Continuous Process Model Refinement from Business Vision to Event Simulation and Software Automation

Bridging Gaps between Stakeholder Communities, Practices, Notations, and Tools

Olaf Zimmermann
University of Applied Sciences of Eastern Switzerland (OST), Rapperswil, Switzerland
olaf.zimmermann@ost.ch

Katharina Luban
University of Applied Sciences of Eastern Switzerland (OST), Rapperswil, Switzerland
katharina.luban@ost.ch

Mirko Stocker
University of Applied Sciences of Eastern Switzerland (OST), Rapperswil, Switzerland
mirko.stocker@ost.ch

Giuliano Bernard
University of Applied Sciences of Eastern Switzerland (OST), Rapperswil, Switzerland
giuliano.bernard@ost.ch

ABSTRACT

Business consultants and software engineers produce and consume process models capturing analysis and design results on different levels of abstraction and at different stages of refinement. Model types commonly found in practice include vision models (current, future), simulation models, and automation models. In this paper, we propose to align and map the terminologies and concepts of these model types to improve stakeholder collaboration. We support this concept mapping with two model transformations to and from discrete event simulation models. We implemented these transformations prototypically (MDSL2JaamSim, JaamSim2MDSL). Our work originates from an industrial case in the FinTech domain. An experimental validation suggests benefits such as effort savings.

CCS CONCEPTS

• **Software and its engineering → System description languages**; **Integration frameworks**; **System modeling languages**; **Orchestration languages**.

KEYWORDS

Business process modeling, discrete event simulation, domain-specific languages, model-driven software engineering, software architecture, API design, enterprise application integration

ACM Reference Format:
Olaf Zimmermann, Katharina Luban, Mirko Stocker, and Giuliano Bernard. 2022. Continuous Process Model Refinement from Business Vision to Event Simulation and Software Automation: Bridging Gaps between Stakeholder Communities, Practices, Notations, and Tools. In *5th International Workshop on Software-intensive Business: Towards Sustainable Software Business (IWSiB'22), May 18, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3524614.3528631

IWSiB '22, May 18, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/3524614.3528631

1 INTRODUCTION

Business domain analysts, simulation specialists, and software architects create process models to capture domain insights, help advance designs, and communicate analysis and design results among various stakeholders from different communities. When modeling as-is and to-be processes, the different roles ask similar questions, e.g., about parallel and alternative flow steps as well as data input and output. The motivation behind these questions differs by stakeholder role: Business analysts and event simulation specialists want to model how enterprises operate; they are interested in the impact of system changes on resource consumption and key performance indicators. Application architects want to analyze and design how business processes can be automated in software fully or partially; integration architects and specialists are concerned about interoperable and guaranteed message delivery between systems.

The stakeholder communities use different terminologies; many competing practices and notations exist. As a consequence, collaboration is tedious. Answering process design questions and aligning models created by stakeholders in different roles is manual, partially repetitive and error-prone work. To the best of our knowledge, tools to transform higher-level business process models (which we call vision models) to simulation models do not exist; the simulation models are usually not used as direct input to software requirements engineering or process automation either.

In this paper, we first review common process modeling practices of business and software stakeholders. Next, we propose a conceptual mapping allowing to advance and refine process models from early stages (in our case, vision models capturing event storming workshop results) to discrete event simulation models and then on to software component interface and flow descriptions (which we call automation models). Our research contributions are:

(a) A mapping of common concepts in business process modeling, event simulation, and software automation.
(b) A Domain-Specific Language (DSL) for vision models.
(c) A realization of the mapping in two transformations.

Our empirical research is motivated by an industry collaboration with a FinTech startup. The proposed transformations are implemented in an open source prototype. They are part of a toolchain featuring the online whiteboard miro, the event simulation package

JaamSim, and tools for the emerging Microservice Domain-Specific Language (MDSL). We validated our research contributions by applying the transformations in action research. We also conducted a controlled self experiment. The mapping from vision model to simulation model is feasible; it can save repetitive modeling effort and improve collaboration. The proposed concepts and open source tools have the potential to close the loop en route to a *continuous process model refinement* practice.

The remainder of this paper is structured in the following way. Section 2 provides background information on discrete event simulation and the use of process models in software engineering and system integration. Section 3 identifies the commonalities in the concepts and terminology used by these stakeholder groups and proposes a mapping for the shared parts. It also presents the implementation of two tool prototypes implementing the mapping in both directions. Section 4 validates and discusses the results. Section 5 summarizes and concludes.

2 BACKGROUND AND RELATED WORK

Software-intensive enterprises change their business models frequently, for instance, in response to market changes. As a consequence, their software systems have to be updated as well. This requires engineering artifacts on different levels of abstraction and refinement to be adapted continuously as the business models and the software evolve. For example, sales and fulfillment activities might be expressed as *business process models* that are first sketched, then designed in detail, and finally made executable. Such models are created for different purposes by different stakeholders. They may serve as a mere documentation and visualization of business procedures, but might also be leveraged to simulate business operations in an event-driven fashion. In the software engineering and application integration, process models may support system decomposition, API design and test, or runtime workflow management. The stakeholders involved in these activities are rather diverse in terms of their educational backgrounds, skills, preferred notations, and tools. If they are not connected well, cross-role analysis and design work become inefficient. For instance, key stakeholders might receive and reply to the same questions multiple times, leading to redundant and/or possibly inconsistent results. E.g., multiple versions of the same sales or fulfillment processes might be modeled, each supporting one stakeholder view. Let us take a closer look at two of these stakeholder communities and their practices now.

