Interoperability of BPMN and MAML for Model-Driven Development of Business Apps

Christoph Rieger(✉)

ERCIS, University of Münster, Münster, Germany
christoph.rieger@uni-muenster.de

Abstract. With process models widely used as means for documentation and monitoring of business activities, the conversion into executable software often still remains a manual and time-consuming task. The MAML framework was developed to ease the creation of mobile business apps by jointly modeling process, data, and user interface perspectives in a graphical, process-oriented model for subsequent code generation. However, this domain-specific notation cannot benefit from existing process knowledge which is often encoded in BPMN models. The purpose of this paper is to analyze conceptual differences between both notations from a software development perspective and provide a solution for interoperability through a model-to-model transformation. Therefore, workflow patterns identified in previous research are used to compare both notations. A conceptual mapping of supported concepts is presented and technically implemented using a QVT-O transformation to demonstrate an automated mapping between BPMN and MAML. Consequently, it is possible to simplify the automatic generation of mobile apps by reusing processes specified in BPMN.

Keywords: BPMN · Business process modeling · Mobile app
Model transformation · Business app

1 Introduction

Business process models have been used for communicating, documenting, monitoring, and optimizing business processes for many years [19]. A wide variety of notations has emerged among which the Business Process Model and Notation 2.0 (BPMN) is best known [11]. However, process models need to be regularly synchronized with actual activities because most of them lack a sufficient level of detail to be directly executable using workflow engines. At the same time, companies are faced with global trends such as digitization and big data which require flexible business processes and new forms of organization in order to adapt to external influences.

At the intersection between process modeling and software specification, it is not possible to fully specify applications using solely BPMN models because of

B. Shishkov (Ed.): BMSD 2018, LNBIP 319, pp. 149–166, 2018.
https://doi.org/10.1007/978-3-319-94214-8_10

their limited focus on the sequence flow of activities. Essential software development perspectives such as data models, user interfaces, and user interactions are missing. In addition, user tasks are vaguely specified and thus hard to transform into adequate representations on screen.

Especially with the increased use of model-driven approaches, domain-specific modeling notations compete against traditional general-purpose modeling notations. Ideally, changes within models propagate automatically to the derived software artifacts, thus enabling a fast reaction to changing requirements without time-consuming development cycles by IT departments. One example is the Münster App Modeling Language (MAML) which provides a graphical notation understandable both for programmers and end users to specify mobile business apps on a high level of abstraction using data-driven processes [13]. Without need for manual programming, the framework automatically generates functional cross-platform app source code for mobile devices.

MAML models integrate the full spectrum of app specification in contrast to the narrower scope of BPMN which only covers a process perspective. Nevertheless, with BPMN as the de facto standard for documenting business processes, the research question guiding this work arises: How can previous process documentation encoded in BPMN models be reused to support cross-platform app specification with MAML? An automated transformation between both notations would therefore be beneficial with regard to consistency and potential time savings: Existing process models could be enriched with app-specific information and app models representing the course of activities could be converted to BPMN for documentation and analysis.

The main contributions of this paper are three-fold and reflected by its structure: Firstly, the capabilities and shortcoming of the BPMN notation are analyzed with regard to the specification of mobile apps (Sect. 3, which also briefly introduces the MAML notation). Secondly, supported workflow patterns for BPMN and MAML are compared to highlight the conceptual differences (Sect. 4). Thirdly, transformations from BPMN to MAML and vice versa are proposed to demonstrate the practicability of the approach (Sect. 5).

2 Related Work

Several approaches which deal with the representation and transformation from process models to executable artifacts have been discussed in scientific literature, e.g., using Petri nets [21] or YAWL models [6]. Their focus lies on supporting the model interpretation by workflow engines. In the context of software development, applications are often conceptually described using layered architectures in order to distinguish the required constituents of data model, business logic, and presentation layer and, consequently, manage complexity. This can be achieved using a combination of generic notations such as UML [7] or domain-specific frameworks [1,3] but not necessarily depicting the application contents as process sequence.

For reasons detailed in the following section, the BPMN notation is not sufficient to represent all aspects of the target application. With regard to process-aware information system modeling, three categories of notations can be distinguished, each exhibiting advantages and shortcomings. Firstly, BPMN is often used to describe the process perspective together with additional notations. For example, Brambilla et al. [2] use BPMN and the WebML language focused on user interaction in order to create rich internet applications, and Traettenberg and Krogstie [19] use the refinement by Diamodl diagrams for dialog specification.

Secondly, the BPMN 2.0 notation itself can be extended to annotate further data using *ExtensionDefinitions* and *ExtensionAttributeDefinitions*. This approach is used by workflow engines such as Camunda [4] but does not allow for visualization of annotated data or new custom diagram elements.

Thirdly, custom notations have been proposed that provide integrated modeling for workflows in general or tailored to specific domains. For example, Kannengiesser et al. [9] propose an approach with a custom notation to combine UI representations with process flows through UI-related task types, and Kalnins et al. [8] combine a graphical notation for process flows with a textual expression syntax to manipulate data.

The MAML notation we use for modeling mobile business apps – i.e., form-based, data-driven apps interacting with back-end systems [10] – falls into the latter category of a domain-specific graphical notation that combines control flow logic, UI, and data perspectives such that all application concerns are specified (see Sect. 3.2). However, the transformation approach does not aim to modify the BPMN meta model but instead extracts the maximum amount of information such that the modeler only needs to enrich missing elements. In this regard, our approach is most similar to the WebRatio BPM tool [3] which transforms BPMN models to the more technical WebML notation.

