptual model development, and steps required to construct Agent-based simulations. Furthermore, Chapter 6 (Tutorial for AMIE test-users) offers a step-by-step guide on how to use the software. Lastly, the Conclusion and Discussion (chapter 7) will outline the research questions, recommendations for future work and reflection.

# 2 Materials and Methods

The Materials and Methods chapter is dissected in modular sections in order to explain and clarify terminologies, concepts, modules that are used throughout the thesis. First of all a more detailed explanation is given on the MAIA framework with respect to ABMS and what means are needed to integrate it into a software package. The second part of the Materials and Methods chapter explains the specific ICT technologies used.

## 2.1 MAIA and ABMS

### 2.1.1 Modeling and simulating socio-technical systems

A simulation can be viewed as a method to analyze complex systems over changes in time and conditions. Socio-technical systems can be seen as Complex Adaptive Systems (CAS) (van Dam, Nikolic, & Lukszo, 2012). Adaptive in the sense there is a network of interaction between physical and social components that responds to selection pressure. Complex in the sense that adaptive cycles takes place where every change in a socio-technical system rewrites the rules of interaction between components, whereas technology, social and economic systems co-evolve.
Nowadays the technological developments in ICT creates opportunities for simulations of such systems that where previously deemed impossible. There have been developments to improve the ABMS process through user interfaces, simulation languages and domain specific simulators. With Internet Of Things (IoT) devices and increased sensor technology, modelers can use data available of the system under study to construct models. Not only physical data is important, but also documents about the system can be used for input data, to determine constraints of the system, or to validate output data (Huang, 2013).
Individuals with their personal values and norms are key components that shape an socio-technical oriented Agent-based Model. Individuals are expressed as 'agents' and reside in an 'environment'. However an agent can also be a technical artifact or a collection of individuals (e.g. a family, a company or a government), referred to as composite agents. An individual or collection of individuals behavior is depended on perspectives, preferences, personal values, resources, capabilities. The environment can be modeled as the institutional setting, which determines the set of rules that coordinates certain activities for sets of agents (Amineh Ghorbani, Bots, Dignum, & Dijkema, 2013). The institutional setting affects not only the agents themselves, but potentially also others that do not belong to the same set of agents. Agent activities are based on a goal (objective) an agent wants to reach, coordinated by rules. In order for a model to come close to a real-life socio-technical system, it is also evident that individuals do not necessarily behave rational at all times. They might have incomplete information that influences their behavior. In terms of ABMS one may argue that the criteria on which agents act can change depending on the available information at a given time. Random non-deterministic choices are sometimes made by agents.
ABMS will always be some steps away from reality, because the modeler decides how the model is built from his or her perspective (e.g. the modeler decides how agents behave under a set of conditions). A modeler would need to understand the system as a whole (what is it's function? What is the interrelation between components?), but at the same time needs to have eye for detail to explain the components in the system.

Next to that, when creating a model it is important to realize the modeler's position. A modeler is a person with lived experience. One may argue that a system's model is described from the modeler's experience. Thus another researcher would explain the system differently. Scientists that analyze real world situations should recognize this inherent subjectivity (Huang, 2013).
The notion of emergence is another important factor for ABMS. Local individual interaction of agents in a given environment leads to global changes in the socio-technical system. This so called emergence cannot be predicted beforehand based on statistics or basic analytics. One may argue that a strict reductionist explanation of STS cannot predict how it would behave under a set of conditions (e.g. changes in policies y with respect to technology x), because the interaction of heterogeneous agents with respect to technology is largely ignored. Neither is a strict holistic view applicable to modeling, as it denies the various individual behavior of agents. ABMS tries to achieve a better understanding of socio-technical systems through the development of building blocks and heterogeneous agents that together describe the socio-technical system. Thus this perspective on modeling and understanding complexities is not strictly reductionist nor holistic.
ABMS provides various insights that can help in decision making, although ABMS has limitations as described shortly in the introduction. These limitations are in part due to the way ABMS are mostly built (Amineh Ghorbani, 2013). First of all social structures are not readily integrated in ABMS and the development of MAIA has been one of the first efforts in integrating these structures. Secondly modeling STS is something different than feeding such a model in an ABMS software application that actually runs the model. Hence social scientist role might be to construct social models of behavior on various levels and engineers supply information on technical artifacts, yet the ICT expertise that is needed to integrate and simulate these distinct concepts properly is not always apparent. This makes it difficult to translate social-technical models into an running ABMS application for analyses. Next to lack of familiarity with computational tools, the way relevant actors are involved as discussed in the introduction is a limitation of the current ABMS practices.

### 2.1.2 Conceptualization – how to go from theory to a description of a system's behavior?

Various frameworks and theories are needed to describe socio-technical systems. Ideally the process of integrating these frameworks and theories lead to a conceptualization of a socio-technical system represented as an Agent-based model. In ICT the notion of conceptualization is widely applied in terms of meta-models. Meta-model is a set of concepts and relations highlighting common properties of models. For example the Extensible Markup Language (XML) is a meta-model format used in ICT. XML based code is for exchange and storage of data in both a human and computer understandable way (w3schools, 2016). Essentially meta-models have a structured description of a model, while at the same time providing a medium for translation from high level languages to low level languages. High level languages are relatively close to human language and is therefore human understandable. Low level languages are commonly used for executable software. Meta-models provide a standardized way of describing models, hence this gives incentives for participatory modeling and simulation. Additionally, meta-models are an integral part of Model Driven Software Development. See section 2.2 for further elaboration on the relation between meta-models and MDSD.

### 2.1.3 The MAIA meta-model

A socio-technical system is shaped by social structures, and has resources and institutions. Institutions influence agent actions through rules, but agents also perform tactical behavior violating norms and rules. However it can be argued that institutions do coordinate agent behaviors to a large extent. The Institutional Analysis and Development framework (IAD) was proposed in 1994 to gain an understanding of the underlying social structures that shape social systems, thereby observing patterns of behaviors and interaction in an given environment. IAD has been developed for over 30 years and used in multiple case studies (Amineh Ghorbani, 2013). The conceptual framework MAIA is built upon IAD and was developed to help modelers conceptualize and implement agent-based models for socio-technical systems (Amineh Ghorbani et al., 2013). The MAIA meta-model is viewed as a description of a STS shaped by social structures in time and space (A Ghorbani, Dijkema, Bots, Alderwereld, & Dignum, 2014). MAIA addresses different components of a socio-technical system. Hence the meta-model is organized according to six structures:

- COLLECTIVE STRUCTURE: These are agents and their attributes. For example attributes can encompass an agent's perspective, preferences, personal values, resources, capabilities.
- CONSTITUTIONAL STRUCTURE: The social context. This is based on rules and conventions alternatively said institutional statements. Institutional statements govern agent behavior/actions to a large extent. The constitutional structure is based on ADICO. ADICO refers to five elements that an institutional statement has: Attributes (the designed role of the statements), Deontic (the set of rules/laws that determine whether it is an obligation, permission, prohibition), aIM (action that has to be taken by an agent), Condition (in which case), Or else (defines what happens if an agent does not comply). Institutional statements can be categorized into three types: rules, norms and shared strategies. If all five elements of ADICO are defined we can speak of a rule. If there is no “Or else” than it is a norm, in this case the consequences of non compliance is not clear. If there is no obligation or sanction than it is a shared strategy that agents may perform.
- PHYSICAL STRUCTURE: The physical aspects of the STS. Such as technical artifacts with their properties (e.g. weight, price, efficiency). This structure also embodies the things that can be done with them (e.g. selling, renting, buying), their behaviors (e.g. aging) and whether the product is public (accessible to all individuals) or private.
- OPERATIONAL STRUCTURE: The dynamics of the system. An agent influences the respective system's state in time and space. The operational structure defines the actions order as each agent has a time frame to perform actions (entity actions) he may be able to execute. What he executes is dependent on 'preconditions' (feasibility e.g. having sufficient money). The action of an agent can in addition be dependent on institutional statements or an agent may go through a decision making process. After execution of an agent's task there is a 'postcondition', which is an update of the system's state (e.g. the loss of money due to expenditures). If an agent cannot perform the actions, because the precondition is not met, there might be consequences for the respective agent or the system.
- EVALUATIVE STRUCTURE: The concepts used to validate and measure the outcome of the system. A model is steps away from reality as mentioned earlier, although it may serve as a valuable measure to assess complex socio-technical systems. From an evaluative perspective a model is not perfect as it may contain bugs in code or conceptualization errors. Therefore the evaluative structure forces a

modeler to be explicit about what patterns of interaction and outcomes of the system is expected. This way the modeler should ask how realistic the model is and whether the model helps answer the research question. A modeler may define certain constraints of the system (e.g. an agent may not have a negative cash balance). In the evaluative structure a modeler specifies the variables that may serve as useful indicators to answer their respective research questions (e.g. the amount of investments by agents).

- ONTOLOGICAL STRUCTURE: Is a container to built an ontology for specific case studies. Meaning that the concepts of a certain domain can be stored and reused.

The MAIA meta-model helps in breaking down and conceptualizing a socio-technical system by structuring complexities, by explicitly defining key characteristics of the system. These six structures outlined above are ought to be filled in by the modeler and this formal structuring makes it a step closer to simulation. With system conceptualization the modeler is forced to think about what components are relevant to include, thereby forcing to address the system boundaries. MAIA provides a medium and source of documentation that would increase legitimacy due to the standardized, structured nature of the meta-model, providing a common language between modelers which serves as a basis for re-use of components (building blocks) in models. A modeler can give interviews and prototype simulations to construct and validate the model with the involvement of stakeholders. In addition MAIA is based on IAD which defines institutions as the building blocks of social structures, thus integrating the notion of social structures.

## 2.2 Model Driven Software Development

MDSD is one of the key concepts used to bring about the development of the MAIA-based platform. MDSD is an approach that allows for rapid software development through the use of application models, alternatively said meta-models (Generative Software Engineering, 2016). These meta-models are used during development, providing means for conceptual software design, the reuse of software components and effective software implementation. Hence MDSD requires meta-models that describe what to model (abstract meta-model), how to make transformation from high level to low level language (set of protocols), what transformation platforms to use (e.g. actual code that transforms model to simulation-ready code) (Amineh Ghorbani, 2013).

With a MDSD methodology, parts of the software can be automatically generated and this fits within the realm of automatic code generation. The benefit of MDSD lays in the fact that repetitive aspects of software development are automated and a more standardized software architecture can be achieved. In the field of ABMS, MDSD is still in its infancy, however concepts of MDSD might prove useful for effective ABMS development.

Model Driven Architecture (MDA) is an industry architecture that utilizes standardized formats (collections of specifications) such as XML and was proposed by the OMG consortium (Object Management Group) in 1999 (Frank Budinsky, David Steinberg, Ed Merks, Raymond Ellersick, 2003). MDA is in a way the standardization of MDSD concepts. MDSD itself is not bound to standards, meaning that with a MDSD approach, any meta-model can be developed, while with MDA meta-models are realized according to specific formats. This distinction is important as the MAIA meta-model is not necessarily bound to formats, thus it suits the notion of MDSD instead of MDA. However several software components that are described in the

following sections are a part of the MDA. Additionally, the MAIA meta-model is translated into a MDA format in order to realize the implementation efforts needed to develop the platform.

## 2.3 Java and object oriented programming

The MAIA meta-model is conceptually rich, however integrating those concepts into a platform is not straightforward and many different options exists to implement the MAIA meta-model. In this thesis Java has been the backbone for the platform development and much of the Java terminology will be used in later sections. In this sub-chapter relevant terminologies are shortly described.
In general Java is a popular programming language for developing software applications (Liang, 2009). The benefit of Java is mostly based on Operating System/platform interoperability. Java can be dissected in three modules: JRE (Java Runtime Environment), JVM (Java Virtual Machine) and JDK (JavaDevelopers Kit). The JVM provides a platform independent way of executing code. The JRE contains libraries that provide functions (e.g. math functions) and also includes the JVM. Most computers have a JRE installed and can therefore run Java-based applications. JDK is used for developing Java applications, that is writing Java source code and compiling them (converting the code to computer readable language). When JDK is installed it is supplied with a JRE to be able to run the developed Java applications. See figure 2.1 for a typical Java application process. The terminology for Java libraries that provide functions is defined as Java Application Interfaces or alternatively said Java frameworks, Java API's. Java API's were used for developing the software application.

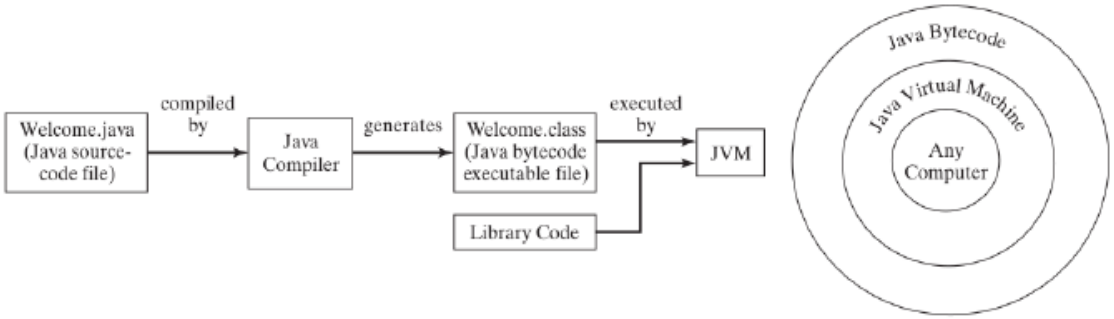



Figure 2.1: A typical Java applications process. The Welcome.Java file contains Java source code (human readable language) of the Java application. A compiler(part of JDK) converts the Java source code to computer readable code(byte code>Welcome.class). Along with library code(part of JRE) the application can be executed by the JVM (part of JRE).

Object oriented programming (OOP) is one of the key concepts used for large application development. The notion on OOP does not only relate to the Java language, but many other programming languages. In general programmers define functions (tasks, procedures, methods) and variables. Variables are symbolic labels associated with a value and whose associated value can be changed. A function is an encapsulation of a specific “thing” a program should do (e.g. a math calculation on two variables). In OOP this encapsulation is extended to code residing in different locations and these code 'snippets' represent classes (Java jargon). Functions can reside in these classes and both classes and functions represent a certain degree of modularity. This explicit breakdown makes programming more transparent and often quicker as no reinvention of the

wheel is needed if functions are already created by other programmers (Liang, 2009). In this case these classes containing functions are referred to as libraries. Additionally classes are commonly used to protect data. For instance, Java developers might define variables inside classes that should not be accessible by others due to security policies. The use of functions from other classes is called inheritance and is used extensively in this thesis. An object is another key term used in programming languages. An object is an abstract representation of a state and behavior of 'something' (e.g. a car object has a state color 'brown' and it's behavior is 'moving'). An object is a member of a class (e.g. class car), however these terms are commonly used interchangeably. In this thesis project the MAIA meta-model is translated to a set of Java classes representing the six MAIA structures at its core following MDA. An instance is an object created from a class and the notion on instances allows users too create their own model based on MAIA. Practically speaking users develop instances of the MAIA meta-model, by using inheritance through developed Java-based Graphical User Interfaces (GUI's). Lastly, the concept of abstract classes and static classes should be shortly addressed. Abstract classes are 'blueprints' that represent the architecture (e.g. general functions) an instance class should have, these abstract classes are MAIA meta-model classes. Instance classes represents specific case studies information. Static classes are also used to define variables that should be shared among agents. A static class is actually an instance by itself, however no instances can be created from it. If variables are defined in a static class, these variables can be used by instances of other abstract classes, which is of importance in ABMS in terms of the sharing of properties.

## 2.4 Eclipse

Eclipse is an open source based software platform developed for tool development and integration. The platform developed is an Eclipse version with custom features that satisfy the needs of ABMS and MAIA. Eclipse is a generic framework that allows for a wide range of extensions on different domains (Frank Budinsky, David Steinberg, Ed Merks, Raymond Ellersick, 2003). It is made out of (1) a platform which defines a framework for extending and building Integrated Development Environment's (IDE's). (2) Java Development Tools (JDT) which signifies ways to develop programs for Eclipse (Java based). (3) The plugin development environment which provides views and editors for the creation of plugins for Eclipse. Eclipse simply works with files and folders and has an API that deals with the creation of projects. In the developed decision support tool, the projects reflect actual Agent-based model creation projects and ABMS experiment-related projects. On the next page a short description of Eclipse terminology.

- PLUGINS: Plugins have all the components needed for creating tools such as code, images, text and a MANIFEST file (called a plugin.xml) that declares interconnections to other plugins.
- RESOURCES: A resource is an eclipse representation of a file or folder that provides capabilities such as change listeners (resource change notifications), markers (error messages) and previous content tracking.
- PROJECTS: A project is a special resource (top-level folder). If it is of a Java "nature", then it contains Java source code.
- SWT: Operating System (OS) independent Java graphics library typically used in Eclipse.
- JFACE: uses SWT, but is a higher level implementation. Provides classes for managing monitors, images. Has an action-framework which can be used to add commands for toolbars, menubars creations.
- WORKBENCH: is an arrangement of views and editors and is implemented with the use of SWT and JFace. Extending Eclipse can be done with the use of "extension points" which allow for new editors, views and perspectives or customization of existing ones.
- PROJECT EXPLORER: part of the workbench and is used to manage resources and projects.
- PACKAGES: is a folder containing Java classes
- WIZARDS: is a GUI interface that is commonly used for the creation of new elements such as resources or projects.

### 2.4.1 Eclipse Modeling Framework (EMF)

Eclipse EMF is aimed at the unification of Unified Modeling Language(UML), Java and XML in order to built integrated software tools (Frank Budinsky, David Steinberg, Ed Merks, Raymond Ellersick, 2003). UML has been used extensively in the field of ICT and is an approach to document, share and communicate model designs across stakeholders and organizations. EMF is part of the MDA and uses model-to-code transformations using design, patterns and templates. EMF is used as a way to develop MAIA-based abstract classes from the MAIA meta-model. A meta-model in EMF referred to as an Ecore model is not necessarily the high-level description usually used with UML, because modeling concepts in EMF are directly related to implementations. EMF is integrated with the JDT of Eclipse which is needed to install and use EMF plugins.
A model in EMF can be UML, XML, EMF-based model or Java code. EMF model glues everything together, because a model in XML can represent the same as a model in Java code. People familiar with programming view models occasionally as unnecessary, because programmers can write code that is a model and implementation at the same time. Or alternatively, in large complicated applications modeling is taken as a requirement, but usually there is a gap between developing the high-level model and the implementation (low-level) of that model. In EMF there is a seamlessly integrated notion of model and implementation and developers can choose to what extent this integration works. EMF models describes what your applications is supposed to do (design). This gap between model and implementation is done through mapping. For more information on EMF please refer to appendix 1 'An Overview of EMF'.

### 2.4.2 Java Emitter Templates (JET)

The EMF framework provides another component called: Java Emitter Templates (JET). JET is an engine that can convert models to SQL, JAVA, XML, Text, HTML formats. JET Templates are used to create implementation classes, meaning that JET provides the translation platform for STS case studies based on the MAIA meta-model (EMF abstract classes). How EMF exactly interrelates with JET will be discussed in detail in the chapter 3 and 4. The JET language is made up out 'scriplets', 'Expressions', 'Directives'. Scriplets is Java code residing in between, '<%' symbols. Expressions are used to insert string values denoted with '{{}}' symbols. Directives are used for specifying output files configurations. Refer to appendix 2 'JET Syntax' for details.

### 2.4.3 Eclipse Rich Client Platform (RCP)

The Eclipse Rich Client Platform provides means for developers to create stand-alone software applications. It is based on the Eclipse and EMF infrastructure. RCP takes on a modular approach and uses the concept of MDSD extensively. RCP version 4 includes simple editors and Cascading Style Sheets (CSS) styling for stand-alone software development. As mentioned the platform developed is an Eclipse version, in which the RCP is utilized to lay out the basic infrastructure on which MAIA-EMF and JET functionalities are added. Refer to appendix 3 for information on the Eclipse Rich Client Platform.

# 3 Model-driven software development for ABMS

The Materials and Methods chapter addressed all the modules required in order to develop the decision support tool. The terminologies that are elaborated in that section will be used for the rest of the thesis report. This chapter addresses how these modules interface which each other and how the MAIA framework is embedded. The MAIA framework is extensive, so far the six main structures are elaborated. For more in-depth details of MAIA concepts and how they relate to ABMS refer to appendix 4 'MAIA Concepts & ABMS'. For more information on what MAIA concepts are integrated in the decision support tool refer to appendix 5 'Platform Implementations of MAIA Concepts'.

## 3.1 The application development approach

The thesis project focuses on the development of an application platform that is able to translate input data from users to computer understandable code. Therefore providing an interface for clients with limited experience in programming to construct an ABMS for STS based on the MAIA framework. As mentioned in the introduction, chapter 2.1.2 and 2.2 there is a need for the transition from high level languages (meta-models) to low level languages (computer understandable code). The MAIA meta-model can be referred to as a Computational Independent Model (CIM), which is only a description of a model at a conceptual level (e.g. what to put in a model). It does not provide details in a low level, computer understandable language. Hence a Platform Specific Model (PSM/domain specific model) is needed which is the realization of a model for a specific platform (e.g. a PSM in the programming language Java). Figure 3.1 shows an overview of the scope of the project. In the following sections a more detailed methodological explanation will be given on the Transformation Platform and how instance model's are constructed.

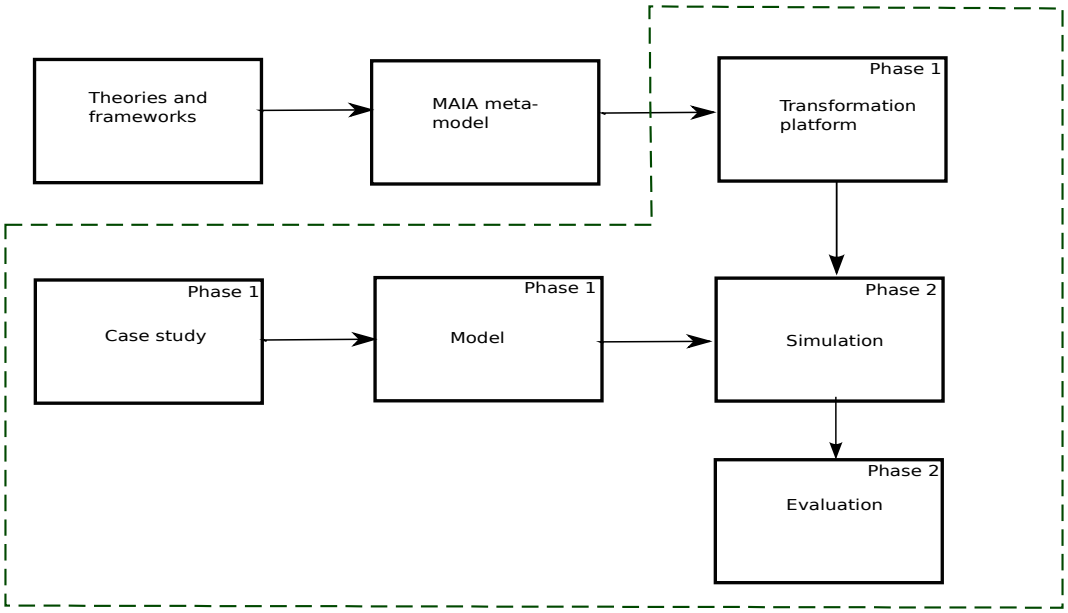



Figure 3.1: Phase 1 embodies the development of the platform (interface): the creation of the MAIA meta-model interface and the development of a transformation platform to executable code. In addition the case study would be developed, hereby a MAIA model is created (based on the meta-model) and this model will be translated to executable code. Phase 2 signifies the evaluation of the model simulation and developed software.

### 3.2 Model development infrastructure – EMF processes

This section focuses on how the MAIA meta-model and MAIA instance models are integrated in the decision support tool using the MDA of EMF and JET. Figure 3.2 shows an overview of the conceptual flow from meta-model integration (black box) to model instance development (black box) to generated code for simulation (grey box and grey circle).

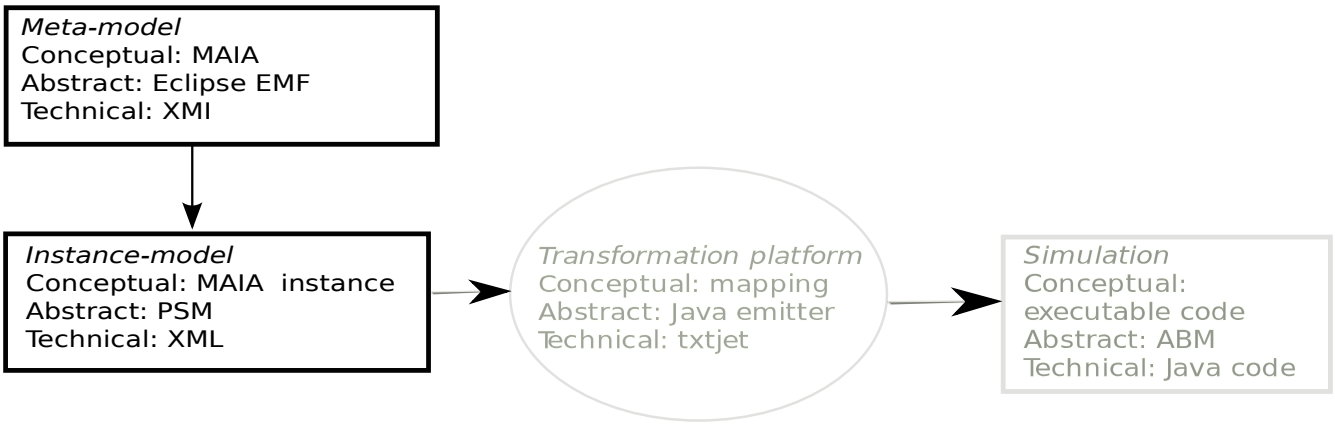



Figure 3.2: The MAIA meta-model functions as a virtual skeleton for case studies based on the MAIA theoretical framework. This meta-model is integrated in Eclipse EMF, resulting in a XMI file that contains all respective information on the theoretical framework. The instance-model depicted below the meta-model box uses this information to create instances of that model. Meaning that it provides methods to fill in the skeleton with information regarding a specific case study. The grayed out boxes will be explained in the superseding section in order to breakdown the processes.

Figure 3.3 on the next page represents a zoomed in view of the Meta-model and instance model integration.

