Validating MAS Analysis Models with the ASEME Methodology

Nikolaos Spanoudakis, Efthymios Floros, Nektarios Mitakidis, Pavlos Delias

preprint

When designing agent-oriented software, engineers consider performance-related non-functional requirements. To this end, performance engineering practices provide a useful toolbox. In particular, simulation of the system's processes appears eminently suitable. However, agent-oriented software engineering methodologies are not directly linked to process simulation features. This paper extends an AOSE methodology for transforming agent roles models to process models, and for streamlining the transformation process towards simulation. Our method allows diverse process model generation, aiming to support the process simulation, and was integrated into a model-driven engineering methodology. We used an established process modelling notation (BPMN) as the target language for the process model, and we are able to deliver a ready-to-simulate model. Through simulation, an analyst can validate specific system requirements and test scenarios of how the system scales beyond the current requirements. Furthermore, because of process models familiarity within the business domain, engineers, managers and stakeholders can seamlessly communicate system designs.

Agent-oriented software engineering; AOSE methodology; process models; BPMN; business process modeling notation; simulation; system validation; scaling; model-driven engineering;

Introduction

This paper aims to discuss and bring forward the concept of combining performance engineering practices with agent-oriented software engineering methodologies. Performance engineering is concerned with ensuring that a system will meet its non-functional requirements for performance (such as latency or resource usage) and scalability (Smith & Williams 2002). Although the idea has been around since 2000 (Rana & Stout 2000) in the context of developing large agent based applications, it is only recently that researchers proposed the idea of using process simulation in order to derive a process's performance measures (Delias & Spanoudakis 2010).

Process simulation allows to quantify performance measures (e.g., cycle time) when specific estimates about the activities are provided (such as processing latency, resource utilization, etc). Through process simulation, execution times can be forecasted along with possible bottlenecks and response of the process to increasing demand. Moreover, a wide range of tools support process simulation (Dumas et al. 2013).

This work extends the open source Agent Systems Engineering Methodology (ASEME) Integrated Development Environment (IDE) by integrating a new method fragment targeting the analysis phase. This method fragment (Mitakidis et al. 2015) allows a Multi-Agent System (MAS) model, an outcome of the analysis phase, to be transformed to a business process model. This model can be employed by an analyst to

validate specific requirements of the modelled system long before its implementation. Process models are well known to clients, business partners and managers, thus, they can have a familiar view of the system being modelled.

Additionally, this work extends the open-source Liveness2XPDL tool provided by a recent research (Mitakidis et al. 2015) by three ways: First, it provides functionality that adheres to the established BPMN 2.0 standard. Second, it enriches the range of the tools that can be used for diverse model generation, and third, it allows capturing the diverse expressiveness of BPMN.

Finally, we discuss how a model-driven AOSE methodology, such as ASEME can be extended to incorporate new method fragments (supported by tools, such as the liveness2XPDL). The Liveness2XPDL fragment supports a wide range of methodologies that have adopted the Gaia roles model (Wooldridge et al. 2000), such as ROADMAP (Juan et al. 2002), Gaia4E (Cernuzzi & Zambonelli 2009) and ASEME (Spanoudakis & Moraitis 2011).

This paper's purpose is threefold:

- show how an AOSE methodology can benefit from adopting features from the performance engineering domain
- propose to AOSE practitioners who use business process models for agent-based simulations (Pascalau et al. 2009; Szimanski et al. 2013) or for communicating them to business people (Onggo 2012), the use of an AOSE methodology
- discuss technical issues on how to use a dashboard for a model-driven AOSE methodology to facilitate the development process

In section two we discuss the background of this work, i.e. the ASEME methodology, the Liveness2XPDL tool and the BPMN language. Then, in section three, we discuss the algorithm for the automatic transformation process and its integration in the ASEME IDE. In section four, we present the modelling process for a sample MAS. Section 5 discusses our finding. Finally, in section six we conclude and discuss future work.

Background

ASEME

ASEME is a hybrid methodology incorporating concepts from the Tropos (Bresciani et al. 2004), MaSE (DeLoach & Garcia-Ojeda 2010) and Gaia (Wooldridge et al. 2000; Zambonelli et al. 2003) methodologies for agent-oriented systems development. Its originality, and added value, lies in being the first model-driven AOSE methodology where the models of a previous phase are transformed to models of the next one and a series of transformations and model refinements takes the developer from requirements elicitation to code generation. Moreover, it allows for integrating diverse agent capabilities and interaction protocols participation behaviour seamlessly in its intra-agent control which is a statechart.

Although ASEME assists the project risk management with the Functionality Graph, where the different technologies and algorithms needed for agent development are presented it lacks a methodological approach to risk apprehension and analysis model

validation, especially when it comes to no-functional requirements such as timing and scaling.

Liveness2XPDL

Liveness2XPDL is a recently developed tool for transforming Gaia liveness formulas to process models aiming to exploit on one hand their capability to be simulated and on the other hand the ease of communication of agent analysis models with business partners as they are familiar with that type of diagrams.

The input of the liveness2XPDL tool is the liveness property of a role. The latter was originally introduced by Gaia (Wooldridge et al. 2000; Zambonelli et al. 2003). It appears in an analysis phase model of Gaia, i.e. the *role model*. Liveness is a model used to define a role's behaviour. It is constructed by connecting the different tasks that a role can perform with specific operators. Briefly, A+ means that activity A is executed one or more times, A^* means that activity A is executed zero or more times, [A] means that activity A is optionally executed, A1.A2 means that activity A2 executes after activity A1 finishes, A1|A2means that A1 or A2 is exclusively executed, A1||A2, means that A1 and A2 are concurrently executed, and, A~ means that A is restarting as soon as it finishes forever.

The output of the tool is a process model in the XML Process Definition Language (XPDL version 2.1). XPDL, a standard supported by the Workflow Management Coalition (WfMC, http://www.wfmc.org) was originally intended as a serialization format for the Business Process Model and Notation (BPMN) standard of the Object Management Group (OMG). However, BPMN version 2.0 came along a rich and native XML serialization format and XPDL started to lose its momentum and support by available tools.

BPMN metamodel

Here we present the aspects of BPMN more important for our work and the metamodel that we used for development. BPMN 2.0 supports a large number of event and task types; however, their enumeration is out of the scope of this paper. Herein, we documented the types that will be used in the rest of this paper:

- Pools: They represent major participants in a process, typically separating different organizations. In the diagram they are represented by orthogonal parallelograms containing a process
- Collaboration: contains the information about the different roles, each of whom is represented by a pool
    - Participant: a system role represented by a pool
- Process: Defines the process in each pool
    - LaneSet: A collection of Lanes. Each pool can contain one or more lanes representing different actors within a specific organization
        - Lane: Representing a single actor or resource that can execute tasks within an organisation. In the diagram it is represented by an orthogonal parallelogram contained in a pool