2.1 Solving Industrial Optimization Problems with Discrete Event Simulation Models

Discrete event simulation (or Discrete Event-oriented Simulation, DES) has been practised for more than twenty years. In DES, system conditions and values change abruptly at discrete points in time due to certain random events. DES can therefore be defined as an interacting set of entities that evolve through different states as internal or external events happen [Robinson 2004].

DES models produce a valid representation of a real-world system that takes the inherent uncertainty and dynamics of the system into account. This is especially useful when the abundance and random distribution of variants and sequences of actions makes the real-world system so complex that the problem cannot be solved satisfactorily by analytical and numerical methods. [Liao et al. 2021] and [Gorecki et al. 2020] for example describe the usage of DES in global production networks and smart production systems. DES can also analyse systems that do not exist yet, such as new constructions or even new business ecosystems. [Evans et al. 2017] point out that simulation can be used "to identify value flows and exchanges, which could reveal opportunities for business model innovations and de-risk experimentation". Thus, DES provides a chance to model economic systems. These systems can be macroeconomic systems such as a national economy or microeconomic systems that include one company or business plan. More recently, DES also has been used to support investment decisions and business plan development for startups [Innosuisse 2020].

Simulation models essentially are evaluation models: A single run of a stochastic simulation model obtained for a particular set of parameters provides a sample of target values. Multiple runs thus generate multiple samples sketching a range of possible outcomes. [Baden-Fuller and Morgan 2010] and [Robinson 2004] recommend so-called "what-if questions" to derive appropriate "sets of parameters", such as "what if we increase the production capacity and extend the service interval?" or "what if the demand doubles and we add another shift?".

It is particularly promising to use simulation models in the field of optimization. The objective of optimization is to successively modify certain input parameters of the model so that improved target values are obtained. One of many exemplary applications is defining an optimal lot size and shift model depending on production capacities, considering volatile demand and service time. The optimization approach entails that a simulation model is evaluated for many parameter combinations. Improved model runtimes massively reduced the computing time in such applications, which is another reason for the increasing popularity of DES and its spreading to the environment of small and medium-sized enterprises.

Methods support systematic and repeatable model creation. For instance, [Swisher et al. 2000] define *sim-opt* as a "structured approach to determine optimal input parameter values, where *optimal* is measured by a function of output variables ... associated with a simulation model". As can be seen in Figure 1, the sim-opt process uses the output values (3) from several simulation runs (2) to evaluate the performance of a certain set of parameters (1). An adjustment and optimization procedure (4) combines the results and past ones to decide upon a new set of input values (1).

[Andradóttir 2006] analysed classes of optimization algorithms to decide on what set of parameters should be sampled in the next iteration of a sim-opt process, and she provides a discussion as to when random search methods (sim-heuristics) converge with near certainty to a globally optimal system design. One of the keys to sim-heuristics is a good understanding of the inter-dependencies of the system. In an industrial or business context, this understanding can only be developed when simulation experts work very closely with business and process owners.

Typically, in a first step, the goals of the simulation have to be agreed on by understanding the business context and by defining issues or questions that the simulation is meant to answer. The next – and arguably most interactive – step is to understand the overall system with its process flow steps and relevant interactions. It is necessary to identify relevant input data and determine

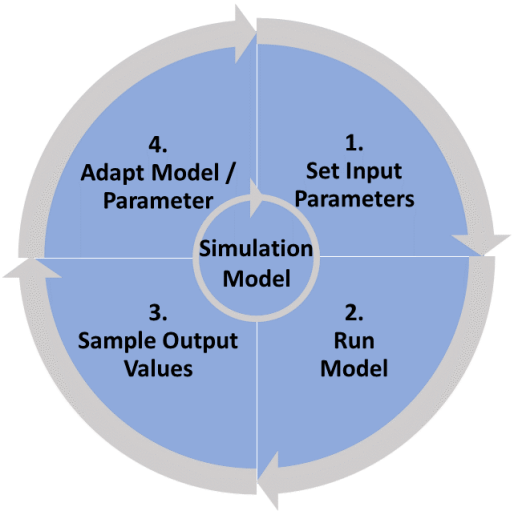

Figure 1: Sim-Opt Process (own presentment)

whether these parameters are fixed or variable. To guarantee a target-oriented output, appropriate sets of parameters are defined at this stage, e.g., with the help of what-if questions (as explained above). Practices applied to gain the necessary insights are interviews and workshops, data analyses, and questionnaires. A common technique that may create data-driven workflow visualizations is *business process modeling*. The resulting process models provide end-to-end views of the business process that help organizations document workflows, surface key metrics, and pinpoint potential problems. We call these models *vision models* in this paper because they may represent a desired future state of operations. They may also capture the currently implemented business processes.

Many different business process modeling notations are available. A common choice is the Business Process Management Notation (BPMN), supported by various tools [Object Management Group 2011; Ruecker 2021].

Based on an agreed vision model, an initial *simulation model* can be created. This is done directly in one of the many existing simulation tools. One of these DES tools is JaamSim [King and Harrison 2013], an open source software that provides an API and creates a configuration file for the model that is human- and machine-readable. Moreover, due to its open source character, JamSim offers the possibility to extend existing or design new model element types, which can satisfy a variety of needs that exceed the capabilities of the standard tool.

