DAI-Labor
TU Berlin

Diplomarbeit

Automated Generation of JIAC AgentBeans from BPMN Diagrams

Fachbereich *Agententechnologien in betrieblichen Anwendungen und der Telekommunikation* (*AOT*)

Prof. Dr.-Ing. habil. Sahin Albayrak
Fakultät IV Elektrotechnik und Informatik
Technische Universität Berlin

Vorgelegt von: **Petrus Setiawan Tan**

Betreuer: Dipl.-Inform. Tobias Küster

Petrus Setiawan Tan
Matrikelnummer: 213933
Stettiner Str. 9
10243 Berlin

Die selbstständige und eigenhändige Anfertigung dieser Diplomarbeit versichere ich an Eides statt.

Ort, Datum	Petrus Setiawan Tan

Abstract

Zusammenfassung

Acknowledgement

Inhaltsverzeichnis

1. Introduction

In this chapter, we will start by introducing the motivation and the goals of this work.

1.1 Motivation

1.1.1 Model Driven Engineering (MDE)

Over the past few years more and more software developers have been adopting the principle of *Model Driven Engineering*(MDE) where they no longer focus on writing programs but on creating a set of models which define the software. By modelling the software, the developer creates documents that provides an abstract view of the software system, independantly from the plattform or a specific programming language, making it understandable for non experts i.e. the stakeholders as well as applicable in different plattforms. These models will then be the basis for the implementation. A significant number of the so called CASE (Computer Aided Software Engineering) tools have been developed to support this methodology. Beside providing support in creating and editing the models, most of these CASE tools are also equipped with transformation features that allows us to transform the model into text or even executeable Programs, thus increasing efficiency in the software development process. We can say that the real benefits of MDE lies in the transformation. By providing a mapping between the model and the code, we can create standardized programs, accelerate development time and minimize faults in writing the code.

1.1.2 MDE in Multi-agent systems

Back in 2007, an MDE-approach has been made in order to bridge the gap between the industry and the multi-agent systems. As a result, a CASE-tool called the ***Visual Service Design Tool (VSDT)*** was developed by Tobias Küster in scope of his Diploma Thesis, which provides the transformation of BPMN (Business Process Modelling Notation) to BPEL(*Business Process Execution Language*) and

JIAC (Java-based Intelligent Agent Componentware) framework. This tool allows agents to be designed using a business process diagram, a model which has already been manifested in the industry. In the scope of this work, a plugin to VSDT will be developed to enrich it's transformation feature with a code generator that will transform BPMN models into executeable Java Code, or JIAC AgentBeans to be more specific.

1.2 Goals

The main goal of this work is to develop an eclipse plugin as an extension to VSDT to enrich its transformation features with a new transformation from BPMN to Java Code or JIAC AgentBeans to be more specific. Because it is nearly impossible to put all implementation details into the model, and to anticipate the possibility, that the generated code will be edited manually, considerations has to be made, such that conflicts should not occur when the transformation is called to a code that has been edited manually.

2. Background

In this chapter, we will discuss the backgrounds of the transformation that should be developed. We will start with JIAC (the target of the transformation), BPMN (the model that is being used), followed by VSDT (the existing modeling and transformation framework to JIAC that should be extended), and JET (the technology that will help us to implement the transformation).

2.1 JIAC(Java-based Intelligent Agent Componentware)

JIAC is a Java-based agent architecture and framework that was developed to simplify the development of software agents. The framework supports the entire software development process of a software agent system, from the design, implementation, and deployment of the system. The JIAC framework provides features such as FIPA compliant communication, Believe-Desire-Intention (BDI) reasoning, strong migration, web-service connectivity and others. Further it provides high security (Common Criteria EAL3, certified by the Federal Office for Information Security of Germany, BSI) and advanced accounting mechanisms, making it suitable for the use in industrial and commercial applications. The JIAC framework comes with a runtime environment and a toolsuite for the creation of agents.