3 Business Process Notations for Mobile App Development

In this section, the most commonly used notation for business process modeling, BPMN, is analyzed regarding the development of process-driven mobile applications, and the domain-specific language MAML is introduced.

3.1 Business Process Model and Notation (BPMN)

BPMN is a control flow-based notation to define a sequence of process steps. The main graphical elements can be subdivided into flow objects (activities, gateways, and events) and artifacts (data objects, groups, and annotations) which are interlinked by connecting objects (sequence flows, message flows, and associations) and organized in roles representing actors or organizations (pools and lanes) [11]. A simplified model depicting the process of writing a thesis is shown in Fig. 1 and contains several typical task types, control flow branches, and data transfer

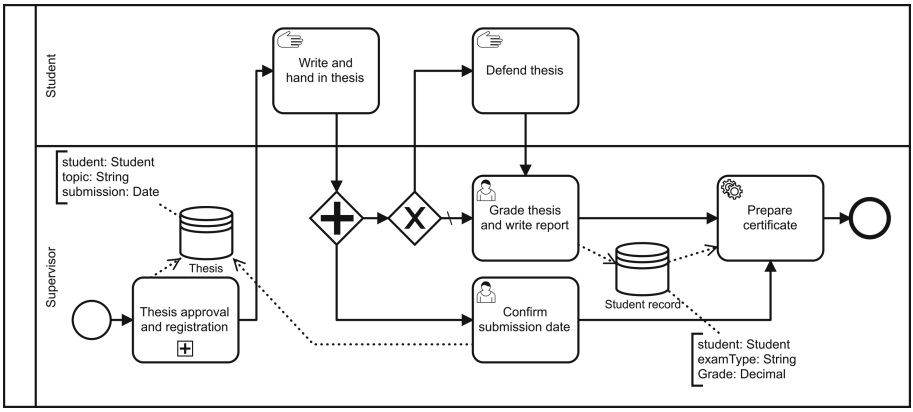

Fig. 1. Sample BPMN model representing a (simplified) process for thesis writing

mechanisms. It is further used to demonstrate how tool support for supervision-related tasks can be provided by transforming it into a mobile app.

Problems of the notation have already been studied in general [12] as well as for its applicability in software development projects [20]. The main criticism arises from deficiencies in business rule specification, symbol overload with partially superfluous or fuzzily delimited concepts, and the abundance of event types. The following paragraphs highlight the issues with regard to model-driven mobile development beyond the use of models for informally sketching process sequences.

Data Layer. BPMN is control-flow oriented and mostly neglects data modeling. Even with the enhancements of BPMN 2.0, the user can only specify the input and output of tasks using process-internal (*Data Input/Data Output/Data Object*) or persistent (*Data Store*) items and additionally distinguish between single items and collections as well as multiple states of the data. However, attributes, data types, and type interrelations – essential for the actual execution of data manipulation tasks – cannot be specified and can only be inferred on a generic level [5].

Business Logic Layer. BPMN models provide a wide range of elements to model the control-flow and conditional paths of execution. Whereas workflow engines support many such concepts, three characteristics of mobile devices complicate a direct utilization of generic BPMN elements.

Firstly, mobile apps focus on user-oriented tasks. In contrast to enterprise contexts in which many steps can be delegated to web services, apps focus on direct user interaction and require a specification of what actions to perform in a single step. Regarding data manipulation, this, e.g., includes whether an item is retrieved, updated, displayed, created, or deleted. Also, mobile-specific functionalities such as sensor/camera access and starting phone calls are not available.

Secondly, mobile devices may experience connectivity issues at any time and for unknown duration, causing unreliable response times if workflow instances are strictly allocated to one device. Similarly, multi-step transactions may lock the whole system.

Thirdly, mobile apps are usually small and inherently distributed systems. Parallel execution of tasks is hardly possible on small screens (and questionable as regards frequent context switches). Even considering a multi-role context in which activities are potentially performed by different users collaboratively, issues arise from the coordination of up-to-date distributed data and conflicting concurrent modifications.

Presentation Layer. Although default representations could be derived from the (unavailable) data attributes to display, a minimal specification of sensible user interfaces also requires the addition of informative texts and means of controlling the user interactions to navigate between views. Neither exists in BPMN.

3.2 Muenster App Modeling Language (MAML)

MAML is a graphical domain-specific language with the purpose of modeling cross-platform business apps. Following the model-driven paradigm, the framework allows for the fully automated generation of app source code for multiple smartphone platforms (currently Android and iOS) as well as a Java-based backend component for app coordination and as interface to other company systems.

The notation is built around five main design goals: (1) code-less cross-platform app creation, (2) a strong domain expert focus in contrast to complex technical notations, (3) a high level of abstraction by modeling data-driven processes, (4) task-oriented modularization using use cases, and (5) a declarative and platform-neutral description of logical steps in contrast to UI-centered configurators. Figure 2 depicts the same scenario as Fig. 1, however, it contains all the information to generate the app.

