Microservice based tool support for business process modelling

Sascha Alpers, Christoph Becker, Andreas Oberweis
FZI Forschungszentrum Informatik
Karlsruhe, Germany
{alpers|cbecker|oberweis}@fzi.de

Thomas Schuster
esentri AG
Ettlingen, Germany
thomas.schuster@esentri.com

***Abstract*— The rise of microservices as architectural pattern creates a bunch of interesting opportunities for software architectures of modelling editors and additional services. Main advantages are the scalability in collaborative distributed scenarios and enhanced possibilities regarding service development and operation. Throughout this article, we will illustrate how modelling editors and additional services can be build based on microservices. Our tooling will focus on business process modelling. We will also strive to highlight how architectures of this kind can enact collaborative modelling techniques, increase reuse of utilized service components and improve their integration into lightweight user interfaces, for example in mobile devices.**

Keywords — microservice; business process; modelling; architecture;

I. Introduction

As a field of science and in industrial practice business process management (BPM) focuses on improvement of corporate performance by managing and improving a company's business processes. Therefore BPM supports the whole lifecycle of a company's business process from identification and modelling to operations [1]. In almost all lifecycle phases information systems support BPM, each system used throughout this lifecycle addresses different user groups (e.g. domain experts, managers as well as technical staff dealing with implementation and operation). Thus tool support is a substantial requirement for BPM. In this article we will concentrate on tooling for modelling of business processes. Tools for process modelling typically incorporate the following functionality: ***a)*** user interfaces for creating and editing models – typically including support for different user specific views; ***b)*** support to model domain or user specific needs (e.g. data or security aspects); ***c)*** since models are often created collaboratively between different users, tools need to offer mechanisms to edit and share models collaboratively (often implemented by attached model repository); ***d)*** to support improvement users need possibilities to analyse and simulate processes; ***e)*** companies need possibilities for process execution, hence modelling tools support model transformation or direct execution.

In order to support all these requirements (and extend them probably with additional features) tools often are integrated in a large tool chain incorporating many different tools like for example the IBM Business Process Manager [2], which consists of a designer, a process repository and a runtime engine, or a single tool suite is used to reproduce this tool chain. In the area of BPM often one big tool suite is employed, and maybe extended by small supportive tools using a plug-in concept. Since large monolithic applications are difficult to maintain [3] and requirements for process modelling are manifold, we believe that with modern possibilities of software architectures (also incorporating the concept of microservices [4]–[6]) business process management tools can be created and maintained more efficiently. The application of a microservice architecture comprises the use of small teams that are in charge of specific services with individual change cycles, thereby adding flexibility to the development process and fostering continuous delivery. Due to inherent loose coupling, these services may easily be grouped to create and adapt process modelling tools according to company and user requirements.

The remainder of this paper is structured as follows: first we will introduce basics of business process modelling. This will be succeeded by a survey about current developments regarding tool support and software architecture in general (microservices). Afterwards we will introduce an architecture to support flexible and scalable modelling tools (Section III). In Section IV we will demonstrate this architecture with implemented mobile tool support. The paper ends with a summary and an outlook on further work.

II. Basics and Related work

Within the next three subsections we provide an overview about the basics of business process modelling and about related work regarding tool support and software architecture. The first subsection will briefly highlight the importance of business process modelling. Afterwards we will illustrate current developments regarding tool support and relate that to common architectures found in BPM tools. Finally, a brief introduction into microservice architectures is given.

A. Business Process Modelling

Business process modelling has developed a lot in recent years. For some time business process models have been utilized by companies in order to capture their processes, thereby providing means of documentation and a common base for discussion. In recent years companies have intensified their efforts to implement these processes using computer systems.

Hence, sufficient tool support is as important as the use of a well-defined and appropriate modelling language because models can fully enfold their advantages only if tools help to manage, develop, optimize and apply them. Today many tools support modelling of business processes in different modelling languages. Most of these tools are built as an integrated software suite, often executed as rich client application. Some examples are ARIS [7], ADOxx [8], HORUS Business Modeler [9] as well as IBM Business Process Manager [2] and webMethods [10]. The last two examples also provide process execution capabilities; however, these are not within the focus of this paper. In addition to these tools there exist general modelling tools that offer additional components to support process modelling; one well-known example is Microsoft Visio [11].

