Linking Domain Models and Process Models for Reference Model Configuration

Marcello La Rosa[1], Florian Gottschalk[2], Marlon Dumas[1], Wil M.P. van der Aalst[2,1]

[1] Queensland University of Technology, GPO Box 2434, Brisbane, QLD 4001, Australia
{m.larosa, m.dumas}@qut.edu.au
[2] Eindhoven University of Technology, PO Box 513, 5600MB Eindhoven, The Netherlands
{f.gottschalk, w.m.p.v.d.aalst}@tue.nl

Abstract. Reference process models capture common practices in a given domain and variations thereof. Such models are intended to be configured in a specific setting, leading to individualized process models. Although the advantages of reference process models are widely accepted, their configuration still requires a high degree of modeling expertise. Thus users not only need to be domain experts, but also need to master the notation in which the reference process model is captured. In this paper we propose a framework for reference process modeling wherein the domain variability is represented separately from the actual process model. Domain variability is captured as a questionnaire that reflects the decisions that need to be made during configuration and their interrelationships. This questionnaire allows subject matter experts to configure the process model without requiring them to understand the process modeling notation. The approach guarantees that the resulting process models are correct according to certain constraints. To demonstrate the applicability of the proposal, we have implemented a questionnaire toolset that guides users through the configuration of reference process models captured in two different notations.

Keywords: reference model, process configuration, variability modeling.

1 Introduction

Business processes like purchasing, recruitment, or customer service processes, are organized in similar ways across companies. To promote the reuse of such processes, Enterprise System vendors provide generic reference process models [4, 16, 17] that can be adapted to the needs of individual companies, thus enabling these companies to leverage proven practices and to avoid designing processes from scratch.

Notations like Configurable Event-driven Process Chains (C-EPC) [13] address the issue of representing and configuring reference process models. Such notations allow users to incorporate multiple process variants into a single configurable process model by means of variation points. By eliminating the undesired variants from the variation points, one can get a configured process model that suits the individual requirements.

Unfortunately, the integration and adaptation of several process variants in such notations often leads to an increase in the model's complexity. Thus, the configuration of a reference process model requires a significant degree of modeling expertise in the particular notation. Moreover, since these notations are designed to capture individual variation points, it is difficult to estimate the impact of high-level configuration decisions on the overall process model, i.e. to determine which variation points are affected by a decision and to what extent. While it is normal to assume that the design-

ers who produce the reference process model itself are familiar with the notation in question, it is less realistic to assume this from stakeholders who have to configure these models (e.g. a logistics or a film production expert).

In this paper we outline a framework that allows users to configure reference process models independently of the process modeling notation employed. We capture the variability of a given domain (e.g. Supply Chain Management) via so-called *domain facts* that form the answers to a set of *questions*. Questions and domain facts are expressed in natural language, thus facilitating the identification and configuration of variants by non-modeling experts. From a given set of facts grouped into questions, we are then able to generate an interactive questionnaire that guides the configuration.

Similarly, we represent this variability at the process level by identifying so-called *process facts*, which relate to the variation points of the configurable process model for the particular domain. We then link domain facts with process facts by means of a *mapping*. In this way, process configuration can be performed by simply answering a set of questions that mask the complexity of the underlying process model. The mapping ensures that the configured process model is semantically correct (e.g. no deadlocks), and consistent with the domain configuration.

The reminder of the paper is structured as follows. In Section 2 we introduce the approaches that are related to and inspired the development of our framework. In Section 3 we present the framework and illustrate it using an application in the area of the Screen Business. We apply the framework to a configurable process modeling notation, namely C-EPC [13] and to a configurable workflow language, namely C-YAWL (an extension of YAWL [1]). In Section 4 we present the approach to link domain and process models and a toolset that supports this approach. Finally, in Section 5 we draw conclusions.

2 Background and Requirements

The separation of process configuration from the context domain has been investigated, among others, by Becker et al. [2,3]. In this approach, adaptation parameters and their possible values are linked to model elements to indicate which sections of the model are relevant or not to a specific application scenario. By assigning values to these parameters, a user can configure a reference model without looking at the process flow. The approach can however be improved by offering guidance to users when assigning values to the adaptation parameters. Moreover, although it is possible to specify local constraints among parameters, no method is provided to check for model-wide consistency that could, e.g., inhibit deadlocks in the process flow or deny parameter settings that are practically not feasible.

To support consistency checking it is possible to use ideas from the CML2 language, which was designed to capture configuration processes for the Linux kernel [12]. It supports the definition of validity constraints based on propositional formulas over so-called symbols. To guide the configuration process, a configuration model in CML2 is composed of questions which lead to a given symbol being given a value.

The use of questions to steer the selection of process alternatives is advocated in [15], where alternatives are depicted as process specializations and the activation of these specializations is linked to conditions expressed as questions. However constraints over questions are not defined and no tool support is offered.

More generally, variability of large software systems has been studied in the field of Software Product Line Engineering (SPLE) [11]. The idea of SPLE is to capture how a collection of available options impact the way a software system is built from a

set of components. Parallels can be drawn between SPLE and reference models. A detailed comparison between our proposal and SPLE approaches is given in [7].