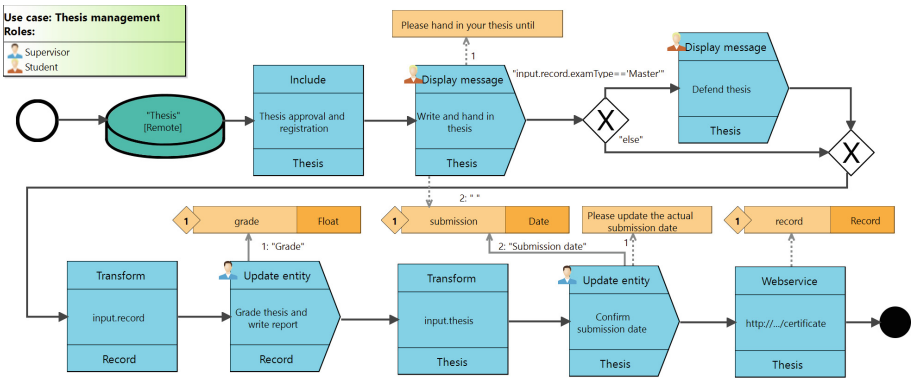

Fig. 2. Sample MAML model with thesis management process equivalent to Fig. 1

Although not necessarily limited to the domain of mobile apps, the notation considers interoperability with mobile device hardware (e.g., using elements for camera usage or sensor access), mobile interaction patterns (such as starting phone calls), and current generators target the Android and iOS platforms. Also, the integrated modeling approach to process flows and data structures may not be well suited for large-scale applications. For a more detailed introduction to MAML the reader is referred to [13,14].

4 Comparison of Workflow Patterns in MAML and BPMN

Russell et al. [17] have collected an extensive list of workflow patterns. As a comprehensive overview on workflow patterns also comprises a data access and resource perspective, respective patterns have been presented in [15,16]. To prepare a possible mapping of concepts, supported patterns need to be assessed first for both notations. An analysis of the BPMN notation is provided in [22]. Therefore, this section evaluates workflow concepts of MAML according to the pattern catalog and highlights the most important differences.

4.1 Control-Flow Patterns

Considering Workflow Control Patterns (WCP), basic patterns include the possibility to depict a *Sequence* (WCP1) of activities, a *Parallel Split* (WCP2) from a single thread of control into multiple threads executed in parallel, the *Synchronization* (WCP3) of multiple parallel activities into a single thread of control, an *Exclusive Choice* (WCP4) to execute one of multiple alternatives according to given conditions, and a *Simple Merge* (WCP5) to consolidate multiple alternative branches. In MAML, WCP1 is given by the corresponding "process connector" construct. However, as explicated in Subsect. 3.1, WCP2 and WCP3 as well as advanced branching and synchronization patterns including inter-workflow parallelism are not supported. Figure 3 depicts how branching the control flow (WCP4) in MAML can be based on a user decision (a) or evaluated based on the state of data objects (b), and can be merged accordingly (WCP5; (c)). Unlike BPMN, implicit branching and merging is disallowed to foster consistency and explicitly indicate control flow variations.

Regarding structural patterns, *Deferred Choice* (WCP16) represents a runtime choice between different branches of which the first executes. In MAML, this is possible to a certain extent: If multiple roles are assigned to one task, (only) the first user will execute the task (d). *Interleaved Parallel Routing* (WCP17) refers to a partial ordering of process step dependencies to be linearized and executed sequentially. *Critical Section* (WCP39) and *Interleaved Routing* (WCP40) are similar patterns that specify a set of subgraphs or individual activities to be executed once in arbitrary sequential order. To clarify the future app structure, MAML currently relies on an explicitly modeled, fixed sequence of activities.

Table 1. Workflow Control Patterns according to [17] in BPMN [22,23] and MAML

Workflow Control Pattern (WCP)		BPMN	MAML
1	Sequence	+	+
2	Parallel Split	+	−
3	Synchronization	+	−
4	Exclusive Choice	+	+
5	Simple Merge	+	+
6	Multi-Choice	+	+/−
7	Structured Synchronizing Merge	+	−
8	Multi-Merge	+	+
9	Structured Discriminator	+/−	−
10	Arbitrary Cycles	+	+
11	Implicit Termination	+	+
12	Multiple Instances without Synchronization	+	−
13	Multiple Instances with a Priori Design-Time Knowledge	+	−
14	Multiple Instances with a Priori Run-Time Knowledge	+	−
15	Multiple Instances without a Priori Run-Time Knowledge	−	−
16	Deferred Choice	+	+
17	Interleaved Parallel Routing	−	−
18	Milestone	−	+/−
19	Cancel Activity	+	+/−
20	Cancel Case	+	+
21	Structured Loop	+	+/−
22	Recursion	−	+
23	Transient Trigger	−	−
24	Persistent Trigger	+	+/−
25	Cancel Region	+/−	+/−
26	Cancel Multiple Instance Activity	+	−
27	Complete Multiple Instance Activity	−	−
28	Blocking Discriminator	+/−	−
29	Canceling Discriminator	+	−
30	Structured Partial Join	+/−	−
31	Blocking Partial Join	+/−	−
32	Canceling Partial Join	+/−	−
33	Generalized AND-Join	+	−
34	Static Partial Join for Multiple Instances	+/−	−
35	Canceling Partial Join for Multiple Instances	+/−	−
36	Dynamic Partial Join for Multiple Instances	−	−
37	Local Synchronizing Merge	−	−
38	General Synchronizing Merge	−	−
39	Critical Section	−	−
40	Interleaved Routing	+/−	−
41	Thread Merge	+	−
42	Thread Split	+	−
43	Explicit Termination	+	+

A *Milestone* (WCP18) represents the ability to execute an activity until a state-dependent execution point is reached. An exclusive choice with state-based condition (b) can be reused to represent this in MAML. BPMN also implements the previous patterns but does not support the concept of workflow state at all.