B. *Architecture of Modelling Tools*

Software architecture as well as the used technology stack may vary depending on vendor and tool version. Many of these tools are implemented by utilization of frameworks like the eclipse rich client platform [12] and eclipse modelling framework [13], some also support web-based modelling editors. However, most BPM tools are based on a client-server-architecture. Depending on the degree of business logic that is comprised within the user client part, rich and thin client architectures can be distinguished (cp. Figure 1).

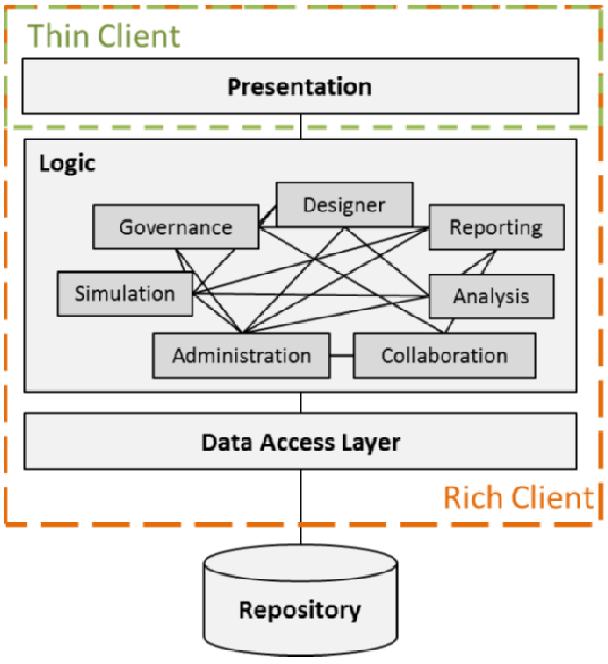


Figure 1: Architecture of BPM Tools

The server side is often implemented as monolithic architecture comprising the data base (rich client) and business logic (thin client). The different components within the business logic typically rely on a common technology stack and communicate via method invocation, this often promotes strong coupling between the components. Considering development of these tools, the following result can be observed: changes of components or integration of supplementary components will result in a rebuild and redeployment (including a phase of integration tests) of the complete business logic layer or even the whole tool.

Furthermore the majority of professional modelling tools support collaborative modelling by usage of shared model repositories. They are often capable of connecting to multiple model repositories (databases). However, these different repositories have to implement the same holistic database schema that is used by the specific application. This imposes increased efforts within the development process for coordinating and implementing changes, especially in large and geographically distributed development teams.

With respect to Abbot’s scalability cube [14] it is obvious that most BPM applications lack possibilities of decent scalability. In most cases a redundant copy of the monolith architecture is deployed in order to scale. In some cases it is possible to partition data across multiple databases adding a second dimension to scale. However, due to the monolith character of BPM applications the third scaling dimension – split according function – is currently not properly addressed, hence leaving room for improvement.

C. *Microservices*

For software architectures of distributed systems the term microservice has gained a lot of attention recently [4], [5]. Microservices can be considered as an architectural style of software systems. In short, microservices are an approach for developing a system through a suite of small sized services. Each service is executed independently (process space), uses its own data (database) and offers lightweight communication mechanisms to other services (often over HTTP or HTTPS). According to this approach services are built around business capabilities, which foster separation of concerns.

Since architectures of large systems are typically organized in layers, teams often consist of specialists who possess skills for a specific layer (like UI, logic and data). This organizational team formation will resemble the system’s architecture – known as Conway’s Law (see Figure 2).

“Any organization that designs a system … will inevitably produce a design whose structure is a copy of the organization's communication structure.” [15]

Once the system has grown to a certain point, and even if it is organized in a bunch of different service modules in the middleware layer, changes or additional services can only be integrated with high effort. The latter is a result of dependencies within the layer and to the layers above and below. If a team changes a single service usually the whole middleware layer has to be built and deployed again. Due to dependencies in e.g. method calls a change is not isolated to a single task. Since a microservice focusses on a single business capability and utilizes a broad implementation stack (including user-interface, storage, and external communication), team organization has to be different. Teams are typically cross-functional, including skills required for design of user-interface, business logic, database, and project management. Compared to the above-mentioned approach services are really loosely coupled