When creating a simulation model for a particular DES tool, the process logic of previously defined models have to be followed. Furthermore, simulation-specific model elements of the chosen DES tool have to be added (for instance, statistics collectors, logger and output viewers). Although many tools assist in creating the models with drag-and-drop and copy-paste functions, the modeling can be a tedious, sometimes repetitive manual task not creating direct value when transferring input models manually. This transfer may introduce errors causing models to be incorrect, which will degrade the quality of the simulation output.

Once the model has been drafted in the simulation tool, it is validated by test runs. In practice, is is often not feasible to prove that a model is valid for its target domain as a whole [Sargent 2010]. As a result, modifications of the model might be required.

If it is considered unnecessary to adapt both the underlying business process model *and* the simulation model, the painstakingly developed business process model might lose importance and finally become outdated. That by itself would not be a problem if the simulation process and the sim-opt process were unidirectional processes. However, the physical world and market dynamics are volatile and continuously changing nowadays, and so are the business needs. Moreover, insights from optimization processes might lead to adapted views. The language to communicate with the users of simulation results is still the business process model (or model), which by then has been superseded and needs a lot of – tedious and error-prone – updating.

As a consequence of the above considerations, an automated interface between high-level business process models and simulation models will be greatly appreciated. Such an interface — as shown in the center of Figure 2 — promises to save time and cost and to reduce the risk exposure caused by manual work.

2.2 Business Processes and Simulation Models in Application and Integration Design

Software architects and system integrators often create models representing static structures and dynamic interactions of systems and their parts. Component walkthroughs and service orchestrations are two examples of such software *automation models* that capture runtime behavior. BPMN, UML activity and sequence diagrams, and Domain-Specific Languages (DSLs) are among the notations commonly used in practice; a number of techniques to craft such models are applied in industry. *Event storming* workshops [Brandolini 2021] are a recent addition to this toolbox, which already contains event-driven process chains, behavior-driven development, object-oriented analysis and design, and various agile engineering practices [Zimmermann and Stocker 2021].

Some enterprise applications include workflow engines with explicit "Process Managers" [Hohpe and Woolf 2003] executing BPMN models (or supporting other languages); an example is Teravis [Lübke and van Lessen 2016]. Even if no workflow engine is used, business-level process models provide valuable input for software architects. For instance, domain model elements and information about external systems that have to be integrated may be found in such models. Volume metrics that help establish quality goals regarding performance and availability can be elicited, and the development of test cases benefits from information about variability, edge cases, and error situations. If such information does not appear verbatim in the models or accompanying text notes, the model content can still help frame related questions to business stakeholders and domain experts.

The simulation models used to solve industrial optimization problems (see Section 2.1) can be particularly valuable according to experience recently gained in action research on a joint industry collaboration project [Innosuisse 2020], introduced in the next section. Both the control flow in these models, but also the input parameters and target values used in the sim-opt process (Figure

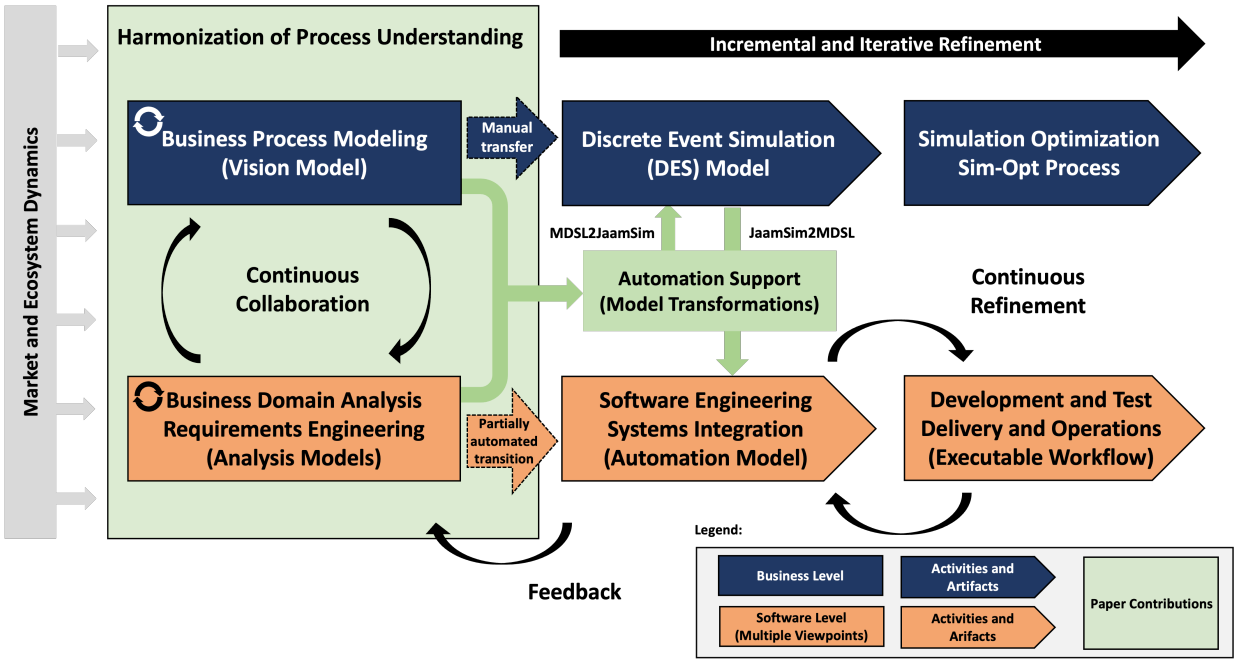

Figure 2: Conceptual Overview: Continuous Process Model Refinement on Business and Software Engineering Level

1) can drive API designs and justify architectural decisions about service design and pattern selection [Zimmermann et al. 2020].