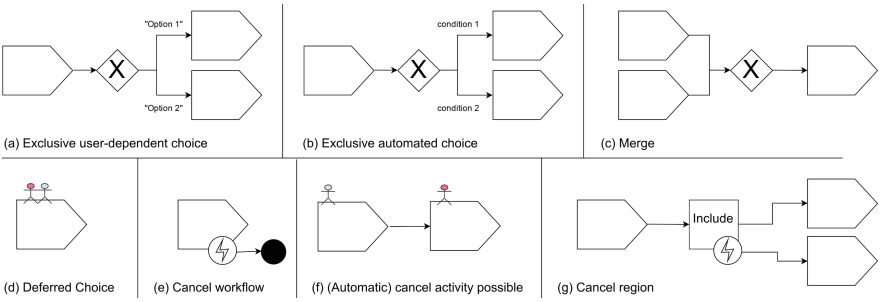

Fig. 3. Basic control-flow and cancellation patterns in MAML

Concerning cancellation patterns, MAML supports the *Cancel Case* (WCP20) pattern (e) to stop the current workflow instance. As tasks are allocated to roles – not individual users – the same process instance might also be started accidentally from multiple devices and automatically canceled for all but the first execution according to the *Cancel Activity* (WCP19) pattern (although not explicitly modeled that can occur in situations such as (f)). *Cancel Region* (WCP25) denotes a subgraph being canceled and can be represented in MAML through a subprocess terminating unsuccessfully (g).

Within a process model, *Arbitrary Cycles* (WCP10) with multiple end points can occur as depicted in Fig. 4(a). Regarding *Structured Loops* (WCP21) to execute activities repeatedly, two representations are possible: A loop is explicitly defined using an XOR element with an expression to test as condition before (for loop; (b)) or after (do-while loop; (c)) the activities are executed. In addition, users may choose multiple elements in a "select entity" step such that subsequent activities are performed for each of the selected elements. In contrast to BPMN, *Recursion* (WCP22) is possible because of passing data to subsequent process elements (d) instead of BPMN's token-based approach to sequence flows. *Persistent Triggers* (WCP24) can be used to start tasks based on a data condition (e). Finally, MAML supports both *Explicit Termination* (WCP43; (f)) and *Implicit Termination* (WCP11; (g)) of workflow instances, although the latter is discouraged to avoid incomplete process branches. Table 1 summarizes the full (+), partial (+/−), or lacking support of the workflow control patterns for BPMN and MAML.

4.2 Data Patterns

A variety of data patterns (WDP) complements the process view of a workflow (cf. Table 2). In MAML, a global perspective is adopted to define the flow of

Table 2. Workflow Data Patterns according to [16] in BPMN [22,23] and MAML

Workflow Data Pattern (WDP)		BPMN	MAML
1	Task Data	+	+
2	Block Data	+	+
3	Scope Data	+	+/−
4	Multiple Instance Data	+	+/−
5	Case Data	+	+
6	Folder Data	+	−
7	Workflow Data	+	+
8	Environment Data	+	+
9	Task to Task	+/−	+
10	Block Task to SubWorkflow Decomposition	+	+
11	SubWorkflow Decomposition to Block Task	+	+
12	To Multiple Instance Task	+	−
13	From Multiple Instance Task	+	−
14	Case to Case	+	−
15	Task to Environment - Push-Oriented	−	+
16	Environment to Task - Pull-Oriented	+	+
17	Environment to Task - Push-Oriented	−	−
18	Task to Environment - Pull-Oriented	−	−
19	Case to Environment - Push-Oriented	+	+
20	Environment to Case - Pull-Oriented	+	+
21	Environment to Case - Push-Oriented	+	−
22	Case to Environment - Pull-Oriented	−	−
23	Workflow to Environment - Push-Oriented	−	+
24	Environment to Workflow - Pull-Oriented	+	+
25	Environment to Workflow - Push-Oriented	+/−	−
26	Workflow to Environment - Pull-Oriented	+	−
27	Data Transfer by Value - Incoming	−	−
28	Data Transfer by Value - Outgoing	+/−	−
29	Data Transfer - Copy In/Copy Out	+	−
30	Data Transfer by Reference - Unlocked	+/−	+
31	Data Transfer by Reference - With Lock	+/−	−
32	Data Transformation - Input	+/−	−
33	Data Transformation - Output	+	−
34	Task Precondition - Data Existence	+/−	+/−
35	Task Precondition - Data Value	+/−	+/−
36	Task Postcondition - Data Existence	−	+/−
37	Task Postcondition - Data Value	−	+/−
38	Event-Based Task Trigger	−	−
39	Data-Based Task Trigger	−	+
40	Data-Based Routing	+/−	+