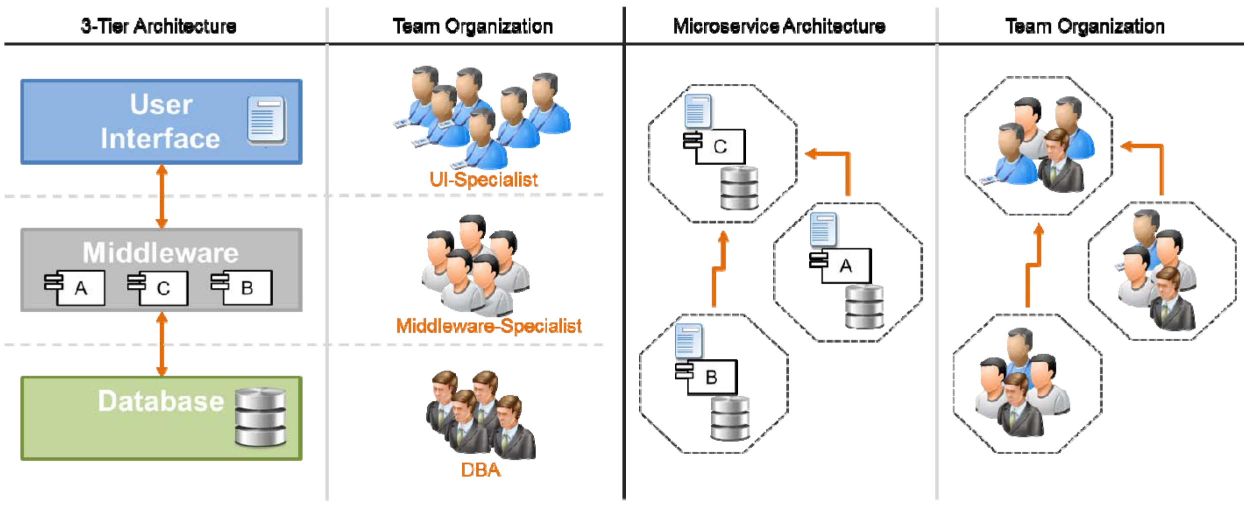


Figure 2: Dependencies between Team Organization and Software Architecture

and the team can redeploy its services whenever needed. Of course this also cannot be done without any cost. The team usually uses fully automated deployment mechanisms to put changes online. Also since a change of a service might affect other services, each team has to monitor service health – hence, the team is responsible for operation (often called devops [4]).

Along with the aforementioned characteristics, microservices foster the following principles [4]:

- Decentralized Governance
- Shared nothing
- Decentralized Data Management
- Smart endpoints & dumb pipes
- Infrastructure Automation
- Evolutionary Design

Evolutionary design seems to be obvious, because a team can change and redeploy its services at any time. More interesting concepts regarding development of modelling tools are shared nothing and smart endpoints & dumb pipes. Shared nothing means that services may use third party libraries but do not share code between each other nor do they share a common state. While shared code would impair the team, any form of shared state would be a limiting factor for system scalability. The difficulty is not only that information would be stored in multiple systems, but also that a state could change and all affected services then would need to synchronize changes immediately.

Smart endpoints & dumb pipes are a major difference between typical service-oriented (SOA, [16]) approaches and a microservice approach. In SOA often a component called Enterprise Service Bus (ESB) is employed to support communication. The ESB usually offers sophisticated services such as message routing and transformation, service orchestration and choreography, and even control of business rules.

Microservices use dumb pipes for communication in order to create application from services which are as decoupled and as cohesive as possible (services in style of UNIX or according to pipes and filters architectural pattern [4], [6]). As result intelligent processing only happens inside services (smart endpoint), while communication uses simple mechanisms (as REST or messaging over a lightweight message bus as RabbitMQ [4]). Message buses used in microservice architecture usually only provide some quality of service guarantees such as reliable asynchronous message distribution.