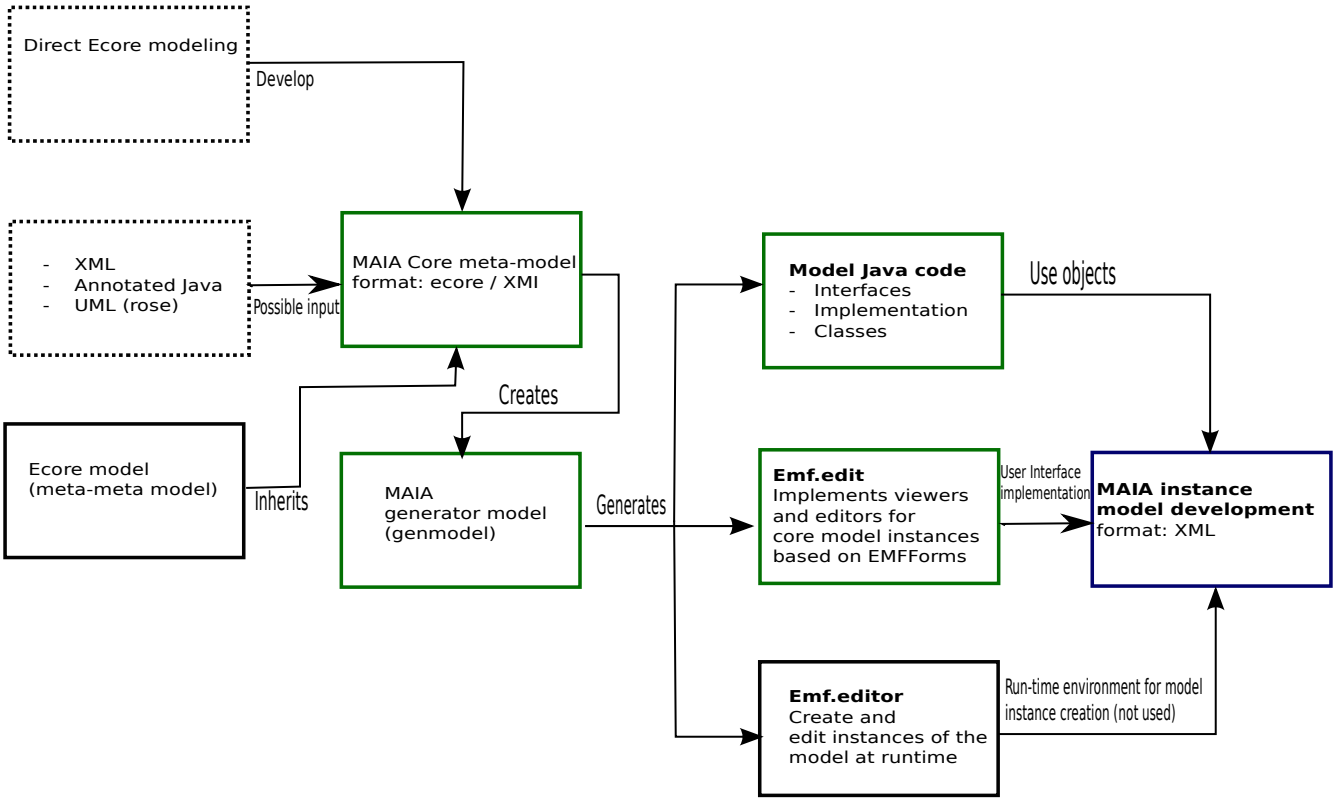



Figure 3.3: This figure shows an overview of developing the MAIA meta-model and exposing it for use. Dotted boxes show potential inputs, while green boxes show embedded processes in the developed application. The blue box is the resulting output of the whole process, which is an actual MAIA instance. Black boxes are not used, but nevertheless are relevant. With EMF, developers can create a so called Ecore/core model. This can be generated based on inputs such as annotated Java, UML or XML files. In addition a core model can be directly modeled with EMF's default visual editor. Once a core model is created, Java code can be generated that is (1) the model code, (2) emf.edit code, (3) emf.editor code. This code is used in order for users to create MAIA instances. Normally emf.editor can be used to start a GUI, however this is fairly low-level and requires object oriented programming skills. Hence through the use of EMFForms (Extension of EMF) instances can be created in a more user-friendly manner with a more straight-forward GUI.

### 3.3 Transformation platform infrastructure – JET processes

This section describes what technical steps are relevant to go from an instance model towards Java-based ABM simulation. Figure 3.4 shows the conceptual flow process again, but the focus is now on the last couple of processes.

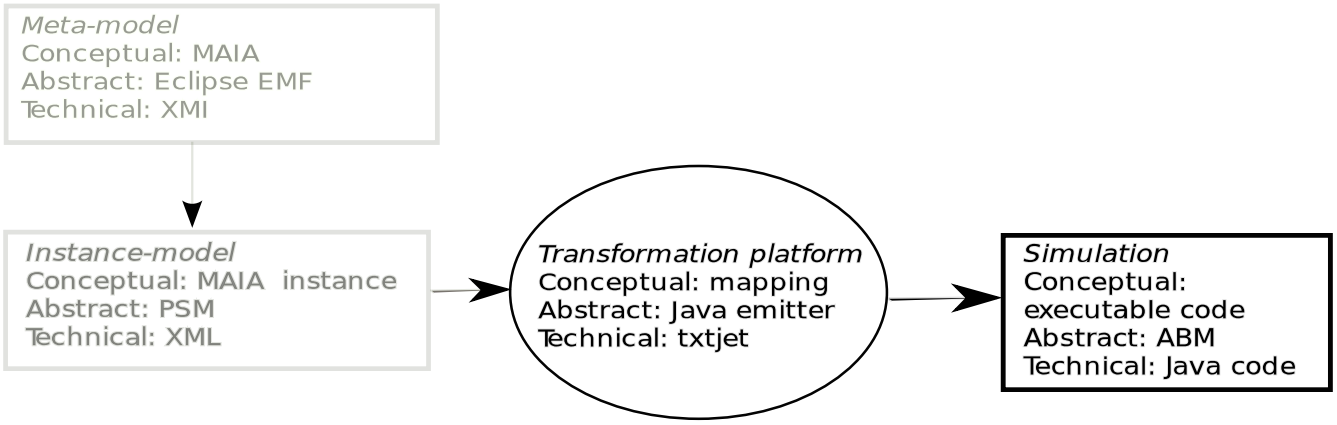



Figure 3.4: In the current implementation Text output is generated based on a custom created translation-model. This translation-model is tied to the MAIA instance model by a custom Java class. This custom class generates Text output that is used at a later step to create actual Java files and packages.

Figure 3.5 below describes a zoomed in version of the transformation platform.

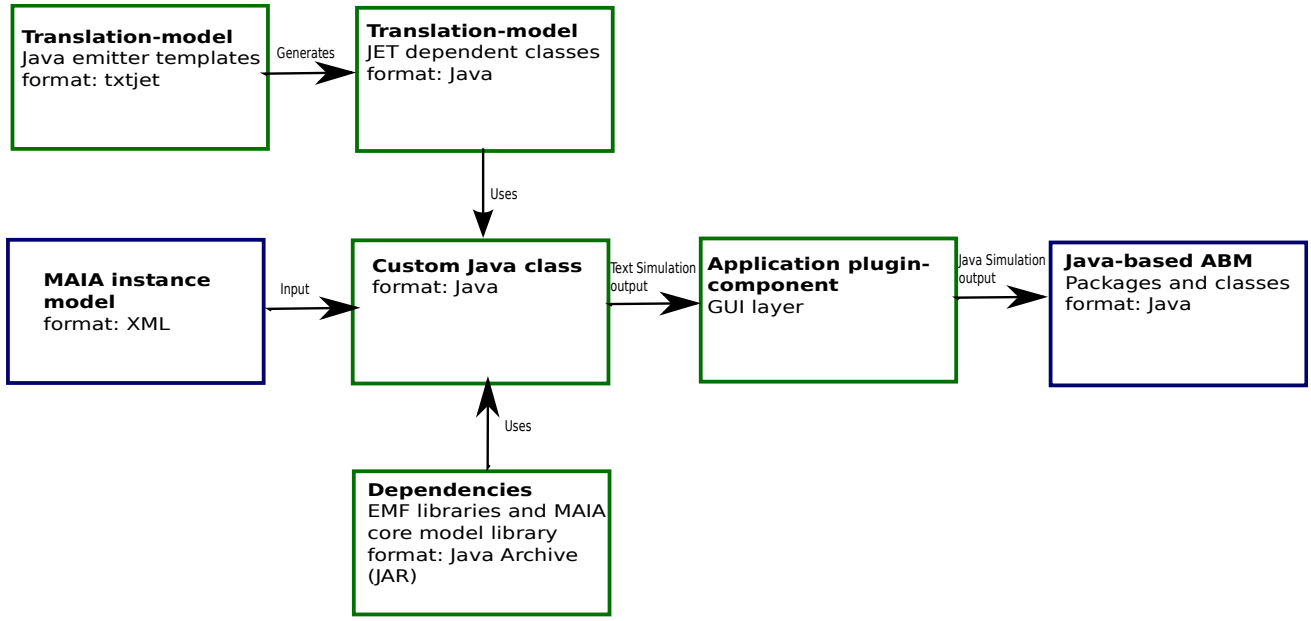



Figure 3.5: The MAIA instance model serves as input to the custom Java class. This class uses the translation-model's Java files to configure Java-based relations with respect to the instance model. The custom Java class is also dependent on EMF and the MAIA core model through Java libraries. The whole process is on the background of the developed application, meaning that users only need to specify the MAIA instance model location and Java code is generated based on that. Additionally just before that step a GUI is presented for runtime information (see box Application-plugin component).The green boxes signify the embedded processes in the application. The final Java-based ABM generated is independent of EMF and MAIA libraries.

JET is a useful technology for automatic code generation. In this thesis JET is used as a supporting tool for creation of Java code from MAIA instance models. Below in figure 3.6 and 3.7 two examples of JET usage.

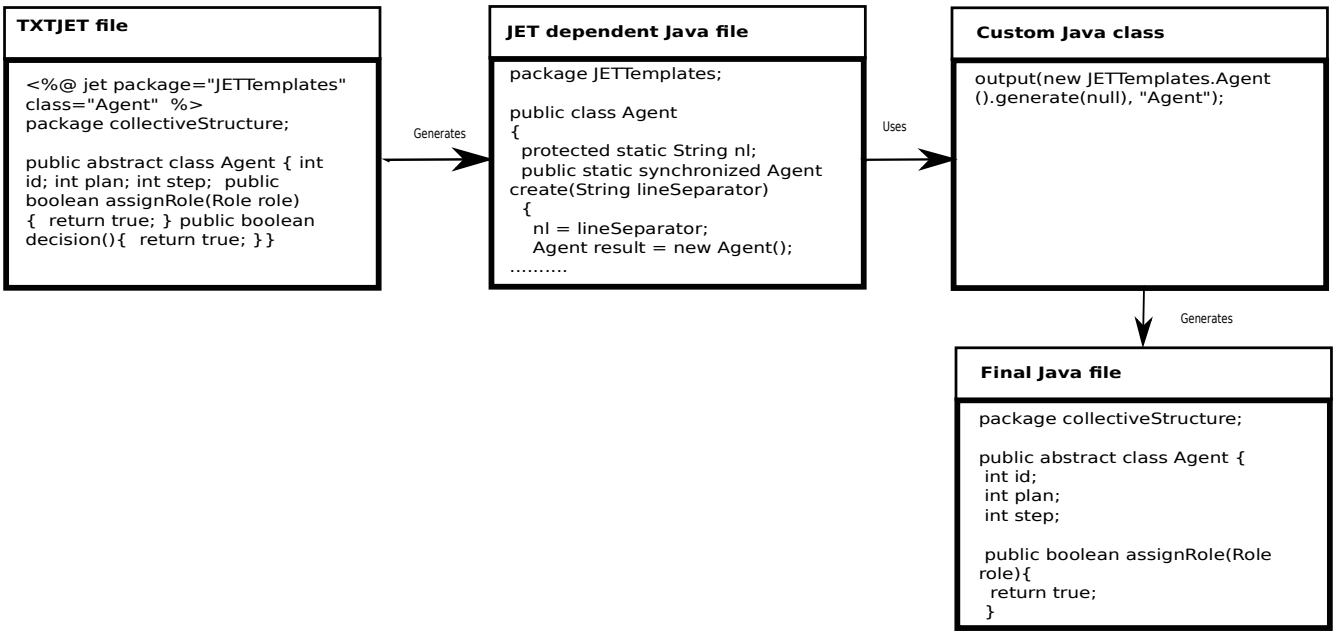



Figure 3.6: The txtjet file contains at first a description on the location in which the output should be set. It specifies the class name and specifies the package along with possible imports if needed (not shown in this example). Under the description settings, the actual contents of the file are projected. In this case it is in abstract class, thus no information from MAIA instance files is needed and this can be hard-coded. The custom Java class uses the output of JET to construct the final Java file. In this case not much is generated besides a standard abstract class.

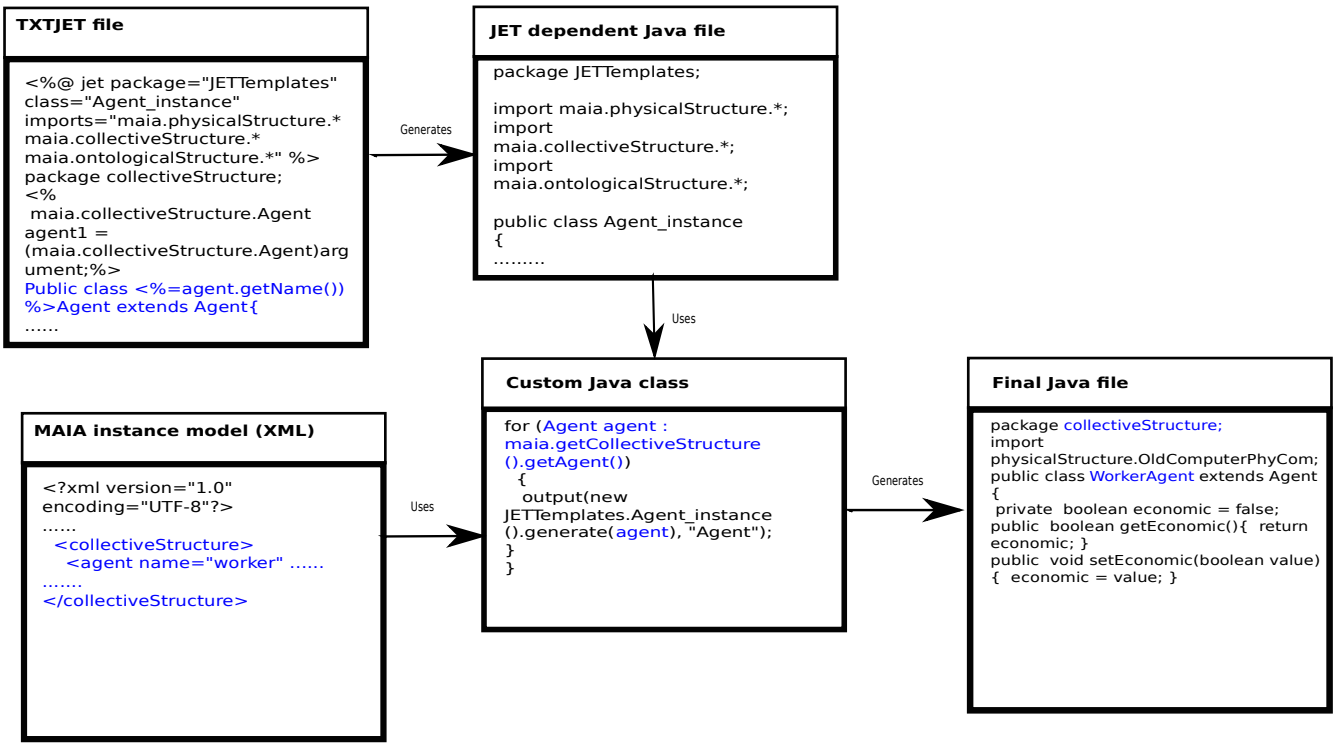



Figure 3.7: The txtjet file instantiates an object of the type Agent. The custom Java class uses the generated Java file from txtjet as a skeleton. This skeleton specifies how the output file should look like (it defines the class and package name, it defines additional type descriptions of MAIA instance data if needed). The actual MAIA instance contains agents that can be parsed into this skeleton, thus creating final Java files for each agent that contains all needed information. This step is performed for all MAIA structures, hence a final Java-based ABM model with multiple packages and classes can be created. See blue text for a mapping process example.

# 4 AMIE - Agent-based Model-driven Integrated Environment

The Material and Methods chapter presented the concepts and components required for application development of the platform. The previous chapter explained in detail the methodology of connecting the software components together. However the software architecture and the software functionality has not been discussed. Therefore, the following sections will address these aspects and this is supported by a case study in chapter 5 that explains the steps from conceptual model development, to implementation, to simulation.
Agent-based Model-driven Integrated Environment (AMIE) was developed as a proof of concept, aimed to demonstrate that MDSD can be used to bring about a platform for ABMS, specifically for STS. AMIE tries to lower the barrier for non-programmers, through structured Agent-based model creation, implementation and simulation in an embedded environment. AMIE functions on multiple OS's such as Linux (tested during development), Windows (tested during development) and Mac. The different versions can be found on GitHub https://github.com/SidneyNiccolson/RCPplatform and the only prerequisite is that Java version 8 or higher is installed on the computer.

## 4.1 Software Architecture

The software architecture of the tool has been developed with Eclipse-based tutorials and the EMF Developer's Guide (Frank Budinsky, David Steinberg, Ed Merks, Raymond Ellersick, 2003). Please refer to appendix 6 'Development MAIA-based RCP' for more in-depth information on the development processes including dependencies for creating the decision support tool. The complete source code with additional third-party tutorials can be found on the GitHub page. The Eclipse Modeling Framework has been used to integrate the MAIA meta-model, in turn instances can be created by an EMF editor (EMFForms) previously illustrated in figure 3.3. The XML instance output can be used by JET to produce simulation ready code as described in chapter 3.3 'JET processes'. All these components together are embedded in the Eclipse Rich Client platform. See figure 4.1 on the next page for a high level abstraction of the AMIE architecture.

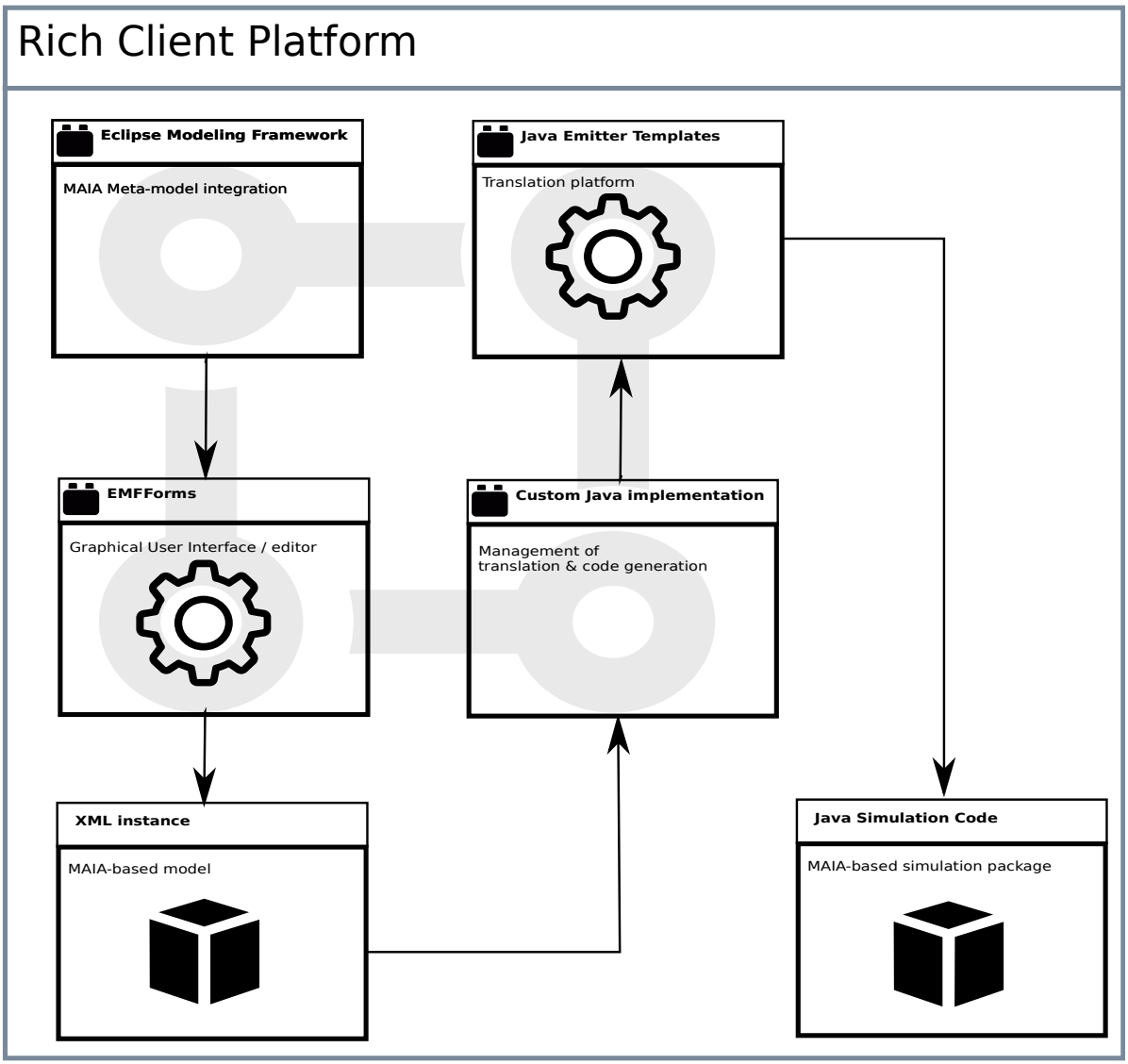



Figure 4.1: A high level abstraction of AMIE architecture.

EMF, EMFForms, JET and custom Java implementations are the modules used to allow for the integration of the MAIA meta-model, the creation of MAIA-based models and MAIA-based simulation packages. Users leverage the EMFForms editor and parts of the Custom Java implementation, while the EMF and JET modules are background processes. In the preceding section more details on the user-side of AMIE.

## 4.2 Software functionality

With AMIE, users interact with the application through interfaces to manage files, experiments and develop models and simulations. A breakdown can be made between the various user processes namely:

- MODEL CREATION. After a MAIA conceptual model has been developed and implementation details have been described, the model can be implemented in the application. Thereby model specific inputs can be given using the EMF model editor (EMFForms). Once the model is finished the model can be used for simulation code generation.
- SIMULATION CODE GENERATION. The format of the model created in model creation is XML. JET parses through this file, using EMF/MAIA libraries in order to create simulation code. This process is shortly interrupted by a dynamic Graphical User-face, that asks the user for runtime simulation information based on the input model. The Dynamic GUI will be further described in the preceding section.
- SIMULATION. Once the code generation process is finished, a bundle of various Java packages are generated representing the MAIA instance model data as a project in AMIE. The bundle of packages forms the Agent-based simulation code and reside in the Project Explorer. The simulation procedure can now be started.
- EVALUATION. During runtime of the simulation, the user is notified the current progress of the simulation. The simulation output is a Comma-delimited File tracking all agent attributes of the simulation, in addition users can specify visuals to be generated during simulation.

Chapter 6 showcases a step-by-step guide on the user interface processes described above. See figure 4.2 on the next page for an overview on how a user typically interacts with the AMIE software.

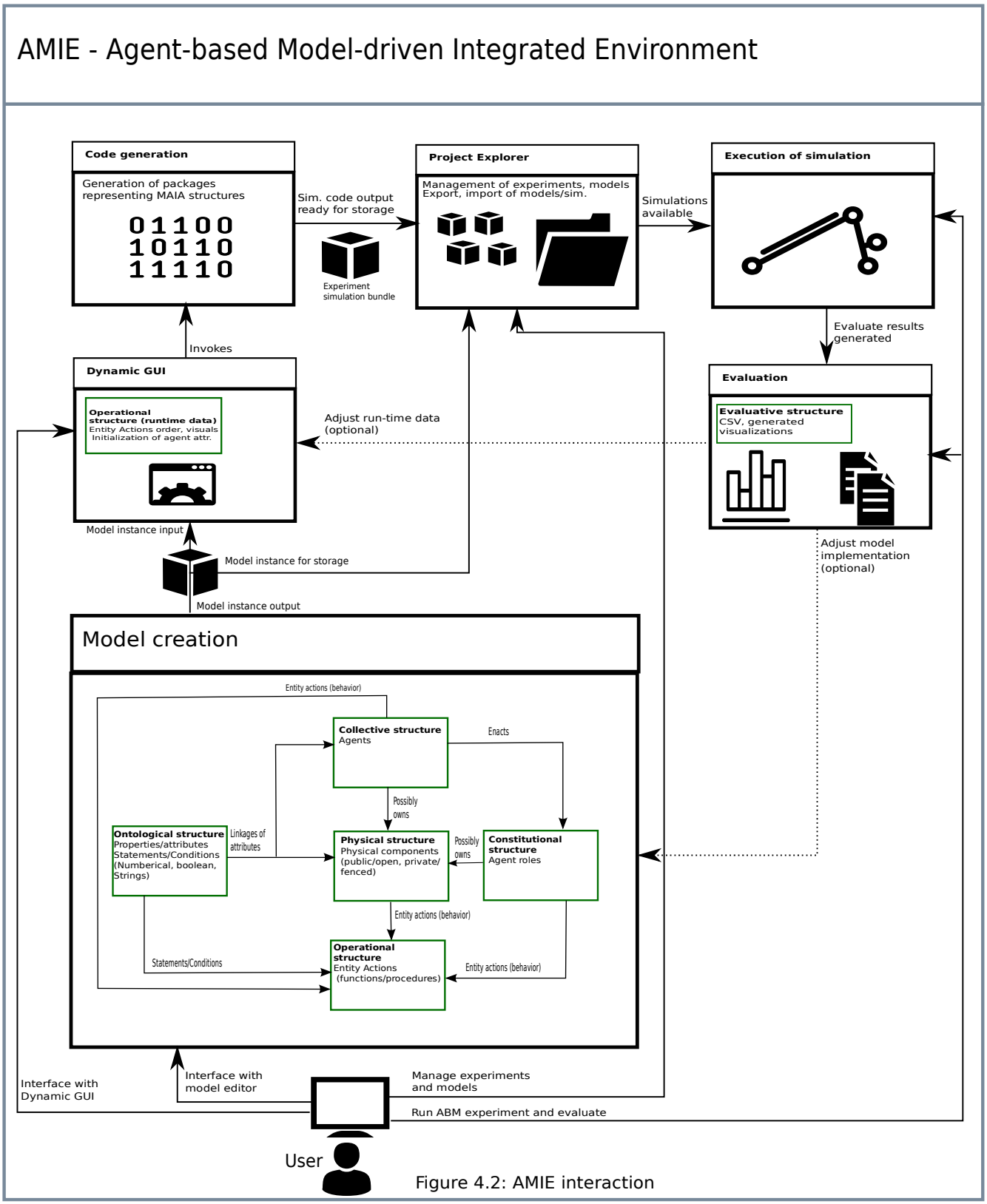



Figure 4.2: AMIE interaction

### 4.2.1 Procedural semantics

In order to construct and operationalize an ABMS with the developed platform, a description of the procedural semantics is needed. In this section of the thesis paper an explanation from a high-level perspective will be given on the procedures of a simulation. These procedures are defined by the user in the model and the supportive Dynamic GUI which has been described shortly in figure 3.5 box 'Application plugin component GUI layer' and in the previous section. If deemed necessary pseudo code will be provided. The procedural semantics of the platform follows for a large part the semantics described in the paper 'Procedural semantics of a MAIA model' (A. Ghorbani et al., 2014). A detailed description will be given on the general operationalization of any ABM created with AMIE which will be referred to as the 'main simulation'. The specific procedures that agents perform, are referred to as 'entity actions'. The '<' and '>' symbols describe the boundaries of what should serve as input specific to a case study, for example if the explanation points to an agent of a case study we denote it as: <Agent>. If an explanation describes a general method it is denoted as 'method name' with a () symbol in the end. Any comments in the pseudo code given will be denoted in italic with a preceding // sign.

