Integrating Material Flow Simulation Tools in a Service-oriented Industrial Context

Jan Fischer, Birgit Obst, Benjamin Lee
Corporate Technology, Automation and Simulation
Siemens AG
Otto-Hahn-Ring 6, 81739 Munich, Germany
jan.fischer@siemens.com, birgit.obst@siemens.com, benjamin.lee@siemens.com

Abstract — **In this paper a generic simulation environment is introduced that allows the integration of commercial and research-based simulation tools in a service-oriented manufacturing environment, such as in Industrie 4.0 factories, to support in flexible reconfiguration and scheduling decisions. For this purpose, the simulation environment is connected to a generic middleware and processes parameterized simulation requests via automated model generation and execution, based on current shop floor conditions. Those simulation requests are formulated in a standardized description language for production systems, in this case a derivative of AutomationML (AML) that was extended by simulation specific data sets. Therefore, the Simulation Environment comprises different tool- and non-tool-specific functionalities and components that allow the integration of simulation tools into industrial IT platform environments.**

Keywords—Discrete Event Simulation; Decision Support; Simulation as a Service; AutomationML; Automatic Model Generation; Platform Integration;

I. INTRODUCTION

Manufacturing companies worldwide are facing enormous pressure to provide higher product variety and more customized products. This also requires companies to make their factories more robust, so that they can effectively manage and quickly respond to change requests or disturbances such as delays, shortages, or resource breakdowns [1]. One promising path forward is the development of agile manufacturing systems that are dynamically reconfigurable and evolvable, enabling self organization and adaption along the system life cycle, facing the challenges of continuously and rapidly changing market conditions and increasingly smaller lot sizes and shorter lead-time and time-to-market requirements [2]. In this context, especially simulation tools are well established means that deliver the necessary functions for the dynamic evaluation of a production system's performance, e.g. in regards to key performance indicator (KPI) assessment and decision making [3]. One of the core ideas of Industrie 4.0 centers on the concept of service-oriented architectures, meaning that required functionalities for the operation and optimization of production systems are provided as services (compare e.g. [4]). Consequently, as simulation based functionalities play a core role in the optimization of operations they should be provided as services as well. However, so far, most commercially available and research based simulation tools require comparably large amounts of effort for model set-up, initialization, parameterization, and adaption especially when considering dynamically changing production conditions. Hence, in order to allow the utilization of conventional simulation tools in the highly flexible and service based context of Industrie 4.0, traditional simulation tools need to be enhanced by additional functionalities such as automatic simulation model generation, automated simulation parameterization, and data conversion.

In addition, the application should be executable by non simulation experts, integrated in a service-oriented architecture, enabling all interface activities for input information pre processing, control logic definition for calculation and optimization, and result access for further treatment [5]. In this paper, a generic *Simulation Environment* (SE) (see Fig. 1) is presented for model integration, definition of simulation execution workflows, and the communication with a middleware for data exchange related to the work within the EU funded project PERFoRM (Production harmonized Reconfiguration of Flexible Robots and Machinery).

II. STATE OF RESEARCH

Against the background described in the introduction, it is obvious that functionalities provided by classical production simulation tools need to be addressed in a standardized form, i.e. via a standardized API and based on standardized data models, if they shall be utilized in a service-oriented architecture. Data model standards, at least on a syntactic level, are provided by description languages as e.g. used for ontologies (e.g. Web-Ontology-Language) or systems modeling (e.g. SysML). For the engineering of manufacturing systems, the meta-standard Automation Markup Language (AutomationML or AML) has gained wide popularity by providing a framework for exchanging data during the