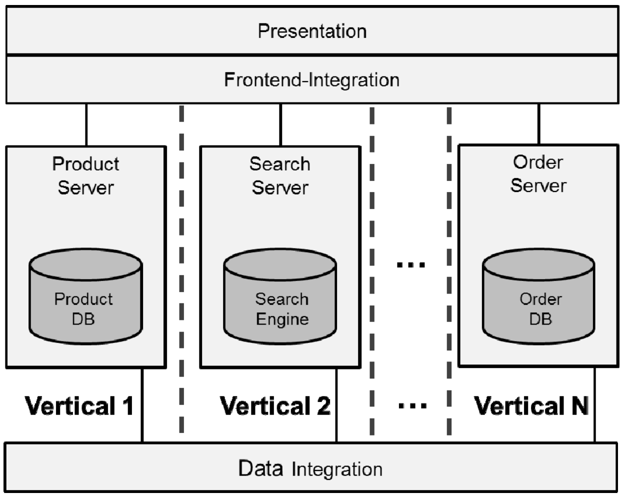


Figure 3: Decomposition according to Business Capabilities and Use Cases

One of the big challenges regarding the implementation of microservice architectures is the decomposition of the system into adequately tailored services. Therefore several different strategies and approaches are available [4]–[6]. One possibility to decompose an application is according to business capabilities or use cases. In contrast to classical monolithic systems where the application is split horizontally (see Figure 2) this approach decomposes the application vertically into cohesive subsystems also called verticals. Figure 3 demonstrates the vertical decomposition using the example of an e-commerce platform [5]. Each vertical comprises all technology tiers (presentation, business logic and data access). Multiple microservices may reside within each vertical. In this example additional layers are required in order to handle the communication between different verticals and integrate the results of different verticals into one page that is returned to the user.

III. ARCHITECTURE

Within this section we propose a microservice-based architecture to create tool support for business process modelling and analysis. Similar to the decomposition strategy presented in Section II.C the application's architecture is decomposed according to entities and resources. This means a service will cover all operations regarding a specific resource (e.g. a Petri Net, organizational resources or rules) [6]. Our resulting microservice architecture will be described below and is depicted in Figure 4.

All services are grouped according to four clusters of service types. These types comprise: ***a)*** editor, ***b)*** management, ***c)*** analysis functionality and ***d)*** presentation respectively. The fourth cluster (presentation) has been added in order to provide rendering and content integration functionality. However, for the reason of simplicity not all available services are displayed in the diagram. Additional services and types might be added at any time to extend or create tool support for specific scenarios. In order to fulfil the requirements of microservice architecture each service of the business process modelling application (see Figure 4) runs decentralized in a separate processes accessing an individual data store. This results in loose coupling and