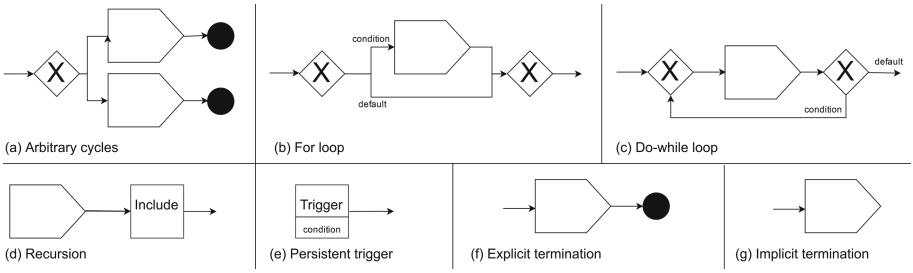

Fig. 4. Iteration, trigger, and termination patterns in MAML

data objects through the process. This represents the *Workflow Data* (WDP7) concept as depicted in Fig. 5(a). Only data required in a certain step is modeled and the state is passed to subsequent process steps, thus effectively also implementing the *Task Data* (WDP1) and *Block Data* (WDP2) patterns. *Case Data* (WDP5) representing objects specific to a process instance, and *Scope Data* (WDP3) denoting custom regions of data visibility can be achieved with local data storage which is transferred to the global data store at a later point of process execution (c). *Environment Data* (WDP 8) from the operating system can also be incorporated, for example by accessing the camera (d) or GPS location (e). Again, patterns related to concurrency are not supported, but the same task may be executed with different data objects in loops (*Multiple Instance Data*; WDP4; (f)).

This broad set of data visibility patterns reaches beyond the capabilities of BPMN which is limited to task, block, and case data through the use of parameters.

Regarding internal data interaction patterns, MAML and BPMN have similar capabilities. Data can be passed from *Task to Task* (WDP9; e.g. (a) in Fig. 5), from Block Task to *SubWorkflow Decomposition* (WDP10; e.g. (d) in Fig. 4), and *SubWorkflow Decomposition to Block Task* (WDP11) by simply connecting the respective process elements. *Data interaction To/From Multiple Instance Tasks* (WDP12–13) are unavailable for lack of concurrency. Also, *Case to Case* (WDP14) data interaction is not intended because of distributed execution on mobile devices with varying connectivity. External data interaction in MAML relies on the global data view and the user being in control of data interactions. Therefore, data can be transferred between task and environment in pull fashion using web services (WDP16; (h) in Fig. 5) as well as push-oriented (WDP15) using web services (i) or implicit/explicit remote data storage (j/k). As task data is available to the subsequent tasks of the case, this also applies to push- and pull-oriented data transfer between the environment and the case/workflow (WDP19–20, WDP23–24). All data is passed by reference (WDP30) without locking mechanisms to support low-connectivity scenarios. In contrast, BPMN supports value- and reference-based data transfer with locks as well as additional

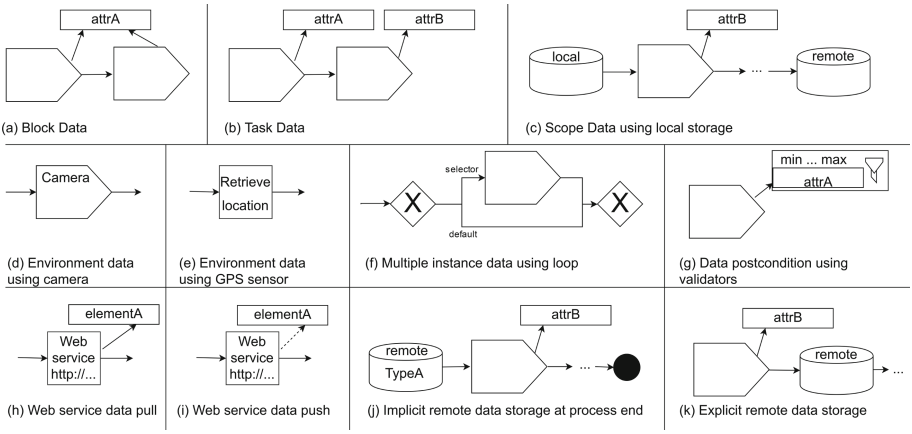

Fig. 5. Data visibility, interaction, and routing patterns in MAML

in-/output transformations, but only regarding task interactions with the environment (WDP15–18, WDP27–29, WDP32–33).

Finally, *Task Preconditions* (WDP34–35) are supported using the exclusive choice construct in MAML (cf. (b) in Fig. 3) but can flexibly validate the value of the data item in contrast to BPMN's more limited check for data existence. *Task Postconditions* (WDP36–37) can be applied equivalently or alternatively using data validators ((g) in Fig. 5). *Data-Based Task Triggers* (WDP39) (cf. (e) in Fig. 4) and *Data-Based Routing* (WDP40) possibilities (cf. (b) in Fig. 3) exist both in MAML and BPMN.

4.3 Resource Patterns

Concerning workflow resource patterns (WRP; cf. Table 3), both notations share the concept of roles to distribute tasks. A role can therefore represent a single actor (WRP1) or a group of resources (WRP2) as depicted in Fig. 6(a) and (b). The role concepts also serves as *Authorization* (WRP4) because distinct apps are created which allow only the execution of tasks related to that role.