978-1-5386-0837-1/17/$31.00 ©2017 IEEE

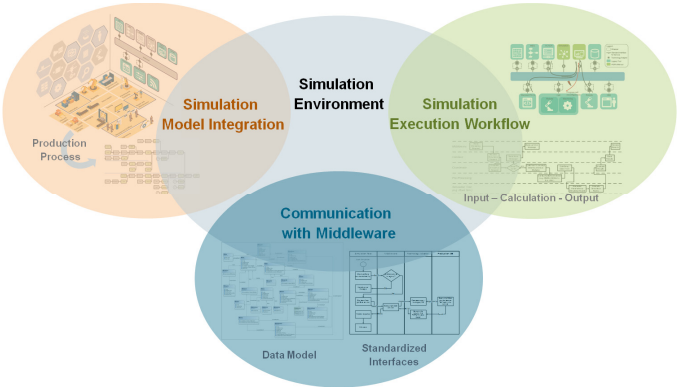

Fig. 1. Functional modules for SE

engineering process of production systems. [6] Consequently, AML is used as the base format for file exchange in this paper. Nevertheless, all of these standards have shortcomings when using them for exchanging data for simulation services as usually no simulation parameters are formally described in these models. Hence, in a first step, these simulation related parameters, such as simulation execution type, simulation time, size of time steps, or termination criteria, need to be integrated in the exchanged data models.

In a second step, an approach is required that translates this standardized and descriptive model of a production system into a formalized and tool specific simulation task, encapsulating the simulation tool based functions as parameterized services – "Simulation as a service". Such approaches can include the automatic generation and execution of simulation models. In this regard, some approaches in current literature exist:

The industrial solution *witness.io* from *Lanner Group Limited* offers a service based simulation in the cloud [7], however, only allows specific simulation model types (WITNESS and BPMN/BPSim) and the utilization of their related simulation engines. The ARUM Project [8] presented an approach for using simulation to guide planning in production environments in which data is retrieved from an intelligent Enterprise Service Bus and then translated via an ontological service. A hybrid approach evaluates strategic level planning activities and then operational level organization of agents, but restricts the analysis to the formalism of mathematical programming.

Fischer et al. [9] and Steimer et al. [10] use SysML based descriptions of production systems in order to generate material flow simulations in *Tecnomatix Plant Simulation*. Although their approach allows the automatic generation of simulations it is limited to a proprietary interpretation of SysML models and only partially includes simulation related parameters. A service interface is not provided. Baudisch et al. also use AML for the partially automatic generation of 3D-kinematic models of production lines in order to conduct simulations in the simulation tool *Process Simulate*. [11] Other approaches generate simulation models for the purpose of virtual commissioning of production lines (e.g. [12], [13]) consequently focusing on automatically controlled production machines or production lines. For the automatic generation they rely on engineering and planning objects/data and/or preconfigured simulation model libraries. In addition, general purpose simulation workflow tools exist, such as *Phoenix Integration ModelCenter [14]*, that allow the automatic and parameterized execution of simulation steps. However, they aim towards the engineering phase of systems, and do not target integration in an operative manufacturing environment.

Consequently, it can be summarized that none of the investigated approaches provides a generic and standards oriented approach for integrating simulations in an operational, service-oriented manufacturing environment. In the following, the development of the generic SE is presented.

III. Simulation Environment

In this section, the development of a concept architecture for the SE is described. First, technical requirements for the SE are identified to provide a concise basis for the development. Then, major components of the concept are reviewed and the implementation of the concept is presented.

A. Context

1) Requirements

The first step in the development of the SE was the elicitation of requirements. The goals for the SE declared in the introduction were reviewed, and then concretized into a set of technical requirements defining constraints on the SE. The major requirements on the SE include:

- The SE must be able to execute a simulation of the production system, and support both discrete event and agent-based simulation paradigms.
- The SE must be able to automatically generate a model of the production system to be simulated.
- The SE must be able to execute a probabilistic simulation of the production system.
- The SE must be able to be triggered manually via user input, as well as automatically via external trigger.
- The SE must be able to accept a planned production schedule, maintenance schedule, control logics for machinery or products, facility topology, and the current status of entities within the facility.
- The SE must be able to calculate meaningful KPIs at the machine, product, and facility levels.
- The SE must be able to start simulation runs based on standardized simulation requests, formulated in an extended AML format. It further needs to present a SOA-compatible service interface for making its functions available as services to the public.

