Pre-Modelled Flexibility for Business Processes

Thomas Bauer
Hochschule Neu-Ulm, University of Applied Sciences, Wileystr. 1, 89231 Neu-Ulm, Germany

Keywords: Business Process, Process Modelling, Build-Time, Process Execution, Flexibility.

Abstract: At process-aware information systems (PAIS), sometimes, a flexible deviation from the rigidly designed process becomes necessary. Otherwise the users would be restricted too much. This paper presents an approach that allows to define the expected flexibility requirements only once already at build-time and apply them at run-time in the PAIS. Compared to dynamic changes during run-time, this has the advantage that the usage of the pre-defined information reduces the effort for the end users at each deviation. In addition, applying flexibility becomes saver; e.g., since user rights can be defined. This paper presents the corresponding requirements, with a special focus on the kind of information that has to be pre-defined at build-time. Thereby, all relevant process aspects were respected and the necessity of the requirements is illustrated with examples from practice.

1 INTRODUCTION

An advantage of PAIS (Reichert and Weber 2012), compared to traditional IT systems, is that the process management system (PMS) guarantees the adherence of the defined business process (BP). Additionally, end users are unburdened from non-productive tasks as searching the right function of the application or the data required in the current process step. With a PAIS, this is performed automatically. PAIS, however, also have disadvantages: Some users dislike their reduced freedom caused by the active and automatic process control. Furthermore, in exceptional cases, restricting the possible execution orders of the process activities may result in situations where sequences are not possible which would be advantageous for the business. This results in disadvantages for the organization.

To avoid such disadvantages, it must be possible to deviate flexibly from the rigidly modelled BP (Schonenberg *et al.* 2007, Redding *et al.* 2009, Dadam *et al.* 2011). A special case of flexibility are deviations that are pre-modelled already at build-time in order to apply them at run-time of the process instances (Pre-Designed Flexibility (Kumar and Narasipuram 2006), Flexibility by Design (Schonenberg *et al.* 2007)). Scientific literature, however, only discusses this categorization. Details of the corresponding requirements and approaches for their realization are hardly content of existing research.

This aspect is the focus of the project CoPMoF (Controllable Pre-Modelled Flexibility). Flexibility of PMS shall be increased, but deviations shall not be defined arbitrary (i.e. completely dynamic) by the users. Instead, predictable flexibility (i.e. deviations eventually required at run-time) is pre-modelled already at build-time. Then, the BP-designer and the BP-owner can evaluate such deviations with respect to their consequences. Furthermore, the required process reliability is guaranteed since only intended deviations are possible and they may only be performed by users with the required rights.

The main advantage of the presented approach, however, is that performing a deviation causes less effort for the end user compared to a dynamic change (eventually it would be even too complicated to define such a change dynamically). Assume, a concept shall be normally controlled by a software developer. But in difficult cases developers are overchallenged with this task. Then, this process activity shall be performed by a software architect of the same project. For this purpose, an alternative actor assignment for this activity was already pre-defined at build-time. Furthermore, it was defined who is allowed to activate this alternative actor assignment.

Dynamic changes (Reichert and Weber 2012) allow to insert new activities into a process instance. Such a functionality is indispensable for the realization of not predictable modifications. For predictable exceptional situations, however, dynamic changes are not well suited since they cause much effort for the user at each single deviation. In the example

Bauer, T.
Pre-Modelled Flexibility for Business Processes.
DOI: 10.5220/0007524605470555
In *Proceedings of the 21st International Conference on Enterprise Information Systems (ICEIS 2019)*, pages 547-555
ISBN: 978-989-758-372-8

explained above, the user would have to create a correct actor assignment which uses (existing) objects of the organizational model. Here, it is more meaningful to expend the effort only once already at build-time; i.e., to pre-model the eventually required actor assignment.

As already mentioned, with respect to pre-modelled flexibility, scientific literature only defines the corresponding category. Until now, this category was not examined in detail. The sole exception is the process aspect control flow. For this aspect, (Bauer 2017) and (Bauer 2018) discuss pre-modelling of flexibility. That means, there exists no answer to the following research question: Which scenarios (i.e. requirements) exist for the other process aspects (Jablonski 1997), where it is advantageous to pre-model flexibility of a BP at build-time, and which information must be provided for this purpose?

