In: Proc. of International Conf. on Business Process Management (BPM 2008), pp. 394–397. LNCS 5240. Springer Verlag.

COREPRO$_{Sim}$: A Tool for Modeling, Simulating and Adapting Data-driven Process Structures[*]

Dominic Müller[1,2], Manfred Reichert[1], Joachim Herbst[2], Detlef Köntges[1,2], and Andreas Neubert[1]

[1] Institute of Databases and Information Systems, Ulm University, Germany
{dominic.mueller, manfred.reichert, andreas.neubert}@uni-ulm.de
[2] Dept. GR/EPD, Daimler AG Group Research & Advanced Engineering, Germany
{joachim.j.herbst, detlef.koentges}@daimler.com

Abstract. Industry is increasingly demanding IT support for large engineering process structures consisting of hundreds up to thousands of synchronized processes. In technical domains, such process structures are characterized by their strong relation to the assembly of a product (e.g., a car); i.e., resulting process structures are *data-driven.* The strong linkage between data and processes can be utilized for automatically creating process structures as well as for (dynamically) adapting them at a high level of abstraction. This paper presents the COREPRO$_{Sim}$ demonstrator which enables sophisticated support for modeling, coordinating and (dynamically) adapting data-driven process structures. COREPRO$_{Sim}$ substantiates the COREPRO approach which provides a new paradigm for the integration of complex data and process structures.

1 Introduction

In the engineering domain, the development of complex products (e.g., cars) necessitates the coordination of large *process structures.* Managing such structures, however, is a complex task which is only rudimentarily supported by current workflow technology [1]. Process structures often show a strong linkage with the assembly of the product; i.e., the processes to be coordinated can be explicitly assigned to the different product components. Further, synchronizations of these processes are correlated with the relations existing between the product components. We denote such process structures as *data-driven.* COREPRO utilizes information about product components and their relations for modeling, coordinating, and (dynamically) adapting process structures based on given (product) data structures. For example, the assembly of a (product) data structure can be used to automatically create the related process structure [2].

The adaptation of process structures constitutes a particular challenge. When adding or removing a car component (e.g., a navigation system), for example, the instantiated process structure has to be adapted accordingly; i.e., processes

[*] This work has been funded by *Daimler AG Group Research* and has been conducted in the *COREPRO* project (http://www.uni-ulm.de/dbis)

as well as synchronization relations between them have to be added or removed. When changing a (product) data structure during runtime, in addition, the running process structure must be adapted on-the-fly, but without leading to faulty synchronizations (e.g. deadlocks). To cope with this challenge, again, our approach takes benefit from the strong linkage between process structure and (product) data structure. Data structure changes can be translated into consistent adaptations of the corresponding process structure [3]. Thus, changes of data-driven process structures can be introduced by users at a high level of abstraction, which reduces complexity as well as cost of change significantly.

This paper sketches COREPRO$_{Sim}$ – a demonstrator enabling the modeling, enactment and (dynamic) adaptation of data-driven process structures. It enhances the COREPRO$_{Modeler}$, our first demonstrator supporting the manual modeling of data-driven process structures [4]. COREPRO$_{Sim}$ has been realized using the *Eclipse Graphical Editing Framework*. It implements an instantiation concept for automatically creating data-driven process structures and a runtime framework for simulating them [2, 3]. Furthermore, COREPRO$_{Sim}$ translates (dynamic) changes of currently processed data structures into corresponding adaptations of the related process structures. Finally, consistency is checked to ensure that dynamic adaptations result again in a sound process structure.

2 COREPRO Framework and Demo Description

The COREPRO modeling framework considers the sequence of states a (data) object goes through during its lifetime. A car component, for example, passes states like *tested* and *released*. Generally, state transitions are triggered when executing the processes which modify the respective object (e.g., *test* and *release*). An *object life cycle* (OLC) then constitutes an integrated and user-friendly view on the states of a particular object and its manipulating processes (cf. Fig. 1b).

Based on a collection of OLCs and their synchronizations, large process structures can be built. OLC state transitions do not only depend on the processes associated with the respective object, but also on the states and state transitions of other objects. As example consider a car prototype, which will be only tested if all subsystems (e.g., engine, chassis and navigation system) are tested before. By connecting the states of different OLCs, a logical view on the data-driven process structure results (cf. Fig. 1d).

Five steps become necessary to create a data-driven process structure using COREPRO$_{Sim}$ (cf. Fig. 1). Step 1 deals with the specification of a domain-specific *data model*, which defines object and relation types, and therefore constitutes the schema for instantiating concrete (product) *data structures*. An object type represents a class of objects within the data model (cf. Fig. 1a), which can be instantiated multiples times (cf. Fig. 1c).

Step 2 (cf. Fig. 1b) is related to the modeling of OLCs. Internally, OLCs are mapped to *state transition systems* whose states correspond to object states and whose (internal) state transitions are associated with object-specific processes. Non-deterministic state transitions are realized by associating different internal