2) Architectural Context

Fig. 2 provides an informational view of the proposed architecture context for the SE, illustrating the separation of the main functionalities of the SE into sub-modules as well as the flow of various information types into and out of the SE: the SE contains a *Simulation Tool* such as Tecnomatix Plant Simulation or AnyLogic that is supported by various

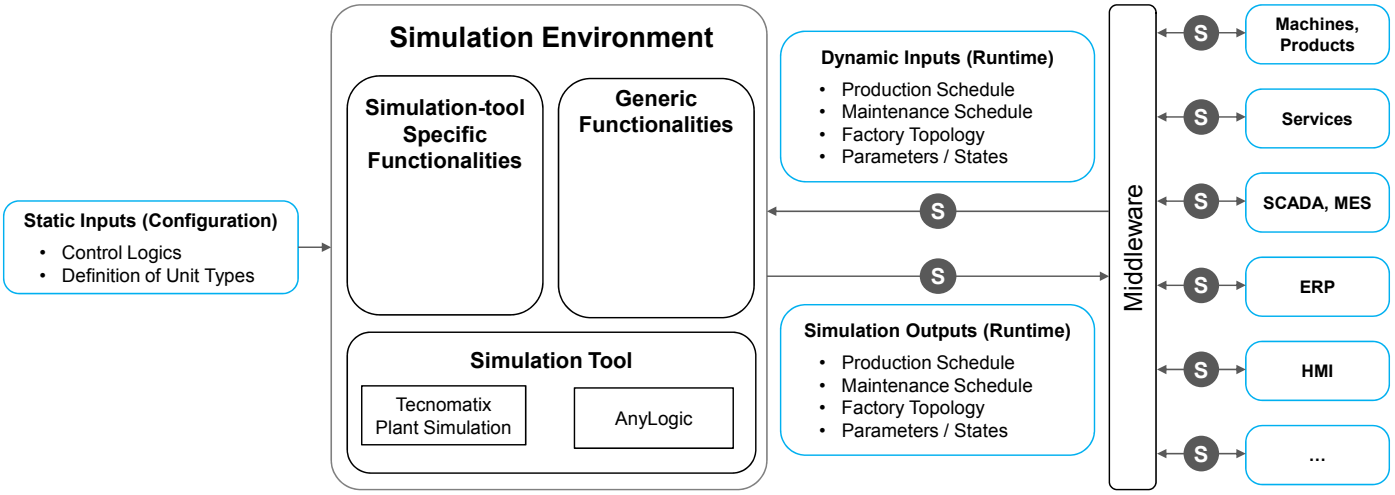

Fig. 2. Context of SE (implementation view).

capabilities. The goal of such an architecture is to capture as much functionality as possible into a set of *Generic Functionalities* that would support a multitude of various simulation tools in an independent manner. However, as it is rarely possible to capture all aspects necessary for a simulation in a completely tool-independent manner, a set of *Simulation-tool Specific Functionalities* are also included.

They are required by the SE in order to correctly process the desired set of information, passing the necessary information to the Simulation Tool at the necessary time, and then processing the results of simulations. The incoming information can be separated into a set of *Static Inputs* which are made available at the time of configuration of the SE, and a set of *Dynamic Inputs*, which are only made available at runtime. The *Static Inputs* describe all information that is necessary for the Simulation Tool, but does not vary with time. For example, this would include the definition of unit types describing the constitutions of the individual entities in the facility, as well as the set of control logics which could be utilized by the entities. On the other hand, *Dynamic Inputs* describe the up-to-date status of the facility being simulated, and may, for example, include planned production or maintenance schedules, or the current status or topology of machinery, products, or workers in the facility. The goal of the SE is ultimately to provide an evaluation of some context. This evaluation is encoded into a set of *Simulation Outputs*, which become available upon termination of a simulation. These outputs could contain, for example, information about optimal configurations, scheduling, and key performance indicators, as well as meta-information about the processes executed.