- Start Event: The start event of a lane indicates the beginning of the process. Each pool can have one such and it is represented in the graphical by a circle
- End Event: Indicates the end of the process. In the graphical model it is represented by a circle with bold outline
- Exclusive Gateway: The XOR Gateway (exclusively one of the outgoing transitions will be followed) is represented by a diamond shape with the "X" character in the middle
- Parallel Gateway: The parallel gateway (all the outgoing transitions lead to activities that will be executed in parallel) is represented by a diamond shape with the "+" character in the middle
- Task: Tasks are represented by a rounded rectangle and correspond to an execution of a system function
- Send Task: A task that sends a message either to a receive task or to a message event
- Receive Task: A task that receives a message either from a send task or from a message event
- Sequence Flow: It is a flow represented with a solid line and arrowhead. It has source and target (at the arrowhead) tasks and defines the control flow in the workflow process
- Message Flow: It is a flow represented with a dotted line and arrowhead and has source and target (at the arrowhead) tasks and defines the message flow between different pools

ASEME IDE Tooling

Motivation

The first versions of BPMN (prior to BPMN 2.0) did not provide any standardization for process diagram interchange between systems. That need was mainly covered by XPDL. Nevertheless, users needed to transform their BPMN models to XPDL in order to exchange them. This transformation activity was not inherently supported by none of the sponsoring organizations (OMG and WfMC), so users had to rely on third parties solutions, that could not guarantee a seamless transformation. When BPMN 2.0 was introduced, an XML-based interchange format was included. In addition, to facilitate interchange of models across various tools, the BPMN Model Interchange Working Group (BPMN MIWG) was set up by OMG.

These bold moves, along with the widespread acceptance of BPMN as a process modelling notation, made BPMN the de facto standard for business process modelling, analysis, and execution. BPMN has gained vendors attention from XPDL, and in some cases vendors are phasing out their XPDL support features. For example, Signavio, and

BIMP (a free simulator of BPMN business process models, http://bimp.cs.ut.ee) require a BPMN model to run a simulation.

Therefore, transforming a liveness formula into BPMN provides the user with the following capabilities, due to interchange potentials:

- Straightforward simulation with a variety of free or commercial tools. In this paper we demonstrate this potential in a case study.
- Visualize differences between the diagram that was derived from the liveness formula (system analysis model), and an already existing model of a business process, see, e.g. (Ivanov et al. 2015). This potential can be used either for *alignment* (descriptive purposes / capturing reality better), or for *auditing* (normative purposes / identifying deviations).
- Find similarities of a process with other processes in a corporate process repository. This is a common use case in mergers/ acquisitions, or in planning process resilience (looking for replicate services) (Dijkman et al. 2011).

Although BPMN has no explicit constraints about what specific resources should be modelled by a pool, and what by a lane, it is largely advised to allocate a pool for every Gaia role. Following the BPMN conventions, communications among roles should therefore be modelled as message flows. The Liveness2XPDL tool already provides this functionality; however, this (generally recommended) approach has a disadvantage: Due to software products' limitations, message flows can not be simulated (Freitas & Pereira 2015). Actually, a BPMN model that includes message flows can be simulated, yet message flows will be ignored. There are two workarounds one could employ:

1 Simulate cases one-by-one and handle manually message flows, as is the case of the Liveness2XPDL tool (Mitakidis et al. 2015). This approach has obvious limitations when many simulation runs must be performed

2 Break the BPMN best practices of messages exchange. This can be achieved by replacing pools by lanes and message flows by sequence flows (and by attaching the proper gateways).

Thus, our goal in extending the Liveness2XPDL tool was twofold, on one hand to directly export native BPMN 2.0 models and on the other hand to allow the analyst to select which of the two cases above should apply to his model.

The New Transformation Algorithm

The transformation algorithm is a recursive algorithm that takes the liveness formula expression elements (right hand side of the formula) from left to right and applies the templates shown in Figure 1, gradually building the BPMN process. For applying templates, keep in mind that the control flows from left to right, thus, if a template follows another, then it is connected to its rightmost element.

The BPMN metamodel that we used for automating the transformation process is the one defined by Camunda (https://camunda.org/) version 7.4. Camunda is an open source organization that works on providing BPMN descriptions to the community. One problem that was challenging in previous work was that different tools implement different BPMN XML according to their owner's needs and thus the interchange of our

produced BPMN models between different tools was prevented. Our BPMN was imported successfully both in Signavio academic and ADONIS:Community Edition 3.0 tools that allow for simulating BPMN models.

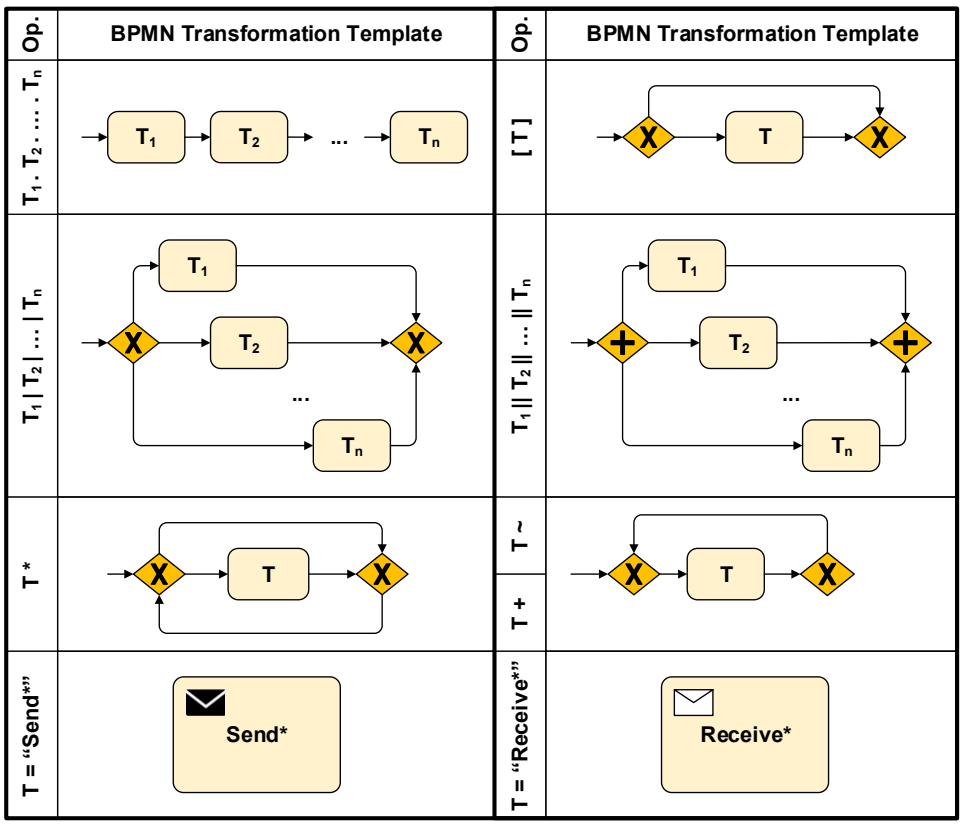

Figure 1. The BPMN 2.0 transformation templates for the liveness formula operators and task names

The ~ and + liveness formula operators have the same template to accommodate the need for the resulting model to pass the Proper Completion test, which demands that each workflow ends with an end event (Van der Aalst 1998). The XOR gateway can be adjusted to always return the flow to the task T. Moreover, in the second version of Gaia there is a case where the authors allow the indefinite operator to be followed by a sequential activity (Zambonelli et al. 2003).