In a fulfillment and logistics business scenario, the architects of a new warehouse management and logistics planning system might inspect the business-level process models to learn about the items (amount, size, other properties) stored in a warehouse and the working hours of staff, to request information about the interface of an external enterprise resource planning system, and to analyze error cases such as "empty warehouse, full order backlog". They also may find information about the transaction rates at which items are delivered and replenished. Business rules subject to auditing such as "no invoice without a shipment, no refund without return of goods" may also become apparent in process models.

2.3 Sample Case: Wearables as Payment Devices

Our collaboration partner from the FinTech industry provides order fulfillment services for wearables such as wrist bands that carry payment tokens (for instance, those corresponding to credit cards). Supply chain configuration is key to make the business profitable. There are many options to design the supply chain, warehouse management, and shipment activities. These options (e.g., for order management, inbound logistics, handling, and packaging) are modelled and simulated. Automation software controls the process execution at runtime, integrating third parties such as banks and credit card providers. On the joint project [Innosuisse 2020], vision models were captured in event-command chains and BPMN; more detailed BPMN models were used to configure the simulations in the DES package JaamSim. MDSL was used for API design in partial process automation. While overlapping in scope, these three types of models were created manually and independently.

2.4 Deficits and Resulting Research Problems

In the current state of the art and the practice, vision, simulation, and automation models as introduced in Sections 2.1 and 2.2 are not connected well, although they represent the same processes. Even if a standard notation such as Business Process Management Notation (BPMN) and the same modeling tools are used for editing and viewing, the resulting models might not be shared or might be unintelligible for the other stakeholder role. Such ambiguities might be viewed as an opportunity to improvise and innovate, but also as a threat causing unnecessary workload and misunderstandings.

According to our experience, one root cause is a missing terminology alignment. For example, terms such as process model, parameter, split/branch, entity, etc. mean different things to business analysts, event simulation specialists, and software architects. If performed at all, transforming process models of different types is tedious and error-prone; full or partial automation is missing.

To overcome the outlined practical collaboration and modeling problems, the following questions have to be answered:

(1) *How do the process modeling concepts and notations used by the business analysis and event simulation communities and the software engineering and system integration communities relate and compare to each other? Do they overlap?*
(2) *Is it possible to map the related and overlapping process modeling concepts for business vision analysis, discrete event simulation, and software automation?*
(3) *Can the conceptual mapping be implemented in model transformations to support partial automation in a continuous business process model refinement practice?*

Our answers to these three questions form the research contributions of this paper; they will be provided in Section 3.

2.5 Related Work

A 2014 survey reported that "design models are not used very extensively in industry, and where they are used, the use is informal and without tool support" [Gorschek et al. 2014]; the report mainly focused on UML. As of 2020, modelling in general is reported to be adopted in practice in certain domains, usage scenarios, and development styles; examples include informal modeling, low-code software development, and model-based engineering of cyber-physical systems [Bucchiarone et al. 2021]. Note that our work concerns both (requirements) analysis and (software) design models, those capturing behavior and systems dynamics in particular.

To the best of our knowledge, a connection or interfaces between business vision models, event simulation models, and process automation models have not been investigated in depth yet; we did not find any model mappings or transformation tools in the literature. "Model Continuity in Discrete Event Simulation: A Framework for Model-Driven Development of Simulation Models" [Çetinkaya et al. 2015] has similar goals but focuses on the engineering process as such. [Yigit and Ulsoy 1989] emphasises the importance of dynamic models in a particular domain but does not cover business analysis (or vision) model transformations. [Hlupic and Robinson 1998] from 1998 comes closest to our work and shares our vision but stops at using simulation to evaluate and compare business processes; it does not propose model mappings or transformations either.

3 RESEARCH CONTRIBUTIONS

Figure 2 illustrates our proposal to align the process modeling practices on the business level and the software engineering level for their mutual benefit and to support the stakeholder collaboration with two model transformations. The third contribution of this paper is a new DSL for vision models (as transformation input/output).

3.1 Alignment of Process Modelling Concepts

As discussed in Section 2, a) DES and b) software engineering share a high-level goal of satisfying stakeholders. They do so in different ways, a) with expressive, useful simulation results that can support business decisions effectively and b) by delivering working software that fulfills its business and quality requirements. To reach these goals, both communities have to understand how a business operates today and wants to operate in the future. In software engineering, this falls under requirements elicitation.

Our sample case suggests that the business process and simulation models are indeed valuable for software engineering, even if created for a different purpose. Under this premise, transferring the models means manual copy-paste or rework; we are unaware of any practice or tool that would bridge the model gap in the current state of the art (see coverage of related work in Section 2.5).

In sprint meetings and retrospectives of the collaboration project outlined in Section 2.3, we developed the terminology comparison and mapping shown in Table 1. The table includes those parts of the terminology alignment we found to be required to map and transform vision, simulation, and automation models.