In the project CoPMoF, an approach with the following properties is developed:

- The requirements shall cover as many scenarios as possible. However, because of the research design, completeness, cannot be reached. In order to identify a large number of requirements, several BP are analysed with respect to their flexibility requirements. These BP are known by the author because of his long-term work in industry and research. Additionally, generally known processes and BP described in scientific literature (e.g. credit applications) were respected.
- The resulting process templates are "enriched" with pre-modelled flexibility. However, they shall stay easy to understand for BP-designers and "normal users". This is especially important for semantic process models (the business view), but also for technical models (process implementation); e.g., to enable users to detect errors in the process models.
- Despite the desired simplicity, the execution semantics of the building blocks for pre-modelled flexibility must be clear, since an easy to understand but vague modelling technique would prevent the execution of process instances by a process engine.
- Finally, only very little effort must result for the end users to trigger a flexible deviation at process execution (run-time).

To close the whole research gap, an approach has to be developed that fulfils all these requirements. This paper addresses the following part of the problem: BP of different domains are presented and examined with respect to the question, which scenarios of predictable flexibility are contained (case studies). Thereby, several requirements and facets are explained, in order to present the scenarios in an exhaustive and understandable manner. That means, the necessity of the requirements is proven with examples from practice. In this paper, the control flow aspect is only mentioned shortly; i.e., the main content are the other process aspects (Jablonski 1997). Detailed solution concepts for the realization of the requirements are not covered in this paper.

Section 2 introduces basic principles of PAIS and explains the challenges. The sections 3 to 5 describe the requirements for the different process aspects. Section 6 discusses related work. The paper concludes with a summary and an outlook.

2 BASICS AND PROBLEM STATEMENT

The first subsection describes pre-modelled flexibility that is typically supported by most approaches and commercial systems for process execution. In Section 2.2 some problem statements are explained at an example scenario from practice.

2.1 Modelling and Execution of Business Processes

PMS consist of a build-time and a run-time component. At build-time, a process template is created that describes the BP. For this purpose, a process graph is modelled which contains activities. Their execution order is determined by edges and conditions. This process template is used at run-time to create process instances. A process engine controls the execution of these process instances. For each currently executable activity instance (often named short: activity) it inserts corresponding items into the worklists of the potential actors. One of these end users selects the item and becomes able to perform this activity (instance). For activity execution, often, the actor has simply to fill a form.

The process aspect control-flow defines the execution order by means of a process graph (cf. Figure 1). Its nodes represent the activities (human tasks performed by users), automatically executed program code, or whole sub-processes. In addition, the process graph contains gateways (rhombuses), which represent Split and Join nodes. Commercial PMS typically offer Split and Join nodes with XOR- (one branch is selected because of its condition), OR- (several branches), and AND-Semantics (all branches are executed). Additionally, loops are supported. Branches as well as loops represent a simple

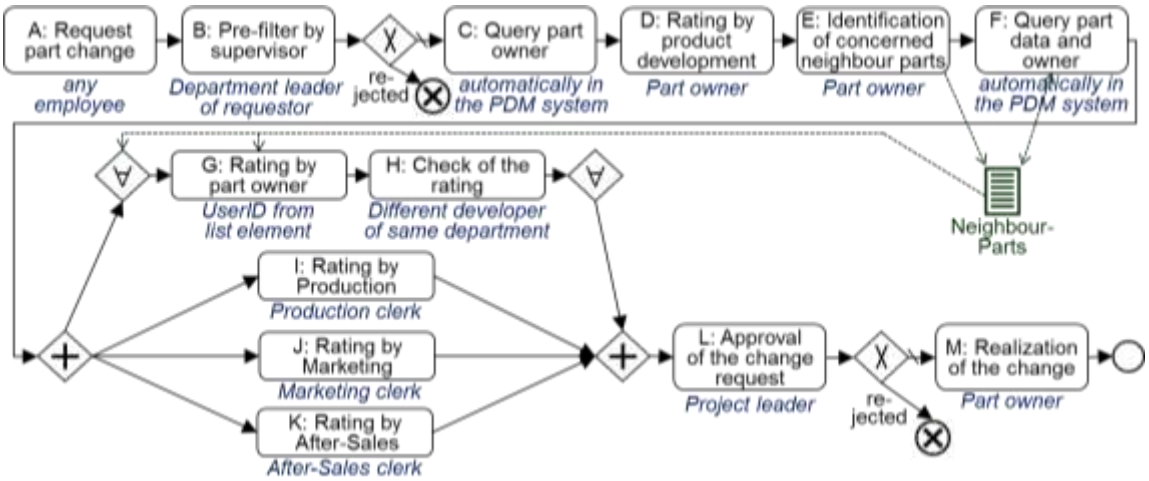

Figure 1: Change Management Process (CMP) for Product Modifications.

form of pre-modelled flexibility, since they have the effect that the set of executed activities and their execution order may differ at each process instance.