In this paper, we follow up on the above approaches and develop an integrated framework for reference process model configuration. The framework has been developed by following a design science approach [10]. By analyzing existing work, we have identified requirements for such an integrated framework. Chief among these requirements is that domain models should be represented separately from process models, in line with the approach of Becker et al. Secondly, domain models should be expressed in terms that are easily understandable by subject matter experts, and that can be directly exploited to guide the configuration process. Following these requirements, we have designed a framework that allows modelers to capture the variability of a given domain by means of a set of ordered questions. The framework provides decision support guidance that helps subject matter experts to answer the question in a way that leads to a valid configuration. An initial outline of this approach is reported in [8]. In this paper, we further extend this initial proposal with concrete mechanisms for linking domain models expressed as questionnaires, with process model elements. We present two such mechanisms in the context of two different modelling languages.

In line with a design science method, we have validated the proposal by means of a software tool implementation which has been tested on comprehensive examples, one of which is shown below.

3 Questionnaire-based Variability Framework

This section presents our framework. First, we show how to represent *domain configurations*. Then, we introduce *process configurations*, i.e., a way of capturing process variability. The next section shows how to integrate both types of configurations.

3.1 Capturing Domain Variability

We propose to depict the variability of a given domain independently of specific notations or languages, by means of a set of domain facts that form the space of possible answers to a set of questions. A domain fact is a boolean variable representing a feature of the domain, e.g. “Tape shoot”, that can be enabled or disabled. Questions group domain facts according to their content, so that all the facts of the same question can be set at once by answering the question. For example, the question “Which shoot media have been used?” allows users to choose the shoot medium between fact “Tape shoot” and “Film shoot”. A fact may appear in more than one question: in this case it is set the first time and its value is preserved in the subsequent questions. A fact always has a default value (*true* or *false*), while it can be marked as ‘mandatory’ if it needs to be set explicitly. For example, fact “Tape shoot” is true by default as the majority of production projects are shot on tape, which is less expensive than film. If a non-mandatory fact is left unset, i.e. if the corresponding question is not answered, the default value can be used instead. This way, each domain fact will always have a value either set explicitly by an answer or by using its default.

To illustrate these concepts, we consider an exemplification of the Post-production process in the area of Screen Business [14]. Fig. 1 presents a possible structure of questions/domain facts to capture the variability in this field. All questions and facts are assigned a unique identifier and a description.

Post-production aims at the editing and technical completion of a screen business project. Depending on the available budget, a project can be shot on tape, on film, or

on both the media. Of the two, film results in a more costly operation due to special treatments for making it visible and permanent. In Fig. 1 this choice is modeled by question q_3 and its facts, while the facts of q_6 and q_7 capture the possible sub-formats of tape, resp. film. The picture cut transfers the editing decisions that are taken on a

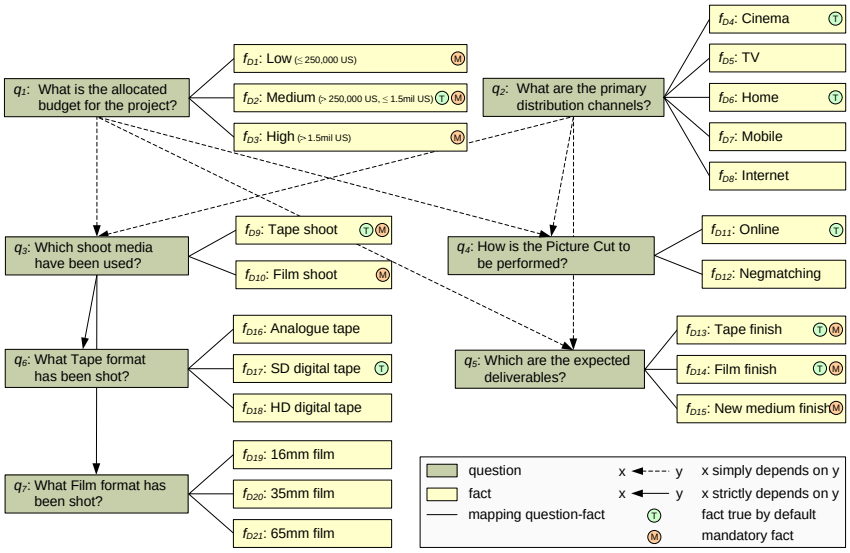

Fig. 1. A possible structure of questions/domain facts for the Post-production example.

low-resolution format in the Offline, to a high resolution format. The cut can be performed in an editing suite (Online), or on the original negative (Negmatching). This choice is captured by the facts of question q_4 and depends on the type of shoot medium. A project can be finished for delivery on tape, film, new medium, or any combination thereof. This is captured by the facts of q_5. The overall finishing process varies on the basis of the finishing media and may involve further tasks according to the choices made for the picture cut. For example, if the cut is done in Negmatching, the project must be finished at least on film. On the other hand, if the cut is only done Online and a film finish is required, the editing results need to be transferred to film in the so-called Record DFM. Similarly, a Telecine transfer is required to transfer the edited film to tape or file if the cut is done only in Negmatching and the project is to be finished on tape or on a new medium (e.g. DVD).