B. Concept

1) Architecture and Components

In order to allow the execution of the introduced SE, a detailed implementation concept was developed. Fig. 3 shows an overview of the internal structure of the SE. It is based on various case-specific and generic components. These components can be clustered into five categories:

(i) An *interface to the middleware* that receives and sends service requests and processes the incoming and outgoing data. Via an API the functions/services of the SE are published to the middleware. The interface can be based on different programming techniques for service-oriented architectures like REST or SOAP. It is directly connected to the coordination module that then evaluates and further processes the incoming or outgoing service request or responses.

(ii) The *coordination module* interprets the service request that was received from the interface and calls the individual sub-components, services, or simulation tools of the SE (e.g. data-conversion, simulation-runs, pre- or post-processing …). The coordinator is thus the main component of the SE and also utilizes decision logics and workflow descriptions in order to connect, combine, and execute the components. In this context, workflows describe the internal sequence of components and simulation tools to be executed including their parameters (see also Fig. 4). Hence, the main task of the coordination module is to connect the incoming service requests with the service case specific workflows stored in the workflow library and then to execute it.

(iii) *Generic components and functionalities* provide services on a rather high level to the SE e.g. in order to post- or preprocess the incoming and outgoing data or to convert the generic data models for simulation requests into simulation tool specific data models (XML-conversion). In addition, a user interface can be provided in order to allow online or offline configuration of the SE or to show results or the current status of the simulation runs. Furthermore, a co-simulation master is provided for the parallel and synchronized simulation of various simulation runs, e.g. material flow simulations on production system level can be coupled with detailed physical simulations on process or machine level. One possibility to conduct co-simulation is described by the FMI/FMU standard [15].

(iv) *Tool-specific- and non-tool-specific libraries and components* provide various functionalities, as e.g. simulation model components in order to build a simulation model, such as processes, workstations, or conveyors. In addition, base models of simulation tools are stored here, so that use case