Often, process variables are used to realize the data flow. They are connected with the input and output parameters of activities (cf. NeighbourParts in Figure 1). When starting an activity, the contents of the variables are passed to the activity. After completion of the activity, its results are stored in process variables. Even complex data types may be used. Typically, objects that are composed of (elementary) data types and lists (arrays) are supported. Since the latter one have a variable length, they may build a basis for flexible process execution.

There exist many requirements for the organizational aspect (Russell *et al.* 2005). Commercial PMS typically allow to create organizational objects (e.g. groups, roles, departments), but some PMS do not distinguish between different types. Organizational objects can be assigned to users. For each activity, an actor assignment has to be defined. This is a "formula" that uses organizational objects (e.g. "role = software developer"). It is used by the process engine to calculate the potential actors of this activity. A corresponding entry is inserted into their worklists or they are informed with an e-mail. One of these persons selects this activity and performs it. Dependent actor assignments offer some type of flexibility, since the corresponding activity is not offered to the same persons at each process instance. Assume, a further inquiry in a business trip application process shall be answered by that person, that has created this application. This results in the dependent actor assignment "same actor as previous Activity X".

In addition, some PMS offer escalation mechanisms. Whenever a pre-defined processing time is reached for a specific activity, for instance, an e-mail is sent automatically to a supervisor or the actor of this activity is modified (automatic delegation).

Often, activities are executed using forms. Some PMS allow to generate such forms automatically based on the input and output parameters of activities. Afterwards they may be adapted manually. It is also possible to realize self-implemented web forms or rich client applications that use an application programming interface (API) of the process engine. Some process engines use web service calls to realize the execution of automatic process steps. For the integration of legacy applications, adapters may be provided. Thereby, flexibility results only from the variety of supported application types and the possibility to use a powerful enterprise service bus (ESB) at service calls.

2.2 Challenges Concerning Pre-Modelled Flexibility

This subsection demonstrates the need for flexibility at an example from practice. As mentioned in Section 1, the focus of this paper is not on dynamic changes used to react to unexpected events. Instead, situations are inspected, which represent exceptional cases, but are predicable. This allows to pre-model an appropriate behaviour already at build-time.

Figure 1 shows a simplified Change Management Process (CMP) as used to request product changes in the automotive domain. The notation is similar to BPMN 2.0 (but extended). With Act. A, an arbitrary employee of the automobile manufacturer may request a change of a vehicle part (e.g. the shape of the engine bonnet). Since the execution of a CMP-instance causes much effort, it can be stopped

with Act. B by a manager. Act. C determines the owner of the concerned part automatically by sending a query to the product data management (PDM) system. In Act. D, this owner rates the effort and the benefits of the change from the viewpoint of the development domain. Then, in Act. E he identifies neighbour parts (e.g. car wing, radiator) that have to be adapted because of the modified shape of the engine bonnet as well. Act. F queries the corresponding part details and part owners and stores these data in the list NeighbourParts (the other process variables were omitted to increase readability).

The rating from the viewpoint of the neighbour parts happens in Act. G by the respective part owner. This activity is instantiated multiple times (once for each neighbour part). The same applies to the check of the rating by another developer in Act. H. With Act. I to K, clerks of several domains are rating (in parallel) whether the change can be realized and estimate the resulting costs. Act. L decides on the approval of the change request and perhaps the parts are changed in Act. M.

The execution of the CMP requires flexibility at several points: The Act. G and H are included within a Multi-Instance-Parallelism. That means, the ∀-Split creates a number of branches that corresponds to the length of the list NeighbourParts. This list was filled by Act. E and F with the required input data and the intended actors. Afterwards, this list may be extended by a user action performed at an arbitrary point in time. However, this does only make sense before the Multi-Instance-Parallelism is finished (i.e. before the ∀-Join). Later on, additional neighbour parts cannot be respected by additional instances of Act. K and L any more.

If the part owner detects during the identification of the neighbour parts (Act. E) that he has made a mistake earlier in Act. D, he may want to correct its output data. For this purpose, he modifies the process variable Rating despite this is not an output parameter of the current Act. E. To allow this, for this variable Rating, it has to be pre-defined with which form or tool it can be modified.