The first column primarily lists primitive "Workflow Patterns" [Russell et al. 2016]. The second column refers to BPMN terms. In the third column, general concepts from programming are used, as well as selected Domain-Driven Design (DDD) patterns such as "Value Object" [Evans 2003]; DTO stands for "Data Transfer Object" [Zimmermann and Stocker 2021]. Application integration terms such as "Pipes-and-Filters" and "Message" come from the pattern language "Enterprise Integration Patterns" [Hohpe and Woolf 2003].

Note that we do not claim that all concepts are semantically identical; some of them correspond to each other rather loosely because they have similar goals in their respective user community.

3.2 A DSL for Business Vision Models

A gray literature review of event storming practices led to previous work on a modeling tutorial.[1] Usually created on real or virtual whiteboards collaboratively, event stormings capture the "what happens when" of a business vision in the form of connected events, commands, artifacts and other, optional concepts [Brandolini 2021].

Only a small subset of the process modeling concepts found in BPMN is adequate in such event-driven vision models. The following example therefore proposes an alternative, novel syntax to capture the result of an exemplary event storming:

```
flow ParallelSplitWithSynchronization
  event FlowInitiated triggers command FlowStep1
  command FlowStep1 emits event FlowStep1Completed
  event FlowStep1Completed
    triggers command FlowStep2 + FlowStep3
  command FlowStep2 emits event FlowStep2Completed
  command FlowStep3 emits event FlowStep3Completed
  event FlowStep2Completed + FlowStep3Completed
    trigger command FlowStep4
  command FlowStep4 emits event FlowTerminated
```

The above flow DSL originates from the domain-driven design tool Context Mapper [Kapferer and Zimmermann 2020] and was revised and adopted for inclusion in MDSL here. Flow DSL and its tools were validated with a set of comprehensive test cases based on workflow pattern primitives from the literature and a translation to BPMN. Feedback from BPMN experts in industry was solicited and incorporated.

3.3 Transformations to/from Simulation Models

We used the conceptual mapping from Section 3.1 to implement mappings between two exemplary notations, Microservice Domain-Specific Language (MDSL) flows as featured in Section 3.2 and the JaamSim configuration format (see Section 2.1 for a brief introduction to JaamSim). These transformations are shown in Figure 2.

Table 2 specifies the mapping implemented in the first one, which is called *MDSL2JaamSim*.[2]. MDSL commands become JaamSim servers, and events are mapped to queues. Composite control flow steps (and/or) are translated into Duplicates and Branches, respectively; join events are mapped to Combines. There is one SimEntity per flow; events and command without a predecessor become Entity Generators, and termination events turn into Entity Sinks (note: all

[1] available at https://contextmapper.org/docs/event-storming/

[2] We created MDSL in previous work to be able to specify service contracts, their data representations, and API endpoints in technology- and platform-independent ways (but providing bindings to contemporary technologies and languages, e.g. OpenAPI/HTTP and ProtocolBuffers/gRPC) [Kapferer and Zimmermann 2020]. MDSL tools (e.g., editor, linter, generators) exist, using Java, Eclipse, and the Xtext framework for DSLs. MDSL and its tools are developed and maintained in an open source project maintained at GitHub: https://microservice-api-patterns.github.io/MDSL-Specification/

Table 1: Process Modeling on the Business Level (Analysis, Design) and in Software Engineering/Application Integration

Process Modeling Concept	BPMN (and other notations)	Automation Models (Prog., Workflow Mgmt., EAI)
Process description	Process model, process map	Workflow, integration flow
Unit of work (atomic)	Activity, task, step	Statement, filter (in Pipes-and-Filters messaging)
Partitioning, containers	Pool, swimlane, subprocess	Aggregate [Evans 2003], component, service, module
Parallel split	Parallel Gateway (AND)	Fork, Recipient List [Hohpe and Woolf 2003]
Inclusive choice	Inclusive Choice Gateway (OR)	Content-Based Router [Hohpe and Woolf 2003]
Exclusive choice	Exclusive Choice Gateway (XOR)	Content-Based Router (with exclusive selector)
Simple Merge	Join	Aggregator [Hohpe and Woolf 2003]
External stimulus/configuration data	Input parameter	In parameters, request Messages
In flight data	Event	Value Object, DTO, Document/Command Message
Result data	Target output	Out parameters, reply queues
Process start	Start event	main methods, Message Construction
Process end	Termination event, stop event	return, Message Consumption (in endpoint)
Asynchronous communication	e.g., messages, signals, thresholds	Signals, interrupts, Event Messages [Hohpe and Woolf 2003]

Table 2: Mapping between Vision Models (here: CML, MDSL) and Simulation Models (here: JaamSim)

Process Modeling Concept	Event Storming Output/Vision Models (CML, MDSL)	DES Model (JaamSim Configuration)
Process	Flow	Configuration with one SimEntity
Start	Initial events and commands	EntityGenerator
Unit of work (active)	Command	Server
Unit of work (passive)	Event	Queue
Parallel split	and operator for commands and events (+)	Duplicate
Choice (inclusive, exclusive)	or/xor operators for commands and events (o, x)	Branch
Merge	Join event (on left side of flow step)	Combine (constrained semantics)
End	Termination events and commands	EntitySink

terms in upper case are JaamSim-specific incarnations of general BPM and DES concepts). Figure 3 shows parts of the generated model in the graphical user interface of JaamSim; the full output is available in the GitHub repository of MDSL.