Whereas work items in BPMN are allocated to resources (WRP14), the mobile context of MAML fosters an on-demand, pull-oriented execution of work items. Work items are *Distributed by Offer to Multiple Resources* (WRP13) and each worker can initiate the immediate execution of work items (WRP23) while having the *Selection Autonomy* (WRP26) to choose tasks from a list of open work items. To limit the amount of context switches in the distributed execution of workflows, MAML implements the *Retain Familiar* (WRP7) pattern which by default starts the next workflow step unless the current user does not comply with any of the assigned roles (c). If all tasks are assigned to the same role or only one role is defined for the use case without explicit assignment (d), this also allows for the *Case Handling* (WRP6) pattern in which all activities are performed by the same individual. Otherwise, subsequent roles are automatically

Table 3. Workflow Resource Patterns according to [15] in BPMN [22,23] and MAML

Workflow Resource Pattern (WRP)		BPMN	MAML
1	Direct Allocation	+	+
2	Role-Based Allocation	+	+
3	Deferred Allocation	–	–
4	Authorization	–	+/–
5	Separation of Duties	–	–
6	Case Handling	–	+/–
7	Retain Familiar	–	+
8	Capability Based Allocation	–	–
9	History Based Allocation	–	–
10	Organizational Allocation	–	–
11	Automatic Execution	+	+
12	Distribution by Offer - Single Resource	–	–
13	Distribution by Offer - Multiple Resources	–	+
14	Distribution by Allocation - Single Resource	+	–
15	Random Allocation	–	–
16	Round Robin Allocation	–	–
17	Shortest Queue	–	–
18	Early Distribution	–	–
19	Distribution on Enablement	+	+
20	Late Distribution	–	–
21	Resource-Initiated Allocation	–	–
22	Resource-Initiated Execution - Allocated Work Item	–	–
23	Resource-Initiated Execution - Offered Work Item	–	+
24	System Determined Work Queue Content	–	–
25	Resource-Determined Work Queue Content	–	–
26	Selection Autonomy	–	+
27	Delegation	–	–
28	Escalation	–	–
29	Deallocation	–	–
30	Stateful Reallocation	–	–
31	Stateless Reallocation	–	–
32	Suspension/Resumption	–	–
33	Skip	–	–
34	Redo	–	–
35	Pre-Do	–	–
36	Commencement on Creation	+	+
37	Commencement on Allocation	–	–
38	Piled Execution	–	–
39	Chained Execution	+	+
40	Configurable Unallocated Work Item Visibility	–	–
41	Configurable Allocated Work Item Visibility	–	–
42	Simultaneous Execution	+	+
43	Additional Resources	–	–

notified about the process instance for *Chained Execution* (WRP39). Also, complying with the *Commencement on Creation* (WRP36) pattern, new use case instances can be initiated for any process whose first task matches a user's role.

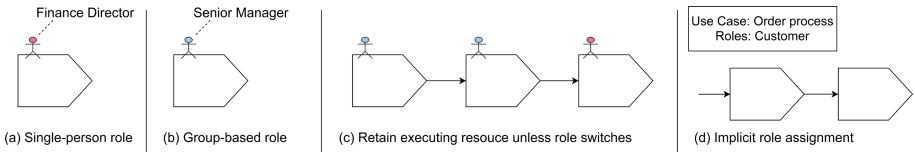

Fig. 6. Resource patterns in MAML

Automatic Execution (WRP11), i.e. trigger-based workflow execution without explicit resource allocation, is possible both in BPMN and in MAML (see (e) in Fig. 4). Also, there are no theoretical constraints on how many instances of a task are executed simultaneously (WRP42) by different users with the same role.

5 Model-to-Model Transformation

MAML models provide integrated representations for process-driven apps but need to be created from scratch. On the other hand, business processes in many organization are documented in BPMN models, although they lack the ability to fully describe mobile applications. To exploit the benefits of interoperability, a model-to-model transformation is presented based on the QVT-O language [18].

5.1 Mapping of Equivalent Language Constructs

Many core elements of the notations have direct correspondences, for instance in the thesis organization system depicted in Figs. 1 and 2. Firstly, both notations have the concepts of roles that can represent individuals or groups of resources. A MAML *use case* is represented as *pool* with a *lane* for each role in BPMN. Conversely, a process element is annotated with the respective *participant*.

Secondly, *user tasks* in BPMN correspond to *process elements* in MAML. A keyword-based matching strategy tries to classify the more specific MAML task type (e.g., *Create entity*) according to its description. Further mappings include:

- *Sub-process tasks* and *service tasks* have equivalent MAML elements named *Include* and *Webservice*
- *Manual tasks*, i.e. application-external actions, can be regarded as *Display message* steps which only output information and allow to proceed
- *Script tasks* indicate automated actions by the process engine. They need to be converted to web services as no arbitrary code can be executed.

– *Call activities* are created only for MAML's location retrieval, camera, and phone calls steps to reflect these global tasks. Other BPMN call tasks are transformed like regular sub-process tasks.

Thirdly, *start*, *end*, *timer*, and *error events* have the same semantics in both notations. Regarding data flows, this also holds true for inputs and outputs using *data associations* in BPMN and *parameter connectors* in MAML.

5.2 Mapping of Related Language Constructs

MAML applications are data-driven and therefore consider data flows together with the process flow such that each process element operates on one or multiple data items which are persisted and globally available after the process completes. Within the transformation, the respective MAML *data sources* need to be retrieved for each activity and annotated as BPMN *data store*. In addition, corresponding *InputSets* and *OutputSets* for an activity's *InputOutputSpecification* need to be created. However, nested attributes cannot be represented in BPMN and are lost during the transformation.

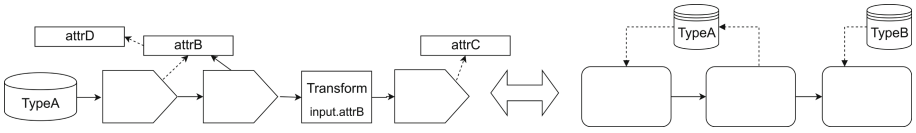

Fig. 7. Data transformation between MAML (left) and BPMN (right)

Conversely, data stores associated with BPMN tasks need to be integrated in the MAML process flow. In case the data item to process differs between tasks, additional *Transform* elements are needed to establish the link (see Fig. 7).