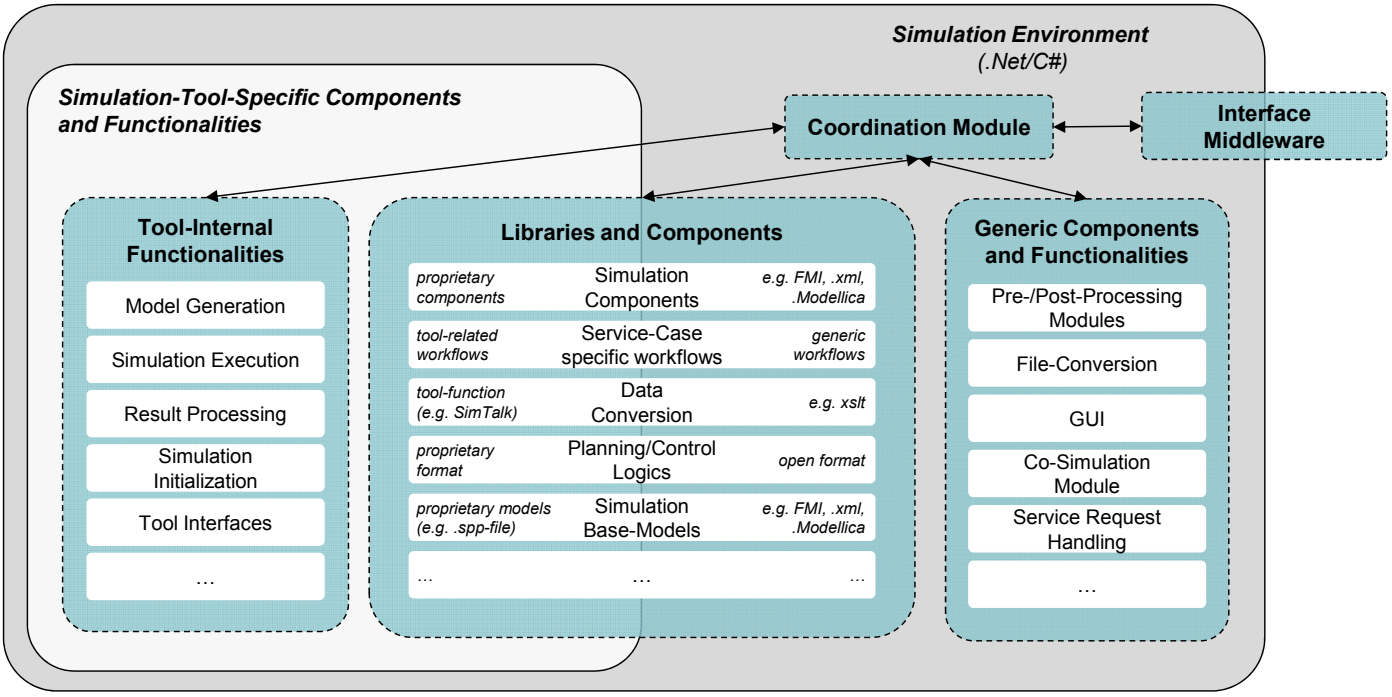

Fig. 3. Internal architecture of SE (implementation view).

specific simulation models can be automatically generated by adapting or parameterizing base models for specific use cases. The libraries are modeled, as far as possible, based on open standards in order to allow a reuse over several tools. Such tool independent and open simulation component descriptions are for example provided by the Modelica standard [16], albeit mainly for the purpose of 1D-multiphysics simulations and not for discrete event factory simulations as mostly intended in this case. However, as not every tool allows this open modeling, it is sometimes inevitable to use tool-specific modeling languages and file formats. Further libraries include various planning and control logics that specify the control of material flow in factories, as e.g. agent based negotiation mechanisms, either as executable base models, as interpretable source code, or as an encapsulated .dll.

Further libraries include data conversion specifications, e.g. in the form of xslt style sheets that define how the generic input files need to be transformed in order to be interpretable by the respective simulation tool that is utilized. Finally, one of the most important libraries is the use-case/service specific workflows library. In each workflow it is specified what sub-components/services and what simulation tools need to be executed in order to provide the output to a higher level service or API-function that is provided by the SE. Consequently, a workflow consists of a sequence of certain execution steps and their parameters that need to be conducted, e.g. 1. preprocessing 2. file conversion 3. simulation model generation ... 8. send results back to requesting service. Furthermore, case specific evaluations of these execution sequences can be contained e.g. in order to make the execution of a certain function dependable on the outcome of the result of a previous function, as e.g. post-processing may only be required if the simulation results are of a certain quality type. Consequently, workflows can be depicted as activity diagrams as described in the following section. They are stored in an open xml format that can be edited by a user via the GUI or can be added via the API.

(v) Finally, *(simulation-) tool internal functionalities* are provided mainly in order to conduct the actual simulation runs. Models are also generated inside the simulation tools if the respective functionality is provided by the tool. E.g. the software Plant Simulation allows the script based execution and generation of simulation models and thus provides the required functionality (see also section IV).

C. Exemplary Workflow

In addition to the structural view of the SE components, their behaviour can be visualized in activity diagrams that depict the logical and time-based interactions of the components. The components are described in these views as so-called swim-lanes. Fig. 4 visualizes an exemplary workflow where a simulation request is received from the middleware and forwarded via the interface to the coordination module. After a validity check of the request, the coordination module generates the required sub-calls and initiates the pre-processing modules, if required, and the simulation itself. Afterwards the required simulation model is updated or generated and the simulation is executed. The results are then sent back to the middleware via the standard interface. In addition to this example, several alternative workflows exist based on the respective simulation request and simulation tool, however their respective workflow will look similar to the one specified in Fig. 4. In the remainder of this section, each of these steps will be described:

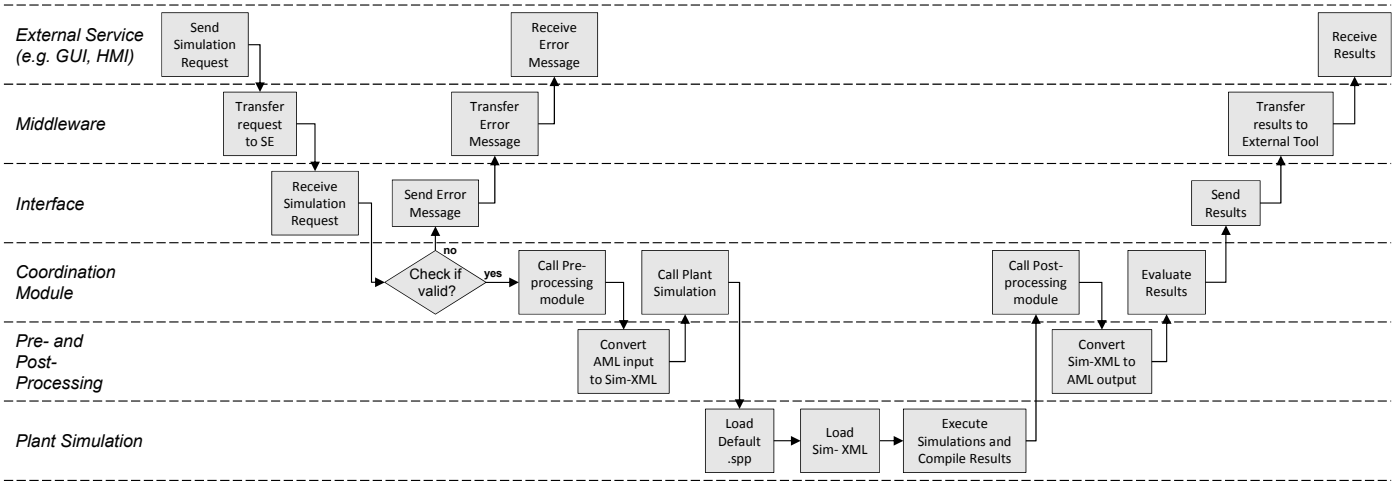

Fig. 4. Exemplary workflow of SE using the simulation tool Tecnomatix Plant Simulation.

In a pre-configuration step (i), that is not depicted here, a simulation expert is required to define *default simulation elements* that describe the entities that will be simulated within the factory context. This consists of defining the types of production equipment that will be utilized, the product types that can be produced, the processes that can be executed using this equipment on these products, the personnel that perform or oversee these processes, and the parameters that define them. Upon initial implementation, this step coincides with the definition of the AML description of the factory. If no legacy models exist, or if no special considerations or models are to be included, then this step requires no additional investment of work than that already required to model the factory components in the AML description of the factory. The simulation elements described above are then *stored in a model library*, such that they can be recalled upon request.

The actual operational workflow starts with a request for simulation (ii) that an external tool or user has made. To do this, the external tool compiles an AMLSimulationResult object that describes the factory state to be simulated, including any planned schedules (*Associated Schedules*) and configurations (*Associated Configurations*) to be evaluated, as well as a list of KPIs (*KPI-List*) by which they should be evaluated. It further contains a unique identifier (*ID*), a description (*Description*) and requested horizons for the simulation (*StartDate*, *EndDate*). Further details about the definition of the object can be found in [17]. The middleware then (iii) receives the request from the external tool, and passes this request onto the SE, which is identified as a registered service. The interface (iv) recognizes the request and passes the AMLSimulationResult object to the coordination module. The coordination module (v) tests whether the requested AMLSimulationResult object is valid. If the object does not contain enough information to fully specify a simulation, then the coordination module queries whether a fully defined default model in the model library exists. If not, then an error message is generated and sent to the interface. Otherwise, the valid AMLSimulationResult is sent to the Preprocessing Module where (vi) the valid AMLSimulationResult is translated into the simulation tool specific file format. Based on this file, the coordination module calls the (vii) simulation tool in order to start the actual simulation runs. As described above, simulation runs are parameterized and can contain several iterations. The simulation tool loads the based simulation model and parameterizes it with the Sim-xml. After the simulation, the results are compiled and via the coordination module the (viii) post-processing is initiated. Here, the tool specific results are transformed back into the middleware conform AMLSimulationResult format. Analogously to the workflow before, the results are then transferred back (ix) via the interface and the middleware to the initial caller of the simulation request.