Interactions like these, which occur among the values of the domain facts, are modeled by a set of *domain constraints* in propositional logic that prune the configuration space. The constraints for the facts of Fig. 1 follow:[1]

C1: $f_{D1} \dot{\vee} f_{D2} \dot{\vee} f_{D3}$
C2: $f_{D1} \Rightarrow \neg(f_{D10} \vee f_{D14})$
C3: $f_{D2} \Rightarrow \neg f_{D10}$
C4: $f_{D4} \vee f_{D5} \vee f_{D6} \vee f_{D7} \vee f_{D8}$
C5: $f_{D4} \Rightarrow f_{D14}$
C6: $f_{D5} \Rightarrow f_{D13}$
C7: $f_{D6} \Rightarrow (f_{D13} \vee f_{D15})$
C8: $(f_{D7} \vee f_{D8}) \Rightarrow f_{D15}$
C9: $f_{D9} \vee f_{D10}$
C10: $f_{D11} \vee f_{D12}$
C11: $\neg f_{D10} \Rightarrow \neg f_{D12}$
C12: $f_{D13} \vee f_{D14} \vee f_{D15}$
C13: $(f_{D16} \dot{\vee} f_{D17} \dot{\vee} f_{D18}) \Leftrightarrow f_{D9}$
C14: $\neg(f_{D16} \vee f_{D17} \vee f_{D18}) \Leftrightarrow \neg f_{D9}$
C15: $f_{D12} \Rightarrow f_{D14}$
C16: $(f_{D19} \dot{\vee} f_{D20} \dot{\vee} f_{D21}) \Leftrightarrow f_{D10}$
C17: $\neg(f_{D19} \vee f_{D20} \vee f_{D21}) \Leftrightarrow \neg f_{D10}$

[1] $\dot{\vee}$ indicates the exclusive disjunction (XOR), a commutative and associative relation.

For example, since it is possible to shoot both tape and film, the facts representing these two features are bound by a logic OR (as per C9). Questions q_1 and q_3 capture the contextual choices for a post-production project, namely the allocated budget and the distribution channel. Their answers can affect the overall project, as shown by the constraints over their facts (C1 to C8). Further on, Negmatching (f_{D12}) cannot be performed if the project has not been shot on film (C11), and since its handcraft style makes it an expensive activity, Negmatching is worthwhile only if it is followed by a film finish (C15). However shooting film (f_{D10}) is only allowed for low/medium budgets (C2, C3), thus limiting the possibility to carry out a Negmatching.

A *domain configuration* is a possible valuation over domain facts that does not violate the constraints.

Order dependencies determine the order in which questions are presented to users. A 'simple' dependency (dashed arrow in Fig. 1) captures an optional precedence between two questions: e.g. q_3 can be posed after q_1 or q_2. A 'strict' dependency (plain arrow) captures a mandatory precedence: e.g. q_6 is posed after q_3 only. This way we can ask the most discriminating questions first, like q_1 and q_2 in Fig. 1, so that subsequent questions are (partly) answered by means of the domain constraints. If, e.g., we answer q_3 with "Film shoot" only, the question about the tape formats (q_6) becomes irrelevant. Dependencies can be arbitrary as long as cycles are avoided.

The above concepts form the definition of a Configuration Model (CM) - a first-class model to capture domain variability. The complete definition of CM can be found in [7]. In the following sections we show how CMs can be applied to support the configuration of reference process models.

3.2 Capturing Process Variability

Generally, notations for configurable process models rely on the concept of variation points to capture a point in the process flow where more than one variant exists. To link a CM to any configurable process model, independently of the notation adopted, we identify each process variant with a so-called *process fact*. A process fact is a boolean variable set to true if the variant it refers to is selected in a given process configuration, and to false otherwise.

Examples of notations for configurable process models are C-EPC [13] and Configurable YAWL (C-YAWL) [5]. C-EPCs extend Event-driven Process Chains (EPCs) [6] with variation points. A variation point can be a configurable connector (extension to the EPC join and split) or a configurable function (extension to the EPC function). A configurable connector can have several variants depending on its type, which can be logical AND, XOR, OR; a configurable function can have three variants: ON if enabled, OFF if disabled and OPT if optionally enabled.

Fig. 2 depicts the post-production process in a C-EPC, where variation points are highlighted by a thicker border (trivial events are omitted). The process starts with the preparation of the footage for edit, which depends on the type of shoot medium being used (tape or film). This choice is captured by the configurable OR-join OR_1. Its behavior can be restricted to an AND connector (if both tape and film are used), to branch SEQ_{1a} (for tape only, this results in branch SEQ_{1b} being deleted), or to branch SEQ_{1b} (for film only, branch SEQ_{1a} being deleted). We identify these three variants with three process facts f_{P1}, f_{P2}, f_{P3}, where f_{P1} is true if both tape and film are chosen, f_{P2} is true if only tape is chosen, and f_{P3} is true if only film is chosen. In a C-EPC a configurable OR can also be configured to an OR or XOR connector, but since these variants are not feasible for this process, we do not assign process facts to them.