Without direct support for concurrency, several workflow patterns can be rewritten to concepts available in MAML. *Parallel gateways* and (equivalent) *implicit parallel splits* are for example transformed to a sequence of activities as depicted in Fig. 8(a). With exclusive gateways to branch process flows, inclusive gateways are mapped to a series of optional steps (b). Similarly, loops (WRP21; (b/c) in Fig. 4) and milestones can be transformed using automatically evaluated, state-based decisions.

5.3 Unmapped Language Constructs

BPMN contains several elements without equivalent representation in MAML. Because unstable mobile connectivity precludes concurrency, BPMN's *conversations* and *choreographies* are unsupported in MAML. In addition, *multiple instance tasks*, *(parallel) multiple events*, and *message flows* are not available in MAML and cannot be mapped to equivalent concepts because of the inherently sequential design of smartphone apps. Similarly, protocols for BPMN's

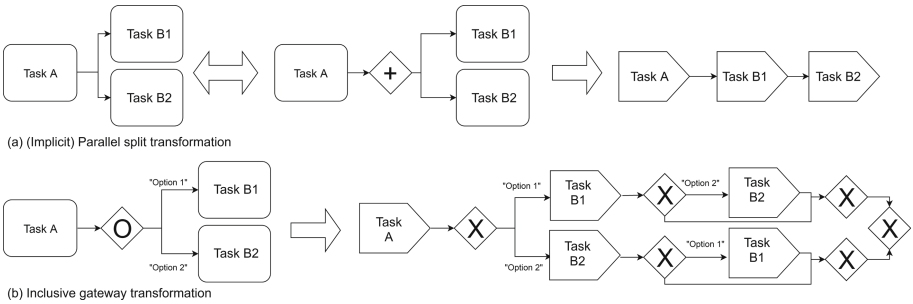

Fig. 8. Gateway transformation from BPMN to MAML

transactional tasks, *compensation*, and *cancel events* contradict the distributed characteristics of mobile systems.

At the current state of development, MAML also does not contain complex control flows for two reasons. Firstly, the model-driven approach aims to actually create executable source code. Features equivalent to BPMN's *signal*, *escalation*, or *termination events* and respective *event-based gateways* need to have intuitive user interactions in the app. Secondly, some elements in BPMN such as *complex gateways*, *business rule tasks*, and *compensation* are less structured or delegate processing to a workflow engine. However, each element requires an adequate in-app representation and a sufficient level of detail in the model to allow for an automated transformation.

On the other hand, some elements in MAML have no adequate representation in BPMN. Most important, custom data types and data objects with nested attribute structures reach far beyond BPMN's capability of specifying primitive key-value pairs as properties of activities. For the same reason, data-driven features such as *computed attributes* and data-dependent variations unknown at design time (*SingleResultEvent*/*MultiResultEvent*) have no equivalent representation in BPMN. Also, interface-related information such as labels, buttons, and field captions is not in the scope of BPMN.

6 Discussion

Section 4 explained the capabilities of MAML and contrasts them to the established BPMN notation. The transformation result of the BPMN model in Fig. 1 is displayed in Fig. 2 (enriched with label texts). As can be seen, many concepts are shared by both notations, although MAML does not strive for being a feature-complete workflow engine. Especially the data layer is currently not well represented in BPMN and has no direct interrelation with data modeling languages such as UML. Workflow engines such as Camunda bypass some of the limitations with custom extension attributes, but data-centric actions still need to be extracted into programmed web services. In addition, the granularity of modeled processes varies strongly and even with a detailed representation,

the modeler might often apply mobile-specific adaptations to the representation and enrich the model with UI elements. We therefore refrain from extending BPMN for our needs. Instead, our transformation eases the import of potentially already existing process documentation and mobile app design. Conversely, small enterprises which have only process documentation on a higher level of abstraction can reuse models used for app generation and export a detailed BPMN representation. As the model-driven paradigm focuses on this primary artifact, the process documentation always remains synchronized with the actual implementation, both regarding process flows and data models which can be derived together with access restrictions directly from the MAML representation (cf. [14] for more details on the data model inference). Particularly for small and medium enterprises, MAML can therefore be seen as a tool supporting digitized business models and increasing quality of operations through custom business apps – eventually even designed by end users within the company. A transformation from and to BPMN thus unburdens departments by simplifying the specification of software and documentation of activities.

Limitations of our work include both improvements to the transformation between BPMN snd MAML as well as its practical application. On the one hand, the evolution of the domain-specific MAML notation may introduce workflow patterns already found in BPMN. Also, previously unmapped language constructs with no adequate representation might be substituted by app-related concepts in future (e.g., establishing conventions for utilizing event-based gateways in the context of notifications). On the other hand, a real-world use in companies with different engagement in process documentation is required to further validate the applicability of the approach and constitutes future work. By analyzing existing models of different granularities, recurring patterns may be uncovered that can also be used for improving the transformation.