#### THE MAIN SIMULATION

The main simulation can be dissected in 5 tasks namely initialize(), main(), analyseData(), createCSV(), generatePlots(). The general initialize() method, initializes the agent types of an ABM. Meaning that in the model's collective structure, <Agent> is initialized. Thereby their amounts and potential variations of specific attributes are specified. For example in a model the agent type <Citizen> might have as attributes <Age>, the modeler might want to have 50 of those agent types, with varying <Age>. The agent types are captured in so called “lists”, which are data structures that hold elements (each element signifies an agent). The box below shows the initialize method in pseudo code. In addition the number of ticks is defined in this method.

```
Initialize()
   set<Ticks> //set the ticks defined by the user in the GUI
   int numberOf<Agent> //set the number of agent types defined by the user in the GUI, e.g. 50
   for numberOf<Agent> //loop over the number of agents specified
        set ID // give the agent an ID (if it's a Role the ID is the same as the Agent enacting the role)
        set variation of <attributes> // give the agent an attribute value in a range, e.g. age 1-70
        append<Agent>      //add the agent to the list of that specific agent type
```

The analyseData() method tracks attributes of the ABM per tick and writes them to a Comma Separated Value (CSV) file. All agent's properties and its physical component's properties are tracked. If visualizations for attributes are defined in the Dynamic GUI a list is constructed for it. And the tracked attributes are appended to that list.

```
AnalyseData()
    get ticks // determine what tick the simulation is in
     for each <Agent>
         get<Attributes> //get the attributes value per tick
         addToPlotList<Attributes> //construct the data to be plotted on the fly
         writeToCSV //write all data immediately to a CSV per tick
```

The createCSV() method returns a CSV object, this object is a pointer to an empty CSV file. In the analyseData() method this pointer is used to write all data immediately to the CSV. The generatePlots() method uses the constructed list of the analyseData() method to generate plots.
The main() method invokes the initialize() method, it defines what agent's should do per tick and invokes all other methods. Below the pseudo code for the main() method.

```
Main()
   printStartOfSimulation //Output to the console that the simulation started
   initialize() // invoke initialize method
   getTicks // get the ticks specified in initialize method
   createCSV() // get the pointer to the CSV in order to write to it
   for each <Tick>
      shuffleAgentList //make sure that per tick the agent entering first is different
      for each <Agent>
           invoke<EntityActions> //call entity actions
      analyseData() //invoke analyseData() each tick
   closeCSVfile  //Close the CSV file ,because at this point all data has been collected
   generatePlots() //as the lists for plotting are constructed we generate them all at once
   printEndOfSimulation //Output to the console that the simulation has ended
```

Notice the shuffleAgentList, agent entity actions are ordered by the user, but this does not create necessarily a deterministic model by default. This is because agent lists shuffle which causes each agent to enter at a different time during a tick. For example this creates the possibility to limit agent actions (e.g. if resources has depleted because other agents already used them, the agent entering at a later point cannot use the resources anymore).

### CONCEPTUALIZING THE NOTION ON AGENTS

So far in this section agents are described as part of the collective structure. In ABMS an agent can be anything, thus the MAIA theoretical framework also describes Roles and Physical Components. Conceptually speaking they can be an agent. For instance the agent <Citizen> might enact the Role of <Consumer>, in that Role he/she might have access to Physical Components <Car>. Those Physical Components have properties <Gas>. In this case the Role of <Consumer> signifies that it is an agent, as it might perform entity actions such as driving the <Car>. A Physical Component might be an <ElectricalGrid> and the properties could be <TotalDailySupply>. If the Physical Component is specified as “OPEN” in the model, this information is freely available by all agent's in the system. The <TotalDailySupply> might be determined by an entity action <Monitor>. The monitoring of the electrical grid might be performed by the Physical Component itself, thus in this case it is an agent. With these examples Roles and Physical Components are both viewed as agents and the code generation of the main simulation will include them and their properties. The difference is that the Physical Component if set to “OPEN” will not be initialized, but is a static class (for more information on static classes see section 2.3 object oriented programming).

### 4.2.2 Error handling

The developed platform has multiple error handling mechanisms in place which are shortly described below.

- Dynamic GUI error handling: In the Dynamic Gui user input is limited to the relevant data input required. For example the number of ticks can only be specified by Integer data types (whole numbers). Number Properties of MAIA agent's can be of data type Double (decimal numbers). Letters (String values) are typically not allowed and cannot be set in these text boxes. If the user does not fill in all required information for simulation, a notification will appear to show that not all data required is filled in.
- The Log Package: Once code has been produced from a MAIA model, a Log Package is created displayed in the Project Explorer. Inside resides a log.txt that tracks how successful the code generation went. If something went wrong it should be visible in this text file.
- Default Java-Eclipse error messages: It cannot be fully prevented that users give wrong input's during model creation. However Eclipse provides general error messages if simulation code is not complete or faulty. These error messages likely reside in the MAIA generated structures and users can open the respective Java classes. Often Eclipse provides a recommendation on how to solve the error message.

# 5 Case study

The previous sections elaborated on the software architecture, functionality, in turn this section will emphasize on a case study. Thereby showcasing how MAIA-based models can be developed and implementation details for using it in the software package AMIE. In order to explain the functionalities of the tool, an example of three subsequent scenarios will be elaborated, showcasing how an electrical grid is influenced by different parameters, such as seasonal changes, behavioral changes and subsidy for renewable energy. This example is not a comprehensive ABM using real-time data. The reason for this approach is that some assumptions will be oversimplified for the purpose of explaining the inner workings of the platform and how the tool can be used for STS model development. Moreover, scenario 1 explores the development of a basic supply and demand model. Scenario 2 explains the impact of subsidy for renewable energy specifically for residential housings. Scenario 3 will showcase the changes in the system if smart-monitoring is widely implemented. Models for all scenarios and output files can be found on the GitHub page. Chapter 6 will encompass a tutorial for AMIE test-users using the case study as an example.

## 5.1 An introduction to the case study

An electrical network (grid) supplies electricity to customers usually on a national or even international level, such a system is a fairly complex type of network. In conventional electrical networks demand drives the supply. Meaning that supply follows demand and electricity generation is centralized (H Gharavi, 2011). Since the emergence of renewable energy the conventional network has become inflexible, mainly due to the intermittent nature of renewable energy. Currently policies, technological development and environmental concerns drive the energy system as a whole towards an efficient and highly dynamic system. Balancing a conventional centralized network gets more and more complicated when the use of renewable energy increases. Hence the concept of smart-grids has been developed over the years that potentially offers new ways of balancing electricity grids. Smart-grids are electrical networks that integrate the demand and supply of electricity intelligently (Clastres, 2011). This means that the behavior and actions of users of smart-grids are to a large extent coordinated, thus leading to an efficient delivery of energy and less energy losses. In addition not only intelligent coordination of demand and supply, but especially the connection of renewable energy sources to smart-grids creates opportunities for more sustainable electrical networks. Currently renewables can lead to fluctuations when connected to electrical grids and extensive management of supply and demand is needed. Smart-grids provide measures for mitigating fluctuations by performing real-time measurements of consumption and power plant outputs, in turn managing the supply and demand.
From a political point of view smart-grids have become more interesting. The European Union has been making changes to the electricity market to meet several new targets for sustainability and deregulations (Clastres, 2011). Business as usual created a situation of high economic efficiency leading to low cost of energy for consumers. However growth of energy demand and concerns on climate change caused decision makers to recognize the need for more active regulation. There are various market models that showcase different degrees of market openings and which model used differs strongly per country. Recent research seems to point towards giving more control to the grid operator as that entity has most control over energy flows (Integrated Pool Market model) (Stacke, 2008). This creates more optimization opportunities, proper

management of renewables and transparent network data. The real-time sensors that smart-grids provide fit with this market model.
Next to the political aspect, from a socio-technological perspective, smart-grids seem interesting as well due to the electrification of transport, the changing roles of consumers towards producers and the emergence of IoT devices. These devices provide readily available interfaces for intercommunication of relevant data, thus possibly supplying information on the daily usage of the device and its power requirements. In terms of electrification of transport, more and more electrical vehicles are developed that in theory can also be used to store energy. In times of over-supply of energy, electrical vehicles may be automatically charged to store the energy. In addition consumers are now able to buy their own PV cells, hence becoming producers and consumers at the same time.
To showcase recent interest in the Netherlands, we see that Amsterdam's local electricity network Alliander has put a large investment in smart-grid technology aimed at the utilization of network sensors and improvement of domestic energy monitoring to trim peaks of electricity use (Amsterdamsmartcity, 2016). Matching supply and demand is a key aspect of such a system.
Many aspects of the smart-grids are uncertain, therefore certain questions arise such as: what implementation efforts are needed to integrate smart-grids in a typical city? What are the current energy requirements? What is the energy mix? Does a smart-grid lead to lower energy use? What are the patterns of energy usage in individuals both currently and in the case of smart-grids? What incentives drive efficient energy usage? What willingness do consumers have to pay more for green energy? It is expected that smart-grids will lead to lower overall energy usage based on efficient management. For example peak demand situations can be balanced with smart-monitoring and energy storage. Costs for smart-grids might be high in the short term with all the investments needed by municipalities and possibly consumers as well. In the long term it is expected that the costs will be lower due to, for example, energy savings, lower costs of smart-grid technology and potential return of investments. Also less tangible developments such as increased knowledge on smart-grids by learning by doing, knowledge sharing may lead to lower costs for installation and maintenance. Although smart-grids offers potentially a more sustainable electricity network, the questions around the actual implementations cannot be answered with a single one size fits all solution, and ABMS might help in answering questions on the dynamics of an electrical network.

## 5.2 Conceptual MAIA-model development: MAIA structures applied

The model that is going to be developed serves mainly the function of instructing the basics on how to develop a MAIA model with regards to the platform, however it is important to still include aspects such as heterogeneity, randomness and agent behaviors. It is also interesting to see what a simple model can describe for conventional and smart-grid based networks. We define the model according to the structures of the MAIA theoretical framework and provide extensions with as goal to go from a simple model towards a slightly more complex one. Assumptions are made to simplify and to set boundaries for the ABM. We depict agents in italic, MAIA concepts capitalized, attributes and physical components capitalized without spaces.

### CONSTRUCTING A BASIC MODEL ON A CONVENTIONAL GRID (SCENARIO 1)

As mentioned a conventional grid is relatively inflexible, demand varies on a daily, weekly, seasonal basis. Supply varies as well and in most cases inflexible power plants are used to supply a base load. For instance

nuclear power plants (class 2) have high costs if they have to shutdown, so they tend to be on all the time. Others such as coal or gas fired power plants (class 1) are more flexible and can follow demand.
Renewable based power plants (class 3) are intermittent and can drive oversupply at times (Balancing Mechanism Reporting Service, 2016). See figure 5.1 to showcase how grid supply and demand fluctuations take place in a month in UK.

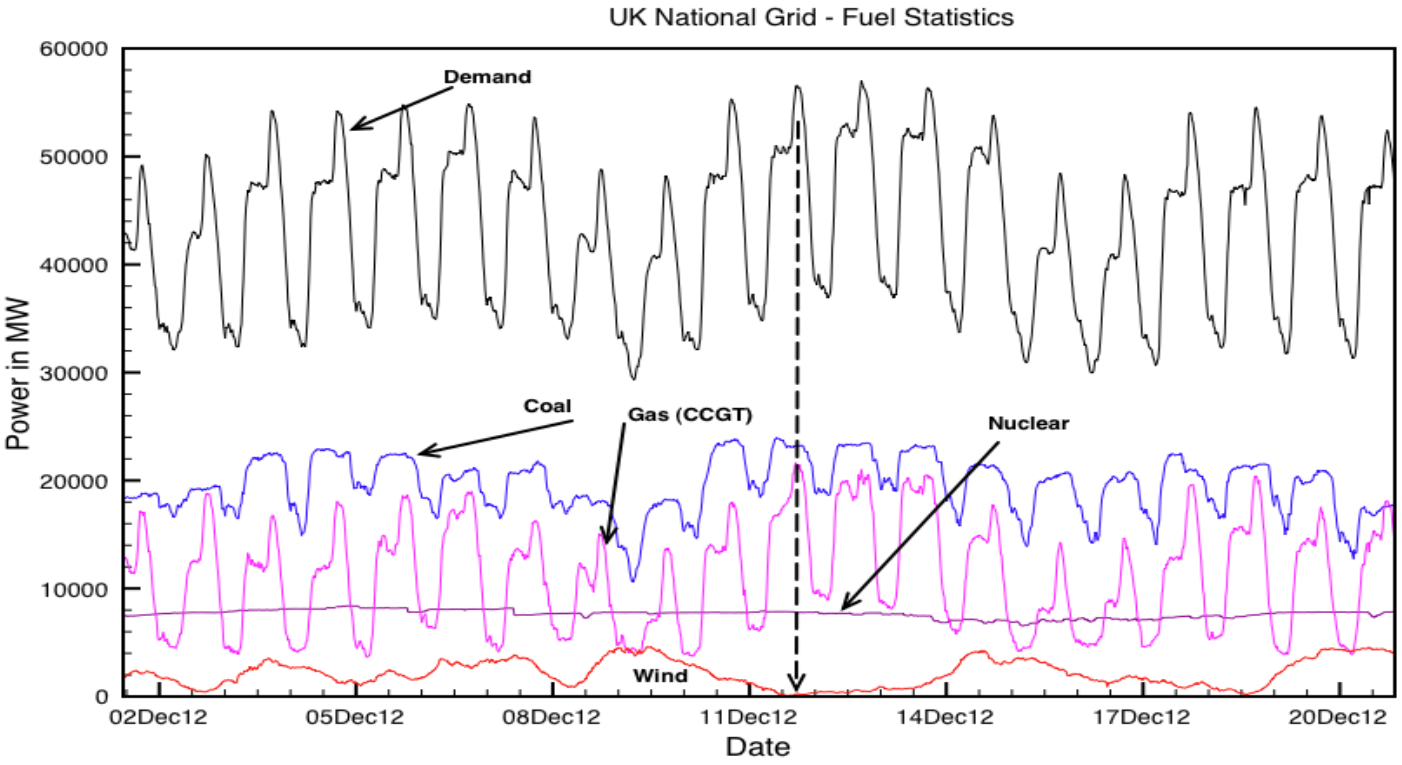



Figure 5.1: Statistics of the supply and demand in UK in December 2012. Coal/Gas follow demand and nuclear provides a base load. If wind energy is high coal/gas plants tend to shut down. (Source: Balancing Mechanism Reporting Service, 2016)

Let's simplify the model that is going to be built and assume the electrical grid only supports a single hypothetical town in the Netherlands of 100 inhabitants. In turn omit any complicated technical aspects of grids and set the time scale to a daily basis. The model should run for a year that is 365 ticks ('days') and is focused on effects of consumer behavior rather than technological developmental changes of the grid. It is important to account for weekly and seasonal changes of demand in this case. The total daily demand will be correlated according to the season and week. Supply will be provided by power plants of only class 1 and 2. Renewable-based power plants will be omitted, but citizens may own their own PV systems. Assumptions will be made based on sources from the United States and countries in West-Europe. To clarify the aim is to discuss how AMIE can be used to develop models and as a case we will study the impact of PV use in neighborhoods by consumers. The commercial sector is included, but their role in this model is relatively small.

### THE COLLECTIVE STRUCTURE

The Collective Structure represents only composite agents namely *active citizens*, *passive citizens,* and three other agents representing a *public-commercial-industrial sector*, a *nuclear* and a *gas/coal* based *powerplant company*. *Active citizens* and *passive citizens* both represent an average household of 5 persons. Properties of *active citizens* are DailyCitizenElectricityUsage, DailyElectricityGeneration and DailyBalance.
*Passive citizens* have only the property DailyCitizenElectricityUsage. The *commercial sector* has the attributes DailyComElectricityUsage and DaysOfEnergyUse. Powerplant companies own physical components which will be described in the Physical Structure.

### CONSTITUTIONAL STRUCTURE

Citizens enact the Role of either *prosumer* (*active citizens*) or *consumer* (*passive citizens*). A *consumer* and *prosumer* both use electricity from the grid, while a *prosumer* also generates electricity. A *prosumer* owns a physical component SolarPanelSet. The power plant companies do not take any Role, we assume that they implicitly are generators of electricity.

### PHYSICAL STRUCTURE

The *nuclear power plant company* owns a NuclearPlant and generates the base load. Its property is the NuclearOutput. *Active citizens* have as a Physical Component a SolarPanelSet which has as properties the DailySolarOutput. The *gas/coal power plant company* has a CombinedPowerPlant as a Physical component with it's property CombinedOutput. The physical structure also contains the electricity *grid*, which is public ('Open'). The *Grid* has as attributes the CurrentAvailableEnergy, NetBalance, AccumulatedOverSupply. The NetBalance in this model reflects the possible oversupply of the *Grid* or undersupply. In the case of undersupply the CombinedPowerPlant needs to satisfy the demand. Figure 5.2 shows a sketch of the model.

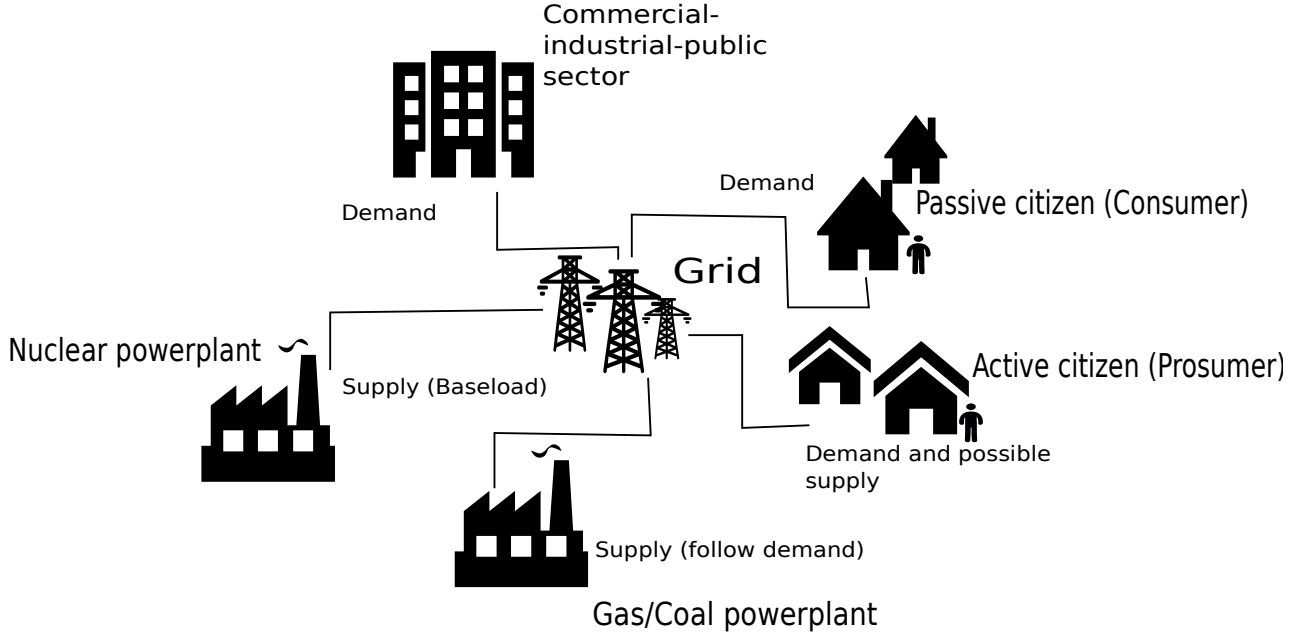



Figure 5.2: A rough sketch of the model.

OPERATIONAL STRUCTURE

This model takes a timespan of one year. A tick signifies a day. We assume that not all demand of the town is from citizens, but a part of the demand is determined by the *public-commercial-industrial* sector.
Realistically a household would in most cases not meet its own total energy demand with a PV system if we take into account factors such as the use of heating and the use of cars (David & Mackay, 2009). In this case the focus is solely on electricity demand excluding the whole of energy demand. If we exclude the whole of energy demand, solar panels of *active citizens* can provide more than enough electricity for the demand in some cases in this model. This is an interesting aspect as cases have already shown that neighborhoods full of PV systems may provide more than is demanded on some days (H Gharavi, 2011). However if the average yearly demand is assessed for households, there are many fluctuations in the demand on hourly, daily and yearly basis. See figure 5.3 for an overview of demand in the total of UK from January to July 2009.

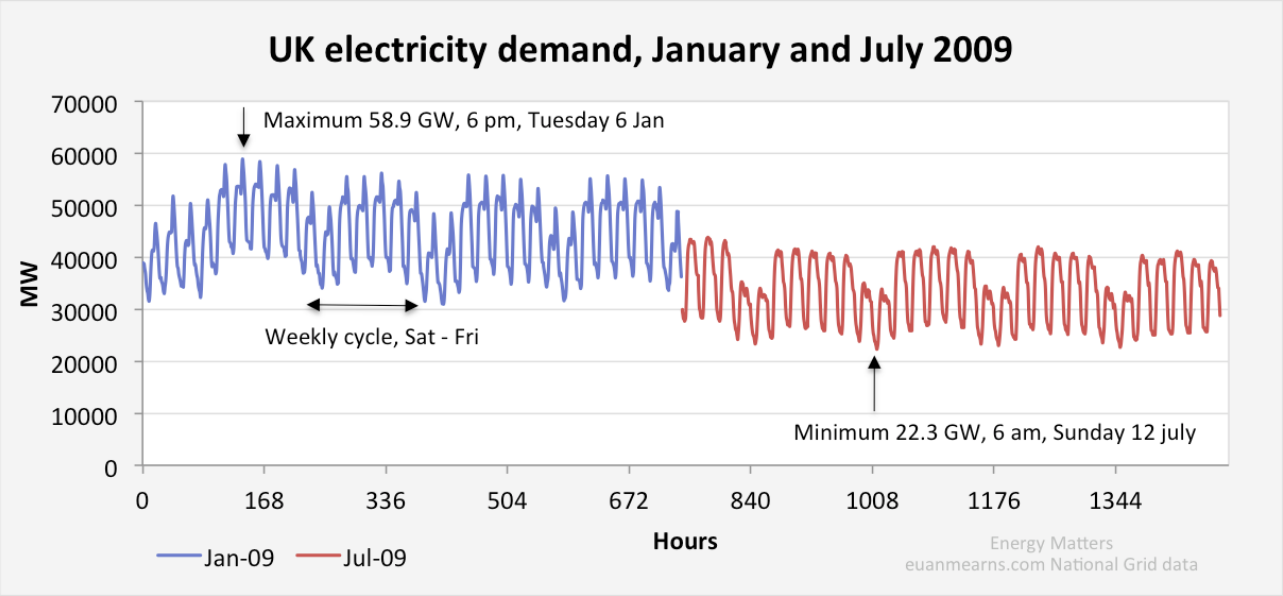



Figure 5.3: Electricity demand in the UK. The weekly cycle shows a lower demand in the weekend and the seasonal cycle shows a higher demand in the winter. (Source: Balancing Mechanism Reporting Service, 2016)

As we want to account for seasonal and daily changes of demand, extensions will be provided in the next sections that includes them. However firstly, a static deterministic model is built in which there are no changes in the system per season or per day. Similarly citizens would have a fixed demand and the same accounts for the *public-commercial-industrial sector*. The SolarPanelOutput is realistically for a large part determined by the sun intensity in a day and the temperature (SunPower, 2016). In the next subsection we account for those factors as well.
In this model we assume that the base load supplied by the *nuclear power plant company* provides enough for the *public-commercial-industrial* sector. The demand of citizens might put the *grid's* CurrentAvailableEnergy in negative within a tick. In turn we assume that the *gas/coal power plant company* follows the demand, thus setting the *grid's* CurrentAvailableEnergy to zero, while updating the *gas/coal power plant company* CombinedOutput property and setting the NetBalance property with the value of the old negative

CurrentAvailableEnergy. If the CurrentAvailableEnergy is above zero due to *active citizens* supplying surplus electricity to the *grid*, we assume that the *gas/coal power plant company* shuts down and does not provide electricity. The CurrentAvailableEnergy would be set to zero at the end of the tick again, but the NetBalance property will be set with the positive old CurrentAvailableEnergy value for tracking purposes.
In each tick the *consumer* simply uses electricity from the grid. The *active citizen* uses electricity and generates electricity. If the generated electricity is high enough they will not use any electricity from the grid and supply any surplus electricity. The *power plant companies* use their physical components to supply electricity to the *grid*. Finally the *grid* monitors the electricity distribution in which the case of oversupply the property AccumulatedOverSupply will be updated.
We can describe the action situations in a more structured way in the following table. For a detailed explanation of entity actions refer to the appendix 7 'Entity Actions In Detail' and its subsection 'Scenario 1 Entity Actions Explained'.

Table 5.1: Agent's coupled with action situations and model simulation updates for one tick in order

| Agent | EntityAction | Perfomer (agent or role) | System updates (attr.=attributes) |
|---|---|---|---|
| *Nuclear Powerplant company* | **supplyEnergyAsNuclearPowerPlant** | *Nuclear Powerplant company* | ++CurrentAvailableEnergy<br>by attr.: DailyNuclearPowerPlantOutput |
| *Public-commercial-industrial sector* | **useEnergyAllSectors** | *Public-commercial-industrial sector* | --CurrentAvailableEnergy<br>by attr.: DailyComEnergyUsage |
| *Passive citizen* | **useEnergyAsConsumer** | *Consumer* | --CurrentAvailableEnergy<br>by attr.: DailyCitizenElectricityUsage |
| *Active citizen* | **generateEnergy** | *Prosumer* | ++DailyElectricityGeneration<br>By attr.: DailySolarOutput |
| *Active citizen* | **useEnergyAsProsumer** | *Prosumer* | +-CurrentAvailableEnergy<br>By attr.: DailyCitizenElectricityUsage & DailyElectricityGeneration |
| *Gas/coal powerplant company* | **followDemand** | *Gas/coal powerplant company* | ++CurrentAvailableEnergy (if needed)<br>by attr.: DailyGasCoalPowerPlantOutput<br>+-NetBalance (depending on oversupply or undersupply) |
| *Grid* | **monitorDistribution** | *Grid* | ++AccumulatedOverSupply (if occurred)<br>0 = CurrentAvailableEnergy (at end of each tick to zero) |

EVALUATIVE STRUCTURE

It is expected that their might be short periods within ticks of power shortages in the case that there are not enough *active citizens,* at the end of each tick this shortage is offset by load following if necessary. The initial distribution of *active* and *passive citizens* determine whether or not their will be power shortage or oversupply of energy for this simple case. We want to analyze the changes of energy flows on a daily and yearly basis. In general in ABMS, thorough analyses should be done in the evaluation of the model, including multiple experimental runs, parameter sweeps, more in-depth tracking of agents. Please refer to chapter 7 for recommendations for improvement of this model for more information.

## 5.3 Simulation implementation details

The hypothetical small town model with a population of 100 uses approximately 995.6 MWh a year (PSO Oklahoma, 2016). Based upon the town's demand we can calculate that 27.3 Kwh per entity is demanded in total on average (995.6 MWh/365 days=2.73 MWh, 2.73/100 entities=0.0273 Mwh), considering the simplified model (PSO Oklahoma, 2016). To validate the magnitude of household (citizens) demand another source is used that explains that on average a household uses 7200 kilowatt-hours per year in the USA (ucsusa, 2016), that is on average 19.7 Kwh per day excluding seasonal changes. In the Netherlands the situation is slightly different as gas is mostly used directly as a source for heating, thus less electricity is needed for heating. We take the number from NUON (one of the dutch electricity companies) for a typical household of 5 persons with three kids, the average electricity use is 12 Kwh a day (Milieu Centraal, 2016).
We assume 24 panels with a average output of 14 KWh per day (Understand Solar, 2016). In terms of supply 995.6 Mwh should be supplied at minimum in the given year.