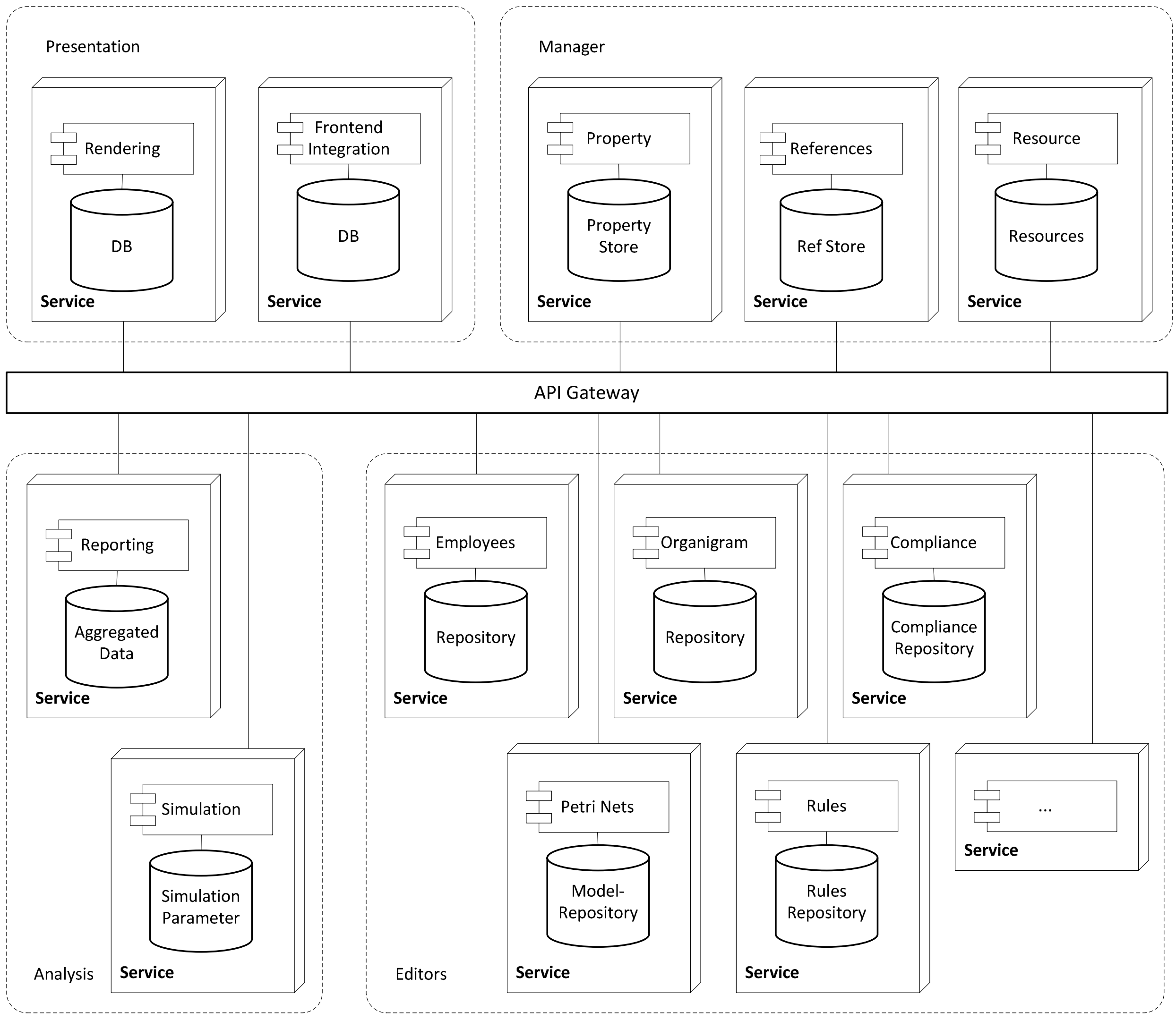


Figure 4: System Development using Microservice Architectural Style

provides the possibility to use a technology stack that is best suited for particular service – in terms of development: each team that develops a service may choose its own technology stack (e.g. Java, C#, Scala or PHP), which best fits their needs. A similar flexibility is achieved with respect to data storage, as there is no overall shared data base schema. Note that data may be stored redundantly in databases of different services. However, following the principle of eventual consistency [4] there is only one service determining the current state of a specific entity. All other services have to pull an entity's current state whenever this is needed.

The client-service communication as well as the communication between different services is implemented using restful web services. Although it is possible to call each service directly, the architecture proposed in this paper additionally comprises an API gateway for lightweight message passing. The API gateway helps to reduce the number of (concurrent) service calls and thereby increasing the performance (cp. [6]). Depending on the granularity the call is either proxied or results in multiple calls to fine grained services. However, in contrast to complex ESBs the API Gateway does not provide any sophisticated routing logic. According to the smart endpoint principle all logic remains within the services.

The UI integration of microservices is a challenging task for which different approaches such as hyperlinks, JavaScript utilizing an asset server as well as Server- or Edge-Side Includes (SSI respectively ESI) are existent. In order to use the business process modelling and analysis functionality different microservices have to be assembled into one page. This is implemented using the presentation microservices depicted in Figure 4. While the rendering microservice provides a skeleton of the page and implements the layout functionality the frontend-integration replaces all placeholders within the page skeleton with content as well as UI-elements delivered from the editor, analysis or management microservices.

As shown in Figure 4 the current architecture offers features covering a broad range of entities and resources relevant to business process modelling (e.g. Petri Nets, Rules and Compliance). In addition to modelling functionality the proposed architecture also comprises analytical functionality as well. This enables the user to analyse and simulate the business process' characteristics based on vast variety of properties. Analysis results can be aggregated by reporting services that provide the foundation for dashboard functionality. Since we utilize a microservice approach tool support may easily be adapted to user requirements, since choice of utilized microservices can be changed at any time and low cost (simply create another presentation service or add additional functional services).

IV. TOOLING SERVICES

Within the next two subsections we illustrate how a Petri Net editor can be implemented using our microservice based architecture depicted in Figure 4. The first section will introduce a set of services required to provide means for basic business process modelling and analysis functionality. Afterwards we demonstrate how these services can be assembled resulting in different editor configurations with different functionality.

A. Microservices for Business Process Modelling and Analysis

Depending on the granularity of decomposition a multitude of microservices is required to build BPM tool support. Since organisations often use a wide range of models to describe and to define structures, processes, data and other perspectives, a bunch of different models and languages can be observed in BPM tooling. We started to create our tool support not only based on microservices but also implemented an iterative tool development, therefore we started to focus on modelling business process models, specifically captured as Petri Net, in our first two iterations.

Within TABLE 1 we display an exemplary set of microservices that have been built and utilized to create tooling in the first place. These services provide basic modelling functionality, such as persistence of process models (Model Storage Service) or a set of modelling shapes (Toolbar Service). Consequently a next step was to create additional microservices to support more sophisticated analysis and reporting functionality, services created with this intent are being listed in TABLE 2.