A second transformation *JaamSim2MDSL* is implemented in plain Java (using the JaamSim API and an MDSL API). It inspects process flow elements in the JaamSim configuration file (e.g., Servers, Queues, Branches, Duplicates) and derives candidate API endpoints with data type definitions for request and response messages from them. In our example, an excerpt from the MDSL output is:

```
API description MDSLToJaamSimPaperExampleContract

data type FlowStep1DTO "placeholder":D<string>

endpoint type MDSLToJaamSimPaperExampleEndpoint
  supports flow FlowTest2
  serves as PROCESSING_RESOURCE
exposes
 operation runFlowStep1
  with responsibility STATE_TRANSITION_OPERATION
  expecting payload "changeRequest":FlowStep1DTO
  delivering payload "updateResult":FlowStep1DTO
  transitions from "anyState" to "FlowStep1Finished"
  emitting event FlowStep1Completed
 // also: runFlowStep2, runFlowStep3, runFlowStep4
 operation calculateFlowTerminated
  with responsibility COMPUTATION_FUNCTION
  expecting payload "flowEntitities":ID<int>+
  delivering payload "flowTerminatedData"
receives event FlowInitiated
```

The code snippet features the part of MDSL that allows API designers to specify endpoint types with operations, including the payload of request and response messages as well as events and state transitions. Here, only two operations are shown, one representing the first flow step (which may be triggered by an external API call) and one representing the computation of simulation results. Steps 2 to 4 in the vision model input and the simulation model generated by MDSL2JaamSim (see Figure 3) are not shown; they look similar.

Data types and an endpoint type representing the simulation entities (exposing create, read, update, and delete operations) are also generated, but not shown. Due to space constraints, we do not describe this second implementation in detail here; source code and output MDSL are available in the MDSL repository [MDSL 2022].

API specifications such as the above MDSL snippet can jump start the implementation of complete automation workflows and/or individual flow steps, for instance in HTTP Web APIs.

4 VALIDATION AND DISCUSSION

Conceptual mapping and implementation for MDSL and JaamSim were developed in three iterations: proof-of-concept demonstrator, Minimum Viable Product (MVP), and open source release.

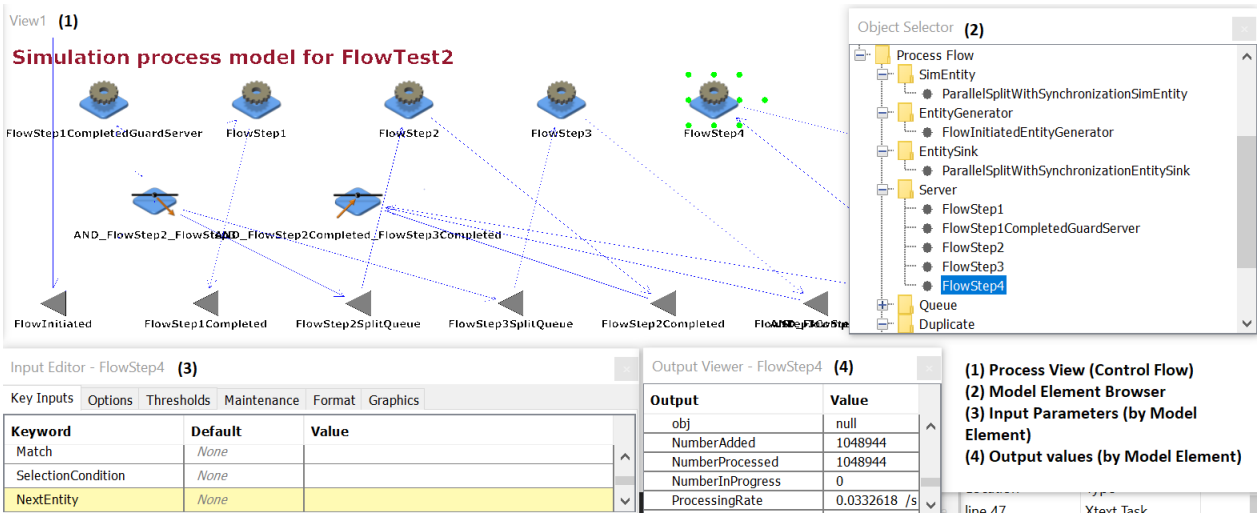

Figure 3: MDSL2JaamSim Output: Configuration of Sample Flow Featuring Workflow Patterns (Selected Views/Model Elements)

The proof of concept had the objective to assess technical feasibility and to solicit early feedback regarding the value, for instance, for DES practitioners on the FinTech project. The MVP was used to validate the correct working and usability of the mapping transformations. The size of the examples grew from a basic "hello world" scenario to a comprehensive unit test suite covering relevant workflow pattern primitives [Russell et al. 2016] to a medium-size vision model and a large, realistic simulation model from the FinTech case (see Section 2.3). This validation setup qualifies combines action research and controlled self experiment to investigate:

Should business process modeling, DES, and software design be treated as disconnected work streams? Or can vision, simulation, and automation models be aligned and serve as mutual modeling input?

4.1 Action Research and Controlled Experiment

Two of the authors of this paper not involved in the implementation validated the MDSL to JaamSim transformation in the MVP version. These early adopters and testers represent our two stakeholder groups: one of them is a DES specialist, the other one a software architect and Web developer. They had not worked with the notations and tools of the other group before. We instructed them to run through a demo script[3] and answer survey questions about concepts implementation, benefits, drawbacks, and practicality of the approach. The test setup uses a slightly larger example than presented above.