7 Conclusion and Outlook

In this paper, the interoperability between the established process modeling notation BPMN and the domain-specific graphical modeling language MAML for mobile-app creation have been assessed. Therefore, MAML was analyzed based on 43 control flow, 40 data, and 43 resource patterns identified for workflows by Russell et al. [15–17]. Although many correspondences exist with regard to control flow, data, and resource handling, differences relating to concurrency, complex control flows, and transactional behavior remain due to different application domains of BPMN using synchronous workflow engines and MAML in distributed mobile environments. Nonetheless, a mapping between many concepts can be found and an implemented model-to-model transformation using QVT-O demonstrates the transferability of concepts. From MAML to BPMN, fully specified models are created. In the opposite direction, additional data- and UI-related elements need to be added due to their absence in the specification of BPMN. Thus, the development process of mobile business apps using MAML can indeed be supported by reusing process knowledge documented in BPMN

models. Regarding future work, a real-world introduction of the approach is pending in order to validate the transformation rules, especially considering the detailedness of input models encountered in practice.

References

1. Barnett, S., Avazpour, I., Vasa, R., Grundy, J.: A multi-view framework for generating mobile apps. In: IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), pp. 305–306 (2015). https://doi.org/10.1109/VLHCC.2015.7357239
2. Brambilla, M., Preciado, J.C., Linaje, M., Sanchez-Figueroa, F.: Business process-based conceptual design of rich internet applications. In: ICWE, pp. 155–161 (2008). https://doi.org/10.1109/ICWE.2008.22
3. Brambilla, M., Butti, S., Fraternali, P.: WebRatio BPM: a tool for designing and deploying business processes on the web. In: Benatallah, B., Casati, F., Kappel, G., Rossi, G. (eds.) ICWE 2010. LNCS, vol. 6189, pp. 415–429. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13911-6_28
4. Camunda Services GmbH (2017). BPMN workflow engine. https://camunda.org/
5. Cruz, E.F., Machado, R.J., Santos, M.Y.: From business process modeling to data model: a systematic approach. In: QUATIC, pp. 205–210 (2012). https://doi.org/10.1109/QUATIC.2012.31
6. Decker, G., Dijkman, R., Dumas, M., García-Bañuelos, L.: Transforming BPMN diagrams into YAWL nets. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 386–389. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85758-7_30
7. de Giacomo, G., Oriol, X., Estañol, M., Teniente, E.: Linking data and BPMN processes to achieve executable models. In: Dubois, E., Pohl, K. (eds.) CAiSE 2017. LNCS, vol. 10253, pp. 612–628. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59536-8_38
8. Kalnins, A., Lace, L., Kalnina, E., Sostaks, A.: DSL based platform for business process management. In: Geffert, V., Preneel, B., Rovan, B., Štuller, J., Tjoa, A.M. (eds.) SOFSEM 2014. LNCS, vol. 8327, pp. 351–362. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04298-5_31
9. Kannengiesser, U., Heininger, R., Gründer, T., Schedl, S.: Modelling the process of process execution: a process model-driven approach to customising user interfaces for business process support systems. In: Schmidt, R., Guédria, W., Bider, I., Guerreiro, S. (eds.) BPMDS/EMMSAD -2016. LNBIP, vol. 248, pp. 34–48. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39429-9_3
10. Majchrzak, T.A., Ernsting, J., Kuchen, H.: Achieving business practicability of model-driven cross-platform apps. OJIS **2**(2), 3–14 (2015). https://doi.org/10.19210/OJIS_2015v2i2n02_Majchrzak
11. Object Management Group (2011). BPMN: http://www.omg.org/spec/BPMN/2.0
12. Recker, J.: Opportunities and constraints: the current struggle with BPMN. Bus. Process Manag. J. **16**(1), 181–201 (2010). https://doi.org/10.1108/14637151011018001
13. Rieger, C.: Evaluating a graphical model-driven approach to codeless business app development. In: HICSS, pp. 5725–5734 (2018)

14. Rieger, C., Kuchen, H.: A process-oriented modeling approach for graphical development of mobile business apps. COMLAN **53**, 43–58 (2018). https://doi.org/10.1016/j.cl.2018.01.001
15. Russell, N., ter Hofstede, A., van der Aalst, W.: Workflow resource patterns. BETA Working Paper Series, WP 127 (2004)
16. Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Workflow data patterns: identification, representation and tool support. In: Delcambre, L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, O. (eds.) ER 2005. LNCS, vol. 3716, pp. 353–368. Springer, Heidelberg (2005). https://doi.org/10.1007/11568322_23
17. Russell, N., ter Hofstede, A., van der Aalst, W., Mulyar, N.: Workflow control-flow patterns: a revised view. BPM Center Report BPM-06-22 (2006)
18. The Eclipse Foundation (2016). QVT Operational: http://www.eclipse.org/mmt/qvto
19. Trætteberg, H., Krogstie, J.: Enhancing the usability of BPM-solutions by combining process and user-interface modelling. In: Stirna, J., Persson, A. (eds.) PoEM 2008. LNBIP, vol. 15, pp. 86–97. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89218-2_7
20. Tuomainen, M., Mykkänen, J., Luostarinen, H., Pöyhölä, A., Paakkanen, E.: Model-centric approaches for the development of health information systems. In: Studies in Health Technology and Informatics, vol. 129 (2007)
21. van der Aalst, W.M.P.: Putting high-level Petri nets to work in industry. Comput. Ind. **25**(1), 45–54 (1994). https://doi.org/10.1016/0166-3615(94)90031-0
22. Wohed, P., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M., Russell, N.: On the suitability of BPMN for business process modelling. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 161–176. Springer, Heidelberg (2006). https://doi.org/10.1007/11841760_12
23. Workflow Patterns Initiative (2017). http://www.workflowpatterns.com/