Once the footage is ready, the project is edited on a low-resolution medium in the

Offline. The editing decisions are then transferred to a high-resolution medium in the *Online* and/or *Negmatching*, depending on the configuration of the OR-split OR_2 (f_{P4}

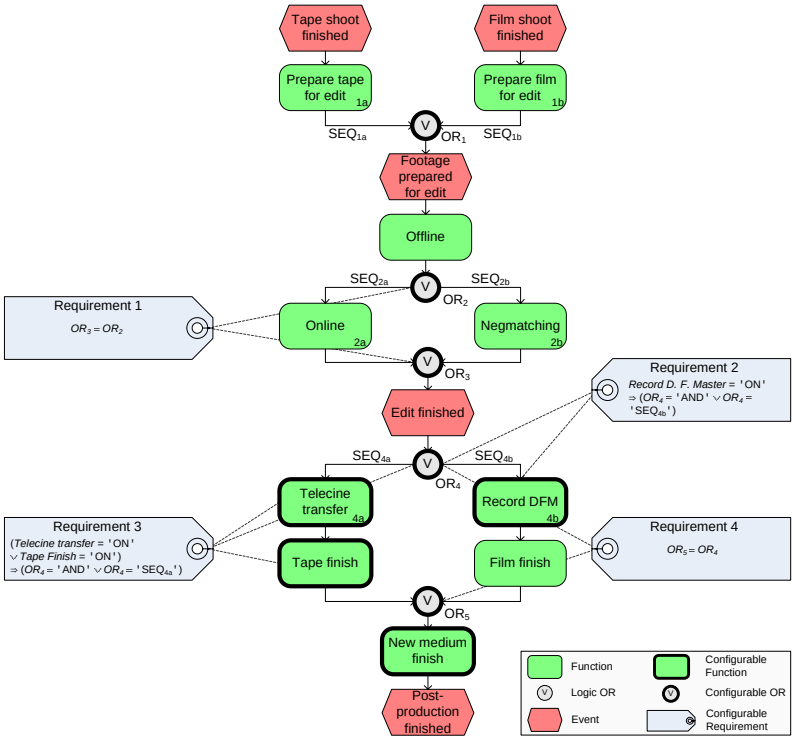

Fig. 2. The post-production reference process model in C-EPC.

for AND, f_{P5} for SEQ_{2a}, and f_{P6} for SEQ_{2b}) and of the OR-join OR_3 ($f_{P7} - f_{P9}$). The possibility to finish the project on any combination of tape, film and new medium is captured by OR_4 ($f_{P10} - f_{P12}$), OR_5 ($f_{P13} - f_{P15}$), and by the configurable functions *Telecine Transfer*, *Record DFM*, *Tape finish* and *New Medium finish*. These functions can be set only to ON or OFF, thus we assign two process facts to each of them. For example, f_{P16} and f_{P17} capture the variants ON and OFF of *Telecine Transfer*.

The same process can also be represented in C-YAWL, an extension of the executable process modeling language YAWL. The YAWL notation is based on conditions and tasks while the logic connectors of AND, XOR and OR are integrated in each task in the form of a join (for the incoming arcs) and a split (for the outgoing arcs). C-YAWL extends YAWL with so-called *ports* as variation points. A task's join has an *input port* for each combination of arcs through which the task can be triggered, whilst a task's split has an *output port* for each combination of subsequent arcs that can be triggered after the task's completion. An input port can be configured as enabled to allow the triggering of the task via this port, as blocked to prevent the triggering, or as hidden to skip the task's execution without blocking the subsequent process. An output port can be enabled to allow the triggering of paths leaving the port, or blocked to prevent their triggering. Like in C-EPC, in C-YAWL we represent

each feasible variant of a port with a process fact, thus, e.g., if a port is always enabled we do not assign any process facts to it.

Fig. 3 depicts the post-production process in C-YAWL with an example configuration for a project shot on tape, edited Online and finished on film and tape. The first task, τ_1, is used to route the process flow according to the shoot media. This task has only one incoming arc from the input condition. Therefore, its join has only one input port which always needs to be enabled, i.e. we do not assign any process fact to its variants. The task's OR-split has three output ports: one to trigger the path to condition *0a* (leading to film preparation), one to trigger the path to condition *0b* (leading to tape preparation) and one to trigger both paths. In this case we assign a process fact to each variant of these output ports as we want to capture the possibility to choose the shooting media. We thus use two process facts for each port: f_{P24} for the variant enabled and f_{P25} for the variant blocked of the first port, f_{P26}, f_{P27} for the second port's variants, and f_{P28}, f_{P29} for the third port's variants. In the example the project is shot on tape, so the port to *0b* is set to enabled (i.e. f_{P26} = true), and the other two ports are set to blocked (f_{P25}, f_{P29} = true).