TABLE 1: BASIC PETRI NET EDITOR SERVICES

Service	Description
Model Storage Service	The Model Storage Service belongs to the editor microservices. When the business process model is passed to this service, the microservice will store it into a business process repository (DB).
Modelling Pane Service	This microservice belongs to the editor services and provides a modelling pane that can be used to model the business process.
Toolbar Service	The Toolbar Service is an editor service. It returns the available tools (shapes) for the toolbar. With regard to the Petri Net Editor these are transitions (rectangle) places (circles) and arcs.
Touch Control Service	The Touch control Service returns a component that translates gestures into modelling shapes adding them to the modelling pane.
PNML Mail Export Service	This is an editor service that transforms the internal representation of the Petri net model into the standardized exchange model before sending it via mail to a recipient.
PNML File Export Service	This is an editor service that transforms the internal representation of the Petri net model into the standardized exchange model before storing it to the file system.

While basic modelling functionality allows capturing business processes further analysis functionality is required to enable detection of failures and optimization of processes. Such functionality is exemplified by a couple of services in TABLE 2. At the moment some analysis services such as reachability and simulation as well as reporting services have been implemented.

While our analysis functionality is currently based on behavioural analysis, we currently implement services to offer structural analysis [17] also. Future service functionality to extend analysis includes checks of the following properties: (structurally) boundedness, WF-Net, soundness and well-handledness. All of our analysis services operate on business process models. Since all analysis service use PNML as input, all services can be reused by any tool, which operates on this standardized format.

TABLE 2: ANALYSIS SERVICES

Service	Description
Reachability Service	This is an analysis service. The service calculates a reachability graph for a given business process model (Petri Net) under a given initial marking.
Token Game Service	The Token Game Service will calculate a set of activated transitions under a given net and a specified marking. Additionally it calculates a forecast.
Simulation Service	This service offers stochastic simulation of Petri Nets. For each simulation experiment a simulation trace log is generated.
Reporting Service	This service provides the graphical representation of analysis results. That means results returned from analysis services are processed to provide clearly arranged aggregations.

Besides microservices that provide model creation and storage functionality (see TABLE 1), additional microservices that integrate controls into user interfaces and display the content accordingly are also required (see Figure 3). This is essential to expose available functionality to end-users. Furthermore, depending on the vast variety of devices (e.g. PC, smart phone or tablet) that is used the user interface requirements may also vary. Due to loose coupling the aforementioned functionality can easily be reused by different rendering and frontend integration services in order to meet the respective UI requirements. Within the next subsection we will highlight how this could be achieved by demonstration of two different user interface configurations.

B. Editor Configurations

The Petri Net editor is currently offered in two configurations, that is a web-based configuration as well as a mobile app equivalent. Thus we empower the users to create and analyse models on different devices meeting their specific requirements.

Due to the microservice architecture it is possible to use different frontend-integration and rendering services for different devices. This means the user interface can be adapted to match the devices' specific requirements. Additionally, the integration services also control the functionality that is available within a specific configuration. Due to the microservice architecture it is easy to bundle an individual set of modelling and analysis functionality into one configuration. In the remainder of this section two configurations (mobile and web-based) with distinct functionality will be presented. The microservices used in the mobile configuration of the Petri net editor have been optimized to work on iOS and Android tablets. The mobile Petri Net editor [18] is based on web technologies like HTML5, CSS and JavaScript.

The major purpose of this editor has been to enable mobile modelling. Therefore the mobile Petri net editor integrates all microservices listed in TABLE 1 except for the service PNML File Export Service. This is due to the missing file system support of iOS devices. In order to improve usability two different kinds of input options have been implemented. The user can either drag and drop the model elements (Toolbar Service) or draw them with customizable gestures (Touch Control Service) as depicted in Figure 6. To support non-modelling-expert users in understanding the control flow a microservice that offers a simulation by token game has also been integrated (see Figure 5). Thus each user can check a control flow by firing enabled transitions.

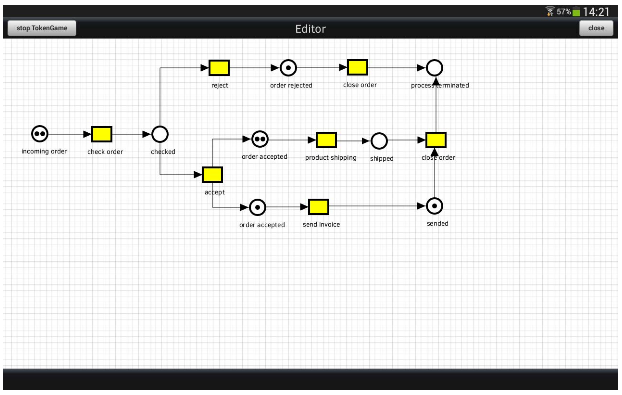


Figure 5: Mobile Petri Net Editor

In order to create the second configuration with manageable effort we reused the same technology stack to create a web-based Petri Net editor that supports standard browser technology on desktop systems (see Figure 7). In contrast to the mobile Petri Net editor the web-based configuration will be extended using microservices to provide sophisticated analysis and reporting (see TABLE 2) as well as file storage functionality (PNML File Export Service).