The theoretical properties of the algorithm are identical to the Liveness2XPDL. It can be easily proved that the transformation is correct using induction and the assumption that if we have a correct BPMN model and replace a task with a correct BPMN fragment, or a well-structured fragment the resulting model is correct (González-Ferrer et al. 2013). The templates are all correct BPMN diagrams (well structured fragments) if they have a start event on their left and a sequence flow to an end event on their right, as every task is on a path from the start event to the end event. Then, for each of these valid models we can easily assert that if we take a random template and replace a task of the model with it

then, again, the model is correct. Then, we hypothesize that after k insertions the model is correct and we insert a new random template. Then, we show that the resulting model is correct. Regarding complexity, we have the case of a recursive function call inside a for loop, indicating that of $O(n^2)$, where n is the number of tasks present in the liveness formulas. A pre-processing function prevents the algorithm from hanging due to an infinite loop in the case that there would be circular references to the left hand side term of a formula from its right hand side expression (or from subsequent formulas) by informing the user for this error.

After transforming the liveness formulas of the roles to individual BPMN models, the tool asks the user to indicate the message passing points (connect sending tasks to receiving tasks).

The reader can see the integration of two roles (1 and 2) using the message passing approach in Figure 2. Whenever a user defines a pair of message sending/receiving tasks the message flow is created. The *Send* task of *Role 1* can be preceded and succeeded by any kind of tasks. The same for the *Receive* task of *Role 2*.

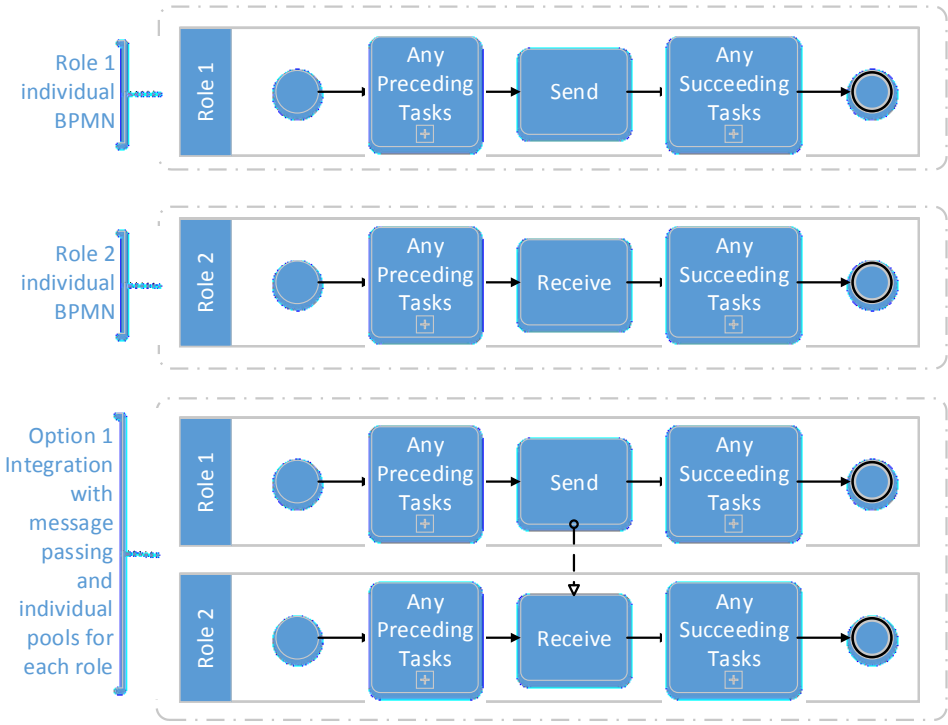

Figure 2. Individual roles integration with message passing

Figure 3 shows the case where we have sequence flows and the different participating roles' pools are integrated in a single pool, each becoming a lane. In this case only the scenario triggering role can have preceding tasks to its *Send* task. The role(s) with *Receive* tasks are not allowed to have preceding tasks. This is a limitation stemming from the fact that in this case (sequence flow) we can only have one *start event*. We retain the

message sending and receiving tasks because in the simulations they will need to capture the network of message transport service delays.

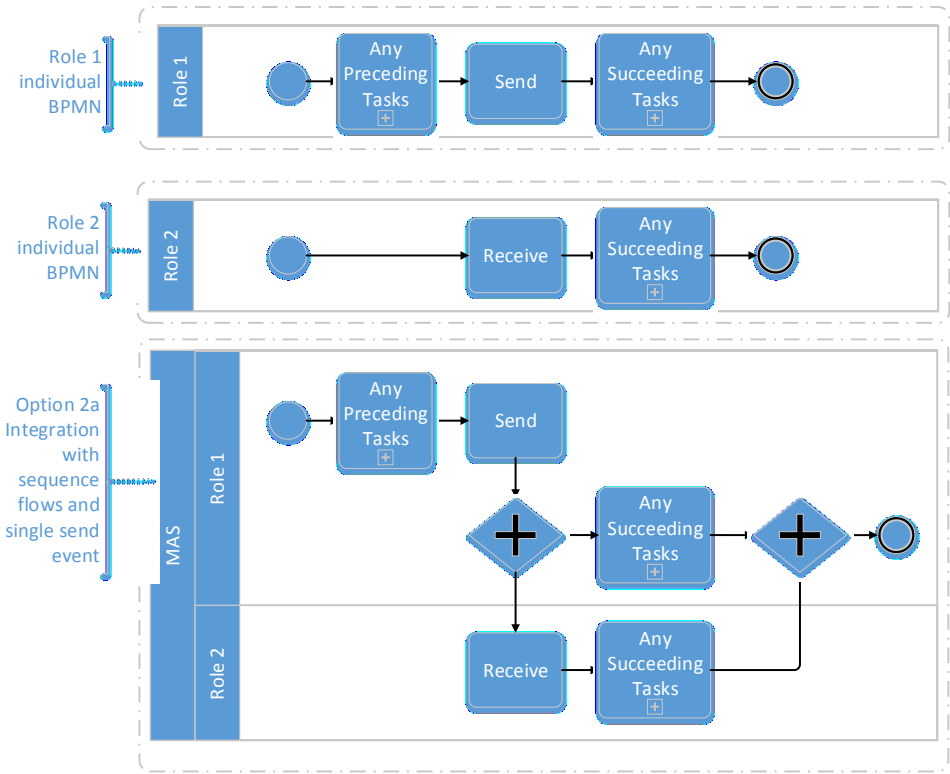

Figure 3. Individual roles integration with sequence flows, the case of a simple message event

Likewise, in Figure 4, in the case of transformation using sequence flows and the exchange of messages the message receiving role cannot have post processing tasks after replying to the triggering role.

ASEME IDE Integration

The ASEME IDE has been built using the popular Eclipse Rich Client Platform (RCP) architecture (http://wiki.eclipse.org/Rich_Client_Platform). The ASEME dashboard view is the plug-in that aggregates the diverse ASEME model views and transformation programs, guiding the developer at the same time from requirements elicitation to code generation. Code generation is supported for the popular JADE platform as well as the C++ language (supporting a generic blackboard architecture for communication between agents as well as and external environments). In Figure 5 the ASEME dashboard view is presented and the reader can notice that there is a main path (indicated by solid lines) and optional paths (indicated by dashed lines).

To integrate the Liveness2XPDL tool to the dashboard we had to turn the application to an Eclipse RCP Plug-in, registering it to the framework.