### TESTING THE MODEL

As mentioned a deterministic model is developed at first, in which in a later stage we are going to include demand and supply dynamics. In this deterministic model we specify the number of agents in the system in a fixed matter. We want to sketch the situation in which we can see small oversupply, due to the PV system. In turn in the extension we are going to change the parameters with uncertainty and supply/demand changes.
With fixed parameters we can calculate the following expected outputs:

- The town: the *grid* should supply in a given year 995.6 Mwh in total. That is 2730 Kwh per day on average (995.6 MWh a year/365 days=2.73 MWh per day).
- Demand *public-commercial-industrial sector*: as mentioned this sector is not the focus, hence we will view the entities making up the sector as a single agent. The *public-commercial-industrial sector* uses 63% of the demand (Glass for Europe, 2016). This is a crude percentage as it reflects the total of energy not only electricity and includes actually everything (e.g. transportation) except households. However for the sake of simplicity this sector would demand .63*2730 = 1719.9 Kwh per day.
- Supply *nuclear power plant*: To offset the demand of the *public-commercial-industrial sector* we assume they supply 110% of that amount. That is 1891.89 Kwh per day (1.1*1719.9). The CurrentAvailableEnergy of the *grid* is now +171.99 Kwh (1891.89-1719.9) within a tick.
- Demand households: for the validation of the initial model, we assume a distribution of 75 *active* and 25 *passive citizens*. Thus 25*12 Kwh per day is 300 Kwh per day as demand per *passive citizen*. We assume oversupply for the PV system as of now, which we could say that *prosumers* supply 2 Kwh per agent per day (14-12=2) to the *grid*. 75*2 is 150 Kwh supply per day.
- Expected output: An oversupply of energy on a daily basis. After the *consumers* used electricity from the *Grid* the CurrentAvailableEnergy is -128.01 Kwh (+171.99-300=-128.01). This is offset by the oversupply of 150 Kwh leading to a small oversupply of 21.99 Kwh per day. Calculating this through all the days of the year we get to an accumulated oversupply value of 8026.35 Kwh in a year which should be showcased in the property AccumulatedOverSupply. The gasCoalPowerPlant will not be used in this case, although the entity action has been defined in case of undersupply. See section 5.4 for simulation results of this model.

Particular in the supply and demand parts we want to introduce heterogeneity. Other scenarios can be developed after the first model. Scenarios that adjust parameters such as the amount of *active* and *passive citizens*, the *powerplant* outputs and DailySolarOutputs, the DailyEnergyUsage by *citizens*. See figure 5.4 for a visual depiction of the model implementation details, according to the MAIA structures.

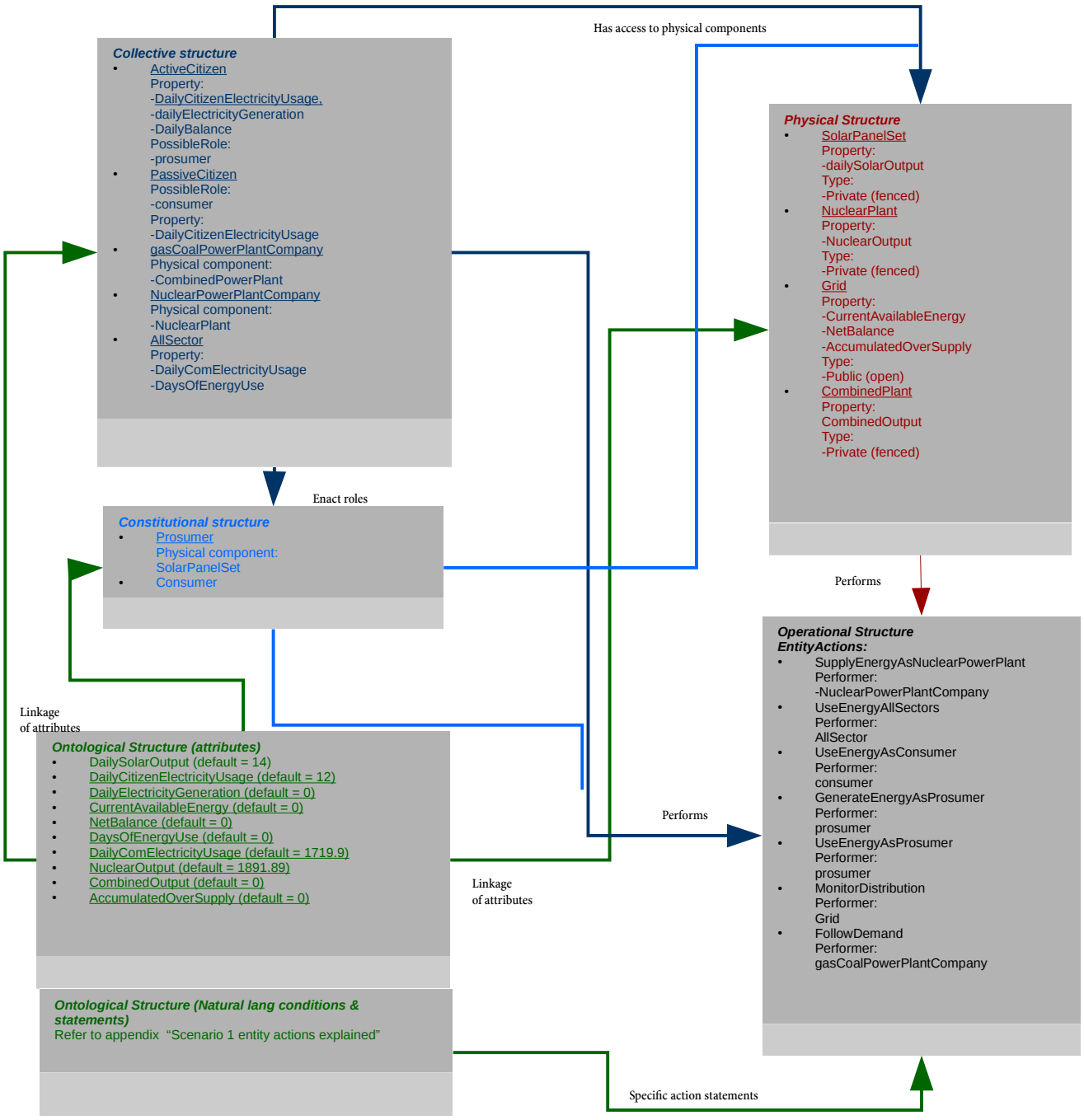



Figure 5.4: MAIA Implementation details

The following subsection elaborates on extensions of scenario 1, followed by scenario 2 and 3. In this stage the model will implement more aspects of ABMS such as variety, randomness and a multitude of attributes effecting the system as a whole. Those attributes should reflect the daily cycle, weekly cycle and seasonal cycle of demand and supply variations. In turn the *Grid's* NetBalance will be influenced considerably. Chapter 5.4 will illustrate the results from these scenarios.

#### DEMAND DECREASE IN SECTORS IN WEEKENDS (EXTENSION SCENARIO 1)

The *public-commercial-industrial sector* (named as AllSector in the platform) is a composite agent, containing institutions. Concerning the weekly cycle these institutions tend to have less demand in the weekends. The reason for this is that commercial and some public (e.g. Universities) sectors are closed (Energy, Environment and Policy, 2016). The attribute of the *public-commercial-industrial sector* DaysOfEnergyUse specifies the day in the week. This ranges from 0-6 in which 5 and 6 depict the weekend. For now we assume that 2% less electricity will be used in the weekends from the grid. See appendix 7 'Entity Actions In Detail' and the table A7.4 'entity action extension (Demand decrease in sectors in weekends)' for details.

#### SEASONAL WEATHER EFFECTS ON DEMAND (EXTENSION SCENARIO 1)

As shown in figure 5.3, the demand is influenced by seasonal weather variations. There is typically more usage in the winter than in the summer in Europe. For the Netherlands we can break down the seasonal weather variations in four seasons (fall, winter, spring, summer). We assume that the winter and fall has more than the average demand of electricity. And summer and spring less than the average demand. As a tick signifies a single day, the breakdown is as follows:

- Ticks 0-88, represents the fall
- Ticks 89-177, represents the winter
- Ticks 178-270, represents the spring
- Ticks 271-364, represents the summer

To implement this feature, an additional attribute SeasonCoefficient is set for agents using electricity. This coefficient represents a factor value, in the model we specify that the coefficient is in fall 1.05 (5% increase), winter 1.1 (10% increase), spring 0.95 (5% decrease), summer 0,9 (10% decrease) respectively. Please refer to appendix 7 'Entity Actions In Detail' table A7.5 for the specific entity action adaptations.

#### VARIATION IN DEMAND PER DAY PER CITIZEN (EXTENSION SCENARIO 1)

Due to the fact that there are differences per day in the use of electricity per *citizen*, there should be a way to reflect these fluctuations in the model. The simplified assumption is made that the variation embodies a maximum of 20% difference of demand for *citizens.* This means that the range minding the default value of 12 Kwh per day can be between 9.6 and 14.4 Kwh per day. Refer to appendix 7 'Entity Actions In Detail' table A7.6 for the implementation.

#### VARIATION IN SOLAR PANEL OUTPUT PER CITIZEN (EXTENSION SCENARIO 1)

As mentioned solar panels do not generate the same amount of electricity everyday, it's output depends on a variety of factors such as temperature, location, type of solar panel, cloudiness (SunPower, 2016). If temperatures are too high a solar panel can become inefficient, on the other hand in the winter there is less

sun hours during a day. Similarly to the variation in demand per day per citizen, we can specify a range of variations while omitting too many technical details. if the DailySolarOutput is on average 14 Kwh, we assume that the output can vary by 10%, which is between 12.6 Kwh a day and 15,4 Kwh a day. Appendix 'Entity Actions In Detail' table A7.7 elaborates on the details of the extension.

### VARIATION IN INITIAL DEMAND PER CITIZEN (EXTENSION SCENARIO 1)

Realistically the usage is different per household as every household does not live in the same type of built environment with the same set of electrical devices. Additionally the solar panels types may be different per *citizens*. With the application platform it is possible to specify an initial variation for the agents. In this case initial properties are added that define the initial property of the specific agent. In this way the model accounts indirectly for heterogeneity in agent's properties. These variations has impact on the variation in demand per day per *citizen* and the variation in SolarPanelOutput per *citizen*. For instance, a specific *active Citizen* might have by default a demand of 15 Kwh/day and in the Role of prosumer a SolarPanelOutput of 13.5 Kwh/day. Considering the variations that we specified as model extensions, what the exact global output of the system will be during a day is uncertain. If we have 75 activeCitizen's with different initial values and daily variations of demand, the impact on the Grid NetBalance cannot be readily predicted. Additionally adjusting the parameters such as the distribution of *activeCitizen*s in the town, would result in a completely different outcome. This showcases the benefit of ABMS as it is possible to develop different scenario's with different parameters and assess emergent behavior or system-wide outcomes.

### SUBSIDY FOR SOLAR PANELS IN HOUSEHOLDS (SCENARIO 2)

Subsidies for solar panel are seen in this model as incentives for *citizens* to invest in solar panels. Thus having subsidy in the model reflects indirectly a larger amount of *citizens* enacting the role of *prosumer*. As no monetary values are implemented in the model, we assume that *passiveCitizens* have a degree of willingness to invest in PV systems. WillingnessToInvest can be allocated as a Personal Value property of *passiveCitizens* with a scale from 0 to 9. We could specify a new agent (e.g. a *govermentRepresentative)* that defines the subsidy, however to keep the model simple we specify a 'policy' property for the *Grid* called FeedInTariff. The FeedInTariff property can be either false or true and represents a payment made to *citizens* for generating their own energy. The payment is not directly implemented in the model, but we assume that the FeedInTariff policy is driving the use of solar panel's for electricity generation. Thus as a precondition for *passiveCitizens* to enact the role of *prosumer,* the model implements that if the WillingnessToInvest is higher than 7 and FeedInTariff is true, *passiveCitizens* will generate their own electricity. If the WillingnessToInvest is too low we could specify that as long the FeedInTariff is in effect the WillingnessToInvest increases each tick as it becomes more and more favorable for those *consumers* to adapt as well in the future. See appendix 7 'Scenario 2 Entity Actions Explained' table A7.8 for more information on the scenario 2 implementation.

### SCENARIO 3 WIDESPREAD ADAPTATION OF SMART-METERS (SCENARIO 3)

Providing subsidy for households leads to an increase in fluctuations in the *Grid,* as more agent's generate electricity and feed surplus electricity into the *Grid.* Cases of oversupply may be prevented by implementing smart-meters in households. As mentioned in the introduction to the case, smart-meters can provide data on real-time consumption and power plant outputs. Considering the model one may argue that *citizens* with smart-meters are aware if there is oversupply or undersupply in the *Grid*. A simple economic based

assumption can be made that in cases of oversupply, usually driven by renewable energy, the price of electricity is low even up to the point of negative prices. In the case of undersupply the electricity price is high, because the demand is higher than the supply at that given point. The basis of this notion is that prices in the power market are typically determined by supply and demand (Clean Energy Wire, 2016). The dynamics of oversupply and undersupply leads to behavioral changes of agents, because they might use less electricity when undersupply is apparent due to high prices of electricity. In turn there will be more use of electricity when the price is low in the case of oversupply. To translate this in the model, it is possible to include an additional Physical Component, the SmartMeter, for *citizens.* The SmartMeter monitors the CurrentAvailableEnergy of the grid. This feature can be implemented within the 'use of electricity' entity actions of *citizens.* In turn if the *Grid's* CurrentAvailableEnergy property falls below zero, we speak of an undersupply. The assumption is made that *Citizens* will use 30% less energy than normal. If the CurrentAvailableEnergy property is more than zero (oversupply) we adjust the behavior of the *citizens* to use 30% more electricity. With this implementation *citizens* adapt to the available electricity on the *Grid*. See appendix 7 “Scenario 3 Entity Actions Explained” table A7.9 for a detailed description of the Entity Actions adjustments.

## 5.4 Simulation results

Scenario 1 showcased a basic model of supply and demand, followed by extensions. Scenario 2 incorporated adaptation of more and more consumers enacting the role of prosumers, while scenario 3 integrated the notion of smart-grids. This section presents the results of these scenarios.

### SCENARIO 1 DETERMINISTIC MODEL RESULTS

The following plots shown in figure 5.5 are generated and saved under the Evaluative Structure package generated after any simulation. As explained in the software functionality, by default a CSV is constructed that tracks all agent properties during simulation. This CSV can be used for further detailed analysis.

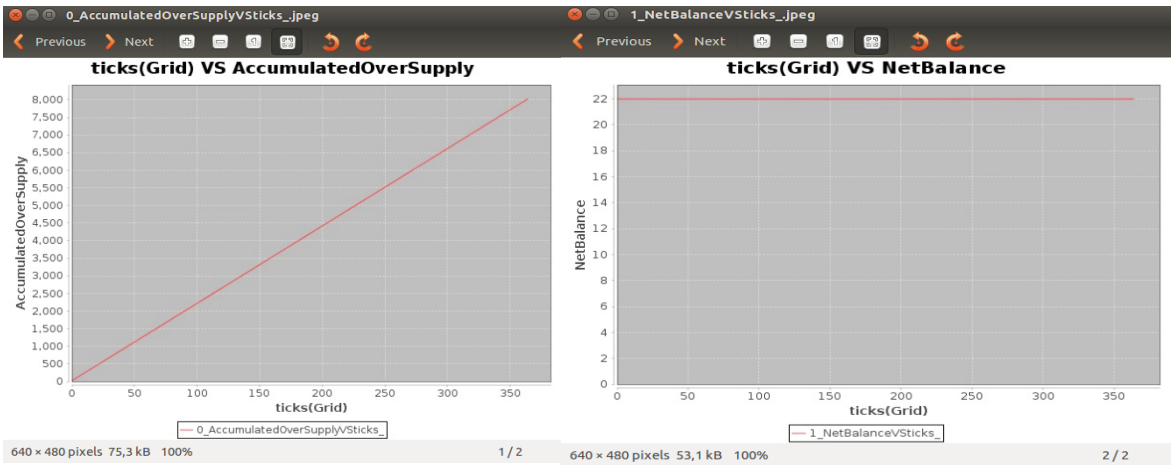



Figure 5.5: Generated plots. Because this is a deterministic model, the plots only show linearity. Analyzing the results of the left plot, we see that the expected output described in 'testing the model' is met by these results.

ASSESSING THE MODEL RESULTS WITH EXTENSIONS

Minding the initialization settings specified in the Dynamic GUI, the results of the simulation of the scenario 1 model with extensions are as follows:

- Figure 5.6. DailySolarOutput for agent *prosumer* ID 929338653 and Daily Citizen electricity demand for agent *activeCitizen* ID 1995265320 per day (tick).

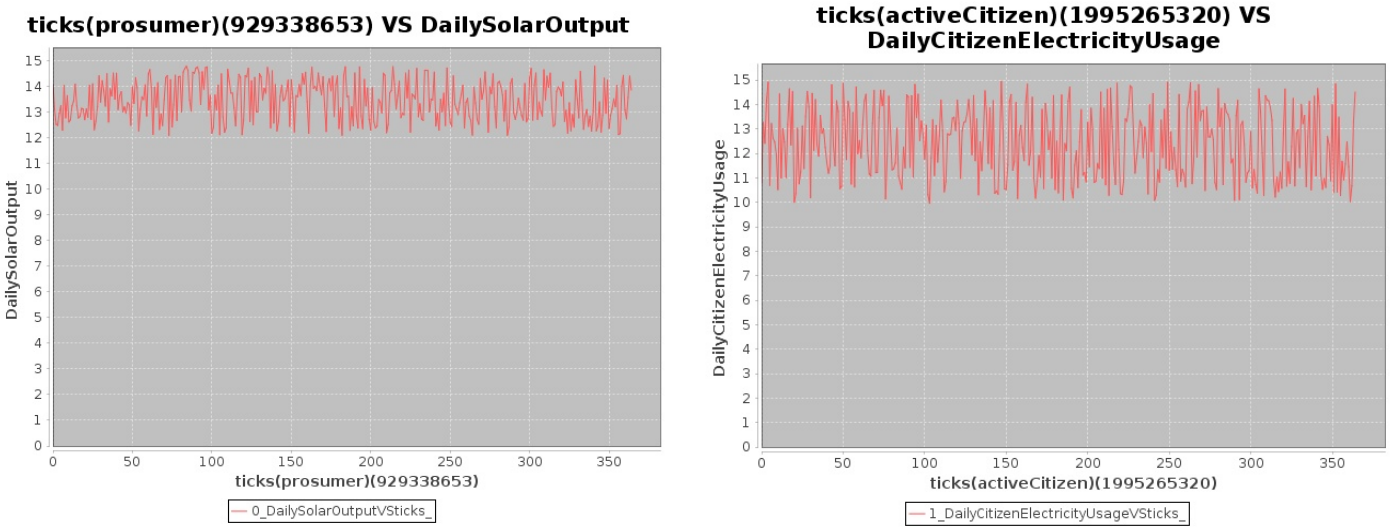



Figure 5.6: Variations in supply and demand of agent's in scenario 1

- Figure 5.7: *Grid's* NetBalance and FollowDemand parameter by *gasCoalPowerPlant.*

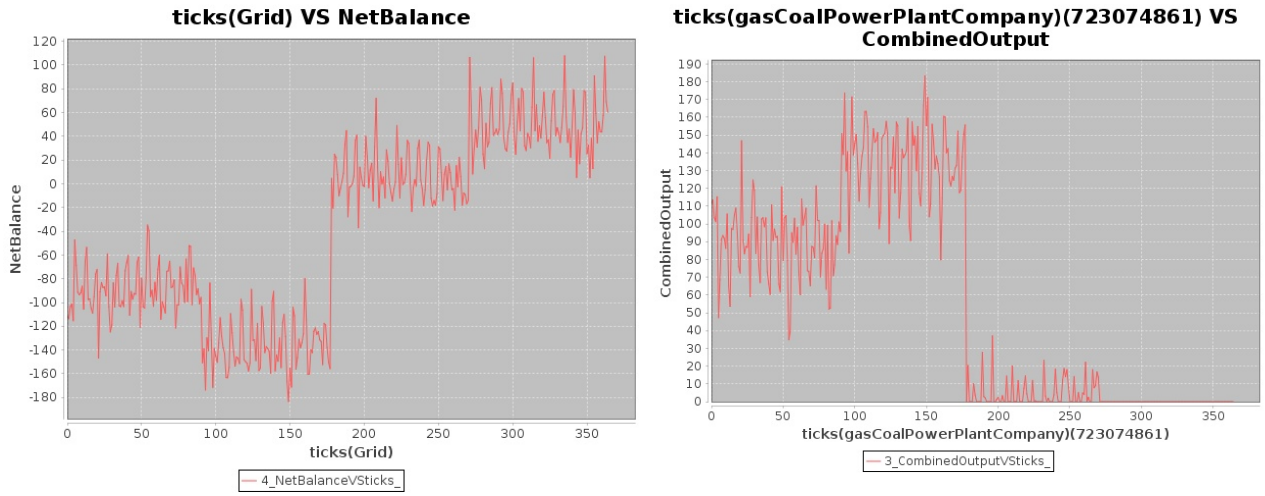



Figure 5.7: (Left) Grid NetBalance. (Right) gasCoalPowerPlant output. In the first two seasons an undersupply is apparent while in the last two seasons oversupply. The output of the gasCoalPowerPlant reflects the load following of demand. The smaller peaks in the NetBalance figure, those below zero, show the decrease of demand in weekends. While above zero the lowest points represents the decrease in weekends.

### COMPARISON RESULTS SCENARIO 1, 2 AND 3

In essence, scenario 2 and 3 are extensions to the basic model as well. In turn we can evaluate the differences in NetBalance output per scenario.
See figure 5.8 for a comparison between scenario 1 and 2. Scenario 2 shows an increase of oversupply.

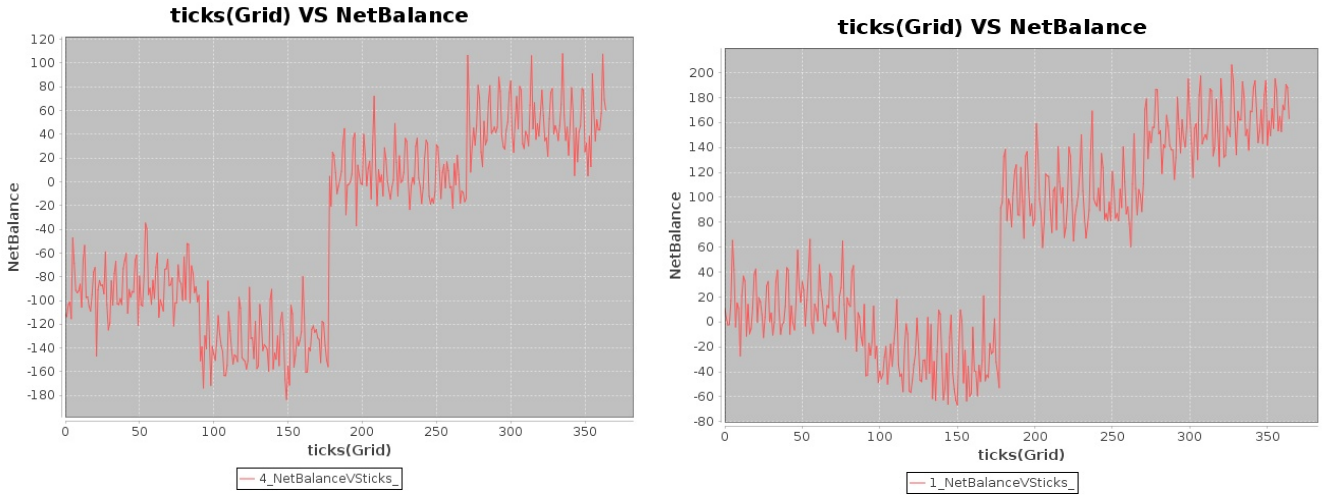



Figure 5.8: (left) results from scenario 1. (right) results from scenario 2.

Figure 5.9 shows that the implementation of smart-metering leads to less extreme fluctuations and supply meets the demand more closely.

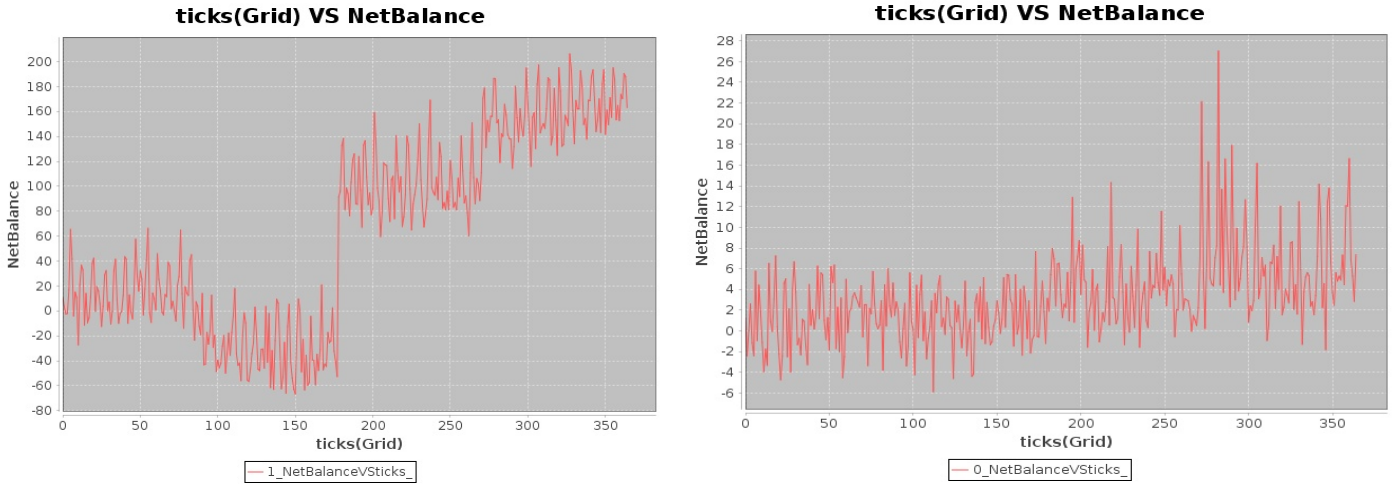



Figure 5.9: (left) Results from scenario 2. (right) results from scenario 3.
In scenario 3 there are still cases of under/over supply, however the magnitude is much smaller.

In the Reflection (chapter 7) section proposals are given on how this case study model can be improved or extended. The following chapter showcases how a simulation can be implemented in the AMIE software.