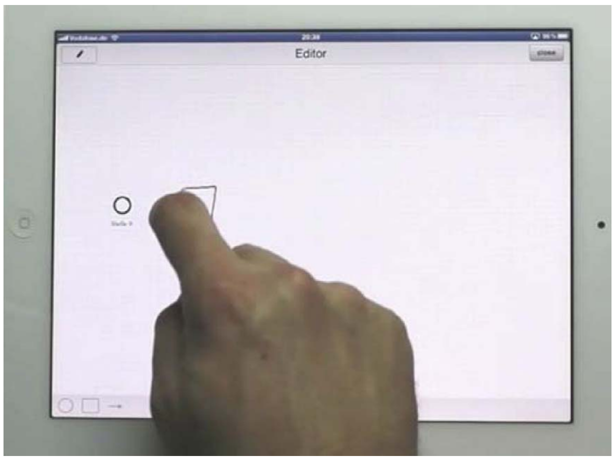
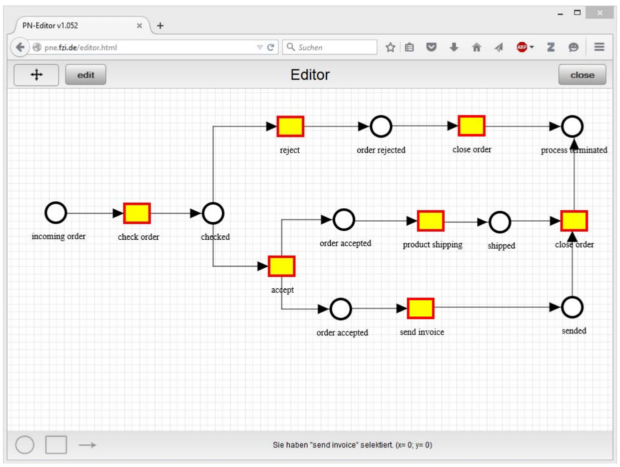


Figure 6: Business Process Modelling using Gestures

As mentioned before not all services suggested in our architecture have yet been integrated in the web-based Petri Net editor. However, a subset of microservices listed in Figure 4 has already been implemented (see TABLE 2). As part of our current development iteration these services are being integrated into the web-based configuration of the Petri Net editor.

At the moment we do not exploit all capabilities that arise from use of microservices in neither of our configurations. For example, all of our backend services are currently built in PHP. Storage of model data is currently only implemented on MySQL databases. Nevertheless all models could be utilized by any other tool that supports standardized modelling of Petri Nets, since we offer services that operate and exchange model data according standardized exchange formats (in this case PNML [19]). As mentioned the current version of our user interface only utilizes backend services implemented in PHP. Due to the pursued microservice approach the integration of services implemented using other languages and technologies is easily possible. Besides user interface development we recently created some analysis services in Java, which we plan to integrate in near future.



Sie haben "send invoice" selektiert. (x= 0; y= 0)

Figure 7: Petri Net Editor (web-based configuration)

V. Outlook and Conclusion

Within this paper we introduced the concept of microservices to tool support in business process modelling and analysis. In Section III we presented microservice architecture for BPM tools in general. The implementation of single aspects according to the suggested architecture has been demonstrated for a modelling tool based on Petri Nets (Section IV). We also outlined how the tool can easily be customized for specific scenarios (user specific, project specific and purpose specific). The high reusability of microservices as well as distributed service execution and data storage support improved scalability according to all 3 dimensions of the Abbot's scalability cube. Thus we created a highly flexible tool. Especially additional or different functionality and increased load by users can be handled more easily than with typical tool architectures (compare Section II). We demonstrated this by two different user interface implementations with different functionality (different use of services).

Future work involves the implementation of further functionality by additional microservices. As next step we will focus on user interface services as well as Petri Net analysis. Additionally we plan to develop services to support other process modelling languages. Since many companies utilize BPMN (hence possess many BPMN-models), we will also support this language in the future. Our plan is to integrate transformation services to import BPMN processes into our Petri Net tooling. As a subsequent step we are planning to integrate native BPMN language support (based on further services). Additionally, we plan to investigate whether there is a need to support a basic modelling language that focuses on the specific need of non-modelling-expert-users (such as domain experts) as described in [20].

