Demonstrating Context-aware Process Injection with the CaPI Tool

Klaus Kammerer, Nicolas Mundbrod, and Manfred Reichert

Institute of Databases and Information Systems
Ulm University, Germany
{klaus.kammerer, nicolas.mundbrod, manfred.reichert}@uni-ulm.de
http://www.uni-ulm.de/dbis

Abstract. Today's enterprises face individual customer expectations, high product variability, and an abundance of regulations. Consequently, they must cope with numerous business process variants, whose design and execution depend on a multiplicity of influencing factors, like, e.g., customer requests, resource availability, compliance rules, or process data. Moreover, already running processes need to be also adjustable to respond to contextual changes, emerging regulations, or new customer requests. With the goal to provide support for process variant management at both design and run time, this demo paper presents the prototype of the context-aware process framework (CaPI). The latter, in particular, enables the sophisticated modeling of process variants based on the context-aware injection of process fragments into a base process. Thus, executed process variants may dynamically evolve during run time, considering the current context of the respective process instance. The CaPI tool was developed based on existing adaptive process management technology. Overall, CaPI enables context-aware process injection, and, thus, the specification of varying processes while providing high process flexibility at run time.

Keywords: Process Flexibility, Process Variability, Process Injection

1 Introduction

In a globalized world, enterprises face various challenges like individual customer expectations, products of increasing complexity, or demanding country-specific regulations. As a consequence, enterprises need to cope with high process variability and a strong demand for process flexibility. In many business processes, the course of action is influenced by an abundance of *process parameters*, like, e.g., external context factors, intermediate results, and process-related events (e.g., successful completion of certain process activities). To deal with this variability, usually, process models comprise complex decisions allowing for alternative courses of actions as well as interdependencies among these decisions. Both the complex decisions and the interdependencies are often hardly comprehensible for process modelers. In addition, an automated, controlled and sound adaptation of (long-running) processes instances is required to address contextual changes, new regulations, or emerging customer requests at run-time.

This paper presents the concept of context-aware process injection (CaPI) as well as the proof-of-concept prototype of the CaPI framework. The latter enables the sophisticated modeling of processes that allow for a context-aware injection of process fragments into a base process during run time. To ease this modeling task, CaPI allows experts to model the possible injections from three different perspectives, i.e., situation-based, location-based, and artifact-based. At run time, the specified process variants are executed to incorporate up-to-the-minute context data and to adjust the process based on the given context. The proof-of-concept prototype was developed based on an advanced adaptive process management technology, which enables dynamic changes of process instances during run time. The CaPI tool presentation will demonstrate the features of the CaPI framework in an integrated and comprehensible way.

2 Application Scenario

To ease the understanding of CaPI, a use case from healthcare is introduced first. Figure 1 shows the simplified process of a patient's medical examination in a medical center [4]. The process contains the activities for ordering and performing a medical examination (e.g., *A2: Order Medical Examination*). Thereby, the latter is performed differently depending on the kind of examination (i.e., *standard, emergency*), the scheduling of examination (*on appointment, same day*), or activities supporting the examination (i.e., the preparation of a patient or its transport). In particular, the process covers four major variants: `V1`: `Emergency Case`, `V2`: `Standard Case with transportation`, `V3`: `Standard Case on Appointment`, and `V4`: `Standard on Appointment with transportation`. For example, there will be a emergency case, if the heart rate of a patient is lower than 50 beats/minute or a laboratory test results smaller than 2.5mmol/l potassium in blood serum.

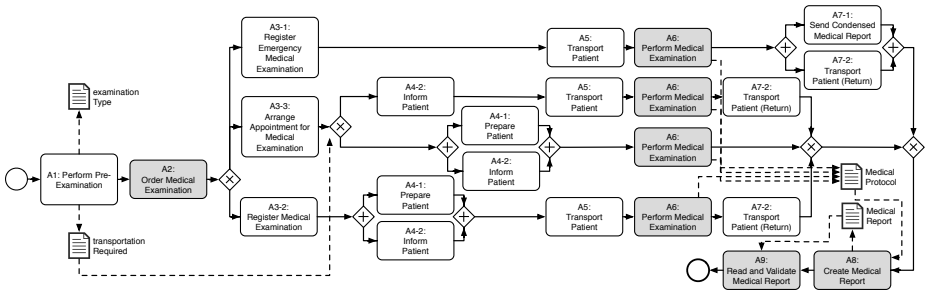

Fig. 1. Process Model of a Medical Examination

For each variant, additional constraints may have to be considered: if variant `V1` shall be performed, an emergency medical examination has to be registered along with mandatory transportation. Before creating a full medical report, a condensed report is sent while transporting back the patient during the execution of variant `V1`. If variant `V4` is executed (scheduled examination with transportation), no preparation is required. Finally, if a patient is transported to the examination room, she needs to be transported back to the ward afterwards.