# 6 Tutorial for AMIE test-users

In this tutorial, scenario 1 of the model was used as an example. Please start the AMIE software by opening the executable. We will go through the implementation of the model in the platform in a step by step manner. If it is preferable to show any log messages, the tool can be opened by a command prompt with the parameters -consolelog. Once the application is opened the following screen is visible in figure 6.1:

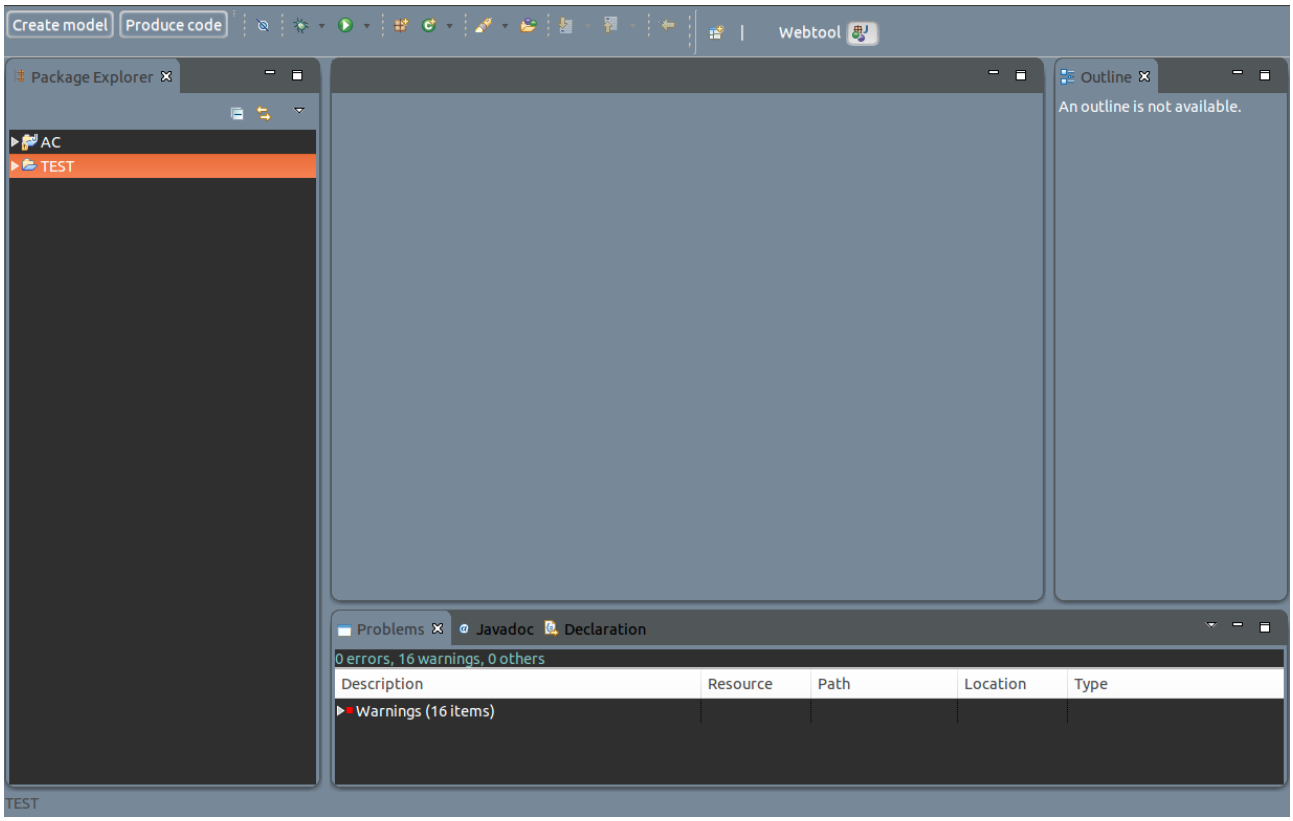



Figure 6.1: the default screen of the MAIA tool. The top bar shows a set of buttons. Only the create model, produce code and the play button (green) are relevant in this section. The left pane shows the package explorer (Project Explorer) in which ABMS projects can be made. The middle pane is for display of editors or Java code. The right pane is an information panel for objects. The bottom pane is showcases general errors or warnings. Upon the start of a simulation (when the play button is pressed) another screen is visible in the bottom pane showing run time information which is defined as the "console".

### STEP 1: CREATE THE MODEL SKELETON

The top left button in figure 6.1 is the create model button. Once clicked a small GUI pops up in which the project name and model name can be specified. See figure 6.2.

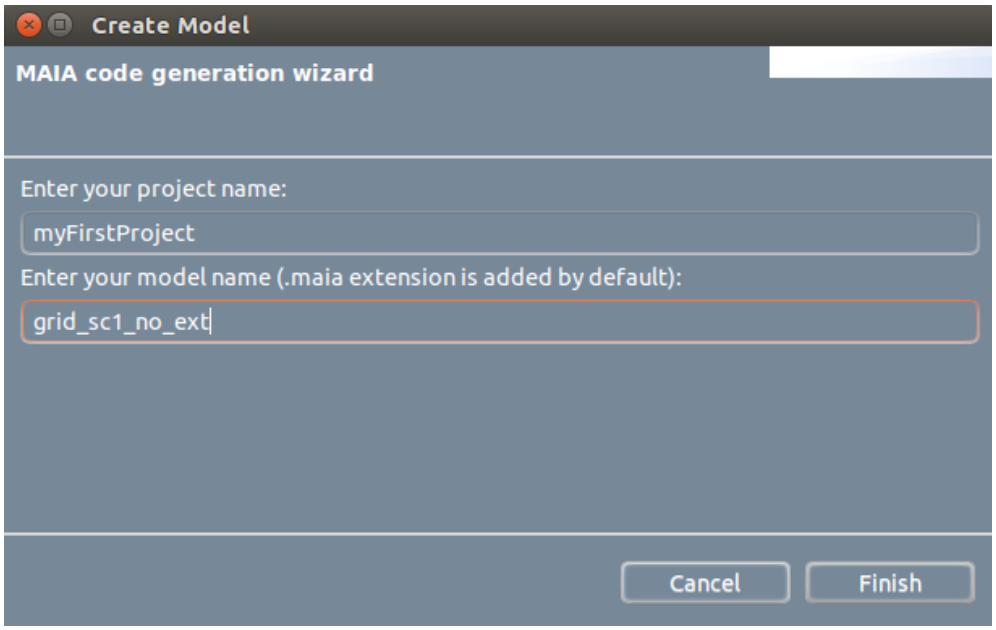



Figure 6.2: creating the basic model skeleton.

### STEP 2: OPEN THE MODEL EDITOR (EMFFORMS)

Once finish is clicked, the Project Explorer will show the project and inside that 'folder' resides the basic model skeleton. By Double-clicking on it, the editor can be opened which is needed to fill in model information. Alternatively right-clicking the file and selecting open with.. > 'MAIA EMF generic editor (recommended)' will open the editor as well.

### STEP 3: USE THE EDITOR TO FILL IN MODEL INFORMATION

Once the editor is open, a symbol saying MAIA is visible. See figure 6.3.

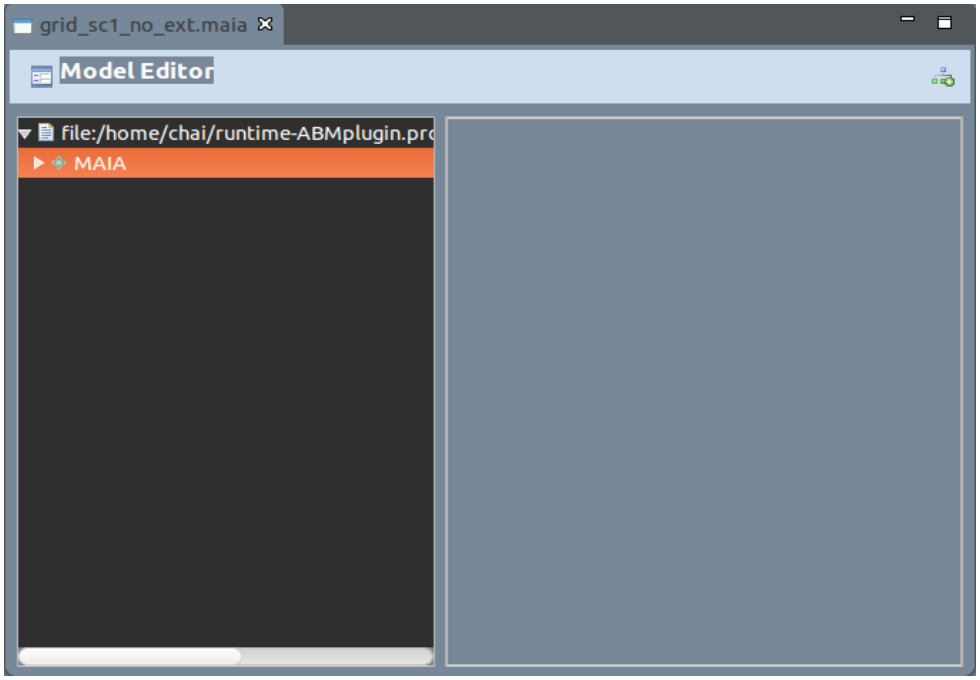



Figure 6.3: The MAIA EMF generic editor. Left pane showcases any MAIA objects users want to create or adjust according to MAIA structures. Right pane is for actual user input.

The MAIA structures become visible by right-clicking the MAIA object. Clicking on a structure will create the structure for the model. In this tutorial only a Collective Structure example will be given. Figure 6.4 on the next page shows the agent *activeCitizen* in the Collective Structure.

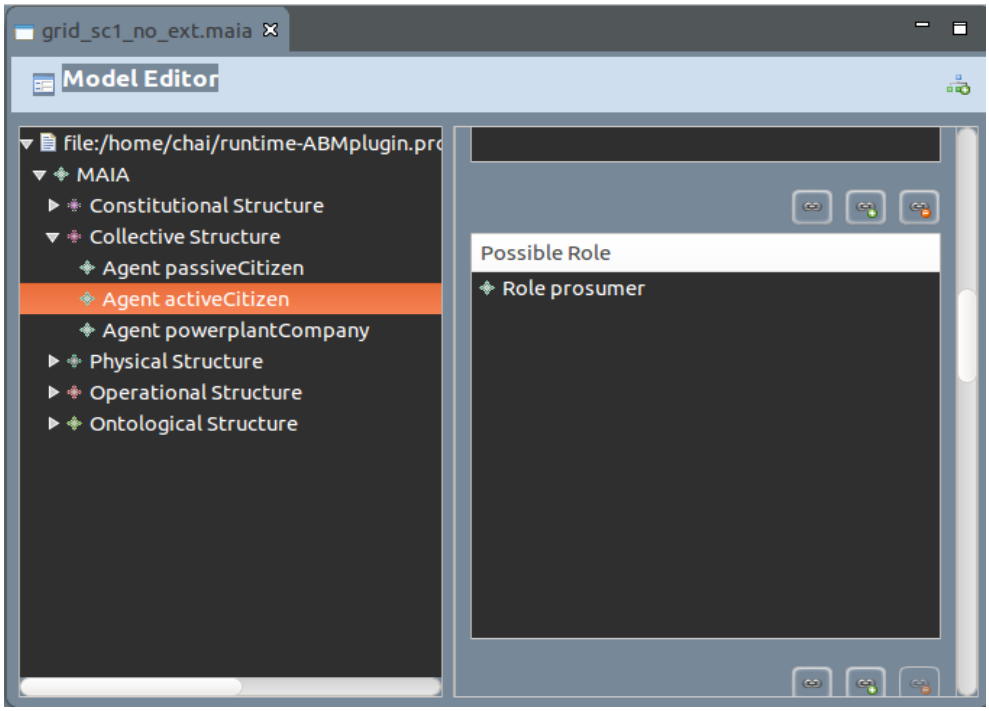



Figure 6.4: The construction of the active citizen. The right pane showcases the possible role this agent might enact, but as this is a scrollable editor users can fill in all agent related data. Values are generally filled in as default values and can be adjusted later on in the dynamic GUI. Additionally three buttons are visible on the right pane. The left one is adding an existing attribute. The middle is creating an attribute from scratch. The right button is for deleting an attribute. If you want to change an attribute double-clicking the attribute allows for adjustments.

With this editor users can create their MAIA-based ABM by simply filling in the boxes with the information of their model. It is advised to end filling in the model with the Operational Structure as that combines all inputs defined in the other structures. The Ontological Structure is used as a container for agent attributes and entity actions which can be allocated to specific agents in the Collective Structure, Constitutional Structure or Physical Structure.

### STEP 4: PRODUCE THE MODEL SIMULATION CODE

Once the model is completely filled in, the user can view the model in the original XML format. Right-click in the Project Explorer on the model > open with.. > text editor. This will show a new text editor as shown in figure 6.5.

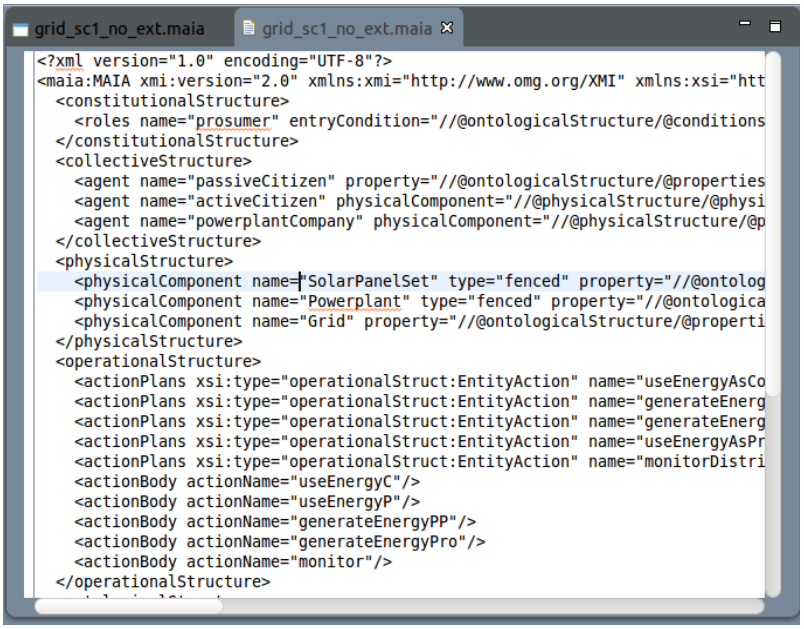



Figure 6.5: The "backbone" of the model. This information is used by the platform to produce code.

Once the model is completed, the Produce Code button can be selected from the top bar of the tool (see figure 6.1). This allows for the creation of a simulation experiment. Similarly to the Create Model button the Produce Code button will open up a small GUI as shown in figure 6.6.

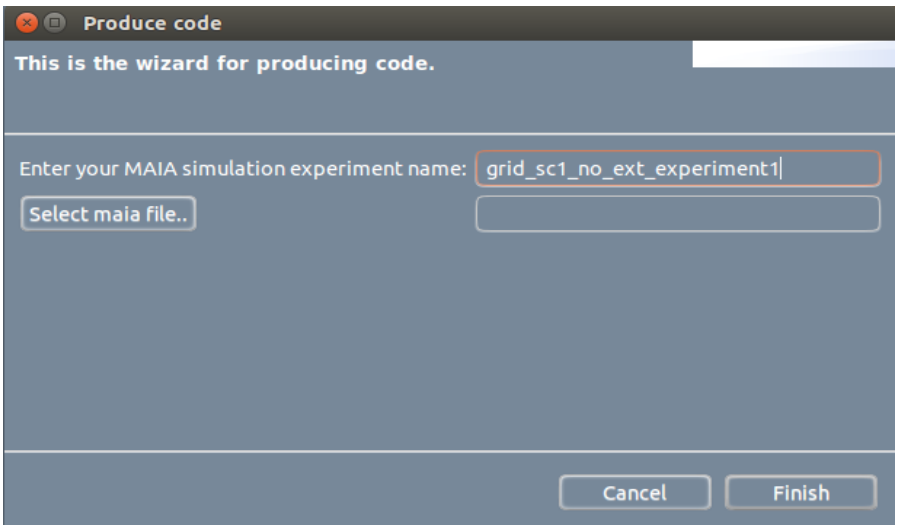



Figure 6.6: the GUI for producing code.

### STEP 5: FILL IN RUNTIME EXPERIMENT DATA

Once the finish button has been pressed (figure 6.6), code is generated on the fly. In addition the Dynamic GUI is shown, that can be used to supply runtime information. See figure 6.7 below.

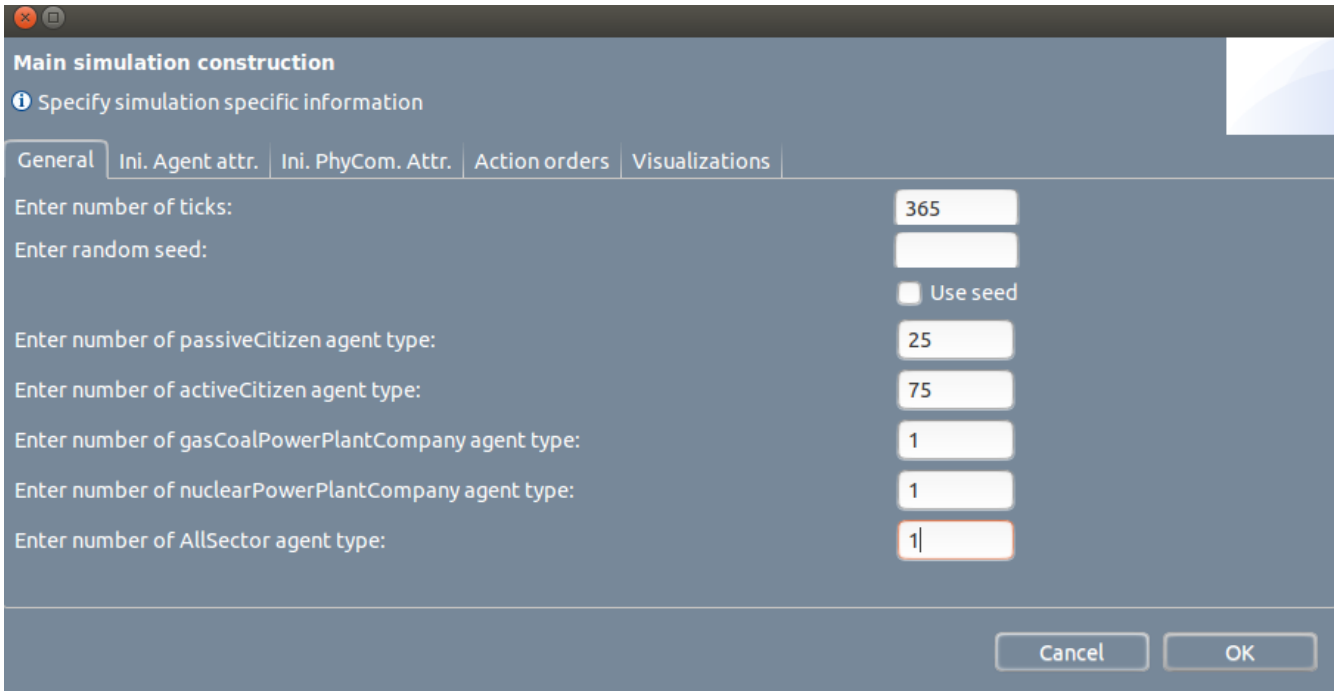



Figure 6.7: The Dynamic GUI. Users can fill in their number of agents, ticks (input required) in the General tab. The random seed settings can be used to replicate any experiment if a seed is set. The Initialization tabs can be used (optional input) to specify initial variation in agent attributes. The Action Orders tab (input required) is used to specify the order of entity actions. The visualizations tab (optional input) can be used to automatically generate plots of runtime results, such as ticks VS agent attributes.

In figure 6.8 an overview of the entity actions order for this specific case study.

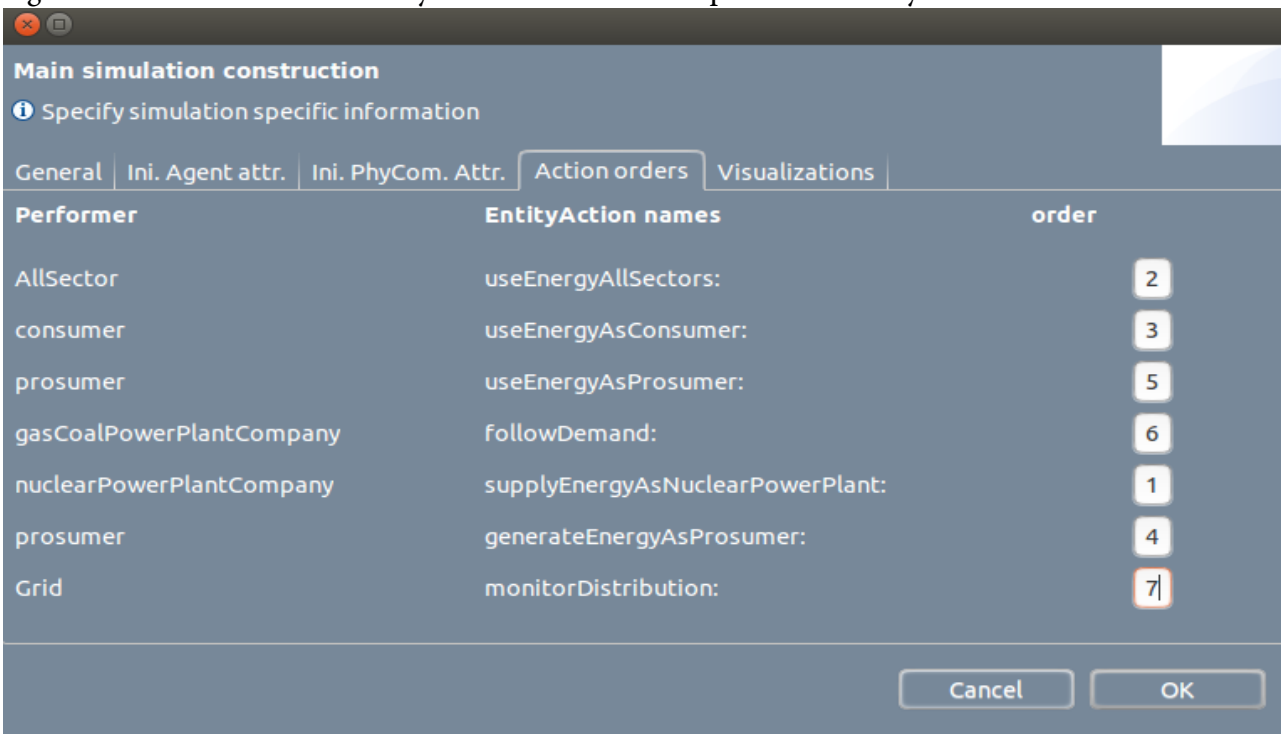



Figure 6.8: Action Order tab for specifying entity actions orders.

### STEP 6: EXECUTE SIMULATION

The Project Explorer will show a new project once all the necessary code has been generated. See figure 6.9.

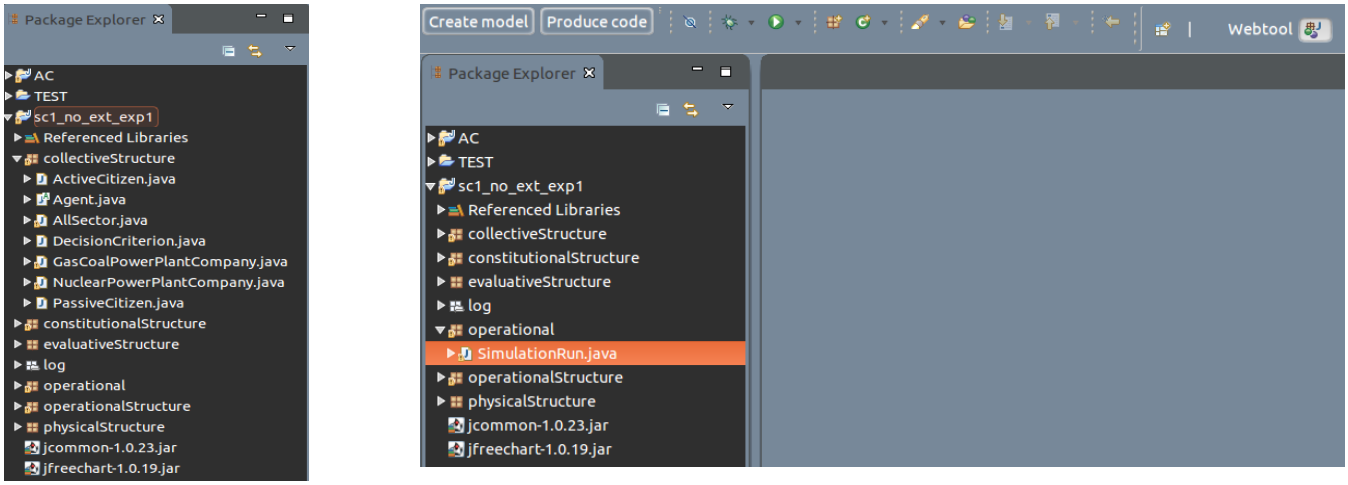



Figure 6.9: The left figure shows the Project Explorer and all generated code inside packages. The log package contains a log text file containing information on the code generation process. The right figure highlights the simulationRun.java code. This code signifies the main simulation described in section 4.2.1 the procedural semantics. If this is selected users can start their simulation by pressing the play (green) button in the top panel.

### STEP 7: EVALUATE RESULTS

Once the simulation is started, the 'console' will be shown displaying run time information. See figure 6.10.

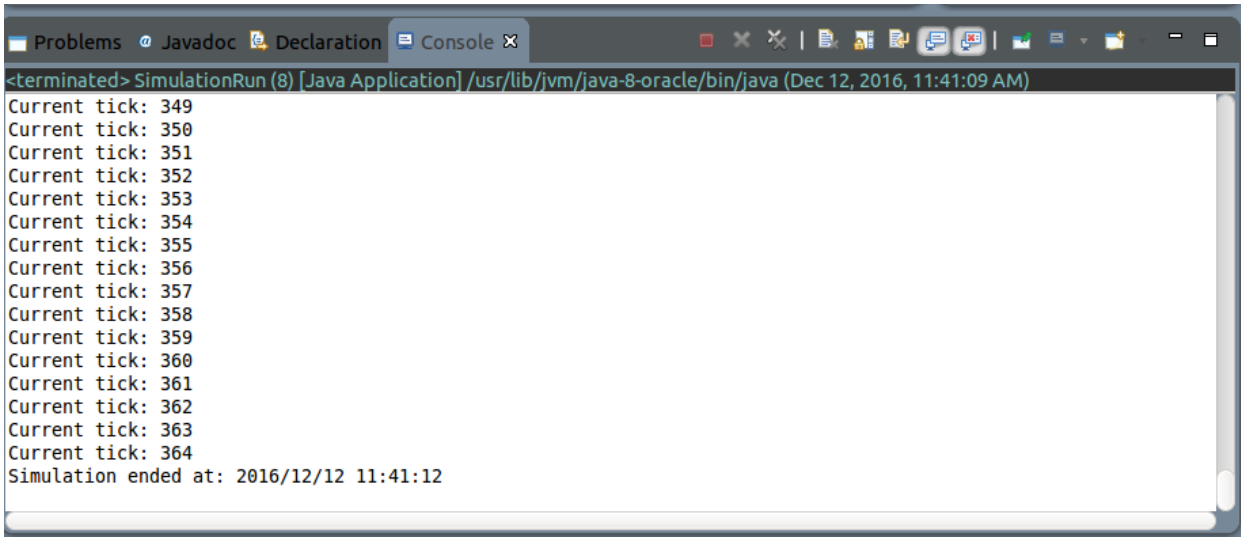



Figure 6.10: Runtime information.

As mentioned earlier, in the Visualization tab of the Dynamic GUI, users can specify plots to be generated during a simulation. These plots are saved under the Evaluative Structure package in the Project Explorer (users need to refresh in the top-left menu option to show them after simulation).

# 7 Discussion and Conclusion

## 7.1 Overview

In this research project we addressed the development of a decision-support tool for the analyses of STS with respect to ABMS. The increasingly larger and complex systems of today's world makes modeling and simulation a significant challenge. Regardless of the challenges, the potential of ABMS has become more and more apparent due to ICT developments, ranging from improvements in computational power to development of IoT devices that transmit real-time data.
The research presented in this thesis revolves around the development of a proof of concept that embraces the concepts of MDSD and automatic code generation. The research that has driven this development is the MAIA theoretical framework. MAIA addressed the usefulness of conceptual models and meta-models for ABMS in relation to STS (Amineh Ghorbani, 2013). The broader context around the research is aimed to provide a platform that brings ABMS within reach for stakeholders, thereby focusing on the usability aspect. Hence a practical approach was taken that elaborates on the technical requirements and implementation details needed to proceed from a MAIA conceptual model towards an actual Agent-based simulation. To decompose the research objective and tackle challenges during the research the following research questions were defined.