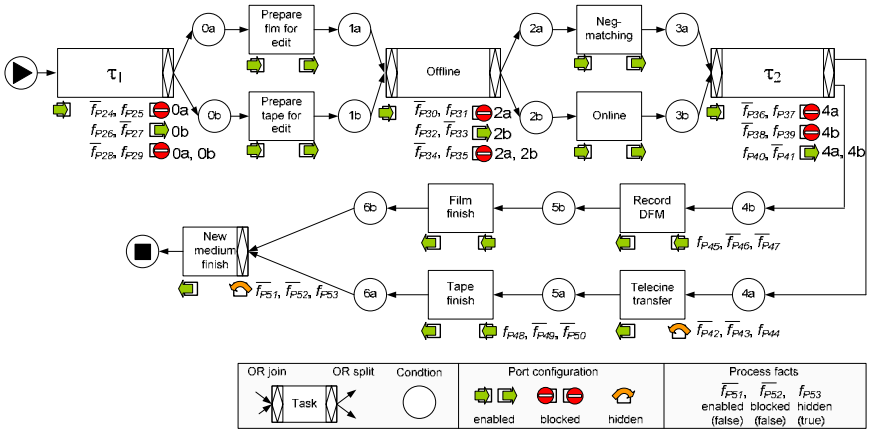

Fig. 3. The post-production reference process model in C-YAWL with process facts overlaid.

An OR-join can only have one input port in C-YAWL. In Fig. 3 the input port of the OR-join of Offline is configured to enabled as this task is always executed. No process fact is thus assigned to its variants. The project is edited Online. So the output port of Offline that triggers condition *2b* is the only one to be enabled (f_{P32} = true). Similar considerations to the OR-join of *Offline* also hold for the OR-join of task τ_2. As the project is finished on tape and film, the output port of the OR-split of τ_2 that triggers *4a* and *4b* is enabled (f_{P40} = true). We do not need *Telecine transfer* and *New Medium finish*, but we want the process to complete. Therefore, their input ports are hidden. To depict this configuration option for input ports, a third process fact is assigned to each input port (f_{P44}, f_{P53}=true). On the other hand, the tasks *Record DFM* and *Tape Finish* are required, thus their input ports are enabled (f_{P45}, f_{P48} = true).

Notations as C-EPC and C-YAWL provide the ability to define configurable requirements to restrict the variants allowed for a variation point. These requirements need to capture the interdependencies of the domain and to preserve the correctness of the model. Process facts, being an abstraction of process variants, are thus subject to

the same requirements. In our framework, however, we need to capture only the requirements for process correctness, as the domain constraints are propagated to process facts by mapping the CM to the configurable process model. Thus our *process constraints* are only a subset of the configurable requirements of C-EPC and C-YAWL. In C-EPC such requirements are annotated to the relevant variation points as labels. In Fig. 2 we only report the requirements that correspond to the process constraints. Req. 1 and 4 ensure a synchronized configuration of the OR splits/joins to prevent the process from a deadlock. This corresponds, e.g., to the process constraints: $f_{P4} \Leftrightarrow f_{P7}$, $f_{P5} \Leftrightarrow f_{P8}$, $f_{P6} \Leftrightarrow f_{P9}$. Req. 2 and 3 guarantee that the configurable functions can be performed in any configuration where they are set to ON. In C-YAWL, the configurable requirements are not directly depicted in the model. For example, to prevent the process from deadlocking, it is required that every C-YAWL task with an input port enabled or hidden has at least one output port enabled; or if a task can trigger a condition other than the final one, there needs to be a task with an enabled or hidden input port which can be triggered by this condition.

A *process configuration* is thus a possible valuation over the process facts that does not violate the process constraints, ensuring the configured model is correct. The process configuration complements the domain configuration. The next section shows how both types of configurations can be related.

4 Linking Domain and Process Variability

We link the domain variability captured by a CM with the variability of a configurable process model, by mapping domain facts to process facts. We call $F_D = f_{D1},...,f_{Dn}$ the set of domain facts and $B_D(F_D)$ the boolean function representing the conjunction of the domain constraints, such that B_D holds for every domain configuration. Also, we call $F_P = f_{P1},...,f_{Pn}$ the set of process facts and $B_P(F_P)$ the boolean function for the conjunction of the process constraints, such that B_P holds for every process configuration. A boolean function $B_M(F_D, F_P)$ creates the mapping of domain and process facts, such that each process fact equals a boolean expression over the domain facts.

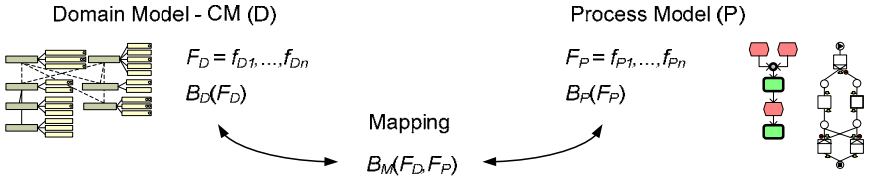

Fig. 4. Mapping configurable domain and configurable process model.