3 Context-Aware Process Injection

To systematically support varying processes requiring (data-driven) run time flexibility, we introduced the approach of context-aware process injection (CaPI) [2]. The key objective of CaPI is to ease the modeling of a process family (i.e., a collection of process variants) at design time and to automate enable controlled process adaptions during run time, e.g., driven by the data becoming available during process execution. By taking the current context of a process into account, CaPI enables the late injection (i.e., insertion) of process fragments into a lean base process in a controlled manner.

The core artefact of CaPI is the *context-aware process family* (CPF) (cf. Fig. 2). More precisely, a CPF comprises a *base process* model (cf. Fig. 2g) with *extension areas* (cf. Fig. 2h), *contextual situations* (cf. Fig. 2e) characterized through *process parameters* (cf. Fig. 2d), a set of *process fragments* (cf. Fig. 2ij) that may be injected at specific extension areas during run time, and a set of *injection specifications* (cf. Fig. 2f). The latter define process fragments to be injected at extension areas of choice at a given contextual situation.

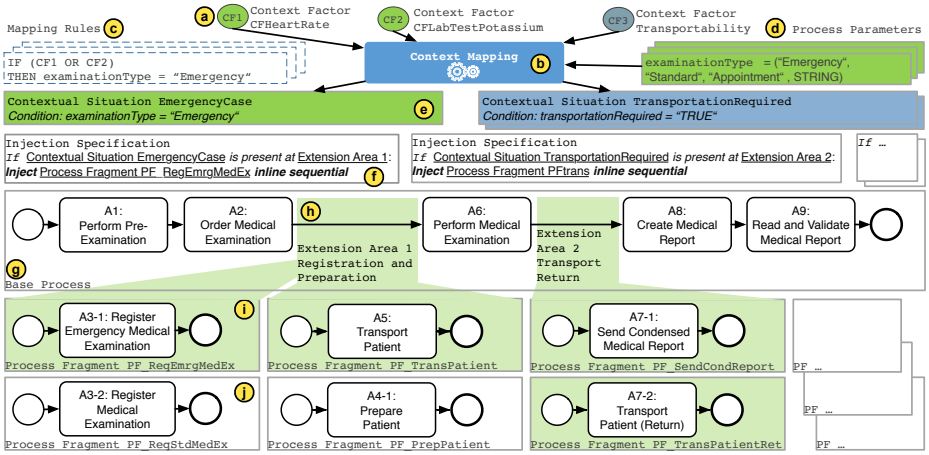

Fig. 2. Illustration of a Context-aware Process Family

Following the *separation of concerns* design principle, the *base process model* solely contains decisions (i.e., branches and gateways) and activities shared by all variants of the process. In particular, these activities need to be known at build time, and must not be changed during run time. By contrast, *extension areas* represent the dynamic parts of the process. Accordingly, first of all, process modelers may focus on modeling the predictable parts of the process and then add the dynamic (i.e., varying) parts of the base process step by step. In particular, extension areas are used to automatically inject process fragments into the base process during run time based on the current *contextual situation* and well-defined *injection specifications*. Moreover, an extension area allows for the dynamic injection of any number of parallel, either the same or different, process fragments. In turn, contextual situations are defined through conditions expressed

in first-order logic, taking *process parameters* as well as data objects of the base process model into account. In this context, process parameters may be linked to dynamic, external factors (e.g., the availability of a resource) influencing the process injection's decision making. When injecting process fragments, CaPI takes care of correct data flow mappings as well: data objects of an injected process fragment are automatically connected to existing ones of the base process.