Having overcome initial setup and installation problems, both testers succeeded in converting the given event storming output to a simulation model with MDSL2JaamSim. They could run through all steps in Figure 2 as instructed, including a model revision. Both of them were able to trace the mapped concepts back and forth. They reported some usability issues and limitations in the implementation (model layout in JaamSim, handling of loops and commands receiving multiple events). These two issues have been resolved in the open source version of the tool. Both testers appreciate the potential to save a significant amount of time in the transformation from a vision model to a simulation model, but were not able yet to quantify these envisioned savings.

Reviewing the current mapping implementation, the horizontal process layout with separate lanes for JaamSim servers, queues, and control flow gateways can be considered good enough as a design starting point generated by a tool prototype. Few overlaps of arrows connecting model elements exist, and the generated model does not conclude any flow semantics from the deliberately more vague MDSL input (that represents the results of informal event storming workshops). During the implementation, it turned out that the Combine concept in JaamSim does not support simple 1:1 pass through semantics; hence, it is not suited to express all Aggregator behavior specified in [Hohpe and Woolf 2003]. For instance, the aggregation strategy "first best" and aggregation algorithms such as "condense data" cannot be expressed easily. Additional model elements such as aggregation queues and servers were required for that. Furthermore, the rather straightforward mapping from events to queues and commands to servers is well suited for most cases, but causes problems for flows with loops and in some edge cases because in JaamSim servers can only wait for input on one queue (this is a limitation of the tool, not an issue with DES concepts and tools in general). Again, additional model elements such as guard servers and extra queues were required to overcome this semantic gap. Such mapping problems are rather usual in application integration, and process modeling tool integration seems no exception in this regard. In our opinion, such known limitations and intricacies would be showstoppers for model transformations with other goals (e.g., those that have to preserve operational semantics and proof their correctness in all cases) , but this is not the case here.

4.2 Discussion of Results

Our process-centric work differs from other model-driven development approaches in that it emphasizes partial automation. Unlike executable UML, for instance, we do not strive for a substitution of

[3] available at ozimmer.ch/practices/2022/02/01/ProcessOrientedServiceDesignDemo.html

design work; our partial automation turns process models and their transformations into design guides (or virtual coaches) and discussion catalysts (and sources of inspiration). The transformations also reduce routine work that is rather dull and error-prone.

As in many modeling tools, model reconciliation, for instance via automatic reverse transformations, remains an open issue. Semantic mismatches between the involved metamodels had to be overcome.

Our conceptual solution is not limited to any product or practice, even if the tool prototype focuses on MDSL and JaamSim. Implementing similar transformations tools would work equally well for other simulation software, assuming that an API or open model import/export capabilities exist (API and open configuration file format are among the reasons we chose JaamSim). The first transformation uses an Apache Freemarker template, ensuring portability (to new versions and other tools). The second one only depends on the APIs provided by the two involved tools; API usage is wrapped to ensure portability.

5 SUMMARY AND CONCLUSIONS

In this interdisciplinary paper, we aligned business process modeling concepts applied in practice. While similar, sometimes identical notations are used to craft (business) vision models, (event) simulation models, and (software) automation models, the content of such models differs widely because serve different purposes in analysis and design on the business and the software level. We implemented two transformations supporting the model alignment, from vision models (for which we proposed a new DSL) to simulation models and from simulation models to automation models.

Aligning terminology can save precious analysis and design time and relieve key stakeholders because they no longer have to answer similar questions from business analysts, simulation experts, and software engineers. Event simulation specialists, software architects, and system integrators can benefit from the models created by the other stakeholder groups, when picking them up and transforming them into their realm. Partially automating transformations improves consistency and stimulates innovation because formerly disconnected communities are enabled to leverage each other's intermediate results We do not aim at full automation here but see the model mappings and their implementation in transformations as productivity accelerators and design assistants. Even if no tools are involved, aligning the process modeling concepts can improve collaboration and avoid unnecessary repetition in the business analysis and software design work.

Opportunities for future work are manifold. On the conceptual mapping level, the data structures and values in the models (including simulation parameters and target values, as well as data transfer objects and database table definitions) could be further investigated and possibly supported by the automated model transformations. Moreover, API testing and mocking could be supported, for instance, by offering simulation capabilities in test tools that leverage generated flow models to analyze the performance impact of API design refactorings.

Acknowledgments. Former OST staff member Stefan Kapferer designed the flow syntax for the application layer support in Context Mapper. [Innosuisse 2020] supported the collaboration project case.

REFERENCES

Sigrún Andradóttir. 2006. Chapter 20 An Overview of Simulation Optimization via Random Search. In *Simulation*, Shane G. Henderson and Barry L. Nelson (Eds.). Handbooks in Operations Research and Management Science, Vol. 13. Elsevier, 617–631. https://doi.org/10.1016/S0927-0507(06)13020-0

Charles Baden-Fuller and Mary S. Morgan. 2010. Business Models as Models. *Long Range Planning* 43, 2-3 (2010), 156–171. https://doi.org/10.1016/j.lrp.2010.02.005

Alberto Brandolini. 2021. *Introducing EventStorming An act of Deliberate Collective Learning*. LeanPub. https://leanpub.com/introducing_eventstorming

Antonio Bucchiarone, Federico Ciccozzi, Leen Lambers, Alfonso Pierantonio, Matthias Tichy, Massimo Tisi, Andreas Wortmann, and Vadim Zaytsev. 2021. What Is the Future of Modeling? *IEEE Software* 38, 2 (2021), 119–127. https://doi.org/10.1109/MS.2020.3041522

Deniz Çetinkaya, Alexander Verbraeck, and Mamadou D. Seck. 2015. Model Continuity in Discrete Event Simulation: A Framework for Model-Driven Development of Simulation Models. *ACM Trans. Model. Comput. Simul.* 25, 3, Article 17 (apr 2015), 24 pages. https://doi.org/10.1145/2699714

Eric Evans. 2003. *Domain-Driven Design: Tacking Complexity In the Heart of Software*. Addison-Wesley.