For example, the variants of OR_1 in the C-EPC process model of Fig. 2 are captured by the process facts f_{P1} (for the variant "tape and film"), f_{P2} (for "tape only") and f_{P3} (for "film only"). At the domain level, this would correspond to answer q_3 (Which shoot media have been used?) with both f_{D9}, f_{D10} = true in the first case, with only f_{D9} = true in the second case, and with only f_{D10} = true in the third case (where f_{D9} is Tape shoot and f_{D10} is Film shoot). Therefore, we map the boolean function $f_{D9} \wedge f_{D10}$ to f_{P1} so that the latter is set to true if and only if f_{D9} and f_{D10} are true. Likewise, we map $f_{D9} \wedge \neg f_{D10}$ to f_{P2} and $\neg f_{D9} \wedge f_{D10}$ to f_{P3}. In this way we can select the proper process variant by checking which of the above expressions holds against a given domain configuration, which is obtained from the answers to the questions of the CM.

The mapping $B_M(F_D, F_P)$ is *valid* if and only if the boolean expressions over the domain facts associated to the process facts of the same variation point are in exclusive disjunction. Given a domain configuration, exactly one variant has to be selected for each variation point. In this way we avoid a domain configuration to lead to zero or more than one variant per variation point, i.e., for example, the domain facts should not require that at the same time a C-EPC function is turned ON and OFF or a port in C-YAWL is blocked and enabled.

An excerpt of the mapping between the CM of Fig. 1 and the process models of Fig. 2 and 3 is given as follows.

M1: $f_{P1} \Leftrightarrow f_{D9} \wedge f_{D10}$ **M2:** $f_{P2} \Leftrightarrow f_{D9} \wedge \neg f_{D10}$ **M3:** $f_{P3} \Leftrightarrow \neg f_{D9} \wedge f_{D10}$

M4: $f_{P4} \Leftrightarrow f_{D11} \wedge f_{D12}$ **M5:** $f_{P5} \Leftrightarrow f_{D11} \wedge \neg f_{D12}$ **M6:** $f_{P6} \Leftrightarrow \neg f_{D11} \wedge f_{D12}$ [...]

M16: $f_{P16} \Leftrightarrow (\neg f_{D11} \wedge f_{D13}) \vee (\neg f_{D11} \wedge f_{D15})$ **M17:** $f_{P17} \Leftrightarrow \neg((\neg f_{D11} \wedge f_{D13}) \vee (\neg f_{D11} \wedge f_{D15}))$ [...]

M24: $f_{P24} \Leftrightarrow \neg f_{D9} \wedge f_{D10}$ **M25:** $f_{P25} \Leftrightarrow \neg(\neg f_{D9} \wedge f_{D10})$ **M26:** $f_{P26} \Leftrightarrow f_{D9} \wedge \neg f_{D10}$

M27: $f_{P27} \Leftrightarrow \neg(f_{D9} \wedge \neg f_{D10})$ **M28:** $f_{P28} \Leftrightarrow f_{D9} \wedge f_{D10}$ **M29:** $f_{P29} \Leftrightarrow \neg(f_{D9} \wedge f_{D10})$ [...]

M42: $f_{P42} \Leftrightarrow (\neg f_{D11} \wedge f_{D13}) \vee (\neg f_{D11} \wedge f_{D15})$ **M43:** $f_{P43} \Leftrightarrow \mathit{false}$ [2]

M44: $f_{P44} \Leftrightarrow \neg((\neg f_{D11} \wedge f_{D13}) \vee (\neg f_{D11} \wedge f_{D15}))$ [...]

Answering a question may affect one or multiple variation points, e.g. the facts of q_1 indirectly affect a number of variation points. Also, configuring a variation point can be determined by answering more than one question, e.g. the process facts that refer to OR_4 in the C-EPC model are affected by q_4 and q_5.

A *valid process configuration* with respect to the domain is given by any domain configuration that leads to a process configuration via a valid mapping, i.e. if the conjunction $B_D \wedge B_P \wedge B_M$ is satisfiable and the mapping is valid. The configuration space is obtained by the intersection of the two configuration spaces (domain and process) via the mapping. By representing the domain variability in a separate model, we can avoid capturing the interdependencies of the domain in the configurable process model, as these interdependencies are represented by the domain constraints in the CM, and propagated to the process model via the valid mapping. Constraints over process facts have thus to deal only with the preservation of the model correctness. As a result, the identification of such constraints becomes simpler.

Once each variation point has been configured with a variant, a set of *actions*, attached to the process fact that captures the selected variant, are performed on the process model to commit a configuration.

In the next section we propose a methodology to constructively define a mapping between a domain and its process model.

4.1 Constructing the Mapping between Domain and Process Model

The first step towards the construction of a mapping is to capture the variability of a given domain by means of a CM. This task should be conducted by the modeller in close collaboration with the domain experts, e.g. the film producers. Similarly, the domain should be represented by a configurable process model in a suitable notation. Process facts should be identified with the variants of the process and process constraints should be defined to preserve the model correctness (these depend on the notation adopted). As a rule of thumb, a fact is meant to represent a variant of the domain or process model. Thus a fact should be considered as such only if it can be freely set before starting the configuration; otherwise it would represent a commonality in the domain or process model, and should be left out.