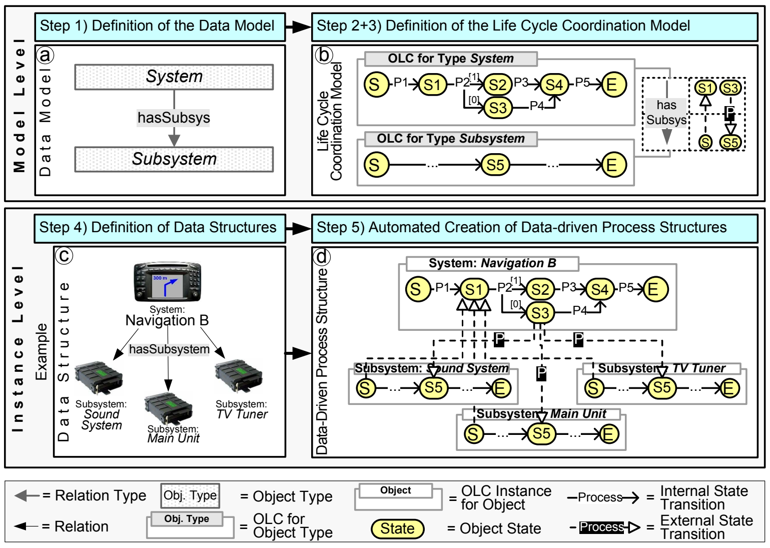


Fig. 1. Procedure for Creating Data-Driven Process Structures

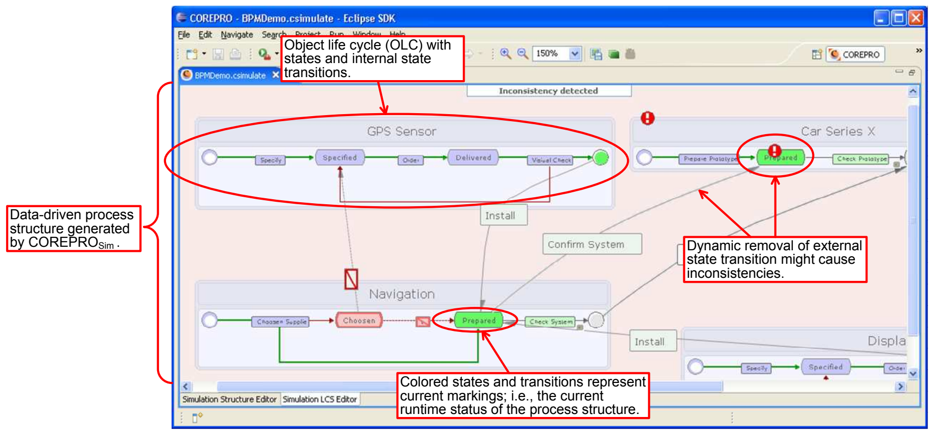
state transitions with same source state and process, and by adding a process result as condition (e.g., `P2` with possible results `0` and `1` in Fig. 1b).

Step 3 deals with the state dependencies existing between the OLCs of *different* object types. In COREPRO, an OLC dependency is expressed in terms of *external state transitions* between concurrently enacted OLCs (which together form the process structure). Like an internal OLC state transition, an external state transition can be associated with the enactment of a process. To benefit from the *strong linkage* between object relations and OLC dependencies, external state transitions are mapped to object relation types (cf. Fig. 1b).

Steps 4 + 5 are related to the instance level. COREPRO$_{Sim}$ allows to instantiate different data structures based on a given data model (cf. Fig. 1c) and to automatically create related process structures (cf. Fig. 1d). A data-driven process structure includes an OLC instance for every object from the data structure. Likewise, as specified in Step 3, for each relation in the data structure external state transitions are added to the process structure; e.g., for every `hasSubsystem` relation in the data structure from Fig. 1c, corresponding external state transitions are added (cf. Fig. 1d).

As result we obtain an executable process structure describing the dynamic aspects of the given data structure (cf. Fig. 1d). To ensure a correct dynamic behavior, COREPRO$_{Sim}$ allows for checking soundness of the process structure based on the concepts described in [2].

When simulating data-driven process structures, COREPRO$_{Sim}$ uses different *markings* to reflect the current runtime status (cf. Fig. 2). We annotate states as well as (internal and external) state transitions with respective markings. By analyzing *state markings*, for example, we can immediately figure out whether


Object life cycle (OLC) with states and internal state transitions.

Data-driven process structure generated by $\text{COREPRO}_{\text{Sim}}$.

Dynamic removal of external state transition might cause inconsistencies.

Colored states and transitions represent current markings; i.e., the current runtime status of the process structure.

Fig. 2. Simulation and Dynamic Change of Process Structure with COREPRO_{Sim}

a particular state of a process structure has been already passed, is currently activated, has been skipped, or has not been reached yet. *Transition markings*, in turn, indicate whether the associated process has been started, skipped or completed. The use of markings further eases consistency checking and runtime status adaptations in the context of dynamic changes of process structures.

To cope with flexibility requirements of engineering processes, we allow users (e.g., engineers) to perform dynamic process structure adaptations. In COREPRO_{Sim}, this is accomplished by automatically translating changes of the data structure (cf. Fig. 1c) into adaptations of the respective process structure [3]. Removing an object, for example, leads to the removal of the related OLC. Such dynamic adaptations must not violate soundness of the process structure. To ensure this, COREPRO_{Sim} constrains changes to certain runtime states (i.e., markings) of the process structure (cf. Fig. 2).

In summary, COREPRO_{Sim} constitutes a powerful demonstrator realizing the concepts developed in the COREPRO project [1–3]. It is currently applied in the context of a case study in the automotive industry. In future, we will extend the current prototype with extensive mechanisms for runtime exception handling and integrate existing data sources and applications.

References

1. Müller, D., Herbst, J., Hammori, M., Reichert, M.: IT support for release management processes in the automotive industry. In: BPM. LNCS 4102 (2006) 368–377
2. Müller, D., Reichert, M., Herbst, J.: Data-driven modeling and coordination of large process structures. In: CoopIS. LNCS 4803 (2007) 131–147
3. Müller, D., Reichert, M., Herbst, J.: A new paradigm for the enactment and dynamic adaptation of data-driven process structures. In: CAiSE. LNCS 5074 (2008) 48–63
4. Müller, D., Reichert, M., Herbst, J., Poppa, F.: Data-driven design of engineering processes with $\text{COREPRO}_{Modeler}$. In: WETICE'07, IEEE (2007) 376–378