The part owner requires a substitute for Act. D and E, since for each part there exists only one part owner. Without such a substitute, the whole process may be delayed unacceptably if this person is on holiday, for instance. This substitute, however, is not the department leader as for other activities of the part owner. This would not be appropriate since Act. D and E are project tasks. Therefore, dependent on the concerned vehicle project (respecting process variable VehicleProject, e.g. contains Golf) a developer of the same project has to act as substitute.

The application program that is used for the rating by a production clerk (Act. I), has to be selected in a flexible manner: Dependent on the person who performs this activity and the software that is installed on his computer, different application programs shall be used. For instance, some users possess a viewer for CAD models, others use the rich client of the production domain with a special visualization of CAD models, some a web form with part pictures, and others an “App” on a mobile device since they often are on the way in production halls. When modelling this activity, it has to be defined (at build-time), therefore, in which cases which application program shall be used.

3 DATA ASPECT

As described in Section 2.1, PMS typically support lists with a variable length that may be used within a Multi-Instance-Parallelism to assign application data and actors to the branches. Furthermore, the following flexibility is required.

3.1 Modify Process Variables (DF-1)

Sometimes, a user shall be able to modify the content of a process variable despite there currently does not exist a corresponding activity in the process. This may be useful; e.g., to correct, supplement, or provide data afterwards.

DF-1a: At build-time it has to be defined whether it is allowed to modify a specific process variable at all and who is allowed to do this. Additionally, it may be necessary to restrict the process area within which such a modification is allowed. Assume that the rating resulting from Act. D of the CMP (cf. Figure 1) contains errors and the execution is proceeded until Act. F in the meantime. A backward jump to Act. D and repeating the activities between D and F would cause unnecessary effort. Instead, the part owner may directly correct (i.e. modify) the rating data. This, however, is only allowed before the ratings of the other domains (Act. I to K) have started since they use this rating as input data.

DF-1b: The change request created in Act. A contains a sketch of the changed part; e.g., a PowerPoint figure or a CAD model. If such a sketch is erroneous it is not sufficient to simply modify text with a form. Instead, an appropriate application program is required. It is necessary to define for each (modifiable) process variable, which application shall be used for subsequent modifications.

3.2 Modify Assignments of Activity Parameters to Variables (DF-2)

The assignment of activity input and output parameter data to process variables shall be changeable as well. Assume that in the CMP currently no part owner is defined for the part that has to be changed (e.g., since he has left the company). The part owner, however, is absolutely necessary as input data for Act. D and E. Otherwise, they cannot be assigned to an actor. In order to solve this problem, for instance, the activity input parameter TaskActorID shall get its content from the process variable DepartmentLeaderID instead from the variable PartOwnerID. For this purpose, it has to be defined at build-time who is allowed to perform such a modification. Additionally, the set of process variables that may be used as a specific input resp. output parameter of this activity may be restricted.

4 ORGANIZATIONAL ASPECT

The PMS uses actor assignments to calculate the potential actors of activities. Thereby, it shall be possible to define powerful expressions. Additionally, a PMS repeats this calculation periodically (refresh) in order to respect changed or new memberships of persons in roles, groups, departments, etc.

If the functionality offered by a specific PMS is insufficient to realize the actor assignment required for an Act. X, the following work-around may be used: The potential actors are calculated by an automatically executed preceding Act. X'. Its output is a list of UserIDs that is stored in a process variable. This variable serves as input parameter of Act. X and the task is offered to these users. A drawback of this work-around is that, after completion of Act. X', this list is not refreshed anymore. Therefore, the following requirements shall be supported directly by the PMS.

4.1 Flexible Mechanisms for Actor Calculation (Org-1)

Org-1a (Calculate Actors using Process Data): It shall be possible to use process variables in an actor assignment. An example for this is Act. D of the CMP (cf. Figure 1): The UserID of the part owner is calculated by the automatically executed Act. C and stored in a process variable. Its content is used to assign Act. D to the right actor. Additionally to such UserIDs, other data may be relevant for actor assignments: The first activity of a credit application process may store the concerned bank branch in the process variable BranchID. The activity “tell decision to customer” shall be performed by a clerk of the same bank branch. This results in the actor assignment “Role=Clerk AND OrgUnit=VALUE(BranchID)”.