The tool was updated to "know" the location of the active user project so that the current SRM model that the user is working can be automatically imported to the tool and the XPDL or BPMN exported models can be saved inside the active project folder.

Moreover, the tool's external dependencies (such as the camunda metamodel) had to be included in the manifest of the new plug-in.

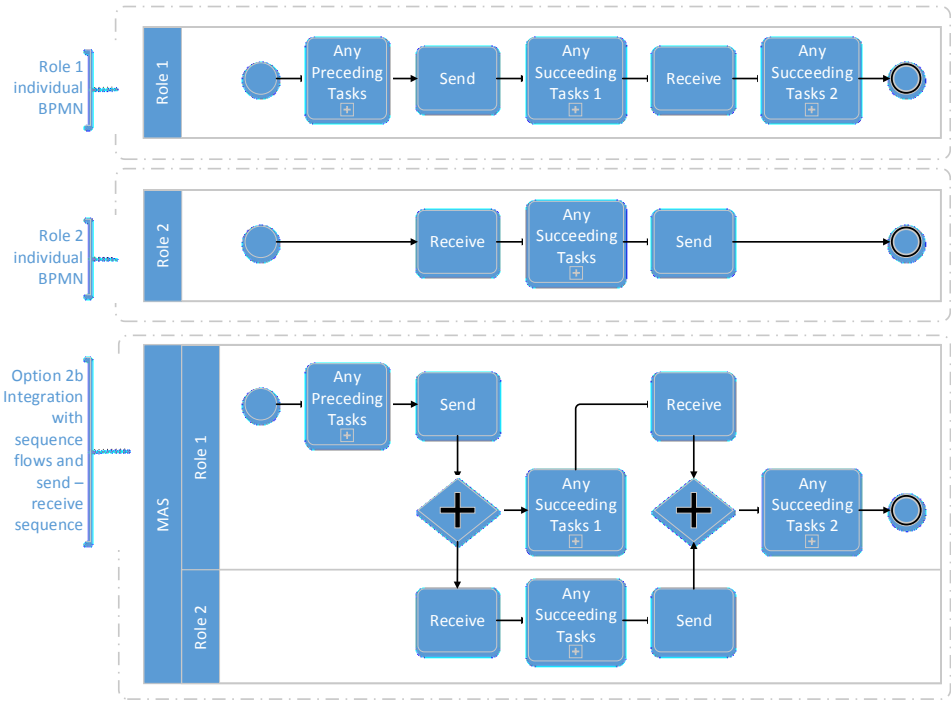

Figure 4. Individual roles integration with sequence flows, the case of message exchanging

A case study

The case study is similar to the ones used by (Delias & Spanoudakis 2010) and (Mitakidis et al. 2015), thus an interested reader can also follow the added value of this work. This case study will use the ASEME IDE form requirements elicitation to the simulation of the analysis phase models.

The first model is the System-Actors Goals model (SAG). There the developer defines the business actors in the application domain and their goals. We consider a brokering scheme where a personal assistant agent on a lightweight device (e.g., a smart phone) requests services from a broker. The broker can service simple requests using a web service or more complex requests using a service provider agent.

Figure 6 shows the SAG model editor in the central window of the IDE with the three identified roles and the goal of the PA to get a service (*GetService* for which it is dependent on the broker) and the goal of the broker to use services of the service provider. At the bottom of the figure the reader can see the dashboard, in which the SAG box has now a model (named *sp_model.sag*). On the left side of the figure at the top the

reder can see the project explorer where the files are all visible and also expandable in terms of their model elements (see the *SAGmodel* expanded from the *my_model.sag* file). The properties window at the bottom left shows the properties of the selected model element. At the time of the screenshot the *GetService* goal was selected and in its properties the reader can see the non-functional requirement for responding within 10 seconds.

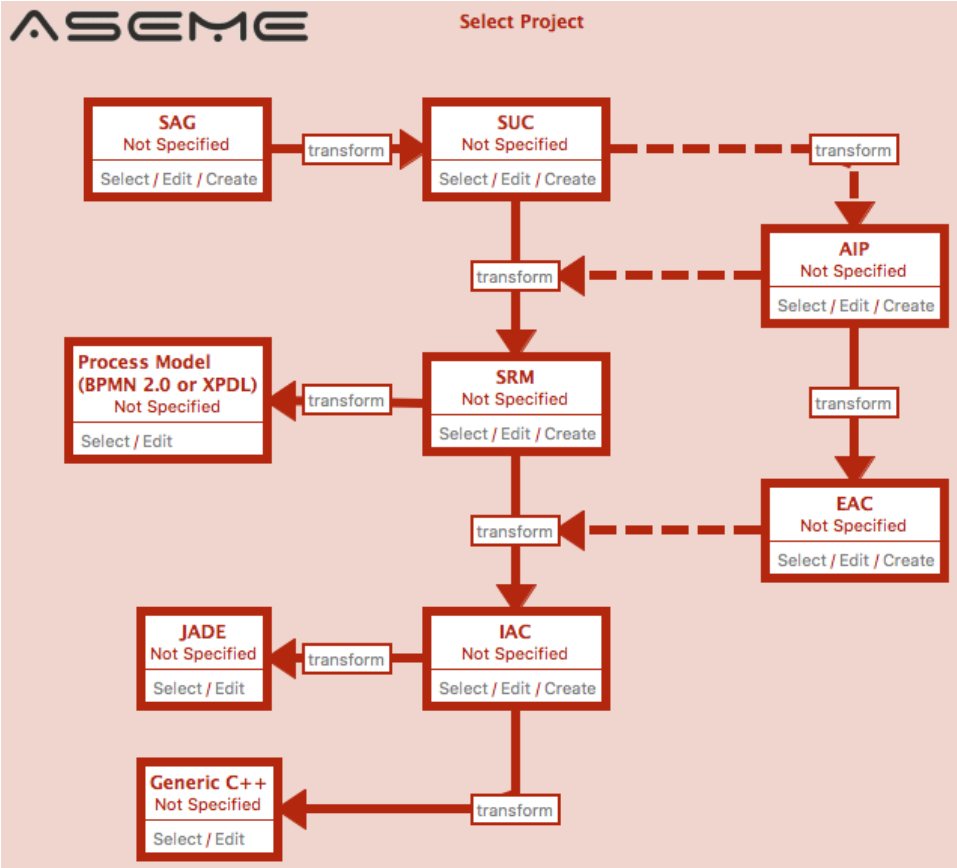

Figure 5. The ASEME dashboard view

After defining the goals and their functional and non-functional requirements the developer clicks the link transform on the dashboard between the SAG and SUC boxes following the arrows that denote the process. The *my_model.suc* System Use Cases model with the respecting *my_model.suc_diagram* diagram are automatically generated and inserted in the project folder. The developer edits the diagram and introduces two abstract roles, the service requester and service provider in order to define a general protocol of interaction. See the task decomposition that took place at this time in Figure 7 using the *include* relationship, as the SUC model is similar to the UML use case diagram.