1. Main research question: How can a platform for rapid development of Agent-based Models and Simulations that utilizes Model Driven Software Development be realized, using a high-level language?
2. Subquestion: What software modules are needed to develop the platform?
3. Subquestion: How can these modules be interlinked, to bring about a workflow-based application?
4. Subquestion:What steps are needed in order to translate socio-technical conceptual models into simulations?

The first subquestion is concerned with the software modules related to MDSD, that is the ingredients needed to develop the Agent-based Model-driven Integrated Environment. The second subquestion is aimed at connecting these ingredients/modules together to bring about a platform for model to simulation transformation. The last question concerns how to use the platform, in turn how MAIA relates to ABMS and how application parameters relate to the MAIA framework. We will answer the subquestions in the following sections. In turn the main question will be answered taking the subquestions into account.

## 7.2 Research Outcomes

### 7.2.1 Research question 2 [components]

The MAIA framework served as a useful framework to integrate in the decision support tool, due to the structured breakdown of the dimensions of STS and the comprehensive documentation on the MAIA-based meta-model. In addition to the methodology on conceptual modeling of STS provided by MAIA, the MAIA meta-model addresses the operational aspects required to translate a conceptual model into an ABMS.
A MDSD approach using MDA represents benefits in software engineering practices, which encompasses the standardization of software development and the reuse of components. It provides a platform for high-level software design that corresponds directly with actual software implementations. The programming language

Java coupled with Eclipse provides sufficient libraries that incorporate MDA, such as the Eclipse Modeling Framework. Additionally software development was relatively effective when using the Eclipse Rich Client Platform and supporting Java graphics libraries.

#### 7.2.2 Research question 3 [connection of components - workflow]

With MDSD there is a need for meta-models, protocols and transformation platforms. AMIE incorporates the MAIA meta-model as an EMF core model. EMF protocols are used to expose the MAIA-EMF core model to an editor allowing for the creation of MAIA instances. A MAIA instance is in essence a XML file, which is used by the Java Emitter Templates engine connected with MAIA-EMF core libraries to bring about Agent-based code generation. The Eclipse Rich Client Platform provided the architecture to realize the needs for MAIA, EMF and JET integration. Custom Java implementations in RCP was needed to allow for functionalities such as invoking model creation and code generation, but also the development of the Dynamic GUI needed to retrieve experiment related user input.

#### 7.2.3 Research question 4 [conceptual models towards simulations]

A conceptual STS model is in essence a high-level description of the system under study. As mentioned earlier a high-level description is not enough to bring about ABMS. MAIA provides a structured framework for the development of conceptual models, while at the same time providing the aspects needed for translation to low-level languages. AMIE leverages the latter point, in order to realize a practical approach from model towards simulation. The case study followed by the AMIE tutorial showed the steps needed from conceptual model, to implementation details, to model integration and simulation in AMIE.

#### 7.2.4 research question 1 [AMIE]

The development of AMIE showed that it is possible to incorporate ABMS related meta-models in a Model-driven Architecture. This study presents a practical methodology for the translation of Computation Independent Models to Platform Specific Models. In regard to this thesis study one may argue that MDSD is valuable for ABMS. However MDSD requires standardized formats such as the MDA. In addition it requires programming languages that support the notion of MDSD. Unfortunate not all programming languages support MDSD. Java is in that sense a forerunner, but this is also accountable to the various Eclipse Frameworks that are based on that language.

### 7.3 Contribution of the thesis study

#### 7.3.1 Participatory ABMS

The overarching context of this research revolves around the notion of bringing ABMS within the reach of stakeholders. The developed decision-support tool provides an integrated environment for creating models and simulations for STS. With the integration of conceptual modeling and translation towards simulation, this platform serves as an advancement towards the realization of participatory ABMS.
The following features of the application make the tool useful for participatory ABMS:

- The tool is a cross platform software application, hence users are not only limited to the Windows Operation System, but can use Linux or Mac as well.

- The MAIA framework embedded in the platform provides a common language between stakeholders. MAIA is described in a high-level language, thus stakeholders with various backgrounds can more easily construct a conceptual model of a specific STS case according to the MAIA structures. This conceptual model can be used for a large extent directly in the platform to construct the instance model for simulation purposes.
- Instance models can be easily shared between users. If a model is finalized a .MAIA XML file is created. This file can be shared throughout different users and multiple versions can be created if desired.
- If model files are shared it is also important that ABMS experiments can be replicated, even if random functions are sometimes used. Thus, a random seed can be set in the Dynamic Gui in order to run the exact same simulation multiple times.
- The integrated environment to construct models and use those models to create simulations, allows for a structured ABMS process. As previously mentioned, ABM modelers tend to mix model with simulation development with no clear boundary. The environment the platform provides is inherently structured and requires users to create Agent-based Models and Simulations in a step-by-step manner.

### 7.3.2 Scientific contribution

#### SOFTWARE ENGINEERING

From a software engineering perspective new knowledge can be gained as this project tries to integrate model-driven development in ABMS. Additionally Agent-based modeling and simulation has been for many years only accessible to researchers experienced in programming. The active involvement of stakeholders might improve Agent-based models due to the added expert information. Including a more detailed picture for instance of technology compositions, current and/or needed infrastructure, consumer expectations, norms and values. The latter includes the explicit incorporation of the heterogeneous set of perspectives of stakeholders. Hence a multi-disciplinary approach that combines concepts of IE, social science and ICT is achieved. AMIE sets itself apart from the conceptual modeling efforts provided by OperA, easyABMS and INGENIAS through the practical approach of translation of conceptual models to simulations. It is therefore comparable with the GAMA platform as is it enables users with limited experience to use a generic tool to bring about ABMS. Additionally AMIE is focused on specifically the construction of agent-based models of socio-technical systems. Concentrating on STS allows for the use of a common language (ontology) such as MAIA.

#### INDUSTRIAL ECOLOGY

IE is a field that does not yet utilize ICT to its full extent (Davis et al., 2010). However within the field of IE the issues and challenges around the understanding and shaping of socio-technical systems has been recognized. The development of a platform for ABMS with respect to STS might aid to the research that has been done in this domain. This research showed how AMIE and the case study can enhance three key concepts of IE namely:

- Systematizing patterns of energy use. Minimize energy loss by simulation of widespread adaptation of smart-meters as showed in the case study.

- Aligning policy to conform with long term industrial system evolution. By creating a decision support system, policies and industrial systems can be better aligned through new insights retrieved from ABMS.
- Creating new action-coordinating structures, communicative linkages, and information. The decision support tool has the potential to bring stakeholders together with the use of the MAIA framework. In addition novel information can be retrieved from the construction of models and simulations.

### 7.3.3 Societal contribution

With the advent of ICT in recent years, the playing field in terms of technology and innovation has been changing rapidly. In addition the way people use technology changes. Traditionally engineers develop technical artifacts from their perspective. The way these artifacts are going to be utilized is for a large part assumed. Concepts such as the rebound effect is not readily assessed beforehand. Hence this interplay between technological and social aspects might become more apparent with participatory ABMS. The ultimate goal is to bring advanced ICT (e.g. simulations) capabilities within the reach of key decision makers. ABMS is currently a scientific tool, but has huge potential as a decision support tool for decision makers. Generally speaking, if the barrier to utilize advanced ICT can be lowered, more stakeholders can be involved, thus they can contribute to improving our understanding of complex systems. Meaning that information is potentially more easily shared and common languages arise in which different stakeholders perspectives can be aligned. In addition simulations of complex systems with the use of machine learning, real-time data can assists decision makers to accommodate policies that enable novel coordination efforts in order to steer societies towards sustainable, resilient communities.

## 7.4 Reflection and future research

This section reflects on the usability of AMIE and addresses recommendations for future research.

### 7.4.1 Technical limitations

In order to address what the drivers and barriers are in terms of the developed platform, aspects that encourage and enable participatory ABMS are viewed as drivers and technical limitations of the tool as barriers. The following technical limitations were defined:

- Not all MAIA concepts are completely implemented in the software application. Appendix 5 describes the platform implementations of MAIA concepts through a set of tables. Certain aspects such as Multi Criteria Decision Making are not readily implemented. Additionally in the Operational Environment only a definition of sequence plans (entity actions) are with the current version possible. For instance a choice plan (Amineh Ghorbani, 2013) cannot be implemented as there is no function to randomize plans.
- A subset of the Java programming language is needed for Entity Actions. The platform is developed for researchers with limited programming experience, however for more complex features of a model such as the entity actions the Java programming language is required. Appendix 7 describes the Java programming language details for the tool. Although the barrier is lower than for full fledged ABMS software such as NetLogo, a comprehensive explanation was needed for defining entity actions as the default model editor does not provide functions to define them in a more user-friendly way.

- Along the line of the previous point, another limitation is that only single statements are currently allowed for entity actions. For instance if there are two preconditions for an entity action of an agent they should be defined in a single statement with Java based operators. The editor allows for multiple statements, however this typically causes errors during code generation.
- Automatic visualizations are limited to generated plots of attributes specific to a single agent. Meaning that it is not possible to select an attribute of one agent and cross-match that with an attribute of another agent. However, ticks can always be selected in combination with an attribute of an agent. In addition agent tracking is limited, because the exact agent of a specific type is selected randomly by default. By setting a seed it is possible to plot all attributes of a single agent if desired, but in turn which agent selected is randomly determined. An option is to make multiple selections with respect to attributes and agents, in turn the same attributes of multiple agents can be analyzed as each selection will select a different agent.

The technical limitations described above may serve as barriers towards using the software application. For instance a high learning curve might be apparent for users when defining entity actions if the programming experience is very limited. The evaluation of a model is now limited to the CSV and the automatic visualization option. Experienced programmers may be able to work with the CSV to filter out desired information, however this barrier can be lowered if more options for evaluations are implemented- such as the export of CSV for detailed simulation selections (e.g. only attributes for one agent type). The MAIA concepts that are not implemented may be implemented in later versions of the tool, but currently this a barrier for more complex and large MAIA-based ABMS. Below are a few proposals on how these barriers can be mitigated:

- Addition of functionalities such as more MAIA concept integrations using JET. As mentioned not all MAIA concepts are implemented, although the mechanisms are in place to realize more MAIA integrations in the future.
- A more user-friendly entity action implementation that lowers the barrier for non-experienced programmers. This can be achieved through changes in the MAIA EMF core model and JET. In which default Java notations currently required will be automatically generated.
- Visualizations implementations can be improved by more transparent agent (ID) tracking mechanisms or additional implementations for cross-matching of agent attributes with other agent attributes.

### 7.4.2 Use of run-time visualizations

The library used for generating visualizations is JfreeChart. JfreeChart provides an API for developing a wide range of plots, however JfreeChart is inherently designed for static graphs for reporting purposes (Java Zone, 2016). Real-time visualizations are possible through manipulation of the API, however the performance is relatively poor. Alternative libraries such as VisualVM (VisualVM, 2016) are designed for optimized real-time plotting and may as well be integrated in a future version of the tool. This would allow for more visual analyses during the simulation. Eclipse also provides extensions for visualizations such as the Graphical Editing Framework (GEF, 2016), Zest (Zest, 2016), Birt (Eclipse Birt, 2016).

### 7.4.3 Comprehensive software evaluation and validation

This thesis study emerged from the need to provide methodologies and tools for participatory ABMS. Many aspects should be considered in doing so and software evaluation and validation is important, unfortunately a comprehensive software evaluation and validation was outside of the scope of the thesis project. For future research it might be interesting to have case studies on large projects with various stakeholders in order to thoroughly test the tool. Additionally this can make the tool more robust if feedback is taken into account for future versions. Moreover, usability testing is a common way to test the ease-of-use of a program usually by doing tests with real users. The developer in turn evaluates what problems and confusions they experience or may encounter. This research focuses mainly on the development of a proof of concept and not a final application product, hence there are aside from the technical limitations other barriers apparent in the application. However it is evident that the tool has been developed for a specific purpose and audience, thus criteria can be developed based on that. The target audience encompasses decision-makers, wide range of scientists (modelers in IE or STS, domain experts), businesses and IE students. This reflects the evaluative nature and scope as the focus is mainly on users with limited programming experience and partially on developers that wish to continue the tool development. A user is typically a person who downloads, installs, configures the application. A developer is a person who writes code that changes the software. Based on the target audience some evaluative statements and questions can be defined that may be used for evaluation and validation (Software Sustainability Institute, 2016):

- VISIBILITY OF SYSTEM STATUS. Does it give users appropriate feedback within reasonable time?
- ERROR PREVENTION. Does it prevent errors in the first place or help users avoid making them?
- MATCH BETWEEN SYSTEM AND THE REAL WORLD. Does it speak the user's language and make information appear in a natural and logical order? Are implementation-specific details hidden from the user?
- USER CONTROL AND FREEDOM. Does it provide clearly marked exits, undo and redo?
- RECOGNITION RATHER THAN RECALL. Does it make objects, actions and options visible and reduce the amount of information a user has to remember?
- FLEXIBILITY AND EFFICIENCY OF USE. Does it offer short-cuts and support macros for frequently done action sequences?
- Are the binary releases packaged for immediate use in a suitable archive format, e.g. .tar.gz, .zip or .jar for Linux, .zip, .jar or .exe for Windows?
- Is the user doc online? Are there any supporting tutorials? Do these list the versions they apply to?.
- Is there design documentation available? How accurate and understandable is it?

These questions cannot be easily answered, and a thorough test phase would be required. Potentially a forum-like setup would be needed to answer user-related questions.

### 7.4.4 More advanced GUI for model editing

Currently the default EMFForms from the Eclipse Modeling Framework is used for users to edit models. EMFForms also provides an API for developing custom editors and this creates the opportunity to make a more user-friendly editor in line with MAIA. Next to a custom implementation of EMFForms it is also possible to create a complete different infrastructure such as a Web-based platform for model creation

(Amineh Ghorbani, 2013). In order to build such an infrastructure knowledge of XML creation and parsing would be needed to make it functional with the simulation and code generation part of the platform.

### 7.4.5 Code generation for other platforms

The code generated in the MAIA Model & Simulation platform is platform specific. In theory the infrastructure defined in the Materials and Methodology chapter can be used for code generation for other platforms. The specific formats that would be required are dependent on the other ABMS platforms, however the benefit for doing so would be that users can use more extensive platforms that provide more features such as real-time visualizations or parameter sweeps.

### 7.4.6 Model extension proposals

The Agent-based model scenarios developed in the chapter 5 do not necessarily reflect the most realistic simulations of electrical grids, but showcases relevant aspects that ought to be considered when developing models for electrical grids. Scenario 1 emphasized more on the platform implementations needed to construct a basic model, while scenarios 2 and 3 went more into detail how the model behaves under different circumstances considering incentives and innovation. Yet chapter 5 has served mainly as an introduction to go from model to simulation. In the end an ABM should include many more components and interactions. Thus the following exploratory concepts can be defined that model, test and/or implement:

- VARIABILITY IN DISTRIBUTION OF ACTIVE CITIZENS (PROSUMERS) AND PASSIVE CITIZENS (CONSUMERS). In the model implementation a fixed amount of *active citizens* and *passive citizens* is used. It might be interesting to do more experiments while changing these distributions.
- USE OF REAL-TIME DATA. With the advent of IoT devices such as Arduino based monitoring systems, more data is available than ever before. It is theoretically possible to read in real-time data on the fly during the simulation. Although this would require a person with sufficient Java programming experience to implement. The platform can in this case be used to develop the basic skeleton of the model in terms of MAIA structures, however the details of the entity action should be manually inserted. Meaning that an algorithm would need to be developed that reads in relevant data and in turn defines certain parameters of the model (e.g. CitizenElectricityUsage).
- STAKEHOLDER ENGAGEMENT IN THE MODELING AND SIMULATION PROCESS. Because the MAIA theoretical framework is aimed at stakeholder involvement, it would make sense to involve domain experts in the process of developing the model. This leads potentially to a model more close to reality and a more comprehensive validation process can take place.
- DIFFERENCES IN DAILY POWER PLANT SUPPLY AND IMPLEMENTATION OF CLASS 3 PLANTS. In the current model power plants are relatively static only supplying a fixed amount of electricity or following demand. It would be interesting to see more dynamics included such as renewable power plants that create larger effects on the system due to their intermittent nature.
- ENERGY STORAGE FACILITIES IN CASE OF OVERSUPPLY BY SOLAR PANELS & VIRTUAL POWER PLANTS. Electrical vehicle's share in the market is increasing and new innovative technology emerges for energy storage. Thus the model could be extended to include energy storage in cases of oversupply. Additionally the concept of Virtual Power Plants is strongly linked with energy storage,

demand management and renewable energy sources. It would be interesting to see how this concept can be integrated in the model.

- DECREASE IN SOLAR PANEL EFFICIENCY. Solar panel efficiency typically decreases in time and it would make sense to include the decrease of efficiency. However as this simulation is aimed to assess a single year the efficiency decrease is negligible. Extending the model to run for multiple years would be of added value in this case, yet this would require more dynamics such as technological developments.
- BREAKDOWN OF SECTORS AS SEPARATE AGENTS. Currently, there is no breakdown in the sectors and it would be interesting to make a distinction of the different types of sectors.
- INCLUSION OF ELECTRICITY PRICE AND TICKS DEPICTING HOURS. Another interesting extension would be to include electricity price and to see if the consumers/prosumers would behave differently to the fluctuations in price. It could potentially be useful to change the ticks from daily to hourly. Hence making it a more complex model that would include the variations in the electricity price per hour, electricity usage and electricity generation.

*Sidney Niccolson*
*Thesis Research Project*
*February 6th, 2017*
*s1650548 (Leiden) - 4452909 (Delft)*

*"It is now highly feasible to take care of everybody on Earth at a higher standard of living than any have ever known. It no longer has to be you or me. Selfishness is unnecessary. War is obsolete. It is a matter of converting our high technology from WEAPONRY to LIVINGRY."*
*-Buckminster Fuller*

*Sidney Niccolson*
*Thesis Research Project*
*February 6th, 2017*
*s1650548 (Leiden) - 4452909 (Delft)*

# Glossary: An introduction to IE

IE embodies the improvement of metabolic pathways of processes and methods in our society in order to meet current and future needs. The name "IE" comes from the analogy between natural systems (biosphere) and industrial systems (technosphere) and that this analogy can be used as an aid in understanding how to design sustainable industrial systems. The term biosphere was first broad up by geologist Eduard Suess in 1875, he defined the biosphere as the place on Earth's surface where life dwells. V. I. Vernadsky (1863–1945) stated that ecology is the science of the biosphere. Moreover Vernadsky places life at the center of the history and functioning of the planet's chemistry. Vernadsky was among the first to argue that organisms actively shaped the atmosphere, oceans and the earth's landscape (Haplopoda, 2001). The biosphere is not merely the Earth's surface where life dwells, Vernadsky tackles the idea of life also influencing its environment by interaction either between organism or organism influencing inorganic compounds. The technosphere can be defined as the world created by man, such as constructions, machines or waste dumps (Richards, 2009). According to Robert Ayers (1989) both the biosphere and the technosphere are systems for the transformation of materials. Ayres gives the example of organisms consuming resources (food), digesting them and producing waste, whereas industries consume material resources, process them and exerts products and waste (Ayres, 2004). The similarity between biosphere and technosphere is based upon the idea that both organisms and firms change the composition of resources in the world. The analogy can be put to productive use by applying lessons from the biosphere to the technosphere and hence accelerating its evolution to a more sustainable and resource efficient state (Ayres, 1989). However Ayres also states that the value of the analogy is limited. This limited value is due to key concepts of the biosphere that does directly correlate with the technosphere. Nevertheless assessing how the biosphere functions and how we can learn from it is of great value. The key principles of industrial ecology can be summarized as follows (O'Rourke, Connelly, & Koshland, 1996):

- Improving the metabolic pathways of processes and methods.
- Creating loop closing industrial ecosystems, which means having as less as possible input and output materials in a system.
- Dematerialize products, for example having a product with the same function with lesser materials. However this is not always necessarily good, because some materials can be scarce or environmentally unfriendly and can therefore have a stronger negative impact on the ecosystem when used in products than others.
- Systematizing patterns of energy use. Minimize energy consumption by for instance energy cascading, meaning using as much of the embodied energy of products at there end of life stage for other products with the least of energy loss.
- Balancing industrial input and output to natural ecosystem capacity.
- Aligning policy to conform with long term industrial system evolution.
- Creating new action-coordinating structures, communicative linkages, and information.

These principles can be are linked to various other related concepts, frameworks and definitions such as the Triple Bottom Line, Life Cycle Thinking, Circular Economy, Closed-loop Supply Chains, Industrial Symbiosis.

In addition certain unexpected indirect effects can have significant environmental consequences, in turn the rebound effects and ripple effects are taking into consideration within the field of IE. The terms that are described in this chapter are shortly addressed in the section below, because they highlight important pieces to the field of IE and to the thesis topic.

### Triple Bottom Line and Sustainable Development

The Triple Bottom Line (TBL) similar to the field IE came to existence under growing concerns on economic growth and its social and environmental consequences. TBL is more focused on guidelines for corporations to ensure that companies not only aim at adding economic value, but also include social and environmental dimensions (Elkington, 2001). The 3P formulation: Planet, People, Profit was developed in which companies ideally should make sure that all three P's are respected in their business models.
Next to TBL, Sustainable Development (SD) is a term widely used in the field of IE. Although SD has been defined in many ways, the most famous definition came from the Brundtland report in 1987 (Brundtland, 1987): "Sustainable development is development that meets the needs of the present without compromising the ability of future generations to meet their own needs". This term addresses societies needs and the limitations that the environment has to meet those needs.

### Life Cycle Thinking

Life cycle thinking is a way to conceptualize environmental issues and how we deal with them, by taking in account the whole life cycle of products/activities (Heiskanen, 2002). Environmental issues are systems extending beyond the boundaries of individual companies and Life cycle thinking proves as a way to make people aware of environmental issues and forces organizations to deal with environmental impacts caused by the whole product chain instead of the single process the respective organization is involved in. The life cycle approach (life cycle thinking) emerged as a conceptual model and led to the emergence of instruments and tools for example Life Cycle Assessment (LCA).

### Circular Economy and Closed-loop Supply Chains

Circular Economy is used as a conceptual approach for industrial systems, wherein industrial systems in general should be restorative or regenerative (The Ellen MacArthur Foundation, 2012). This means a shift from linear production mechanisms using unsustainable materials and energy sources towards circular production processes utilizing renewables and minimizing waste through the redesign of materials, products, systems and business models. Next to the redesign, we can also think of shifts in roles of relevant actors related to industrial systems such as the role of citizens that may change (Hobson, 2015). These circular production processes can be seen as a system that implements closed-loop supply chains. Closed-loop supply chains differ from forward supply chains because in a traditional forward supply chain the customer functions as the end of the processes, whereas in closed loop supply chain there is value to be recovered from the consumer (Guide & Wassenhove, 2003). Taking this definition broader, value can be recovered throughout the production supply chain by utilizing waste and by-products.

### Industrial Symbiosis

Industrial Symbiosis (IS) is a collaboration between separate organizations involving not only physical exchange of materials, energy, water, and/or by-products, but also exchange of knowledge and assets. This exchange of resources in an industrial system can lead to the creation of novel knowledge and this exchange should be applied in a sustainable, effective and measurable way, with the aim of mutual benefit for the involved organizations (Chertow, 2000; Lombardi & Laybourn, 2012). IS is more of a practical concept that can be used by companies to cooperate in turn exchange flows (closing supply chains if possible) and knowledge.

### Rebound effect

The rebound effect is a counteracting effect of improved efficiency of technologies. For example increased energy efficiency leads to lower costs of energy services and in turn effects consumer behaviour (Herring & Roy, 2007). Energy saving by increasing efficiency may seem to reduce environmental impact as less resources would be needed to generate energy, however lower cost of energy services can lead to higher consumption levels. In this way increased consumption levels can counteract the initial environmental benefit of more efficient systems. This can be denoted as direct rebound effects, the indirect effect can be explained in the way that consumers may buy more, more functional or larger products. The rebound effect not only accounts for energy systems but also for other forms of consumption that can give environmental problems (Hertwich, 2005).

### Ripple effect

The ripple effect is very much in line with the rebound effect that occur in socio-technical systems and can be explained as:

- Indirect effects that are a consequence of overestimation of technology to mitigate climate issues and underestimation of environmental impacts (Arvesen et al., 2011).

The use of reductionist approaches with narrow system boundaries and emphasis on technical artefacts led to an incomplete assessment of social and economic aspects in which technology is embedded. Six issues are described by Arvesen, Bright & HertWich:

- Transitioning to clean energy causes climate impacts.
- Unclear cost benefits of energy efficiency due to:
  - negative costs such as market failure (failure of demand), conflicting assumptions of optimal behaviour according to technician than to individual end-use.
  - Rebound effect. If there is high price elasticity there is a large rebound effect.
- Fossil fuel with Carbon Capture and Storage (CCS) technology and renewables may lower system wide effect. If efforts on GHG emissions are hindered by fossil fuel lock-in; CCS will force a longer life span of fossil fuel usage.
- Lack of absolute decoupling (absolute decrease in impact, if GDP grows). Usually there is relative decoupling (per unit of product less energy used for example).
- Interconnections of environmental pressures (e.g. high risk of problem shifting). Due to complexity we tend to implement reductionist approaches.

- Underestimation of future demands. New categories of demand arise and grow which cannot be predicted beforehand.

### Relevance of Information Communication Technology (ICT) to IE

Now that these aspects within the field of IE has been described, we can see that many of these conceptual frameworks and definition intertwine with each other. In this thesis ICT plays a very large part and the question remains how can ICT contribute to the field of IE and why is it important? The vision of utilizing Information Technologies within the field of IE was first coined as Industrial Ecology 2.0 by Chris Davis and colleagues in 2010. With the advent of ICT, sharing information on vast amounts of data has become possible. The Internet already serves as the medium for knowledge exchange, and some fields in academia have already taken up the possibilities that current software (open source and licensed) provides. For example Bio-informatics emerged as a field that combines biological data with computer science to gain a better understanding of the functions of organisms . It has to be mentioned that appreciable amounts of data already exists relevant for IE, however that it is largely a challenge of interlinking data. Enabling collaboration, standardization and dealing with resistance due to privacy issues are key to successfully reaching a community-driven, collective knowledge web within the field of IE. As mentioned by Chris Davis creating feedback loops in which information that is gathered can be reused and replicated is of great value. In addition the quality of information can be improved through continual peer reviews. Concluding that ICT can contribute to IE in the collection, processing, curating and sharing amounts of data and knowledge. Unfortunately one may argue that within the field of IE that this can be improved (Davis et al., 2010). Additionally the way data is generated can be improved by using state of the art IT technologies.

To come back to the notion of socio-technical systems with regard to IE, we could argue that there is an inherent need to understand and deal with the complex nature of these systems. The complexity embodies uncertainties that are related to governance, rebound effects, ripple effects. ICT can serve as medium to model and simulate complexities that may lead to a better understanding and clearer goals for actors towards steering socio-technical systems. Ideally moving away from linear to closed loop supply chain while embracing the concepts of the Circulair Economy, TBL, IS, SD.

# Appendix 1: An overview of EMF

In this section a short description will be given on EMF terminology. This section might be useful for developers that want to use EMF.