A typical JIAC Application consists of *AgentNodes*, *Agents*, and *AgentBeans* (see also Figure 2.1). An *AgentNode* is a Java VM providing the runtime infrastructure for agents, which consists of discovery services, white and yellow pages services, as well as communication infrastructure. A JIAC application consists of one or more AgentNodes. Normally, there is one AgentNode per physical machine. The AgentNode comes ready-to-run, but can be adapted to the needs of the target environment and can also be extended by additional components, so-called AgentNodeBeans.
Each AgentNode may run several *Agents*. Agents provide services to other agents and comprise lifecycle, execution cycle and a memory. An agent can use infrastructure services in order to find other agents, to communicate to them and to use their

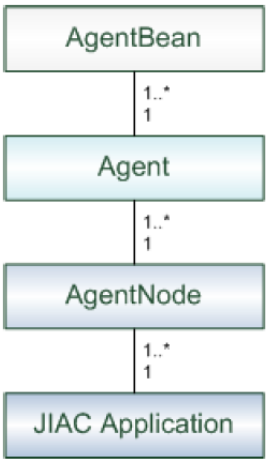

Figure 2.1: Jiac Basic concepts and their structural relationships [4]

services. Skills and abilities of the agent can be extended by so-called AgentBeans. *AgentBeans* is the mean to implement the functionality. They are plugged into agents and provide services (so-called Actions) to other agents. AgentBeans have a lifecycle.

2.2 BPMN(Business Process Modelling Notation)

BPMN [5] is a standard Notation for modelling business processes, initially published by the BPMI which is later adopted by the OMG(Object Management Group). A business process diagramm (as seen in Figure 2) can be compared to UML's activity diagram.

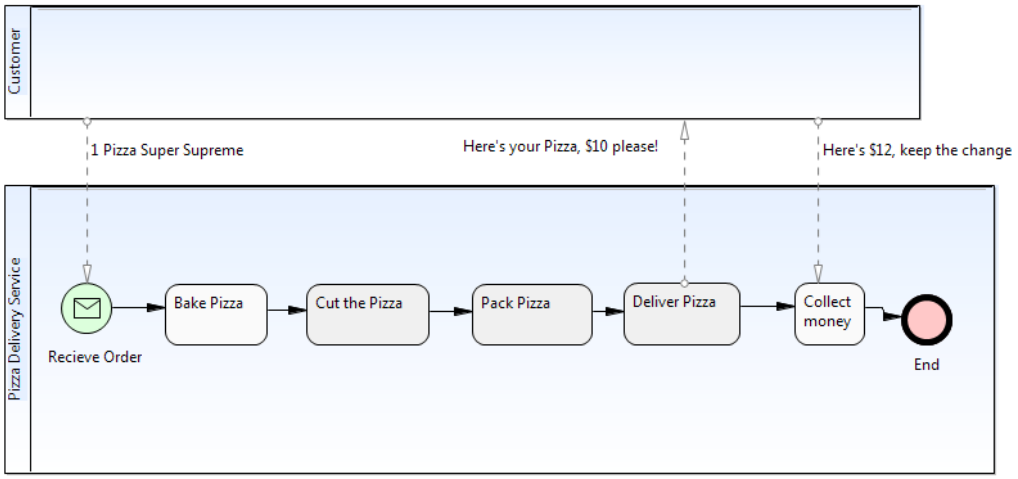

Figure 2.2: A simple Business Process Diagramm

BPMN was made to provide a notation that is understandable by all business users, creating a brige for the gap between the business process design and the process implementation. With the mapping of BPMN to Agents, we hope to be able to increase the spreading of the multi agent systems in the business world.

To provide a rough overview on how Agent technology can support the implementation of Business Processes let us take a look on this mapping example of BPMN

elements to Agents. A Pool in a Business Process Diagram can represent an Agent, which are able to communicate with other agents (another pool) through messages (represented with the BPMN MessageFlows). Agents can react to Events. To fully implement the transformation a detailed mapping is needed and will be developed in the scope of this work.