At Multi-Instance-Parallelisms it may be necessary to respect the sequential number i of the currently executed branch. Each rating by a part owner (Act. G) has to be performed by that person, that was determined by Act. F in the PDM system and stored in the list NeighbourParts at index position i (if currently the branch number i is executed). But the determination of UserIDs by the automatically executed Act. C and F has a disadvantage: Changes of part owners that occur later than the execution of these activities are not respected any more. Therefore, service calls that determine UserIDs should be triggered directly by the actor assignments of the Act. D, E, and G; i.e., it shall be possible that an actor assignment contains such a service call. These actor assignments are re-calculated (refreshed) periodically by the process engine. This, again, triggers the service call with the result that the currently valid potential actors are determined.

Org-1b (Alternative Actor Assignment): It may be predicable that the regular actor assignment is not appropriate for an activity in all exceptional cases. For instance, the regular potential actors may be overburdened sometimes, what shall be compensated by involving additional actors from other business domains. Another example is that an activity shall be performed by different actors in special cases; e.g., since the “regular actors” would be overchallenged with this task (cf. Section 1). For such activities, it shall be possible to define alternative actor assignments already at build-time.

Additionally, it has to be defined who is allowed to activate such an alternative. This may be the actual actors of preceding activities, all potential actors of the concerned activity, or the process owner (administrator). Switching to an alternative actor assignment shall be even possible after the process engine has inserted the corresponding items (that are based on the original actor assignment) into the worklists. Then, switching to the alternative actor assignment automatically triggers a re-calculation (refresh) of the potential actors. Therefore, it becomes effective immediately.

4.2 Substitutes (Org-2)

Only one single person (the part owner) is allowed to perform Act. D, E, and G of the CMP. If he is ill,

on holidays, or on a business trip the whole process execution will be delayed. Therefore, substitution rules have to be defined for these activities.

Org-2a (Definition with Rules): Rules for the calculation of substitutes are pre-modelled at build-time. They shall offer the same powerful functionality as actor assignments and may use organizational objects (e.g. roles, departments). Therefore, changes in the organizational model automatically result in updated substitutes. The actor (i.e. the part owner) of Act. D, for instance, may be substituted by all persons with "Role = Developer AND member of the same project as the regular actor". It is not sufficient that one single person can be defined as substitute, since multiple substitutes may be required to distribute the work load. In case of a substitution, they all become potential actors and one of these persons can decide to perform the activity.

Org-2b (Substitution Dependent on Activity): The substitutes of a person may Depend on the concerned activity or activity type. The part owner is substituted by a colleague of the same project (see above). At project-independent tasks (e.g. ordering of office supplies) his supervisor acts as substitute. Therefore, the substitution rules have to be defined at build-time as part of the process template; i.e., as rule valid for a single activity (type) or the whole process template. It is not sufficient to tell the process engine (independent from the process context) who are the substitutes of a person.

Org-2c (Configurable Behaviour): It shall be possible to configure the behaviour of the substitution rules. This concerns the following topics:

- The process designer defines when a substition shall be acitivated. This may happen i) if a single regular actor is absent, ii) if all regular actors are absent, or iii) if a given number or quota of absent actors is reached. Here exists a conflict of aims between the avoidance of extrem work-loads for the remaining regular actors and the requirement that the regular actors shall perform their activities if this is possible someway.
- It can be defined that a substitute may be substituted himself. Then, multiple "stages" of substitutions are applied by the PMS.
- It may be meaningful that a substitute is no longer allowed to perform a specific activity if the original (i.e. substituted) actor returns. This may concern only not started or even already running activities.

Many of these requirements cannot be fulfilled by current commercial PMS (if they offer substitutions at all).

4.3 User Actions (Org-3)

Users must be able to perform unplanned actions that concern the organizational aspect. For instance, the set of the potential actors of an activity has to be modified. Such actions are partially described in (Russell *et al.* 2005) as well. In the following, however, the focus is on the question, what has to be pre-modelled for such a user action.

Org-3a (Delegation): With a delegation, the regular actor transfers the activity to another person. Thus, it appears in a different worklist. A PMS has to offer powerful mechanisms for delegations: It shall be possible to delegate a task to multiple persons. For this purpose, some kind of "actor assignment" may be used. Assume, for instance, that a team leader wants to delegate an activity to some team members which are very skilled in the given context. Alternatively, he may delegate the activity to all team members. The latter case causes only little effort if a corresponding rule was already pre-defined at build-time ("Role = Clerk AND same team as regular actor") that may be used for delegations.