- EMF model: Is essentially a class-diagram, a simple model of classes and represent data of the application.
- Class definitions (Java and UML) are represented as complex types definition if using XML
- Attributes (UML) or methods/functions (Java) are represented as nested elements declarations in XML
- E-core meta-model: A model to describe the meta-information. Which is the MAIA meta-model in this thesis project.
- Core model: An instantiation of the E-core meta-model is called a core model. These are actual case studies.
- Eclass: a modeled class (a name to a set of functions together)
- Eattribute: has names and types similar to UML attributes.
- Ereference: the link/association between classes
- Edatatype: can be primitives or object types such as java.util.Date.

E-core models can be developed through Java interfaces, XML schema, UML. With UML you have multiple options 1 direct editing with the graphical editor, option 2 import from UML (rational Rose).
Other features are persistence (data storage on physical drives) and serialization (translating data or objects into a state or structure so that it can be stored). This storage is either in a file or transmitted over network and reconstructed on the opposite end. To create a model with Java, Java interfaces are needed together with annotations. The serialization of core models for persistence is done by XMI (meta-data exchange). It is a standard for serializing meta-data. After E-core model construction, Java code is generated. It contains (1) interface classes and (2) implementation classes (instances)with functions/methods that correspond to those interface classes. The concept of notifiers is also important, it notifies any users that there is a change in state (e.g. if there is a set method it changes a variable, a notifier may be called). Also (3) packages (4) factory are generated. The generated factory includes a create method for each class (instance). Packages are used to return and set values of attributes/variables. Lastly and (5) XML schema for the model and (6) plugin manifest file is generated, so the E-core model can be used as an eclipse plugin,
Normally the generation is not enough. It is expected to add methods and instance variables to the code. To go back to notifiers they do not only observe, but also provide ways to do something with the observed change that is to extent the behavior (so called adapter's). Automatic generation creates many objects (classes and other related objects), which can easily be used to create instances. And data storage is done through resources. EMF is related to the Object Management Group (OMG) in which UML, XMI, MDA are part of.
XMI: connects XML with modeling. XMI is a meta-model in itself in which XML is the instance. Model Driven Architecture (MDA) is the focus in the OMG group, it is aimed at full life cycle application development. That is data and application integration, so that formats, languages all connected with each other true precise mapping.

# Appendix 2: JET Syntax

Similarly to appendix 1, this section is aimed for programmers that wish to utilize JET and want to have a better understanding of it. Below a description of JET syntax.

- <%=methodPrefix%> : if you want to generate for instance the Java code "private foo", then in JET it will be denoted by 'private <%=methodPrefix%> foo'
- <%maia.operationalStructure.Plan entity = (maia.operationalStructure.Plan)argument;%> : object declaration example (entity).
- <%@ jet package="JETTemplates" class="EntityAction_instance" imports="maia.physicalStructure.*" %> : output java file with imports
- <%=<object>.<method>%> : for any object to use its methods this is an option
- <% for (Plan plan : entity.getActions()){ %> <%=plan.getName()%> <% }%> : For loop example. In this case entity is an object ActionSituation.java but that object is of type EList Plan.
- <%maia.operationalStructure.EntityAction test = (EntityAction) entity;%> : class casting without validation of the object.
- <%if (entity instanceof maia.operationalStructure.EntityAction) { maia.operationalStructure.EntityAction test = (maia.operationalStructure.EntityAction) entity;}%> : class casting with validation of the object.
- <%// get object Plan %> : set comments

# Appendix 3: Eclipse RCP

Recent Eclipse software packages including the RCP take on a MDSD approach. Thus Eclipse RCP is strongly dependent on a application model (E4XMI). The application model is made out of visual (windows, parts views,editors)) and non visual elements (commands, handlers). Each model element is made out of attributes describing the look and feel of the respective element. However the actual UI widgets are still defined in source code, which can be displayed in a visual element.
Model elements of the application model are connected to classes by URI's. For Java classes a 'bundleclass' with prefixes component is used and for resources a 'platform' component is used. Lets assume a part that contains a URI pointing to a Java class. The Java class contains actual code that make up the logic that is shown in a part. While the part itself may be embedded in a window layout.

- Windows: are the empty canvasses that can contain other elements
- Parts: UI components which allow to navigate and modify data
- Views: are parts, but are typically used for working on a particular set of data. e.g. the project explorer can be used to explore projects of the application.
- Editors: are parts used to modify single data elements. e.g. the Java editor is used to edit Java files.

Parts can be assigned to windows or used directly, but they can also be grouped together:

- Part shash container: All children parts are shown at the same time. The container data attribute can be used to assign a given weight to parts, thus some are then bigger than others.
- Part stack: Has children parts which only one is shown at a given time (has tabs).
- It is also possible to have a part shash container with a part stack in it.

A perspective is a container for sets of parts, for instance for manipulating data one might have a perspective as well as one for visualizing data. In terms of Java classes, at the moment there is a pointer to a Java class in the runtime environment an object of that class is made. Commands are used to set actions on events, however a command is only an abstract explanation of an action, there are no implementation details. The behavior of a command is described via handler. If a command is selected, the runtime determines the relevant handler for the command. The handler refers to a specific class (e.g. Menu item with command reference < command < Handler < Java class).
RCP has many features such as bundling a JRE with the application, and creation of tools for multiple platforms.

# Appendix 4: MAIA Concepts & ABMS

The transformation of high level to low level language requires a mapping of MAIA concepts to simulation modules. Table A4.1 below signifies the simulation elements required in Agent-based models in relation to MAIA concepts. Beware that not all MAIA concepts are integrated in the tool and that some concepts are differently handled.

Table A4.1: Linkages of MAIA concepts with ABMS elements (source: (Amineh Ghorbani, 2013))

| **Simulation Element** | **Related MAIA Concepts** |
|---|---|
| Agent | The agent concept in MAIA represents agents in the simulation. The agent is extended with the responsibilities and objectives of the potential roles he may enact. Also, the physical components that an agent may possess are included in the definition of the agent. |
| Agent List | For each MAIA agent type, we define a list. All public physical components are also stored in lists. |
| Scheduling Mechanism | This is not present in the MAIA meta-model and would need to be added to the simulation. |
| Space | The action arena, institutions and physical connections provide the operational, institutional and physical spaces in the simulation respectively. |
| Main Model | Initialization is not provided by MAIA. However, the evaluative structure of MAIA which addresses the goal of the simulation with a set of variables, builds the result section of this element. |

A further breakdown of each simulation element in relation to MAIA concepts is described below.

- THE AGENT, AGENT LIST AND ACTION-ARENA: In MAIA, an agent is a decision making entity. He has information on physical components (Agent list) and which role he may be performing in the action arena (environment). He must comply to institutions with respect to whether it is a rule, norm or shared strategy. In the case of shared strategy, there are no consequences if the institutional statements are not followed, hence irrational behavior might take place. Additionally an agent may also perform Multi Criteria Decision Making (MCDA). MCDA is a way of determining preferences and weighting them next to each other (e.g. what are the agent preferences, what are the properties of the technical artifact the agents wants, what do other consumers do?). Typically an agent in ABMS

has three phases at each time step: entering the simulation (enter), executing its plans (execute), leaving the arena (exit). Figure A4.1 gives a sketch of an agent and its processes of behavior.

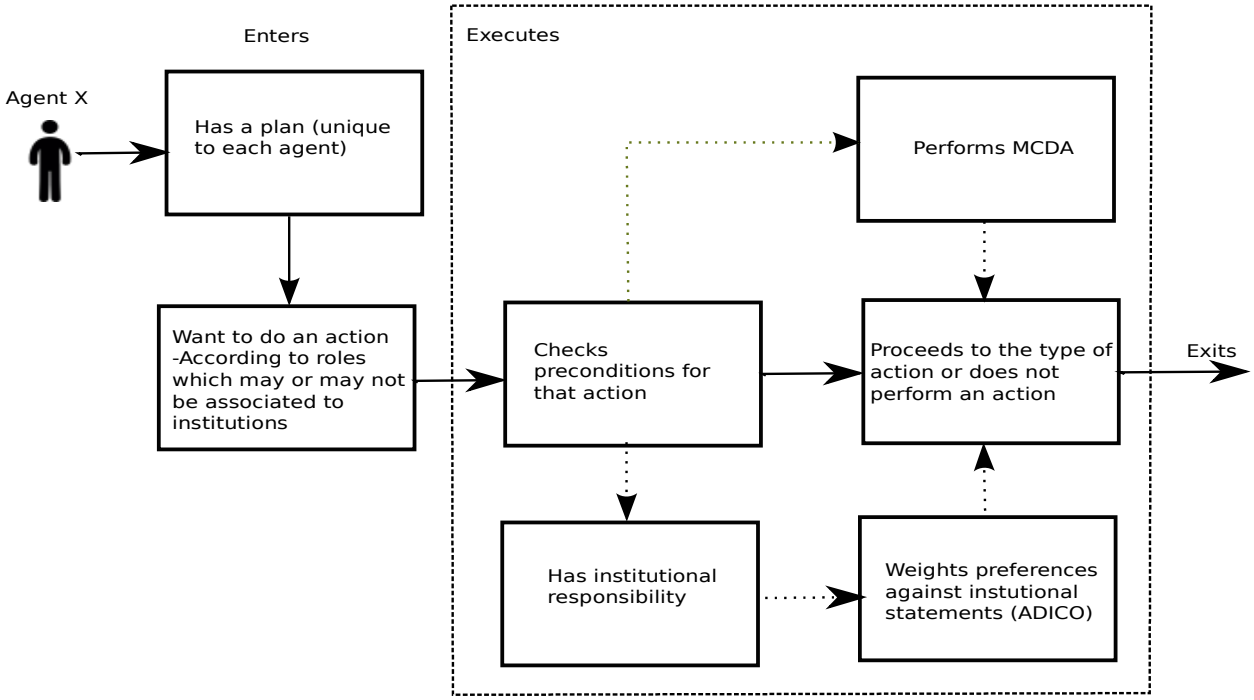



Figure A4.1: Agent's behaviors. The dashed arrow lines denote optional steps taken by agents.

- THE ORDER OF ACTIONS/SCHEDULING MECHANISM: In order to allow for emergent behavior, agents do not perform actions in a specific order. The MAIA meta-model does not provide the order of actions either. The way this step is defined can be different per meta-model, however typically the order of agents entering the action arena is systematically randomized. In addition agents may or may not perform actions they come across as this allows for unanticipated behavior. One may argue that agents should perform their actions simultaneously, however this creates complexities for the evaluation of the model. Basically the modeler wants to track how the system evolves over time, and if agents perform their actions simultaneous it becomes quite difficult to analyze the emergent behavior.
- SPATIAL AND LOGICAL REPRESENTATION OF AGENTS / THE SPACE: In figure A4.1 there are preconditions and certain institutional responsibilities. This signifies certain operational, constitutional structures. Constitutional in the sense of institutional statements and operational in terms of agent behavior according to agent's plans (which may or may not involve institutional responsibility). The physical structure is also denoted in this simulation element, this can be visual elements such as coordinates that place agents in the action arena. But can also be definitions on public components that are shared between agents.

- THE MAIN MODEL: Signifies the initialization process (e.g. what starting variables do agents possess) and uses the scheduler mechanism to control actions taken in each time step.
- THE EVALUATIVE STRUCTURE: This structure is not represented by one of the simulation elements as it is for a large part outside of the scope of simulation. However the monitoring of variables and parameters (e.g. starting cash balances and how it evolves) and an analysis of relations between variables and agents is important and is specified during conceptualization.

# Appendix 5: Platform Implementations of MAIA Concepts

Considering the limited time frame of the thesis project, not all MAIA concepts could be integrated in the application platform. As showcased with the case study a running simulation can be developed with only a subset of the MAIA concepts. To provide a clear overview for further extensions a table is constructed that describes what is readily integrated, what is partially integrated and what should be integrated in the future in terms of MAIA concepts. For more details on the specific MAIA concepts refer to Dr. Amineh's Ghorbani dissertation (Amineh Ghorbani, 2013) and the Results chapter of the thesis.

Table A5.1: The MAIA collective structure

| MAIA Concept | MAIA attribute | Completely integrated in platform | Integrated in JET code generation | Represented in EMF, but to be developed for platform | Notes and remarks |
|---|---|---|---|---|---|
| *Agent* | *Name* | √ | | | |
| *Agent* | *Personal value* | | | X | Properties can embody personal values |
| *Agent* | *Physical component* | √ | | | |
| *Agent* | *Possible Role* | √ | | | |
| *Agent* | *Intrinsic behavior* | √* | | | *Implicitly implemented in agent entity actions. Do not directly fill this in with the GUI, but through entity actions. |
| *Agent* | *Property* | √ | | | |
| *Agent* | *Information* | √ | | | Assigns a value to a variable that the agent already has . Can also be simulated through Physical component (OPEN) attribute as done in the case study. |
| *Agent* | *Decision* | | | X | Can be simulated through entity actions |
| *MCDA* | *All* | | | X | Can be simulated, but a comprehensive entity action condition needed |
| *Personal Value* | *All* | √ | | | |

Table A5.2: The MAIA constitutional structure

| MAIA Concept | MAIA attribute | Completely integrated in platform | Integrated in JET code generation | Represented in EMF, but to be developed for platform | Notes and remarks |
|---|---|---|---|---|---|
| *Role* | *Name* | √ | | | |
| *Role* | *Objective* | | √ | | Not used in operationalization of simulation. Specifically the entryCondition method generated is not called. |
| *Role* | *Institutional capability* | √* | | | *Implicitly implemented in role entity actions. Do not directly fill this in with the GUI, but through entity actions. |
| *Role* | *Institution (typeof strategy)* | | | X | |
| *Role* | *Entrycondition* | | | X | Can be implemented in precondition of role entity action |
| *Role* | *Physical component* | √ | | | |
| *Role* | *Information* | | | X | Can be simulated through Physical component (OPEN) attribute |

Table A5.3: The MAIA physical structure

| MAIA Concept | MAIA attribute | Completely integrated in platform | Integrated in JET code generation | Represented in EMF, but to be developed for platform | Notes and remarks |
|---|---|---|---|---|---|
| *Physical component* | *Name* | √ | | | |
| *Physical component* | *Type* | √ | | | |
| *Physical component* | *Affordance* | | | X | |
| *Physical component* | *Behaviour* | | | X | Can be simulated through entity actions in case of |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | OPEN component type |
| *Physical component* | *Property* | √ | | | |
| *Connection* | *Begin node/end node* | | | X | |
| *Composition* | *Begin node/end node* | | | X | |

Table A5.4: The MAIA ontological structure

| MAIA Concept | MAIA attribute | Completely integrated in platform | Integrated in JET code generation | Represented in EMF, but to be developed for platform | Notes and remarks |
|---|---|---|---|---|---|
| *Natural long condition (statements)* | *All* | √ | | | This is key for complex operationalization as the full Java language can be used to denote statements. |
| *Number property* | *All* | √ | | | |
| *String property* | *All* | √ | | | |
| *Boolean property* | *All* | √ | | | |
| *Formula* | *All (Andformula, orFormula etc.)* | | √ | | Used by objectives |

Table A5.5: The MAIA operational structure

| MAIA Concept | MAIA attribute | Completely integrated in platform | Integrated in JET code generation | Represented in EMF, but to be developed for platform | Notes and remarks |
|---|---|---|---|---|---|
| *Entity action* | *Name* | √ | | | |
| *Entity action* | *Action body* | | | X | Not needed as the name of entity action already signifies the action body |
| *Entity action* | *Post condition* | *√ | | | *Only a single statement allowed, use ";" to denote multiple or use Java operators |
| *Entity action* | *Pre condition* | *√ | | | *Only a single statement allowed, use Java operators |

| | | | | | |
|---|---|---|---|---|---|
| *Entity action* | *Post condition not do* | *√ | | | *Only a single statement allowed, use ";" to denote multiple or use Java operators |
| *Entity action* | *Performer* | √ | | | |
| *Entity action* | *Role enactment* | | | X | Linking of agents and roles are already integrated if performer is specified as a role type |
| *Entity action* | *Decision making* | | | X | Can be simulated through natural lang conditions |
| *Entity action* | *Institution* | | | X | |
| *Plan* | *All types* | √ | | X | Only one type of plan is implemented: the sequence plan. In which order of entity actions can be specified in the GUI |
| *Action situation* | *All types* | | | X | Same as the plan remark, entity actions orders can be specified sequential. Who performs what is not deterministic as the Agent list shuffle each tick. |
| *Action arena* | *All types* | | | X | Same as the plan remark, entity actions orders can be specified sequential. Who performs what is not deterministic as the Agent list shuffle each tick. |

Table A5.6: The MAIA evaluative structure.

| MAIA Concept | MAIA attribute | Completely integrated in platform | Integrated in JET code generation | Represented in EMF, but to be developed for platform | Notes and remarks |
|---|---|---|---|---|---|
| *Ontological concept variable* | *All* | | | X | |
| *Property variable* | *All* | √ | | | Properties dynamics can be tracked through either specifying visualizations in the GUI or through the default generated CSV. |

| | | | | | |
|---|---|---|---|---|---|
| *Personal value variable* | *All* | | | X | As mentioned in the collective structure table, the personal values can be represented as Agent attributes. |

# Appendix 6: Development MAIA-based RCP

This section serves as a means to address some important aspects for the tool development. As mentioned in the Materials chapter it is a RCP application. The development environment is:

- Eclipse SDK Version: Neon.1 (4.6.1)
- Build id: M20160907-1200
- JRE used: Java 8 oracle

The documentation that follows is mainly based on the following third party tutorials:

- [1] Vogella's Rich Client Platform Tutorial
- [2] Diksmetric's tutorial eclipse RCP e4 with 3.x views like project explorer, properties
- [3] EMF Forms Editors
- [4] Eclipse multi-platform's tutorial (A Brief Overview of Building at Eclipse )
- [5] Creating Eclipse wizards

All these tutorials can be found on https://github.com/SidneyNiccolson/RCPplatform or online.

Take away message from tutorial [1]:
A RCP can be created straight away through the wizard for RCP app creation or
an Eclipse plugin can be created first, in turn secondly a product is created, thirdly a feature should be created that holds the plugin (enter the feature as content of the product). Inside of the plugin resides the application.xml. As feature will be content to the product, you would have to change the product configuration to feature and include at contents the features we need (remove version dependencies).

A combination of tutorial [1] and tutorial [2] has been used to construct the basic application. If in tutorial [2] the hello RCP is missing just use the headless RCP template and find the Java files needed online for hello RCP template. Below the steps taken to built the application:
1: Install everything with tutorial [1]
2: Follow tutorial [2]
3: Make sure that dependencies are met, and you can export the product.
4: Follow tutorial [3], but add your own emf core model (MAIA) to the product (see next bullet points).

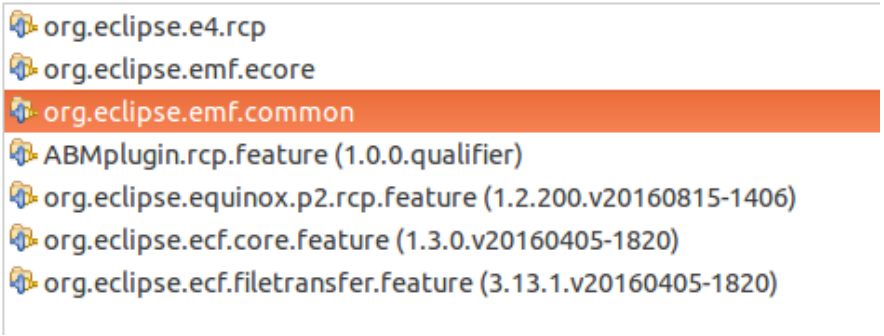
- Change the extentions in plugin.xml to “maia” instead of “task”.
- If not yet installed, install Eclipse modeling framework SDK
- install EMF plugin for Eclipse
- Create a new "Empty EMF Project" in eclipse (I called mine 'MAIA', but I suppose you can name it what you want)
- -Copy 'tools.ecore (or what ever .ecore file I sent you previously) ' to the 'model' folder of the new project (I renamed it to maia.ecore, but I don't think that is significant)
- Create a new EMF Generator Model (right-click the model folder > new > other > find EMF Generator Model in the list; locate it in the model folder, and point it to the .ecore in the workspace)
- It automatically opens the genmodel; right-click the top element (maia), and select (subsequently) "Generate Model Code", "Generate Edit Code", "Generate Editor Code".

- Add all MAIA folders to the dependencies tab of the plugin.xml of your product
- Satisfy all dependencies through validation of the product (RCP function). You can add them either in feature.xml (advised) or add dependencies in plugin.xml.

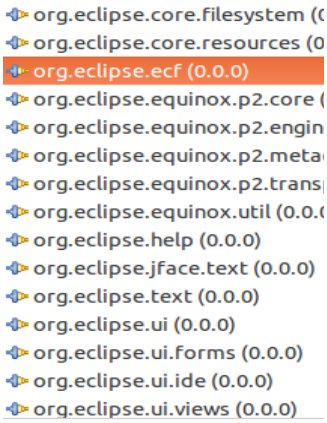
5: Create a multi-platform feature: This is based on tutorial [4].
6: Implement logic such as wizards that can create models or generate code functionalities: This is based on tutorial [5] (also see subsection "Creating wizards" in this appendix).

### Common errors:

- GENERAL MISSING DEPENDENCIES. Error: 'Application Id not found'. Means that the application dependencies is not added to the product dependencies (not in sync). To solve this in a quick way you can run configurations and select the application and add required plugins. This is not the right solution as you should add all these dependencies. In the product content tab insert this:




Create a feature project. In the feature.xml add the plugins! (creation of feature project is taken from Vogella tutorial), and add these dependencies (there are more than only these, see GitHub source code):




As a check always validate your product(RCP has an option for doing so), to see if dependencies are missing. Workaround method: to find the right dependencies, you can use run configurations and select certain plugins. If that make the application working than you are missing some that are by

default de-selected there (you can reset run configurations by deleting the entry and restarting the product). Therefore you need to add iteratively dependencies in one of the three directories or all (it is recommended to start with only feature.xml and check validation of export product each time to make sure the dependencies are met). For example if you want to include EMFforms then there are many org.eclipse.emfforms.* libraries that need to be added.
Below a list of important dependencies:

```
org.eclipse.core.*
org.eclipse.sdk > for JET options
org.eclipse.jdt.*
org.eclipse.search
org.eclipse.ui.console
org.eclipse.ui.intro
org.eclipse.debug.ui
```

- MULTI-PLATFORM ERRORS: while exporting to multiple platforms, I encountered an issue with Java version (errorcode: 13). this is probably due to the Eclipse product using a different Java version by default. This can be also 32bit or 64bit. As a solution you can try another windows version export than the one used or client side solution is to adjust the eclipse.ini to change the JVM used by the system.
- PRODUCT WORKS IN RUN-TIME ENVIRONMENT, BUT NOT AS EXPORTED APPLICATION: This can be solved by adding to the built tab in the manifest.mf a library called "." (simply dot). By having this you can link it to the src folder which is needed by the application.
- UNABLE TO LOAD CLASS ORG.ECLIPSE.EMFFORMS.SPI.EDITOR.GENERICEDITOR: This means that we need to find this class somehow, meaning we have to add ECP dependencies. At least the following one: install ecp SDK e4 (target feature) from the ECP/emfforms site (tutorial [3]). Add this dependency to the plugin itself (we have three dependency entries plugin.xml, feature.xml, product (contents)).
- JET FOLDERS ARE NOT CREATED ERROR: Files used for populating projects can not have the same name (capital or non-capital). There for the Jet templates need to have as package a non-used folder name.

### Creation of wizards and sending additional files:

Based on tutorial [5] the take away message is:
It is required to create an Eclipse plugin that uses the org.eclipse.ui.newWizards extension point. You can define your own category or use an existing one once you find the category ID. To create a new project wizard rather than a new resource wizard, you need to set the "project=true". Also, the plugin must contain a class that implement org.eclipse.ui.INewWizard. Clicking on the class link from the plugin.xml editor will do the trick. That class must do All The Work in the performFinish override, and must return true to indicate that it actually did its thing and the wizard can close. This is where you create files, directories, set natures, and so forth.

## Useful starting code:

```
//This is where I want the JET project to be created by default, by overriding performFinish you can create custom
projects
        @Override
        public boolean performFinish() {
                //set monitor to track progress
                NullProgressMonitor pm = new NullProgressMonitor();
                // create empty project
                IWorkspaceRoot root = ResourcesPlugin.getWorkspace().getRoot();
                IProject project = root.getProject("MyProject");
                try {
                        project.create(pm);
                        project.open(pm);
                } catch (CoreException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                try {
                //set Java nature
                          IProjectDescription desc = project.getDescription();
                           desc.setNatureIds(new String[] {
                              JavaCore.NATURE_ID});
                           project.setDescription(desc, pm);
                //set output folder for classess (bin folder)
                           IJavaProject javaProj = JavaCore.create(project);
                           IFolder binDir = project.getFolder("bin");
                           IPath binPath = binDir.getFullPath();
                           javaProj.setOutputLocation(binPath, null);
                           //custom libaries to add to the builtpath
                           String filename = "org.eclipse.emf.common-2.9.0.v20130528-0742.jar";
                           InputStream is;
                           is = new BufferedInputStream(new FileInputStream(filename));
                           IFile file = project.getFile(filename);
                           file.create(is, false, null);

                           IPath path = file.getFullPath();
                         //add custom folder
                           IFolder folder = project.getFolder("templates");
                           IFile file2 = folder.getFile("hello.txtjet");
                           if (!folder.exists())
                                    folder.create(IResource.NONE, true, null);
                           if (!file2.exists()) {
                                    byte[] bytes = "<%@ jet package=\"Templates\"  class=\"Role_instances\"
%>package constitutionalStructure;public class inspectionOfficer extends Role{}".getBytes();
                                    InputStream source = new ByteArrayInputStream(bytes);
                                    file2.create(source, IResource.NONE, null);
                                }
                        //set JRE
                           Set<IClasspathEntry> entries = new HashSet<IClasspathEntry>();

                           entries.addAll(Arrays.asList(javaProj.getRawClasspath()));

                           IVMInstall vmInstall= JavaRuntime.getDefaultVMInstall();

                           LibraryLocation[] locations= JavaRuntime.getLibraryLocations(vmInstall);

                           for (LibraryLocation element : locations) {

                           entries.add(JavaCore.newLibraryEntry(element.getSystemLibraryPath(), null, null));
                           entries.add(JavaCore.newLibraryEntry(path, null, null));
                           }

                           javaProj.setRawClasspath(entries.toArray(new IClasspathEntry[entries.size()]), pm);
                } catch (CoreException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                          //IProject newProject = getNewProject();
```

```
                catch (FileNotFoundException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        System.out.println("Working Directory = " +
                              System.getProperty("user.dir"));
                }
                return true;
        }
```

### Adding files to root directory of RCP app

Sometimes we want to expose files to the RCP app to be used by any wizards. In the development of the decision support tool for instance the default MAIA skeleton for construction of instance models needs to be exposed to the RCP app. Below an example of how files can be exposed:
Create a new feature [say org.me.helloworld.files.feature] Under the new feature, we create a file called "hi.html" and a folder called "myfiles" In the "build.properties" page remove "bin.includes = feature.xml" line, then add "root=file:hi.html,myfiles".
Finally, we go back to add "org.me.helloworld.files.feature" feature to the product file of the RCP app using "Contents" tab. Then Export the RCP product, now we can see "hi.html and myfiles folder" under RCP product root directory.

### Create output for potential error messages and error prevention on Windows

You can start in cmd and then <tool>.exe -consolelog to get the output of the application.
Make sure that the MAIA tool resides in a folder path with no empty spaces in directory names.

### Default plotting mechanism library

A simulation project constructed with the decision support tool has one single dependency, automatically added by the RCP application itself. That library is JfreeChart and is used for the tool's integrated default plotting mechanisms.