IV. Application

The introduced simulation environment was prototypically implemented and tested by integrating *Tecnomatix Plant Simulation* as an exemplary simulation tool. Plant Simulation is a simulation tool that can perform material flow simulations of discrete event systems, mainly used for factories or production systems. It provides features that allow simulation in a dynamic and flexible manufacturing context as e.g. object oriented modelling, predefined simulation model components, configurable and customizable components via the internal scripting language *SimTalk*, or various file-, inter-process-, and database-interfaces (compare e.g. [18]). For this purpose,

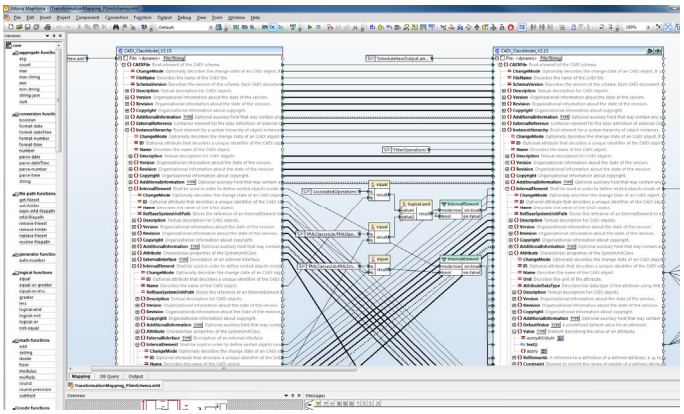

Fig. 5. Extraction of the mapping between the AML-file and the Plant Simulation tailored xml-file displayed in Altova MapForce.

```
For each Element in XML-Input-Table Do
    Create Simulation-Object of type Element.Type with name Element.Name
    For each Parameter in Sub-Table of Element
        Identify variable type (e.g. Boolean, integer, string)
        Set Element.parameter-name.value := stringToVarType(parameter.value)
    End {For-Parameters}
End {For-Elements}
```

Fig. 6. Pseudo Code for automatically generating material flow simulation models in Plant Simulation.

an AML based data model containing a simulation request and a simplified production system topology was transformed into a Plant Simulation tailored xml-file using *Saxon* as the processing engine and *Altova MapForce* for xslt file creation (see Fig. 5). The xml-file is then imported as a nested string table object into Plant Simulation. In order to automatically generate the model, this table is iteratively looped and the simulation objects are individually created and parameterized using createObject- and typeConversion-functions (see Fig. 6). After the model generation, the simulation run(s) are automatically executed and the results are transformed back into the AMLSimulationResult format.

Although well applicable, the approach contains some disadvantages as e.g. simulation-tool specific functions can only hardly be triggered with generic data models. Furthermore, the definition of tool specific stylesheets can be time consuming, especially if more complex and case dependant transformation rules need to be implemented. In addition, AML files for larger data structures, especially production schedules, tend to become unnecessarily big, due to the rather large AML file overhead. Nevertheless, the SE provides the required function-wrapping and file-conversion mechanisms as intended and thus successfully allows the automatic and service based execution of simulation runs and thus supports the integration of simulation tools in highly dynamic and service oriented manufacturing environments.

V. SUMMARY

In this paper, a Simulation Environment (SE) was presented in order to execute material flow simulations in a standardized and service based manufacturing context. The SE allows the integration of industrially available simulation tools in a wrapper concept that allows the automatic generation of simulation models and their service based execution, e.g. via a REST interface. For this purpose, a detailed implementation concept was introduced, describing requirements, architecture, and workflows as well as an exemplary application of the SE by integrating the material flow simulation tool Tecnomatix Plant Simulation. The generic SE further allows users to execute a configured simulation for their factory. Internally, the SE contains a data model of simulation based elements describing components that can be instantiated to describe a particular simulation. Further research will focus on the integration of further simulation tools, the extension and standardization of the utilized data models, and the coordinated execution of several simulation tools for specific, tailored simulation service requests, either sequentially or in parallel, i.e. in a co-simulation approach.