At least, it must be possible to define at build-time, whether a delegation for a specific activity is allowed at all. Whenever this is necessary for process safety (e.g. to respect compliance rules), the set of target persons of the delegation may be restricted. This restriction may be realized by an organizational expression similar to an actor assignment.

Org-3b (Modification of Actor Set): The set of potential actors of an activity may be modified by adding or removing persons. All other potential actors keep this function. Such a modification may even occur, before the concerned activity is ready to start.

At least, it has to be defined who is allowed to perform such a modification. Similar as at Org-3a, the set of persons that may be added can be restricted. Again, the usage of this function becomes very comfortable if rules were pre-defined at build-time, which may be used to add or remove persons at run-time. Assume, a chief physician detects at a prior interview with the patient that this is a very complicated case. Therefore, for instance, he wants to remove all assistant physicians as potential actors of a later treatment activity. Since such cases occur frequently, the modification-rule "remove actors with role Assistant-Physician" was pre-defined already at build-time. At run-time, the chief physician simply has to activate this rule.

Org-3c (Deallocation): Users select activities (items) from their worklists for execution. Such an actor may detect afterwards, that he does not want or

is not able to perform this activity. Then, he deallocates the activity; i.e., it can be selected by the other potential actors from their worklists again.

For each activity it may be defined whether such a deallocation is allowed at all. Furthermore, the point in time, until that a deallocation is possible, shall be configurable. The following activity states are meaningful for this purpose:

- A deallocation is only possible before the execution of that activity starts.
- A deallocation is allowed after starting the activity, but only before intermediate results were created (and stored within the PMS). This allows the actor to look at the activity input data (the details) in order to decide whether he wants to perform this activity. After starting work on this activity and storing intermediate results, a reallocation is no longer allowed.
- A reallocation is even allowed after partial execution of the activity and transmission of intermediate results to the PMS. In this case, it has to be defined additionally i) whether these intermediate results shall be discarded or ii) whether the next actor may use the results as input data of the activity; i.e., he may continue the (already performed) work.

5 ACTIVITY EXECUTION

(Commercial) PMS normally contain a client implementation that displays the worklists of the users. Additionally, this client displays the forms that are used to perform the activities. Often, web clients as well as rich clients are supported. It is also possible to develop own clients that use the application programming interface (API) of the process engine. With respect to activity execution, the following requirements are not fulfilled by many PMS.

5.1 Application Types (App-1)

Arbitrary types of applications shall be useable for activity execution. Connecting them with the PMS should cause only little effort. For this purpose, it may offer adapters. Then, the only remaining effort is to configure these adapters appropriately at build-time.

App-1a: It may be necessary that a specific application is usable for the execution of an activity. Therefore, no type of application shall be excluded. For instance, a specific document type must be processed with a specific text processing program (e.g. MS Word) or CAD tool. Normally, such "stand-alone applications" cannot be integrated into the given PMS client. Nevertheless, the stand-alone application must be usable at activity execution. It shall be started automatically (after selection of the corresponding worklist item) and its input data are transferred to the program. Finally, the results (output data) have to be transferred back to the process engine.[1]

App-1b: Functions that belong to an external server system with separate data management (e.g. SAP ERP) shall be usable as activity implementations as well. For this purpose, the process engine sends a message with the input data to the external server. Then, this server offers the task to the appropriate users; e.g., by notifying them with e-mails or by realizing its own worklists. Additionally, an integration with the worklists of the PMS may be required. After completion of the activity, the external server transmits the output data back to the process engine.

App-1c: A PMS shall support mobile clients as well. Mobile devices differ very much from each other (compared to PCs). Therefore, the properties of their mobile devices may influence the set of the potential actors of an activity. These actors shall be selected, for instance, based on the type of their mobile device (smart phone, tablet computer, or laptop), the size of the display, the current location, or the state of charge. That means, the set of "all possible" potential actors is restricted based on such criteria. Therefore, for each activity, the required properties of the mobile devices must be definable at build-time.

5.2 Different Applications for the Same Activity (App-2)

App-2a: It shall be possible to use multiple different applications as implementation of one activity. All of them have the same interface (i.e. input and output data) but they differ in their behaviour (i.e. user front end). The application program, that is used in fact, shall be selected in a flexible manner, for instance based on the skills of the current user, his preferred or used client type (web or rich client), the software installed on his computer (e.g. MS Word or Open Office Writer), or his type of device (PC or smart phone, cf. App-1c). The selection of this application program shall be performed with rules (expressions) that were pre-defined at build-time. At

[1] The Program Execution Client of IBM MQ Series Workflow (IBM 1996) proofs that this is possible in general.

run-time, they use process instance data and data concerning the current actor.