### Important Java classes

There are a few key Java classes that are not part of the default Eclipse RCP implementation, but are part of the custom logic applied to develop the tool. These classes are also available on https://github.com/SidneyNiccolson/RCPplatform and can be found in the following folder on GitHub RCPplatform/workspaceNeon1/ABMplugin.rcp/src. Below a description of these Java classes and their functions:

- JETTemplates classes (Java dependent classes in figure 3.5): These classes reflect the JET templates developed with Eclipse-JET. On GitHub the original templates can be found as well. As mentioned in the Methodology Chapter these classes are part of the MAIA framework and are used during parsing.
- ABMplugin > rcp classes: These classes reflect custom Eclipse RCP implementations based on tutorial [1] and [2]. The ApplicationWorkbenchWindowAdvisor.java class is important for setting the title of the and default size of the application. The application.e4xmi is the model of the application it self (as Eclipse RCP is model-based). ABMplugin.product is the actual product that links all classes together

and can be used for export of the application along with run-time configurations. Build.properties and MANIFEST.MF are Eclipse based configuration files (e.g. for configuring dependencies).

- CreateModel classes: The CreateModelShowWizard.java fires up the GUI for creating models. The CreateModelPage.java are the actual contents of the GUI. The CreateModelHandler.java implements all logic needed such as invoking the CreateModelPage.java contents and creating a project with a basic .maia skeleton.
- PartHandling: The RenderWebPage.java is used in the perspective Webtool for firing up the MAIA web-tool developed by Dr. Amineh Ghorbani.
- ProduceCode classes: These classes are really the core of the application. The MainSimulationDialog.java is the Dynamic GUI that takes in dictionaries (HashMaps) created by reading in MAIA instance model XML data and in turn dynamically creates the user interface. The ProduceCodeHandler.java generates all code needed for simulation. Hereby it reads in MAIA instance model XML data, generates MAIA instance code, creates dictionaries for the Dynamic GUI, retrieves data from the Dynamic GUI and finally constructs the main simulation class. The ProduceCodePage.java reflects the basic wizard content when the produce code button is launched. The ProduceCodeShowWizard is used to fire up the small GUI before the Dynamic GUI is invoked (which is invoked at performFinish() method in ProduceCodeHandler.java).
- MAIA resources (MAIA, MAIA.edit, MAIA.editor): The general MAIA resources (folders) contain the MAIA EMF based classes. These are dependencies of the application.
- ABM plugin rcp feature and plugin rcp files feature: As the basic MAIA skeleton needs to be supplied when creating models these features are added to supply that file.

*Sidney Niccolson*
*Thesis Research Project*
*February 6th, 2017*
*s1650548 (Leiden) - 4452909 (Delft)*

# Appendix 7: Entity actions in detail

Since entity actions determine the main features of any MAIA-based ABMS, a more detailed explanation will be given in this section. Entity actions can be assigned to agents (Collective Structure), components (Physical Structure), roles (Constitutional Structure). Entity actions embodies what each agent does per step (tick). Every default platform notation (Java based) will be colored blue with a larger font. Every case specific example will be specified with '<' and '>' symbols. Because this platform is based on the Java language, that programming language can be used directly.

## An introduction to entity actions

The order in what agent performs entity actions is defined in the dynamic GUI. The agent <Worker> enacting the role of a <Consumer> serves as a simple (though not necessarily realistic) example to describe entity actions. The <Worker> has as attributes <Money> and <AbleToWork>. In the role <Consumer> the agent utilizes a car to drive to work. We can specify two entity actions namely <DriveCar> and <Work>. Entity actions have "preconditions", "postconditions" and "postconditions not do" and those attributes have "statements". We first specify that the <Worker> in the role <Consumer> should <DriveToWork>. A "precondition" statement may be <Worker.getMoney() - 1000>, the "postconditions" might be <Consumer.getCar.getGas() – 1 ; <Worker.setAbleToWork(true);>.
A "postcondition not do" would be <Worker.setAbleToWork(false);>.
In the entity action <Work> the precondition is <Worker.getAbleToWork()> (true or false), a postcondition might be <Worker.setMoney(Worker.getMoney()+10;)>. The order of the entity actions should be specified in this case as firstly <DriveToWork> followed by <Work>. Below a table that gives an overview of this simple case.

Table A7.1: a description of entity actions for a simple hypothetical case study

| Agent | Role | EntityAction | Property update |
|---|---|---|---|
| Worker | Consumer | DriveToWork | Gas: decreases<br>AbleToWork: true/false |
| Worker | XXX | Work | Money: increases |

Below a table of the rules for any statements and examples on how to put statements together.
Table 2: Types of conditions and possible statements as example. Notice that for physical components of the "OPEN" type these example notation is slightly different (no .get or .set needed).

| Type of condition | Rules | Statement collective structure agents or roles example | Statement physical component (OPEN) example |
|---|---|---|---|
| Precondition | Can be completely omitted (default is true at that point). | Worker.getMoney() >= 1000 | Grid.TotalDailySupply >= 3 |
| Postcondition | Should always be set. Specify ";" sign at the end. | Worker.setMoney(Worker.getMoney()+10); Worker.setAbleToWork(true); | Grid.TotalDailySupply = Gird.TotalDailySupply + 1; |
| Postcondition not do | Can be completely omitted. Specify ";" sign at the end. | Worker.setMoney(1000); | Grid.TotalDailySupply = 3; |

In general with the semicolon notation (requirement by default also for one statement) multiple statements can be defined in the "postcondition" and "postcondition not do". In the case of precondition no semicolon is needed at all, but Java operators can be used. Beware that numerical values can be used without any specification, but letters (denoted in Java as Strings) should be within quotes (e.g. if <Worker> has property willingness you would denote it as Worker.setWillingness = "High"). For operationalization it is recommended to capitalize all attributes, as the current platform version does not support non-capitalized attributes.
The following Java operators can be used inside of statements:

- < : smaller than (use for numerical values). E.g. Worker.getAge() < 10.
- <= : smaller than or equal to (use for numerical values). E.g. Worker.getAge() <= 10.
- > : higher than (use for numerical values). E.g. Worker.getAge() > 10.
- >= : higher than or equal to (user for numerical values). E.g. Worker.getAge() >= 10.
- || : or statement. E.g. Worker.getAge() < 10 || Worker.getMoney() > 1000.
- && : and statement. E.g. Worker.getAge() < 10 && Worker.getMoney() > 1000.
- <attribute>.equals(): Assess if letters (Strings) are the same.
  E.g. Worker.getWillingness().equals("High")
- !<attribute>.equals(): Assess if letters (Strings) are not the same.
  E.g. !Worker.getWillingness().equals("Low")

## Scenario 1 Entity Actions Explained

The following tables describes how the entity actions are implemented for the case study in scenario 1.

Table A7.3: entity actions for the base scenario

| Entity action | Performer | Statements |
|---|---|---|
| supplyEnergyAsNuclearPowerPlant | NuclearPowerPlantCompany | **PostCondition:**<br>//set the starting value of CurrentAvailableEnergy<br>• Grid.CurrentAvailableEnergy = nuclearPowerPlantCompany.getNuclearPlant().getNuclearOutput(); |
| useEnergyAllSectors | AllSector | **PostCondition:**<br>//use energy from the grid<br>• Grid.CurrentAvailableEnergy = Grid.CurrentAvailableEnergy – AllSector.getDailyComElectricityUsage(); |
| useEnergyAsConsumer | Consumer | **PostCondition:**<br>//use energy from the grid<br>• Grid.CurrentAvailableEnergy = Grid.CurrentAvailableEnergy - passiveCitizen.getDailyCitizenElectricityUsage(); |
| generateEnergyAsProsumer | Prosumer | **PostCondition:**<br>//generate energy<br>• activeCitizen.setDailyElectricityGeneration(prosumer.getSolarPanelSet().getDailySolarOutput()); |
| useEnergyAsProsumer | Prosumer | **PostConditions:**<br>//use from grid if necessary otherwise supply the remainder of the usage<br>• Grid.CurrentAvailableEnergy = Grid.CurrentAvailableEnergy - (activeCitizen.getDailyCitizenElectricityUsage() - activeCitizen.getDailyElectricityGeneration() ) ;<br>//update DailyDemandBalance value<br>• activeCitizen.setDailyDemandBalance(activeCitizen.getDailyElectricityGeneration()- activeCitizen.getDailyCitizenElectricityUsage()); |
| followDemand | gasCoalPowerPlantCompany | **PreCondition:**<br>//check whether it is needed to follow demand<br>• Grid.CurrentAvailableEnergy <= 0.0<br>**PostConditions:**<br>//set net balance according to current available energy<br>• Grid.NetBalance = Grid.CurrentAvailableEnergy;<br>//update plant output to the demand needed |

| | | |
|---|---|---|
| | | • gasCoalPowerPlantCompany.getCombinedPowerPlant().setCombinedOutput(-Grid.CurrentAvailableEnergy);<br>//simplify the model by simply setting the currentAvailableEnergy to zero<br>• Grid.CurrentAvailableEnergy = 0;<br>**PostConditions not do:**<br>//reset to plant output as it should not run in this case<br>• gasCoalPowerPlantCompany.getCombinedPowerPlant().setCombinedOutput(0);<br>//set net balance to current available energy<br>• Grid.NetBalance = Grid.CurrentAvailableEnergy; |
| monitorDistribution | Grid | **PreCondition:**<br>//check whether there is oversupply<br>• Grid.CurrentAvailableEnergy > 0.0<br>**PostCondition:**<br>//set oversupply value as this condition is only executed in case of oversupply<br>• Grid.AccumulatedOverSupply += Grid.CurrentAvailableEnergy; |

Table A7.4: entity action extension (Demand decrease in sectors in weekends)

| Entity action | Performer | Statements |
|---|---|---|
| useEnergyAllSectors | AllSector | **PreCondition:**<br>//check if it is weekend<br>AllSector.getDaysOfEnergyUse() >=5<br>**PostConditions:**<br>//use energy from the grid with 5% less demand<br>• Grid.CurrentAvailableEnergy = Grid.CurrentAvailableEnergy – (AllSector.getDailyComElectricityUsage()*0.95);<br>//Specify that a day has passed<br>• AllSector.setDaysOfEnergyUse(AllSector.getDaysOfEnergyUse()+1);<br>//Make sure that the days of energy use are reset<br>• if (AllSector.getDaysOfEnergyUse() >= 7) {AllSector.setDaysOfEnergyUse(0);};<br>**PostCondition Not Do:**<br>//use energy from the grid (default)<br>• Grid.CurrentAvailableEnergy = Grid.CurrentAvailableEnergy – AllSector.getDailyComElectricityUsage();<br>//specify that a day has passed<br>• AllSector.setDaysOfEnergyUse(AllSector.getDaysOfEnergyUse() + 1); |

Table A7.5: entity action extension (seasonal weather effects on demand)

| Entity action | Performer | Statements |
|---|---|---|
| adaptToSeasonA | activeCitizen | **Precondition:**<br>//if not in summer<br>• Ticks.getTick() < 271<br>**PostConditions:**<br>//if fall<br>• if (Ticks.getTick() <= 88) {activeCitizen.setSeasonCoefficient(1.05);}<br>//if winter<br>• else if(Ticks.getTick() > 88 && Ticks.getTick() <= 177) {activeCitizen.setSeasonCoefficient(1.1);}<br>//if spring<br>• else if(Ticks.getTick() >= 178) {activeCitizen.setSeasonCoefficient(0.95);}<br>**PostCondition Not Do:**<br>// if the year has passed just reset the tick, because this model only runs for a single year<br>• if (Ticks.getTick() > 365) { Ticks.setTick(0); }<br>//set summer coefficient<br>• else{activeCitizen.setSeasonCoefficient(0.9);} |

Attribute SeasonCoefficient (default=1) has been added to the model in this extension. If the SeasonCoefficient is defined it needs to be used in the useEnergy entity actions of the agents. For example Grid.CurrentAvailableEnergy = Grid.CurrentAvailableEnergy – activeCitizen.getDailyCitizenElectricityUsage(); is changed to Grid.CurrentAvailableEnergy = Grid.CurrentAvailableEnergy – (activeCitizen.getDailyCitizenElectricityUsage() * activeCitizen.getSeasonCoefficient);
This implementation should as a final model be added throughout all the other agent entity actions that demand electricity.

Table A7.6: entity action extension (Variation in demand per day)

| Entity action | Performer | Statements |
|---|---|---|
| dailyVariationA | activeCitizen | **PostConditions:**<br>//specify default value<br>• double default = activeCitizen.getIniCitizenElectricityUsage();<br>//create random parameters between 0.8 and 1.2 (coefficient)<br>• Random randomizer = new Random(); double randomX = randomizer.nextDouble()*(1.2 - 0.8) + 0.8;<br>//use the coefficient derived above for setting the electricity usage<br>• activeCitizen.setDailyCitizenElectricityUsage(default*randomX); |

This implementation should as a final model be added for passiveCitizens as well. There could be an implementation for the sectors, yet different assumptions can be made on to what extent the usage variates in a day. An initial property has been added in this case called IniCitizenElectricityUsage defining the initial property value.

Table A7.7: entity action extension (Variation in solar panel output per citizen)

| Entity action | Performer | Statements |
|---|---|---|
| generateEnergyAsProsumer | prosumer | **PostConditions:**<br>//specify default value<br>• double default = prosumer.getSolarPanelSet().getIniDailySolarOutput();<br>//create random parameters between 0.9 and 1.1 (coefficient)<br>• Random randomizer = new Random(); double randomX = randomizer.nextDouble()*(1.1 - 0.9) + 0.9;<br>//determine electricitygeneration property with coefficient and SolarPanelOutput<br>• activeCitizen.getSolarPanelSet.setDailySolarPanelOutput(default*randomX);<br>• activeCitizen.setDailyElectricityGeneration(prosumer.getSolarPanelSet().getDailySolarOutput()); |

## Scenario 2 Entity Actions Explained

Two additional properties are added to the model in scenario 2 namely WillingnessToInvest (number property as personal value: during initialization this range is 0-9) for *passiveCitizens* and (boolean property: set to true by default) FeedInTariff for the *Grid*. Additionally the ElectricityGeneration property is now also allocated to *passiveCitizen* instead of only *activeCitizen.* It is possible to specify a single prosumer role and integrate the conditions described in scenario2, but for clarity purposes an additional role prosumerAsPassive is defined. See table below for a description of the new entity action.

Table A7.8: scenario 2

| Entity action | Performer | Statements |
|---|---|---|
| generateEnergyAsProsumerPassive | prosumerAsPassive | **PostConditions:**<br>//specify default value<br>• double default = prosumerAsPassive.getSolarPanelSet().getIniDailySolarOutput();<br>//create random parameters between 0.9 and 1.1 (coefficient)<br>• Random randomizer = new Random(); double randomX = randomizer.nextDouble()*(1.1 - 0.9) + 0.9;<br>//determine electricitygeneration property with coefficient and SolarPanelOutput |

| | | |
|---|---|---|
| | | • prosumerAsPassive.getSolarPanelSet().setDailySolarPanelOutput(default*randomX);<br>• passiveCitizen.setDailyElectricityGeneration (prosumer.getSolarPanelSet().getDailySolarOutput()); |
| useEnergyAsConsumer | consumer | **PostConditions:**<br>//use from grid if necessary otherwise supply the remainder of the usage<br>• Grid.CurrentAvailableEnergy = Grid.CurrentAvailableEnergy - ((passiveCitizen.getDailyCitizenElectricityUsage()- passiveCitizen.getDailyElectricityGeneration()) ; |

## Scenario 3 Entity Actions Explained

In scenario 3 the SmartMeter Physical Component is added for *citizens.* In theory the SmartMeter does not do much other than monitoring the *Grid's* CurrentAvailableEnergy. A property of the SmartMeter is the GridBalance that reflects the CurrentAvailableEnergy. The GridBalance determines the behaviour of electricity usage by *citizens.* We need to implement this feature in the electricity usage Entity Action, because during that action the CurrentAvailableEnergy changes. Below in the table a description of a Entity Action that implements this feature.

Table A7.9: scenario 3

| Entity action | Performer | Statements |
|---|---|---|
| useEnergyAsProsumer | prosumer | **PostConditions:**<br>//determine GridBalance property<br>• activeCitizen.getSmartMeter.setGridBalance( Grid.CurrentAvailableEnergy);<br>//determine if GridBalance is positive or negative and implement the usage accordingly (by adapating the seasonCoefficient)<br>• If (activeCitizen.getSmartMeter().getGridBalance() > 0) {activeCitizen.setSeasonCoefficient(activeCitizen.getSeasonCoefficient() + 0.3);}<br>• else {activeCitizen.setSeasonCoefficient(activeCitizen.getSeasonCoefficient() - 0.3);}<br>//use from grid if necessary otherwise supply the remainder of the usage<br>• Grid.CurrentAvailableEnergy = Grid.CurrentAvailableEnergy - ((activeCitizen.getDailyCitizenElectricityUsage()*activeCitizen.getSeasonCoefficient()) - activeCitizen.getDailyElectricityGeneration()) ; |

| | | |
|---|---|---|
| | | //update DailyDemandBalance value<br>• activeCitizen.setDailyDemandBalance((activeCitizen.getDailyElectricityGeneration()*activeCitizen.getSeasonCoefficient())-activeCitizen.getDailyCitizenElectricityUsage()); |

# 8 References

Aldewereld, H., Boissier, O., Dignum, V., Noriega, P., Padget, J., & others. (2016). Social Coordination Frameworks for Social Technical Systems.

Allenby, B. R., & Graedel, T. E. (1993). *Industrial ecology*. Prentice-Hall, Englewood Cliffs, NJ.

Amsterdamsmartcity (2016). Retrieved 18 October, 2016 from https://amsterdamsmartcity.com/projects/city-zen-smart-grid-in-amsterdam-nieuw-west

Arvesen, A., Bright, R. M., & Hertwich, E. G. (2011). Considering only first-order effects? How simplifications lead to unrealistic technology optimism in climate change mitigation. *Energy Policy*, *39*(11), 7448–7454. http://doi.org/10.1016/j.enpol.2011.09.013

Axtell, R. L., Andrews, C. J., & Small, M. J. (2001). Agent-Based Modeling and Industrial Ecology. *Journal of Industrial Ecology*, *5*(4), 10–13. http://doi.org/10.1162/10881980160084006

Ayres, R. U. (1989). Industrial metabolism. *Technology and Environment*, *1989*, 23–49.

Ayres, R. U. (2004). On the life cycle metaphor: Where ecology and economics diverge. *Ecological Economics*, *48*(4), 425–438. http://doi.org/10.1016/j.ecolecon.2003.10.018

Balancing Mechanism Reporting Service (2016). Retrieved 18 October, 2016 from https://www.bmreports.com/bmrs/?q=help/about-us

Bichraoui, N., Guillaume, B., & Halog, a. (2013). Agent-based Modelling Simulation for the Development of an Industrial Symbiosis - Preliminary Results. *Procedia Environmental Sciences*, *17*, 195–204. http://doi.org/10.1016/j.proenv.2013.02.029

Borrás, S., & Edler, J. (2014). Introduction: on governance, systems and change. In *The Governance of Socio-Technical Systems Explaining Change*. Cheltenham, UK: Edward Elgar Publishing, Inc. Retrieved from http://www.elgaronline.com/9781784710187.00010.xml

Brundtland, G. H. (1987). Our Common Future: Report of the World Commission on Environment and Development. *Medicine, Conflict and Survival*, *4*(1), 300. http://doi.org/10.1080/07488008808408783

Chertow, M. R. (2000). Literature and Taxonomy.

Clastres, C. (2011). Smart grids: Another step towards competition, energy security and climate change objectives. *Energy Policy*, *39*(9), 5399–5408. http://doi.org/10.1016/j.enpol.2011.05.024

Clean Energy Wire (2016). Retrieved 3 October, 2016 from https://www.cleanenergywire.org/factsheets/why-power-prices-turn-negative

David, C., & Mackay, J. C. (2009). *Sustainable Energy — without the hot air*.

Davis, C., Nikolic, I., & Dijkema, G. P. J. (2010). Industrial ecology 2.0. *Journal of Industrial Ecology*, *14*(5), 707–726. http://doi.org/10.1111/j.1530-9290.2010.00281.x

Drogoul, A., Amouroux, E., Caillou, P., Gaudou, B., Grignard, A., Marilleau, N., … Zucker, J.-D. (2013). Gama: A spatially explicit, multi-level, agent-based modeling and simulation platform. In *International Conference on Practical Applications of Agents and Multi-Agent Systems* (pp. 271–274).

Eclipse Birt (2016). Retrieved 4 December, 2016 from http://www.eclipse.org/birt/

Elkington, J. (2001). Enter the Triple Bottom Line. *The Triple Bottom Line: Does It All Add Up?*, *1*(1986), 1–16. http://doi.org/10.1021/nl034968f

Energy, Environment and Policy (2016). Retrieved 3 October, 2016 from http://euanmearns.com/electricity-supply-and-demand-for-beginners/

Frank Budinsky, David Steinberg, Ed Merks, Raymond Ellersick, T. J. G. (2003). *Eclipse Modeling Framework: A Developer's Guide*.

García-Magariño, I., Gómez-Sanz, J., & Fuentes-Fernández, R. (2009). INGENIAS Development Assisted with Model Transformation By-Example: A Practical Case. In Y. Demazeau, J. Pavón, J. M. Corchado, & J. Bajo (Eds.), *7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2009)* (pp. 40–49). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-00487-2_5

Garro, A., & Russo, W. (2010). easyABMS: A domain-expert oriented methodology for agent-based modeling and simulation. *Simulation Modelling Practice and Theory*, *18*(10), 1453–1467. http://doi.org/http://dx.doi.org/10.1016/j.simpat.2010.04.004

GEF (2016). Retrieved 4 December, 2016 from https://eclipse.org/gef/

Generative Software Engineering (2016). Retrieved 29 December, 2016 from http://documentation.genesez.org/en/ch01s01.html

Ghorbani, A. (2013). *Structuring Socio-technical Complexity; Modelling Agent Systems using Institutional Analysis*. TU Delft, Delft University of Technology.

Ghorbani, A., Bots, P., Dignum, V., & Dijkema, G. (2013). MAIA: a framework for developing agent-based social simulations. *Journal of Artificial Societies and Social Simulation*, *16*(2), 9.

Ghorbani, A., Dijkema, G. P. J., Bots, P., Alderwereld, H., & Dignum, V. (2014). Model-driven agent-based simulation: Procedural semantics of a MAIA model. *Simulation Modelling Practice and Theory*. http://doi.org/10.1016/j.simpat.2014.07.009

Ghorbani, A., Dijkema, G. P. J., Bots, P., Alderwereld, H., & Dignum, V. (2014). Model-driven agent-based simulation: Procedural semantics of a MAIA model. *Simulation Modelling Practice and Theory*, *49*, 27–40. http://doi.org/10.1016/j.simpat.2014.07.009

Glass for Europe (2016). Retrieved 2 October, 2016 from http://www.glassforeurope.com/en/issues/faq.php

Guide, V. D. R., & Wassenhove, L. N. (2003). *Business aspects of closed-loop supply chains* (Vol. 2). Carnegie Mellon University Press Pittsburgh, PA.

H Gharavi, R. G. (2011). Smart Grid: The Electric Energy System of the Future, *99*(6), 917–921.

Halog, A., & Manik, Y. (2011). Advancing integrated systems modelling framework for life cycle sustainability assessment. *Sustainability*. http://doi.org/10.3390/su3020469

Haplopoda, D. (2001). Book reviews, (2000), 231–240. http://doi.org/10.1111/j.1468-2346.2012.01079.x

Heiskanen, E. (2002). The institutional logic of life cycle thinking. *Journal of Cleaner Production*, *10*(5), 427–437. http://doi.org/10.1016/S0959-6526(02)00014-8

Herring, H., & Roy, R. (2007). Technological innovation, energy efficient design and the rebound effect. *Technovation*, *27*(4), 194–203. http://doi.org/10.1016/j.technovation.2006.11.004

Hertwich, E. G. (2005). Consumption and the rebound effect: An industrial ecology perspective. *Journal of Industrial Ecology*, *9*(1–2), 85–98. http://doi.org/10.1162/1088198054084635

Hobson, K. (2015). Closing the loop or squaring the circle? Locating generative spaces for the circular economy. *Progress in Human Geography*, 0309132514566342-. http://doi.org/10.1177/0309132514566342

Huang, Y. (2013). Automated Simulation Model Generation.

Java Zone (2016). Retrieved 4 December, 2016 from https://dzone.com/articles/real-time-charts-java-desktop

Kraines, S., & Wallace, D. (2006). Applying Agent-based Simulation in Industrial Ecology, *10*(1), 15–18.

Le Page, C., Becu, N., Bommel, P., & Bousquet, F. (2012). Participatory Agent-Based Simulation for Renewable Resource Management: The Role of the Cormas Simulation Platform to Nurture a Community of Practice. *Journal of Artificial Societies and Social Simulation*, *15*(1), 10. http://doi.org/10.18564/jasss.1928

Liang, Y. D. (2009). *Introduction to Java programming: brief version*. Pearson Prentice Hall.

Lombardi, D. R., & Laybourn, P. (2012). Redefining Industrial Symbiosis: Crossing Academic-Practitioner Boundaries. *Journal of Industrial Ecology*, *16*(1), 28–37. http://doi.org/10.1111/j.1530-9290.2011.00444.x

Milieu Centraal (2016). Retrieved 18 oktober, 2016 from https://www.milieucentraal.nl/energie-besparen/snel-besparen/grip-op-je- energierekening/gemiddeld-energieverbruik/

O'Rourke, D., Connelly, L., & Koshland, C. P. (1996). Industrial ecology: a critical review. *International Journal of Environment and Pollution*, *6*(2/3), 89–112. Retrieved from http://web.mit.edu/dorourke/www/PDF/IE.pdf

PSO Oklahoma (2016). Retrieved 3 October, 2016 from https://www.psoklahoma.com/info/facts/Facts.aspx > derived calculation (18,916,965*100)/1,900,000

Richards, J. P. (2009). *Mining, society, and a sustainable world*. *Mining, Society, and a Sustainable World*. http://doi.org/10.1007/978-3-642-01103-0

Software Sustainability Institute (2016). Retrieved 2 October, 2016 from https://www.software.ac.uk/resources/guides-everything/software-evaluation-guide

Stacke, F. (2008). Integrated pool / bilateral / reserve electricity market operation under pay-as-bid pricing, 1–7.

SunPower (2016). Retrieved 18 October, 2016 from https://us.sunpower.com/home-solar/solar-cell-technology-solutions/winter-solar-panel- performance-and-maintenance/

The Ellen MacArthur Foundation. (2012). Towards a Circular Economy - Economic and Business Rationale for an Accelerated Transition. *Greener Management International*, 97. http://doi.org/2012-04-03

ucsusa (2016). Retrieved 19 October, 2016 from http://www.ucsusa.org/clean_energy/our-energy-choices/how-is-electricity-measured.html#.WCrwyswrJEQ

Understand Solar (2016). Retrieved 30 November, 2016 from, http://understandsolar.com/calculating-kilowatt-hours-solar-panels-produce/

van Dam, K. H., Nikolic, I., & Lukszo, Z. (2012). *Agent-based modelling of socio-technical systems* (Vol. 9). Springer Science & Business Media.

VisualVM (2016). Retrieved 4 December, 2016 from https://visualvm.github.io/

W3schools (2016). Retrieved 20 December, 2016 from http://www.vogella.com/tutorials/EclipseRCP/article.html

Zest (2016). Retrieved 4 December, 2016 from https://www.eclipse.org/gef/zest/