2.3 VSDT(Visual Service Design Tool)

The VSDT [?, ?, ?, ?] is a well-equipped BPMN editor that is independent of any specific target language. Thus, while the usual transformation to BPEL, as well as other transformations, is included, the VSDT can easily be extended with additional export functionality targeting other languages. It was developed in early 2007 as a Diploma Thesis at the TU Berlin and since then it has been continuously extended. Since it's initial development, VSDT's transformation framework is designed to be extensible and reuseable. This allows the development of a new transformation to be easier. For this purpose the transformation process is subdivided into several stages:

1. *Validation*: Validate the input model.
2. *Normalisation*: Prepare the input model for transformation.
3. *Structure Mapping*: Convert the input model to a block-like structure.
4. *Element Mapping*: Perform the actual mapping, create target model.
5. *Clean Up*: Remove redundancies, improve readability, etc.

Due to the fact that the validation, normalisation and structure mapping are mostly independent from the target language, the standard mapping provided for these stages are reuseable, which makes it possible to implement a new transformation by specifying the element mapping only. Figure 3 shows the UML Class Diagram of the transformation famework with the example transformation to BPEL.

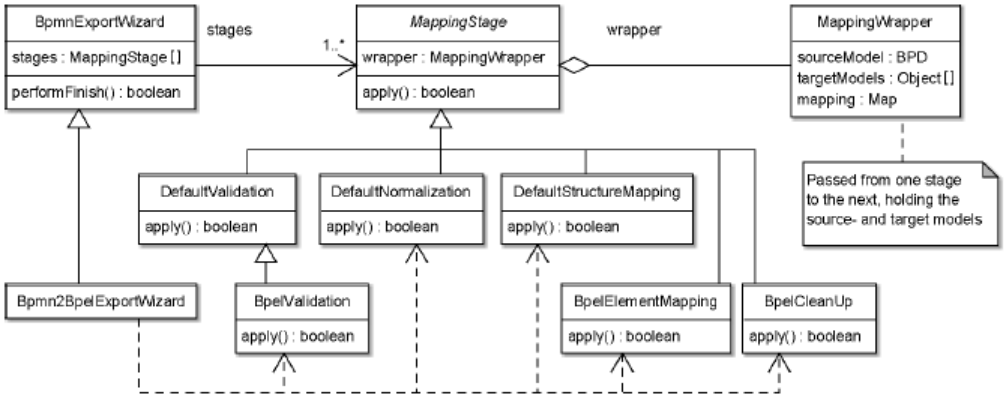

Figure 2.3: Essential classes of the transformation framework, including the BPEL case.[?]

2.4 JET(Java Emitter Templates)

JET[**?**] is a code generating framework, developed as a part of the Eclipse Modelling Project [**?**]. JET uses templates to transform the model into any kinds of code artifacts such as java, html, xml or even plain text.

JET templates uses a JSP-like syntax which makes it easy to write and understand. A set of JET templates is called transformation. It is possible to build this transformation with a main template which acts as a visitor and runs through the model, and this main template will then use other templates which handle a specific element of the model. For example, in UML to Java transformation you can have special templates that handles the package, class, variables and methods.

2 different JET-Versions

There are currently 2 different JET-Versions in the Eclipse Modelling Project. The older Version allows us to generate text with an Object as argument. In the template, one can type cast the argument variable into the class of our model.
The translated JET-Template will have the method
public String generate(Object argument){ ... }

In the updated version of JET, also called JET2, some workspace and java related "tag-libraries" are provided, enabling us to do the transformation without using the Java and Eclipse API. Unfortunately, in this Version the model has to be an xml-File. To directly transform the BPMN directly from it's xml-Nature will be harder and the template will be confusing, therefore a decision has been made to implement the mapping using the existing transformation framework where an intermediate model class of the AgentBean will be created, and then this intermediate model into Java code using the older version of the JET Transformation. More details on the implementation will be discussed in chapter 5.