App-2b: Even after deployment of the process template, it shall be possible to create further implementations of an activity. Again, rules define their usage; i.e., their connection with the process template. Such implementations and rules shall be useable even for already running process instances (Late Binding). Therefore, the deployment of the activity implementations and the corresponding selection rules must happen independently from the business process (template). At build-time, it must be possible to determine that they shall be deployed immediately and shall be valid for an already deployed process template as well.

App-2c: A special case of an activity is a composed activity (subprocess). Even for composed activities it shall be possible to select the subprocess with rules at run-time (cf. App-2a), instead of assigning a fixed subprocess at build-time. Again, it may be necessary that additional subprocesses are created (i.e. modelled) after the deployment of the (father) process and after the creation of process instances (cf. App-2b). The corresponding selection rules are defined afterwards and these rules and the new subprocess must be deployed separately from the father process template.

6 RELATED WORK

(Kumar and Narasipuram 2006) present different categories of flexibility for BP. The category that corresponds to the CoPMoF approach is called "Pre-Designed Flexibility". The categories are refined in (Schonenberg *et al.* 2007) with the resulting categories "Flexibility by Design" and "Flexibility by Underspecification". A literature review[2] has shown that, until now, it was hardly examined what shall be pre-modelled at build-time in order to reach much flexibility and low effort for the users at run-time.

(Reichert and Weber 2012) suggests to use exception handling based on events and exception handlers to treat special cases: An event is assigned to single activities or whole process regions. If it occurs at run-time (throw) an exception handler is executed (catch). This is similar to a try-catch-block in programming languages and well suited to handle technical errors; e.g., the crash of an activity program. It may also be used to change a resource assignment; e.g., delegation of an activity to a more appropriate actor (Org-3a) (Russell *et al.* 2005). (Bauer 2009) concerns flexibility for the organizational aspect as well. It presents several requirements and concepts concerning substitution rules (cf. Org-2).

Case Handling (Aalst *et al.* 2005) is an approach for knowledge intensive BP, with the focus on data. The users (Knowledge Workers) know all data and have the possibility to change them at any time (cf. DF-1). Changing data is performed with forms. The state of a process instance results from the content of its data objects. They determine the activities that are currently executable; i.e., the control-flow is not modelled explicitly. The users decide (autonomously) to execute, skip, or repeat activities.

Some publications address late binding resp. late selection (cf. App-2). (Graml *et al.* 2008) allows to select a subprocess with rules at run-time (App-2c). Different pre-modelled subprocesses may contain different realizations of an application program (App-2a). (Adams *et al.* 2006) realizes activities by "Worklets". Dependent on the context of the process instance, they define which activity implementation shall be used. Worklets may be changed at run-time, therefore, the application resp. GUI of a process step may be adapted. A similar technique is used by commercial PMS with service orientation; e.g. IBM Business Process Server (IBM 2017): The execution of an activity results in a service call, which triggers an ESB flow (Erl 2005, Buchwald *et al.* 2009). This flow may contain branches and conditions. Therefore, it is possible to call different realizations of an activity with a selection criterion that depends on the context of the process instance. Since the deployment of the ESB flow happens independently from the BP template, it may be substituted at any time and the change becomes operative immediately. Therefore, such PMS offer an appropriate basis for the realization of App-2a to c. (Weber *et al.* 2008) presents "Pattern for Predefined Change". These pattern allow to model that specific decisions or definitions shall not happen until run-time. This enables, for instance, late selection and late modelling of process fragments (App-2c). (Pryss *et al.* 2016) offers some flexibility at the selection of actors that use mobile clients (App-1c). The potential actors of an activity may be selected based on the type of the mobile device, its current location, or its charging state.

[2] The search was performed with the following terms, all in combination with business process: flexibility by design, pre-designed flexibility, flexibility build-time, flexibility data flow, flexibility organization, flexibility activity. Furthermore, (Reichert and Weber 2012) as "overview book for flexibility in BP" was examined with respect to hints to relevant approaches.