Following this approach, CaPI enables dynamic configurations and changes of varying processes in a controlled way during run time. By solely enabling insertions of process fragments, CaPI allows process modelers to focus on the commonalities of all variants (base process) and the varying process parts instead of struggling with a complex process model that captures all variants. Furthermore, process modelers may directly integrate contextual influences into the modeling of variants. Thereby, complex external context factors are abstracted by meaningful process parameters and reusable contextual situations. In turn, CaPI is able to cope with context-driven run time changes based on the late evaluation of contextual parameters at given extension areas. In this context, the automated construction of a consistent data flow between the injected process fragments and the underlying base process mitigates the efforts of involved users. Finally, CaPI empowers process activities to seamlessly read and write data. A discussion of related work can be obtained from [2].

4 Demonstration

We developed the CaPI prototype to enable empirical studies. Furthermore, the technical feasibility of the described CaPI framework, was demonstrated in a case study we conducted in the automotive and electronics industry. In particular, from this case study we received valuable feedback regarding both the framework and the proof-of-concept prototype [2].

The CaPI prototype is based on a well-defined architecture (Fig. 3). It is realized based on Java EE 7 technologies and comprises a web-based frontend, which enables domain experts to intuitively model CPFs (*CaPI Modeler*), as well as the following components; representing the CaPI core run time functions: *CPF Repository*, *CaPI Monitor*, *CaPI Control*, and *Context Integrator*.

Using the web-based frontend based on Google Polymer, a domain expert may model the CPFs. First, she needs to specify the mappings of context factors to process parameters–and mappings of process parameters to contextual situations accordingly. Then she creates injection specifications, putting together the CPF components, i.e., extension areas, contextual situations and process fragments, via *drag and drop*. For this purpose, we implemented three different perspectives a domain expert may take to create an injection specification (see our screencast).

To properly execute modeled CPFs, the CaPI prototype integrates AristaFlow BPM Suite–an advanced adaptive process management technology [1]. The latter enables the modeling, deployment, and execution of well-structured processes. Furthermore, it provides sophisticated change operations to adapt running process instances at run time [3]. Thus, AristaFlow provides the basic execution platform required to enable the sound injection of process fragments.

CaPI Control interprets the CPF specifications to continuously monitor the execution of the process. Therefore, CaPI Control receives any status updates of activities from AristaFlow and confirms the start of every activity in the base process. When reaching an extension area, CaPI Control evaluates the contextual situation, and then injects the appropriate process fragments during run time.

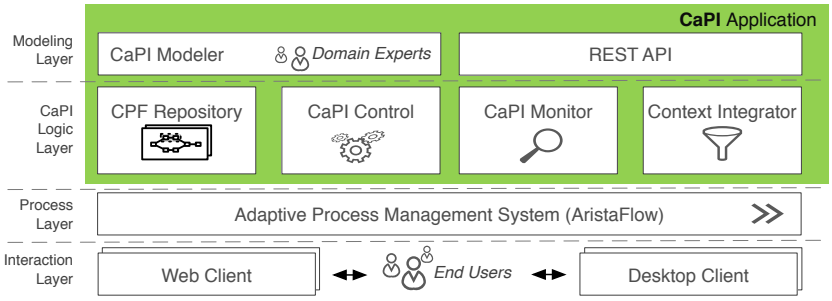

Fig. 3. Overview of the CaPI Architecture

The CaPI modeling and execution functions are illustrated in the screencast along the presented application scenario (cf. Sect. 2). This screencast can be retrieved in the following link: bpmdemo2017.capeframework.com

5 Conclusion

Taking the context of a process instance into account, CaPI allows for the controlled injection of process fragments into the given base process. Based on well-defined *extension areas*, control and data flow correctness of the process is ensured after injecting any process fragment. Moreover, CaPI reduces the complexity of specifying all the variants of a process family. This demo presents the current state of the CaPI proof-of-concept prototype as well as the feasibility of the CaPI framework using a real-world process from healthcare. Next, we want to mature the CaPI tool as a foundation for future empirical studies.

References

1. Dadam, P., Reichert, M.: The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support - Challenges and Achievements. Computer Science - Research and Development 23(2), 81–97 (2009)
2. Mundbrod, N., Grambow, G., Kolb, J., Reichert, M.: Context-Aware Process Injection: Enhancing Process Flexibility by Late Extension of Process Instances. In: Proc CoopIS 2015. pp. 127–145. No. 9415 in LNCS, Springer (2015)
3. Reichert, M., Dadam, P.: ADEPTflex - Supporting Dynamic Changes of Workflows Without Losing Control. J Intelligent Information Systems 10(2), 93–129 (1998)
4. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies. Springer (2012)