[2] Fact f_{P43} can be omitted since it represents a commonality, being always *false*.

Once domain facts, process facts and their constraints have been identified, it is possible to define the mapping by means of a two-way impact analysis:

- *from domain to process*: e.g., given a domain fact, we need to estimate what are the implications of setting such a fact to true/false in the process model;
- *from process to domain*: e.g., given a variation point, we need to consider which domain facts are impacted by configuring such a point with a variant.

Although a mapping is valid, the process constraints might restrict the configuration space of the domain so as to deny some domain options, as a result of the application of the mapping. This situation may lead to problems when it comes to communicate such restrictions to the stakeholders, and as such should be avoided. In these cases, either the mapping or the process model should be modified, supposing the domain cannot be changed. On the other hand, the mapping might be so restrictive that no correct process model is allowed at all. In these cases, process facts and constraints can be used to evaluate the feasibility of the domain constraints and of the mapping, and to suggest their revision.

4.2 Tool support

To establish the practical feasibility of our framework, we have implemented a toolset[3] that provides end-to-end support for reference process model configuration. Each tool is a stand-alone application responsible for a specific task in the configuration process, from the collection of the answers via questionnaires, to the release of a configured process model. Fig. 5 provides an overview of the toolset's architecture.

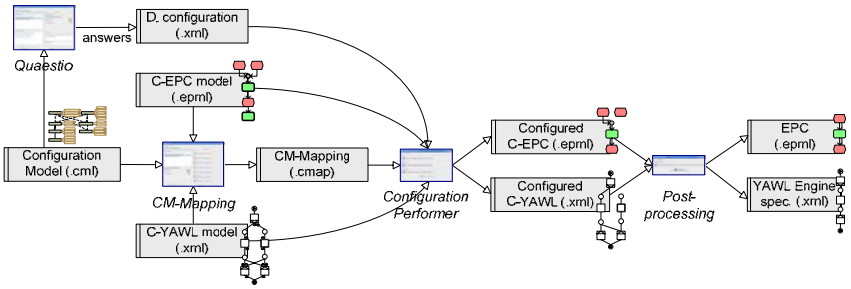

Fig. 5. The software architecture of the tools implemented.

Quaestio accepts an XML serialization of a CM as input and generates a questionnaire that guides the configuration interactively by posing only the relevant questions in an order consistent with the order dependencies. The tool also prevents users from entering conflicting answers to subsequent questions, by dynamically checking the domain constraints. Questions can be answered explicitly or by using the default values, and they can be rolled back. Quaestio embodies a SAT solver[4] based on Shared Binary Decision Diagrams (SBDDs) [9], to handle propositional logic formulae. Algorithms based on SBDDs can efficiently deal with systems made up of around one million of possibilities [9]. Quaestio can thus scale with CMs made up of thousands of domain facts and around one million of possible configurations.

The *CM-Mapping* tool allows designers to define mappings between the domain facts and the process facts of a C-EPC or C-YAWL net, whose serialization is taken

[3] Downloadable from `http://sky.fit.qut.edu.au/~dumas/ConfigurationTool.zip`
[4] Downloadable from `http://www-verimag.imag.fr/~raymond/tools/bddc-manual`

as input. This tool uses the SBDD calculator to check the consistency of the domain constraints and of the process constraints, i.e. if they can be satisfied and if each fact can be freely set. It also verifies the validity of the generated mapping, checks for redundancies, and shows possible restrictions of the single configuration spaces (domain and process) that may occur from the application of the mapping.

The *Configuration Performer* takes as input a domain configuration generated by Quaestio, a serialization of a C-EPC or C-YAWL reference process model, and the corresponding mapping. It outputs an intermediate format to represent the configured net, where each variation point has been marked with one variant. This artefact is then post-processed by a tool that implements a derivation algorithm to generate a syntactically correct EPC or YAWL model.

To achieve the configuration over process facts shown in Fig. 3, we use the following domain configuration $\sigma_D = \{\neg f_{D1}, f_{D2}, \neg f_{D3}, \neg f_{D4}, \neg f_{D5}, \neg f_{D6}, f_{D7}, f_{D8}, f_{D9}, \neg f_{D10}, f_{D11}, \neg f_{D12}, \neg f_{D13}, \neg f_{D14}, f_{D15},...\}$. From the domain configuration we realise that the process configuration of Fig. 3 is only feasible for medium/high budgets projects. Fig. 6 shows the configured YAWL process obtained by the post-processing tool.

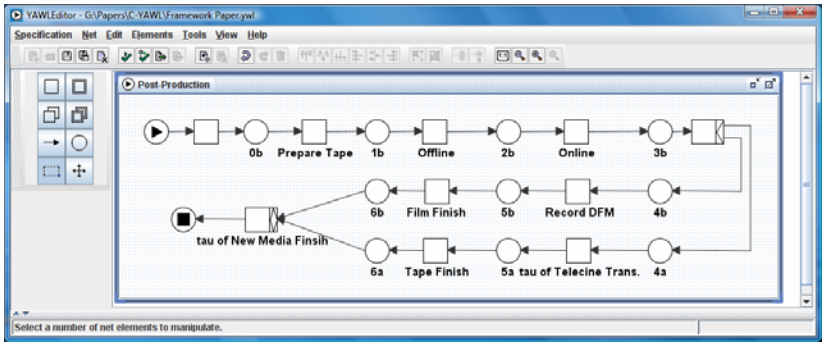

Fig. 6. The YAWL process model derived from the configuration of the model in Fig. 3.