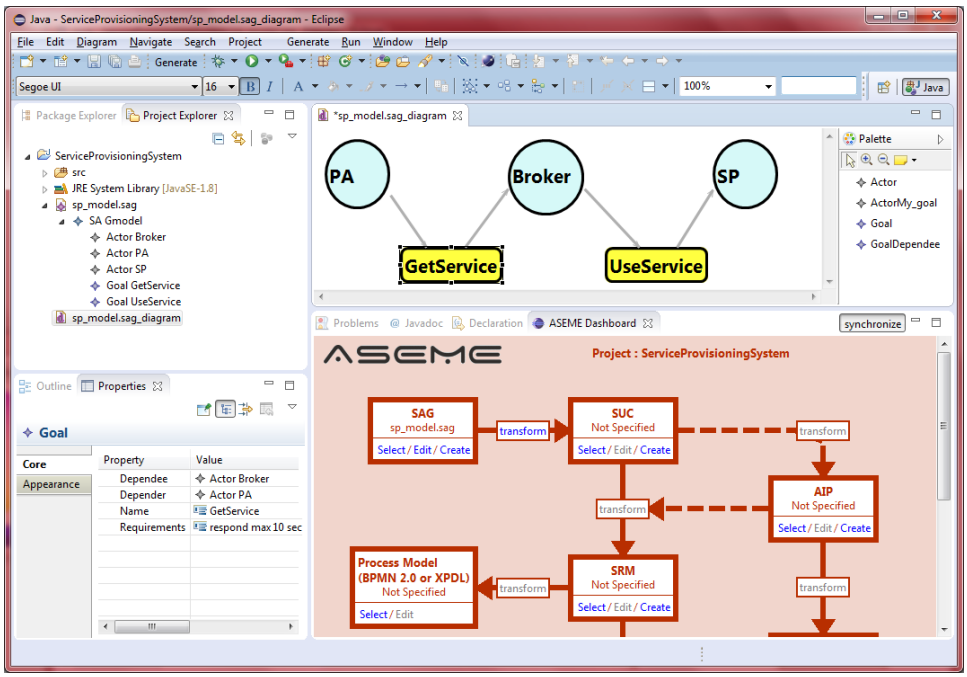

Figure 6. The SAG model editor

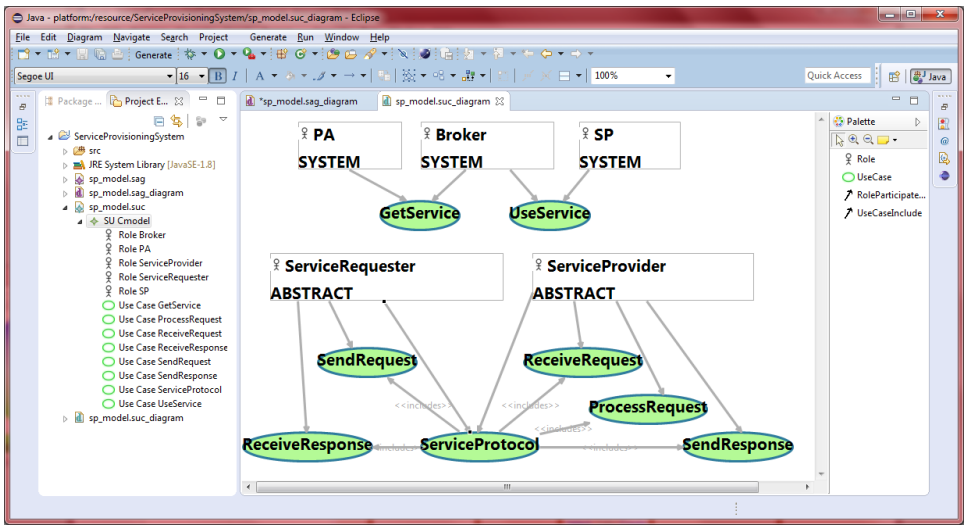

Figure 7. The SUC model editor

Next is the transformation from SUC to AIP. The automatically generated model is shown in Figure 8. There is a liveness property for each participant in the protocol and this is where they write their process. When initialized the model shows the use cases of each participant as tasks connected with unknown operators (*OP?*). The developer writes

the rules for engaging and expected outcomes in free text and then refines the formulas to depict the actual process followed by each participant. The refined model is shown in Figure 9.

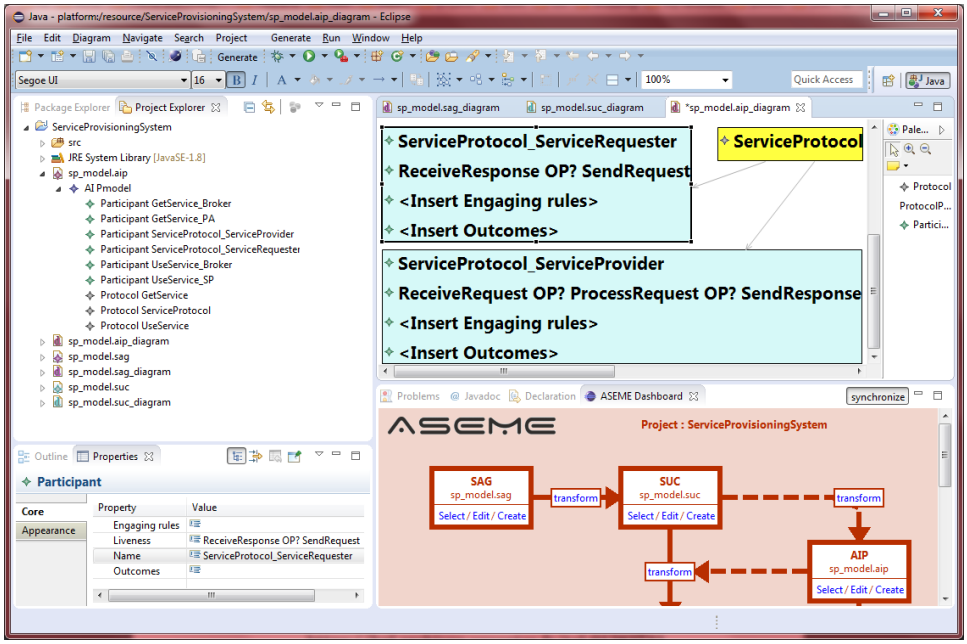

Figure 8. The automatically generated AIP model for the service protocol

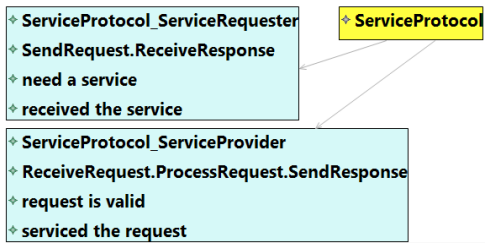

Figure 9. The refined AIP model for the service protocol

The next step is to define the concrete roles of the system participants. We will see how we can do that by re-using the abstract service protocol. We click the link *transformation* between the SUC and SRM boxes to initialise the System Roles Model of ASEME, which is similar to the Gaia roles model. ASEME supports another graphical view at this stage, the Functionality Graph. The *my_model.fg* file is shown in Figure 10. The developer has a monitor here for the system roles, their capabilities (capabilities originate from use cases that include others, they are decorated with the puzzle icon), activities (activities originate from use cases included by others, they are decorated with the process icon) and functionalities (functionalities correspond to activities and define the technology or algorithm related to each of them, they are decorated with the gear icon). See how roles can share capabilities, i.e. the PA and Broker share the capability to

participate in the service protocols as requesters (*ServiceProtocol_ServiceRequester* capability). The broker has a specific broker service process request capability that includes the activity *ServiceMatch* that uses a semantic service matching functionality.