ACKNOWLEDGMENT

The PERFoRM project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 680435.

REFERENCES

[1] E. Westkämper, T. Rist, L. März, H. Hezel: „Die Wandlungsfähigkeit der Fabriken“, Deutsche Fachkonferenz Fabrikplanung, 1998, Offenbach

[2] T. Bauernhansl, U. Dombrowski: „Einfluss von Industrie 4.0 auf unsere Fabriken und die Fabrikplanung“, Deutscher Fachkongress Fabrikplanung, 2016, Ludwigsburg

[3] M. Zapp: „Simulationswerkzeuge zur Rekonfiguration von Produktionsanlagen“, Productivity management 15 (1), 2010, pp.35-38 ISSN: 1868-8519

[4] T. Erl, Service-oriented architecture: a field guide to integrating XML and web services. Prentice Hall PTR, 2004

[5] S. Arya, M. Yadav: „Service-Oriented Architecture: Concepts and Challenging Issues”, International Journal of Advances Research in Computer Sience and Software Engineering, vol. 4 (7), 2014 ISSN:2277 128X

[6] N. Schmidt, A. Lüder „AutomationML in a Nutshell“, AutomationML e.V. Office, https://www.automationml.org/o.red/uploads/dateien/1447420977-AutomationML%20in%20a%20Nutshell_151104.pdf, 2015

[7] WITNESS.IO https://io.witness.cloud/Experiment/GetStarted, last accessed 2017-06-08

[8] P. Leitão, J. Barbosa, P. Vrba, P. Skobelev, A. Tsarev, D. Kazanskaia: "Multi-agent system approach for the strategic planning in ramp-up production of small lots”. In 2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 4743-4748.

[9] J. Fischer, R. Willers, C. Sinnwell „Materialflusssimulation für die CTPS-Entwicklung“, In M. Eigner, W. Koch, C. Muggeo: „Modellbasierter Entwicklungsprozess cybertronischer Systeme“; Springer Vieweg, 2017.

[10] C. Steimer, J. Fischer, M. Cadet, H. Meissner, J.C. Aurich, N. Stephan: SysML-basierte Planung cybertronischer Produktionssysteme in frühen Entwicklungsphasen. In: S.-O. Schulze, C. Tschirner, R. Kaffenberger, S. Ackva (eds.): Tag des Systems Engineerings 2016. Carl Hanser Verlag, München, 2016.

[11] T. Baudisch, V. Brandstetter, J. C. Wehrstedt, M. Weiß, T. Meyer: „A central multiperspective data model for the automatic generation of simulation models for virtual commissioning”, In: Tagungsband Automation. VDI 2017

[12] G. Wünsch: Methoden für die virtuelle Inbetriebnahme automatisierter Produktionssysteme. Forschungsberichte IWB, Band 215. Herbert Utz Verlag, 2008

[13] M. Oppelt, G. Wolf, O. Drumm, B. Lutz, T. Baudisch, J.C. Wehrstedt, A. Krause, L. Urbas: „Automatische Generierung von Simulationsmodellen für die virtuelle Inbetriebnahme auf Basis von Planungsdaten. Vorstellung eines generischen Konzepts und einer prototypischen Implementierung“, In: Tagungsband Automation. VDI 2014

[14] Phoenix Integration, ModelCenter® Integrate, http://www.phoenix-int.com/modelcenter/integrate.php, last accessed 2017-04-12

[15] Modelica Association; Functional Mock-up Interface for Model Exchange and Co-Simulation, https://www.fmi-standard.org/downloads, v.2.0, 2014

[16] Modelica Association; The Modelica Specification; https://www.modelica.org/documents; v.3.3; May 2012

[17] PERFoRM-Project: “Specification of the Generic Interfaces for Machinery, Control Systems and Data Backbone”, Del.2.3, Sept. 2016

[18] S. Bangsow, „Fertigungssimulationen mit Plant Simulation und SimTalk." Anwendung und Programmierung mit Beispielen und Lösungen“, 2008, Carl Hanser Verlag München Wien.