7 SUMMARY AND OUTLOOK

PMS must allow to deviate from the rigidly modelled process. Otherwise, these systems are not usable in practice. Dynamic changes are one way to realize such deviations. For predictable deviations, however, this results in too much effort for the end users and may cause errors. To avoid such disadvantages, predictable special cases and exceptions should be pre-modelled already at build-time. This paper presents corresponding requirements and examples from practice. An intended impact is to motivate tool manufacturers to support the described scenarios in commercial BP modelling tools and process engines.

The generalisability and relevance of the presented scenarios has to be verified with further practical examples from other domains. Furthermore, they have to be complemented with additional requirements for pre-modelled flexibility. Some of the presented concepts are not available in today's process modelling languages (e.g. alternative actor assignments). Therefore, such situations are probably not captured in existing process models, despite they exist in reality. This problem may be solved by the usage of different research methods (e.g. expert interviews).

Detailed solution concepts still have to be realized for the identified requirements. Furthermore, a prototypical realization and a case study concerning usability may be necessary.

REFERENCES

Aalst, W.M.P. van der, Weske, M., and Grünbauer, D., 2005. Case Handling: A New Paradigm for Business Process Support. *Data & Knowledge Engineering*, 53 (2), 129–162.

Adams, M., *et al.*, 2006. Worklets: A Service-Oriented Implementation of Dynamic Flexibility in Workflows. *Proc. 14th Int. Conf. on Cooperative Information Systems*, 291–308.

Bauer, T., 2009. Substitution Rules for Task Actors in Process-oriented Applications. *Datenbank-Spektrum*, 9 (31), 40–51 (in German).

Bauer, T., 2017. Requirements for Pre-modelled Flexibility for the Control-Flow of Business Processes. *Proc. Informatik 2017, Workshop zum Stand, den Herausforderungen und Impulsen des Geschäftsprozessmanagements*, Chemnitz, 799–813 (in German).

Bauer, T., 2018. Execution Semantics for Jumps in Business Processes. *Datenbank-Spektrum*, 18 (2), 99–111 (in German).

Buchwald, S., Bauer, T., and Pryss, R., 2009. IT Infrastructures for Flexible, Service-oriented Applications. *Proc. 13. GI-Fachtagung Datenbanksysteme in Business, Technologie und Web*, 524–543 (in German).

Dadam, P., Reichert, M., and Rinderle-Ma, S., 2011. Process Management Systems. Only a bit Flexibility will not be enough. *Informatik-Spektrum*, 34 (4), 364–376 (in German).

Erl, T., 2005. *Service-Oriented Architecture - Concepts, Technology, and Design*: Prentice Hall.

Graml, T., Bracht, R., and Spies, M., 2008. Patterns of Business Rules to Enable Agile Business Processes. *Enterprise Information Systems*, 2 (4), 385–402.

IBM, 1996. *FlowMark Installation and Maintenance: Version 2, Release 2, Document Number SH12-6260-00*.

IBM, 2017. *Business Process Manager V8.6.0: 2017* [online]. Available from: https://www.ibm.com/support/knowledgecenter/en/SSFPJS_8.6.0 [Accessed Zugriff am 25 Jan 2019].

Jablonski, S., 1997. Architecture of Workflow Management Systems. *Informatik Forschung und Entwicklung, Themenheft Workflow-Management*, 12 (2), 72–81 (in German).

Kumar, K. and Narasipuram, M.M., 2006. Defining Requirements for Business Process Flexibility. *Workshop on Business Process Modeling, Design and Support, Proc. of CAiSE06 Workshops*, Luxemburg, 137–148.

Pryss, R., *et al.*, 2016. Context-Based Assignment and Execution of Human-Centric Mobile Services. *Proc. IEEE 5th Int. Conf. on Mobile Services*, 119–126.

Redding, G., *et al.*, 2009. Modelling Flexible Processes with Business Objects. *Proc. IEEE Conf. on Commerce and Enterprise Computing*, Wien, 41–48.

Reichert, M. and Weber, B., 2012. *Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies*: Springer.

Russell, N., *et al.*, 2005. Workflow Resource Patterns: Identification, Representation and Tool Support. *Proc. Int. Conf. on Advanced Information Systems Engineering*, 216–232.

Schonenberg, M.H., *et al.*, 2007. *Towards a Taxonomy of Process Flexibility (Extended Version)*: Eindhoven University of Technology.

Weber, B., Reichert, M., and Rinderle-Ma, S., 2008. Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems. *Data and Knowledge Engineering*, 66 (3), 438–466.