Besides assigning capabilities to roles and functionalities to activities the developer can further refine the livenesses of the roles (including the liveness of the protocols participants whose capabilities it aggregates).

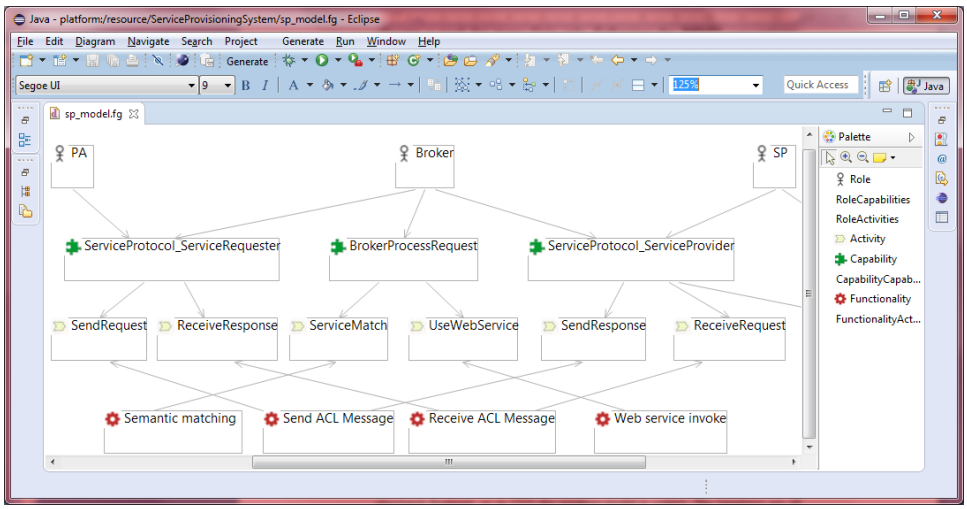

Figure 10. The Functionality Graph model for the system roles

When the roles are ready ASEME normally finished the analysis phase and transforms the liveness models to statecharts to go to the design phase. Using the SRM to Process model transformation (from the SRM to the Process model box in the dashboard, see Figure 5) the developer can use the tool that we integrated to the ASEME dashboard to transform the liveness models to process models.

In Figure 11, the developer has clicked the relevant *transform* link and the *Liveness to XPDL or BPMN Transformation Application* has been launched. Its main window is shown at the bottom right of the figure. It has automatically opened the SRM model and found the three system roles. The developer selects as many as he/she wants (all three in this case) and selects to transform them to BPMN from the *Transform* menu item.

The *Inter-role Messages Definition for bpmn* dialog has been launched (at the center top of the figure) and there the developer has selected to connect the *SendRequest* task of the broker to the *ReceiveRequest* task of the service provider role. At the bottom-left of the figure the reader can see the liveness of the PA role.

After the developer has defined all the message send and receiving pairs (the tool automatically suggests the *Receive** possible receiver tasks to a *Send** task, where * is any string) the BPMN model is outputted in XML format. Figure 12 shows a portion of the generated *mas.bpmn* file. The reader can notice the definition of a *collaboration* with the three *participant*s and two defined *messageFlows*.

Now the developer can import the BPMN file to any compatible tool for further analysis. For this use case we used the Signavio editor for graphically viewing the files and the BIMP simpulator for running some simulations.

The two possible files generated for this service requesting scenario by a personal assistant are depicted in Figure 13 and Figure 14. The latter was the one used for simulations.

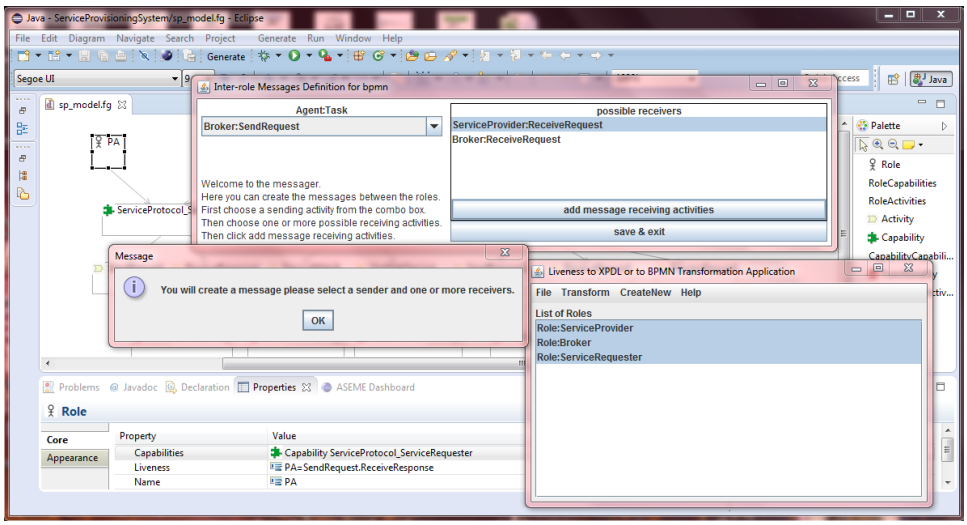

Figure 11. Transforming SRM to BPMN

Figure 12. An extract of the BPMN XML file

The importance of defining the performance of a system with multiple roles leads us to simulate the system's behavior. The results will guide the software engineer and the project manager to decide on the effectiveness of the system, because not only they review the response of each agent role in different simulation scenarios, but also they have the ability to distinguish clearly flow model hazards, such as deadlocks or bottlenecks.