In terms of microservice evaluation we plan to utilize different technology stacks in order to highlight the full potential of this approach. Further work will also focus on the aspect of scalability. We plan to test different configurations in order to investigate most efficient scalability dimension. Moreover we plan to integrate a lightweight message broker like Vert.x [21] or RabbitMQ [22] as API Gateway to route the messages between the microservices.

References

[1] W. M. P. van der Aalst, A. H. M. ter Hofstede, and M. Weske, "Business Process Management: A Survey" in *Business Process Management*, W. M. P. van der Aalst and M. Weske, Eds. Springer Berlin Heidelberg, 2003, pp. 1–12.

[2] "IBM Business Process Manager" 13-Aug-2015. [Online]. Available: http://www-03.ibm.com/software/products/de/business-process-manager-family. [Accessed: 13-Aug-2015].

[3] S. Sarkar, S. Ramachandran, G. S. Kumar, M. K. Iyengar, K. Rangarajan, and S. Sivagnanam, "Modularization of a Large-Scale Business Application: A Case Study" *IEEE Softw.*, vol. 26, no. 2, pp. 28–35, Mar. 2009.

[4] J. Lewis and M. Fowler, "Microservices" *martinfowler.com*. [Online]. Available: http://martinfowler.com/articles/microservices.html. [Accessed: 21-Jun-2015].

[5] G. Steinacker, "Scaling with Microservices and Vertical Decomposition" *dev.otto.de*.

[6] C. Richardson, "Microservices: Decomposing Applications for Deployability and Scalability" *InfoQ*. [Online]. Available: http://www.infoq.com/articles/microservices-intro. [Accessed: 04-Mar-2015].

[7] A.-W. Scheer, "ARIS Toolset: A software product is born" *Inf. Syst.*, vol. 19, no. 8, pp. 607–624, Dezember 1994.

[8] D. Karagiannis and N. Visic, "Next Generation of Modelling Platforms" in *Perspectives in Business Informatics Research*, J. Grabis and M. Kirikova, Eds. Springer Berlin Heidelberg, 2011, pp. 19–28.

[9] F. Schönthaler, G. Vossen, A. Oberweis, and T. Karle, *Business Processes for Business Communities: Modeling Languages, Methods, Tools*. Springer, 2012.

[10] "Software AG - webMethods" [Online]. Available: http://www.softwareag.com/de/products/az/webmethods/default.asp. [Accessed: 18-Aug-2015].

[11] "Microsoft Visio" [Online]. Available: https://products.office.com/visio. [Accessed: 05-Jul-2015].

[12] L. Vogel and M. Milinkovich, *Eclipse Rich Client Platform*, 3rd ed. Hamburg: Lars Vogel, 2015.

[13] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework, 2nd Edition*, 2nd ed. Addison-Wesley Professional. Part of the Eclipse Series series, 2008.

[14] M. L. Abbott and M. T. Fisher, *The Art of Scalability*. Upper Saddle River, NJ: Pearson Education, 2009.

[15] M. E. Conway, "How Do Committees Invent?" vol. Datamation magazine, 1968.

[16] Z. Mahmood, "Service oriented architecture: tools and technologies" *benefits*, vol. 6, p. 7, 2007.

[17] T. Murata, "Petri nets: Properties, analysis and applications" *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[18] S. Alpers, E. Eryilmaz, S. Hellfeld, and A. Oberweis, "Mobile Modeling Tool Based on the Horus Method" in *International Workshop on Advanced Information Systems for Enterprises*, Tunis, 2014, pp. 65–71.

[19] M. Weber and E. Kindler, "The Petri net markup language" in *Petri Net Technology for Communication-Based Systems*, Springer, 2003, pp. 124–144.

[20] J. Hess, C. Reuter, V. Pipek, and V. Wulf, "Supporting end-user articulations in evolving business processes: a case study to explore intuitive notations and interaction designs" *Int. J. Coop. Inf. Syst.*, vol. 21, no. 04, pp. 263–296, Dezember 2012.

[21] "Vert.x" [Online]. Available: http://vertx.io/. [Accessed: 05-Jul-2015].

[22] "RabbitMQ" [Online]. Available: https://www.rabbitmq.com/. [Accessed: 05-Jul-2015].