5 Conclusion and Outlook

This paper presented a framework and a toolset for the configuration of reference process models. The main innovation of the proposal is that the configuration is not driven directly by a (configurable) process model, but rather by a model that captures the variability of the application domain. The domain model is structured in terms of questions, facts and constraints that reflect the decisions that need to be made during configuration. From a domain model, an interactive questionnaire is generated that guides domain experts through the process of configuring the process model without requiring them to understand the process modeling notation.

The domain model is then linked to a configurable process model, so that once all configuration decisions are made, an individualized process model can automatically be generated. The variability of the configurable process model is also captured in terms of facts and constraints. By merging the process constraints with the domain constraints, we obtain a unified set of constraints that is used by the configuration tool to ensure that the model generated from the configuration decisions is always correct.

Capturing reference process models in terms of facts and constraints allows us to abstract away from the modeling notation. We have demonstrated the applicability of

the proposal on two configurable process modeling notations: one intended for analysis (C-EPCs) and another intended for implementation (C-YAWL). We have tested the approach on several examples, particularly in the area of film post-production.

In the current framework, process facts and constraints need to be manually extracted from a process model. We are working on applying formal analysis techniques to automate this extraction. Furthermore, we are conducting experiments with screen business experts to empirically evaluate the applicability and impact of the proposed configuration approach on the modeling process.

References

1. W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4): 245–275, 2005.
2. J. Becker, P. Delfmann, and R. Knackstedt. Adaptive Reference Modelling: Integrating Configurative and Generic Adaptation Techniques for Information Models. In *Reference Modeling. Efficient Information System Design through Reuse of Information Models*, pp. 23–49. Berlin et al. 2007.
3. J. Becker, P. Delfmann, A. Dreiling, R. Knackstedt, D. Kuropka: Configurative Process Modeling – Outlining an Approach to Increased Business Process Model Usability. In *Proceedings of the Information Resources Management Association Conference*. New Orleans LA, USA. 2004. pp. 615-619.
4. T. Curran and G. Keller. SAP R/3 Business Blueprint: *Understanding the Business Process Reference Model*. Upper Saddle River, 1997.
5. F. Gottschalk, W.M.P. van der Aalst, M.H. Jansen-Vullers, and M. La Rosa. *Configurable Workflow Models*. BETA Working Paper 222, TU Eindhoven, 2007.
6. G. Keller, M. Nüttgens, A.W. Scheer. *Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK)*. Veröffentlichungen des Instituts f˙ur Wirtschaftsinformatik, University of Saarland, Saarbrücken, 1992.
7. M. La Rosa, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. *Variability Modeling for Questionnaire-based System Configuration*. Preprint # 7992, Queensland University of Technology, 2007. Available at http://eprints.qut.edu.au/archive/00007992.
8. M. La Rosa, J. Lux, S. Seidel, M. Dumas and A. ter Hofstede. Questionnaire-driven Configuration of Reference Process Models. In *Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE)*. Trondheim, Norway, June 2007.
9. S. Minato, N. Ishiura, S. Yajima. Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean function Manipulation. In *Proceedings of the ACM/IEEE Conference on Design Automation (DAC)*, pp. 52–57, 1990.
10. C. L . Owen. Design Research: Building the Knowledge Base. *Design Studies*, 19(1): 9-20, January 1998.
11. K. Pohl, G. Böckle, and F. van der Linden. *Software Product-line Engineering – Foundations, Principles and Techniques*. Springer, Berlin, 2005.
12. E. S. Raymond. *The CML2 Language*. http://catb.org/esr/cml2/cml2-paper.html, 2000.
13. M. Rosemann and W.M.P. van der Aalst. A Configurable Reference Modelling Language. *Information Systems*, 32(1): 1–23, March 2007.
14. S. Seidel, M. Rosemann, A.H.M. ter Hofstede, and L. Bradford. Developing a Business Process Reference Model for the Screen Business - A Design Science Research Case Study. In *Proceedings of the 17th Australasian Conference on Information Systems (ACIS)*, Adelaide, Australia, 2006.
15. P. Soffer, B. Golany, and D. Dori. ERP modeling: a comprehensive approach. Information Systems, 28(6): 673–690, September 2003.
16. S. Stephens. The Supply Chain Council and the Supply Chain Operations Reference Model. *Supply Chain Management - An International Journal*, 1(1): 9–13, 2001.
17. C. Taylor and C. Probst. Business Process Reference Model Languages: Experiences from BPI Projects. In *Proceedings of INFORMATIK 2003 – Innovative Informatikanwendungen*, Band 1, pg. 259–263, 2003.