3. State of the art

A similar approach (designing agents behaviour with processes and transforming it into Java Code) has been developed by the Telecom Italia with their JADE-extension called WADE (Workflows and Agents Development Environment). While Jade was developed to simplify the implementation of Software Agents, WADE extended JADE with a workflow engine, making it possible to create Agents that executes tasks defined as workflows.

3.1 JADE (Java Agent DEvelopment Frameork)

JADE is an application framework and a middleware written in Java, which support the development of software agents. The framework provides distributed runtime environments, agent and behaviour abstractions as well as communications between agents and discovery mechanisms. We can say that it's role is very similar to JIAC.

3.2 WADE (Workflows and Agents Development Environment)

WADE [?, ?, ?] is a software plattform built on top of JADE. One of the key component of WADE is the WorkflowEngineAgent which extends the basic Agent class of the JADE library with an ability to execute workflows represented in a WADE specific formalism. A WADE Workflow is actually a Java Class, thus it can be edited and managed as Java classes and can contain pieces of code which is needed to implement the process. With WOLF, a development environment that comes with WADE, developers can edit the Workflow graphically as well as textually. The code view and the graphical view of the workflow are kept in sync. Despite having all the advantages of a Java code, the WADE workflow is rather simple and not so expressive as the BPMN.

4. Mapping BPMN to Jiac AgentBeans

The element mapping from BPMN to Jiac AgentBeans is created based on the existing mapping to JiacV - JADL script. In comparation to a JADL script, an AgentBean is written completely in Java, enabling more possibilities in mapping concepts such as intermediate Event handling. This chapter will provide a tabular overview of the mapping.

Business Process Diagram

Business Process Diagram	
Pool	A pool is currently mapped to a Java-File containing a Jiac AgentBean.
Process	The process content in a pool is mapped to a Workflow Method in the generated AgentBean. Depending on the type of the StartEvent, an Action might be exposed, enabling this method to be invoked as a service. This method will contain a Script which is generated from the flowObjects contained in the process.
Lane	Lanes will not be mapped in this mapping

Events

For intermediate Events, an EventHandlerThread will be generated and started. If the intermediate Event is attached to an activity, the activity will also be started as a Thread.

Start Events	
Timer	The generated process will be started in the execute() method of the JIAC AgentBean
Message	If the implementation is a Service, an action will be created, and the process can be started through a service call (invoke). If the implementation is a Message Channel, a Space Observer will be attached to the agents memory, and the process will start as soon as a message is recieved.
Rule	Rule start events are not regarded in the current mapping
Link	
Multiple	If a pool has multiple start events, each start event will be mapped according to it's respective trigger
Intermediate Events	
Rule	not regarded in the current mapping
Timer	A TimerEventHandler will be created and started.
Message	A MessageEventHandler will be created and started.
Link	...
Multiple	each event will be mapped according to it's respective trigger
Cancel	...
Compensate	...
Error	If attached to an activity, the method generated from the activity will be called in a try-catch block
End Events	
Message	If implementation is Message Channel, a message with the payload will be written to the channel
Link	
Multiple	
Cancel	
Compensate	
Error	The generated service will throw an exception
Terminate	

Activities

Basically, an Activity-method will be created for each activity.

Activity	
Standard Loop	...
Multi Instance Loop	...
With Event Handler	the method generated will be started parallel to an EventHandlerThread. When the activity is completed, the EventHandlerThread will be terminated. In contrary, if the Event is trigerred before the activity is completed, the Thread performing the activity will be interrupted.