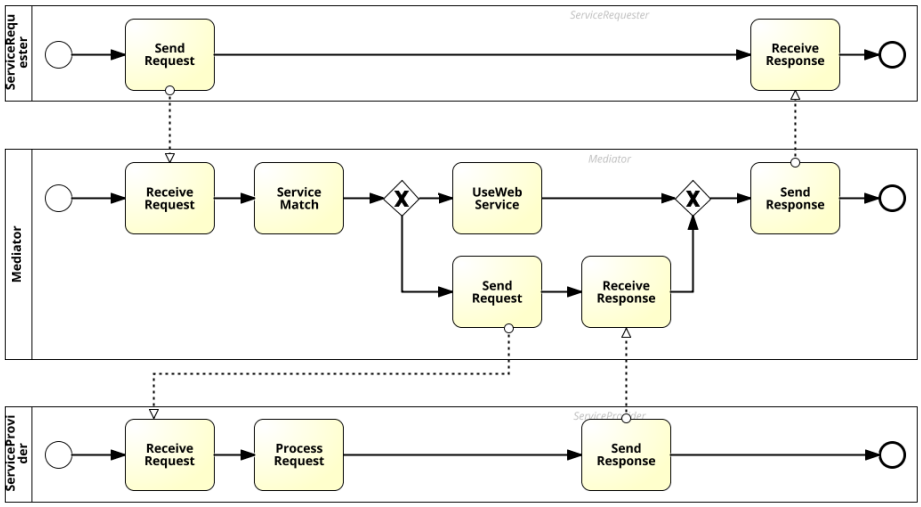

Figure 13. The messages-based BPMN model

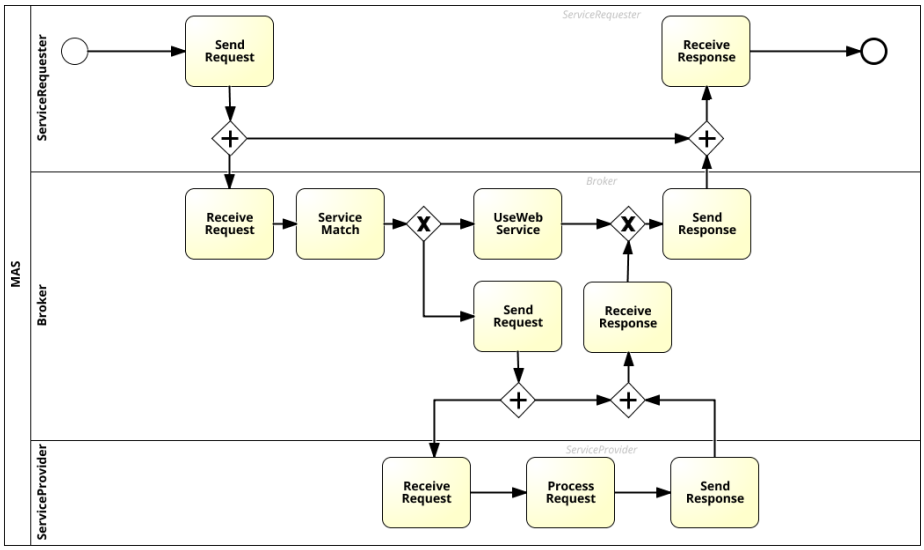

Figure 14. The sequence flow-based BPMN model

The Signavio academic online tool offers validation of the generated BPMN model, syntax check and simulation capability. The BIMP simulator helps the user in defining a complete experiment with the following attributes:

- Number of simulation instances
- The time between two simulation instances

- The duration of the simulation
- The number of each role instances
- The duration of the task (can be fixed or follow a distribution such as normal or exponential)
- The percentage weight of different paths in XOR

The setup of our experiment is shown in Table 1. These figures are extracted by execution times of prototype algorithms, measurements of network speeds and data access speed. Each scenario run for 10,000 service request and the XOR gateway was set at an equal probability to follow either path.

The results of our experiments are depicted in Figure 15. We decided to display a series of figures to show the performance gain in the average cycle time and the max cycle time and the utilization of the participant roles as we change the number of role instances. The different systems have acceptable average cycle times, however the interested reader will notice that the system can scale to servicing requests in an acceptable way (respond within 10 seconds, see the relevant requirement in Figure 6) when requests come at an average of one every two seconds. To have that quality of service for the worst case we need three brokers and two service provider instances. Thus, using this approach the developer can have a clear view on the number of agent types he/she will need to instantiate during system deployment.

TABLE 1. The setup of the trials

Task	**Distribution**	**Mean**	**Standard Deviation**	Performer
SendRequesttobroker	Normal	0,024	0,063	PA
ReceiveResponsefromBroker	Normal	0	0	PA
ReceiveRequestfromPA	Normal	0,002	0,002	Broker
ServiceMatch	Normal	0,254	0,112	Broker
UseWebService	Normal	2,639	1,113	Broker
SendResponsetoPA	Normal	0,007	0,006	Broker
SendRequesttoSP	Normal	0,007	0,006	Broker
ReceiveResponsefromSP	Normal	0,024	0,063	Broker
ReceiveRequestfromBroker	Normal	0,024	0,063	ServiceProvider
ProcessRequest	Normal	2,92	1,3	ServiceProvider
SendResponsetoBroker	Normal	0,007	0,006	ServiceProvider

Discussion and Conclusion

This paper shows an original result for an AOSE methodology, i.e. the methodological approach to address the issues of validating system properties even from the analysis phase. Such properties are the instances of agents needed for a desired level for the system quality of service, an indication on how the system can scale and the

identification of bottlenecks, i.e. situations where a resource of the system is slowing it down.

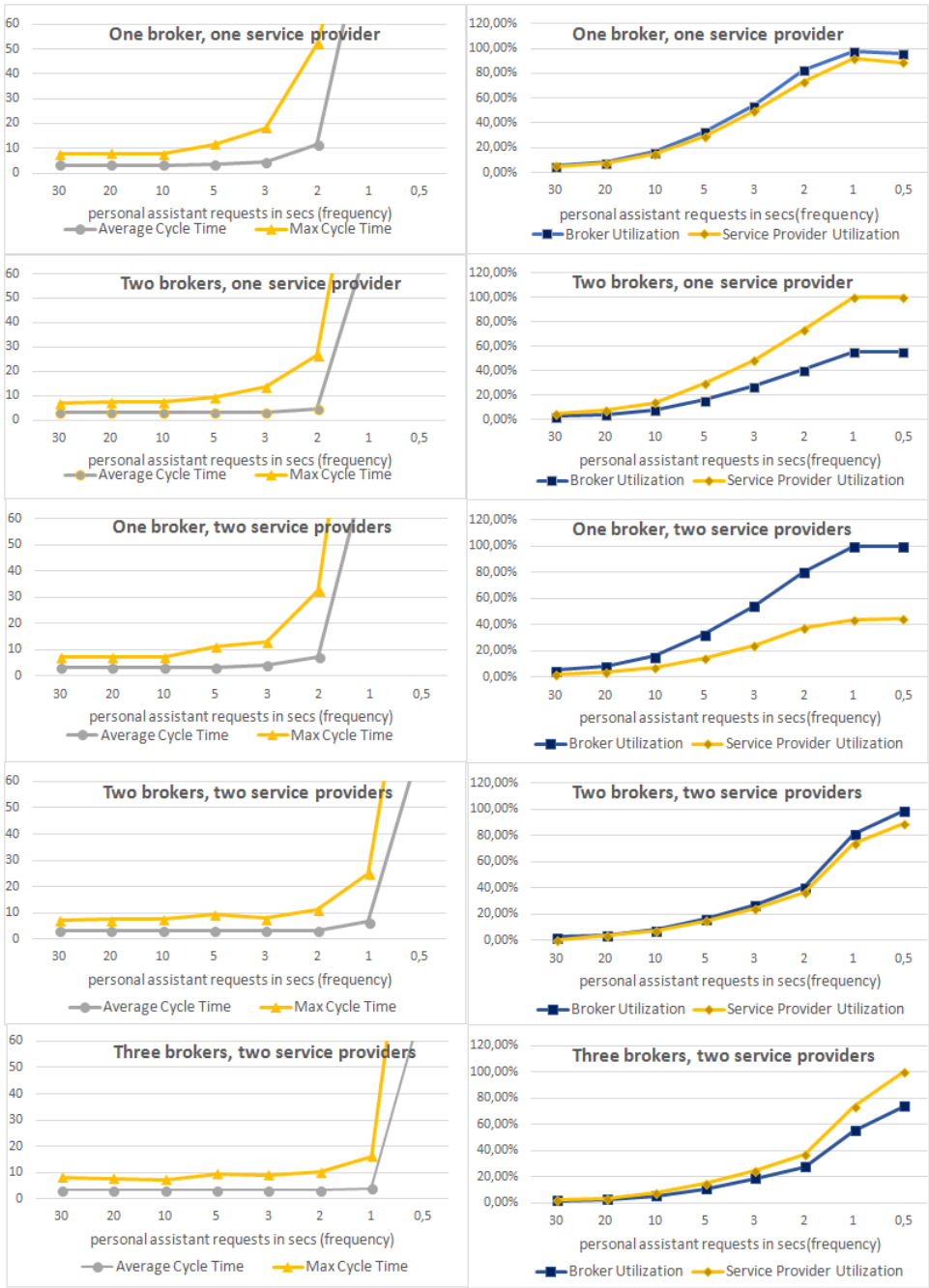

Figure 15. The results of the simulations

To achieve this result we utilised and extended a recently published result, i.e. the Liveness2XPDL tool. We extended it in order to export native BPMN models and allow for transforming messages exchange to sequence flows in the process model. Although there are some limitations the results that we obtained outperform previous works. For example we can now simulate MAS automatically instead of manual execution of models. Moreover, we integrated it in the ASEME methodology's IDE and showed how it can be seamlessly exploited by modelers that use ASEME.