Steve Evans, Doroteya Vladimirova, Maria Holgado, Kirsten van Fossen, Miying Yang, Elisabete A. Silva, and Claire Y. Barlow. 2017. Business Model Innovation for Sustainability: Towards a Unified Perspective for Creation of Sustainable Business Models. *Business Strategy and the Environment* 26, 5 (2017), 597–608. https://doi.org/10.1002/bse.1939

Simon Gorecki, Jalal Possik, Gregory Zacharewicz, Yves Ducq, and Nicolas Perry. 2020. A Multicomponent Distributed Framework for Smart Production System Modeling and Simulation. *Sustainability* 12, 17 (2020), 6969. https://doi.org/10.3390/su12176969

Tony Gorschek, Ewan Tempero, and Lefteris Angelis. 2014. On the Use of Software Design Models in Software Development Practice: An Empirical Investigation. *J. Syst. Softw.* 95 (sep 2014), 176–193. https://doi.org/10.1016/j.jss.2014.03.082

V. Hlupic and S. Robinson. 1998. Business process modelling and analysis using discrete-event simulation. In *1998 Winter Simulation Conference. Proceedings (Cat. No.98CH36274)*, Vol. 2. 1363–1369 vol.2. https://doi.org/10.1109/WSC.1998.746003

Gregor Hohpe and Bobby Woolf. 2003. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley.

Innosuisse. 2020. From Wearable to Wallet: A Novel Simulation and Automation Ecosystem for Payment Card Digitization: Project number 39597.1 IP-SBM. https://www.aramis.admin.ch/Grunddaten/?ProjectID=47264

Stefan Kapferer and Olaf Zimmermann. 2020. Domain-Driven Service Design. In *Service-Oriented Computing*, Schahram Dustdar (Ed.). Springer International Publishing, Cham, 189–208.

D. H. King and Harvey S. Harrison. 2013. Open-source simulation software "JaamSim". In *2013 Winter Simulations Conference (WSC)*. IEEE, 2163–2171. https://doi.org/10.1109/WSC.2013.6721593

Shuangqing Liao, Adrian Rüegg, and Roman Hänggi. 2021. Deriving a global production network type in times of uncertainty – a simulation based approach. *Die Unternehmung* 75, 4 (2021), 552–575. https://doi.org/10.5771/0042-059X-2021-4-552

Daniel Lübke and Tammo van Lessen. 2016. Modeling Test Cases in BPMN for Behavior-Driven Development. *IEEE Software* 33, 5 (2016), 15–21. https://doi.org/10.1109/MS.2016.117

MDSL. 2022. Microservice Domain-Specific Language (MDSL) Repository. https://github.com/Microservice-API-Patterns/MDSL-Specification

Object Management Group. 2011. Business Process Model and Notation (BPMN): Version 2.0. https://www.omg.org/spec/BPMN/2.0

Stewart Robinson (Ed.). 2004. *Simulation: The practice of model development and use / Stewart Robinson*. Wiley, Chichester.

Bernd Ruecker. 2021. *Practical process automation: Orchestration and integration in microservices and cloud native architectures*. O'Reilly, Sebastopol, CA.

Nick Russell, Wil M.P. van der Aalst, and Arthur H. M. ter Hofstede. 2016. *Workflow Patterns: The Definitive Guide*. The MIT Press.

Robert G. Sargent. 2010. Verification and validation of simulation models. In *Proceedings of the 2010 Winter Simulation Conference*. 166–183. https://doi.org/10.1109/WSC.2010.5679166

James R. Swisher, Paul D. Hyden, Sheldon H. Jacobson, and Lee W. Schruben. 2000. Survey of simulation optimization techniques and procedures. *Winter Simulation Conference Proceedings* 1 (2000), 119–128.

A.S. Yigit and A.G. Ulsoy. 1989. Controller design for rigid-flexible multibody systems. In *Proceedings of the 28th IEEE Conference on Decision and Control,*. 665–673 vol.1. https://doi.org/10.1109/CDC.1989.70201

Olaf Zimmermann and Mirko Stocker. 2021. *Design Practice Reference - Guides and Templates to Craft Quality Software in Style*. LeanPub. https://leanpub.com/dpr

Olaf Zimmermann, Mirko Stocker, Daniel Lübke, Cesare Pautasso, and Uwe Zdun. 2020. Introduction to Microservice API Patterns (MAP). In *Joint Post-proc. of the 1st and 2nd International Conf. on Microservices 2017/2019 (OASIcs, Vol. 78)*, Luís Cruz-Filipe et al. (Ed.). Dagstuhl, Germany, 4:1–4:17. https://doi.org/10.4230/OASIcs.Microservices.2017-2019.4