5. Implementation of the transformation

In this chapter, some details of the transformation will be presented. As mentioned before in section 2.3, the transformation process in the VSDT is divided into 5 stages. We've also mentioned that the default validation and structure mapping provided by the transformation framework are reuseable. The stages of the implemented transformation to Jiac AgentBeans is shown on Figure 5.1

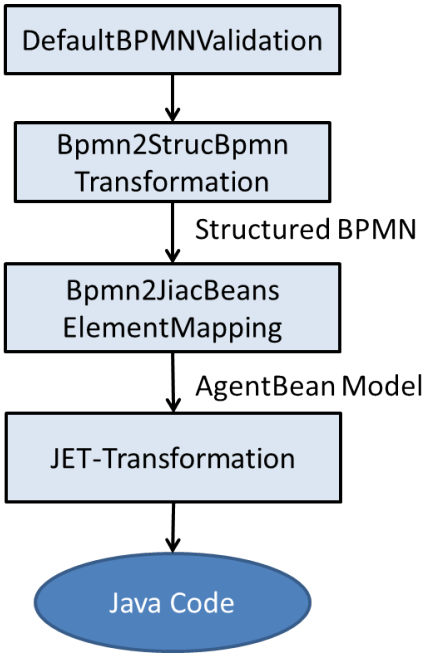

Figure 5.1: Transformation stages

6. Fazit

Literaturverzeichnis

[1] Fabio Bellifemine, Giovanni Caire, Agostino Poggi, and Giovanni Rimassa. Jade: A software framework for developing multi-agent applications. lessons learned. *Inf. Softw. Technol.*, 50:10–21, January 2008.

[2] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. JADE - a FIPA-compliant agent framework. In *Proceedings of the Practical Applications of Intelligent Agents*, 1999.

[3] Giovanni Caire, Danilo Gotta, and Massimo Banzi. Wade: a software platform to develop mission critical applications exploiting agents and workflows. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*, AAMAS '08, pages 29–36, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[4] CC ACT, DAI-Labor, TU-Berlin. *JIAC - Java Intelligent Agent Componentware*, 06 2010. Version 5.1.0 Manual.

[5] Object Management Group. Bpmn specification v2.0. `http://www.omg.org/spec/BPMN/2.0/`.

[6] Benjamin Hirsch, Thomas Konnerth, and Axel Heer. Merging agents and services the jiac agent platform. In Amal El Fallah Seghrouchni, Jrgen Dix, Mehdi Dastani, and Rafael H. Bordini, editors, *Multi-Agent Programming:*, pages 159–185. Springer US, 2009. 10.1007/978-0-387-89299-3_5.

[7] Eclipse Modelling Project Homepage. `http://www.eclipse.org/modeling/`.

[8] JADE Homepage. `http://jade.tilab.com/index.html`.

[9] JIAC Homepage. `http://jiac.de/?id=35`.

[10] VSDT Homepage. `http://jiac.de/?id=30`.

[11] WADE Homepage. `http://jade.tilab.com/wade/index.html`.

[12] JET. `http://www.eclipse.org/modeling/m2t/?project=jet#jet`.

[13] Tobias Küster. Development of a visual service design tool providing a mapping from BPMN to JIAC. Master's thesis, Technical University of Berlin, Faculty of Electrical Engineering and Computer Science, 2007.

[14] Tobias Küster and Axel Heßler. Towards transformations from BPMN to heterogeneous systems. In Massima Mecella and Jian Yang, editors, *BPM2008 Workshop Proceedings*, 2008.

[15] Tobias Küster, Marco Lützenberger, Axel Heßler, and Benjamin Hirsch. Integrating process modelling into multi-agent system engineering. In Michael Huhns, Ryszard Kowalczyk, Zakaria Maamar, Rainer Unland, and Bao Vo, editors, *Proceedings of the 5th Workshop of Service-Oriented Computing: Agents, Semantics, and Engineering (SOCASE) 2010*, 2010.

[16] Giovanni Caire (Telecom Italia S.p.A). *WADE User Guide*. Copyright ©2010, Telecom Italia, July 2010.