Of course, the AOSE community has been studying and using business process models for quite some time, see, e.g., a work for improving a process model representing the behavior of agents (Szimanski et al. 2013), or another for proposing a method for transforming BPMN models to agent-oriented models in the well-known Prometheus methodology (Dam & Ghose 2012), or even another that provides a mapping of BPMN diagrams to a normalized form checking for certain structural properties, while the normalized form can itself be transformed to a petri-net that allows for further semantic analysis (Endert et al. 2007).

All these works can now be aligned with an AOSE methodology and be integrated in an overall development process, i.e. that of ASEME. In this way, a practitioner can transform the process model outputted from ASEME to a system specification using the Prometheus methodology notation (Dam & Ghose 2012) and continue using that methodology. Another might be interested in checking certain structural properties of the process model (Endert et al. 2007).

Our future work is about further extending the tool to allow for more diverse scenarios. An interesting direction is to couple simulation tools with testing. Thus, the design can be proved or even improved as design models and functionality capabilities are exercised long before the final software product is made available (Pries & Quigley 2010). Another interesting path is that of the modern field of Human-Agent Collectives (Jennings et al. 2014), based on BPMN.

References

Van der Aalst, W.M.P., 1998. The application of petri nets to workflow management. *Journal of Circuits, Systems and Computers*, 8(1), pp.21–66.

Bresciani, P. et al., 2004. Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3), pp.203–236.

Cernuzzi, L. & Zambonelli, F., 2009. Gaia4E: A Tool Supporting the Design of MAS using Gaia. In *Proceedings of the 11th International Conference on Enterprise Information Systems (ICEIS 2009), Volume SAIC, Milan, Italy, May 6-10*. pp. 82–88.

Dam, H.K. & Ghose, A., 2012. Agent-Based Development for Business Processes. In N. Desai, A. Liu, & M. Winikoff, eds. *Principles and Practice of Multi-Agent Systems*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 387–393.

Delias, P. & Spanoudakis, N., 2010. Simulating Multi-agent System Designs Using Business Process Modeling. In *8th European Workshop on Multi-Agent Systems (EUMAS 2010)*. Paris, France.

DeLoach, S.A. & Garcia-Ojeda, J.C., 2010. O-MaSE: a customisable approach to designing and building complex, adaptive multi-agent systems. *International Journal of Agent-Oriented Software Engineering*, 4(3), pp.244–280.

Dijkman, R. et al., 2011. Similarity of business process models: Metrics and evaluation. *Information Systems*, 36(2), pp.498–516. Available at: http://www.sciencedirect.com/science/article/pii/S0306437910001006.

Dumas, M. et al., 2013. *Fundamentals of Business Process Management*, Berlin, Heidelberg: Springer Berlin Heidelberg.

Endert, H. et al., 2007. Towards a Mapping from BPMN to Agents. In J. Huang et al., eds. *Service-Oriented Computing: Agents, Semantics, and Engineering*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg.

Freitas, A.P. & Pereira, J.L.M., 2015. Process simulation support in BPM tools: The case of BPMN. Available at: http://repositorium.sdum.uminho.pt/handle/1822/39192.

González-Ferrer, A., Fernández-Olivares, J. & Castillo, L., 2013. From business process models to hierarchical task network planning domains. *The Knowledge Engineering Review*, 28(02), pp.175–193. Available at: http://journals.cambridge.org/abstract_S0269888912000410.

Ivanov, S., Kalenkova, A. & WMP Wil Aalst, van der, 2015. BPMNDiffViz : a tool for BPMN models comparison. Available at: http://www.narcis.nl/publication/RecordID/oai:library.tue.nl:797560.

Jennings, N.R. et al., 2014. Human-Agent Collectives. *Communications of the ACM*, 57(12), pp.80–88.

Juan, T., Pearce, A. & Sterling, L., 2002. ROADMAP: extending the gaia methodology for complex open systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems part 1 - AAMAS '02*. New York, New York, USA: ACM Press, pp. 3–10.

Mitakidis, N., Delias, P. & Spanoudakis, N., 2015. Validating Requirements Using Gaia Roles Models. In M. Baldoni, L. Baresi, & M. Dastani, eds. *Engineering Multi-Agent Systems, Third International Workshop, EMAS 2015, Istanbul, Turkey, May 5, 2015, Revised, Selected, and Invited Papers*. Lecture Notes in Computer Science. Cham: Springer International Publishing. Available at: http://link.springer.com/10.1007/978-3-319-26184-3.

Onggo, B.S.S., 2012. BPMN pattern for agent-based simulation model representation. In *Proceedings Title: Proceedings of the 2012 Winter Simulation Conference (WSC)*. IEEE, pp. 1–10.

Pascalau, E., Giurca, A. & Wagner, G., 2009. Validating Auction Business Processes using Agent-based Simulations. In *Proceedings of 2nd International Conference on Business Process and Services Computing (BPSC2009), March 23-24, Leipzig, Germany*.

Pries, K.H. & Quigley, J.M., 2010. *Testing Complex and Embedded Systems*, CRC Press. Available at: http://www.amazon.com/Testing-Complex-Embedded-Systems-Pries/dp/1439821402.

Rana, O.F. & Stout, K., 2000. What is scalability in multi-agent systems? In *International Conference on Autonomous Agents*. Barcelona, Spain: ACM, pp. 56–63. Available at: http://portal.acm.org/citation.cfm?id=336595.337033 [Accessed October 2, 2010].

Smith, C.U. & Williams, L.G., 2002. *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*, Addison-Wesley.

Spanoudakis, N. & Moraitis, P., 2011. Using ASEME Methodology for Model-Driven Agent Systems Development. In D. Weyns & M.-P. Gleizes, eds. *Agent-Oriented Software Engineering XI*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, pp. 106–127.

Szimanski, F. et al., 2013. Improving Business Process Models with Agent-Based Simulation and

Process Mining. In S. Nurcan et al., eds. *Enterprise, Business-Process and Information Systems Modeling*. Lecture Notes in Business Information Processing. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 124–138.

Wooldridge, M., Jennings, N.R. & Kinny, D., 2000. The Gaia Methodology for Agent-Oriented Analysis and Design. *Autonomous Agents and Multi-Agent Systems*, 3(3), pp.285–312.

Zambonelli, F., Jennings, N.R. & Wooldridge, M., 2003. Developing multiagent systems: The Gaia methodology. *ACM Transactions on Software Engineering and Methodology*, 12(3), pp.317–370.