Requirements-Driven Design and Configuration Management of Business Processes

Alexei Lapouchnian[1] Yijun Yu[2] John Mylopoulos[1]

[1] Department of Computer Science, University of Toronto,
Toronto, ON M5S 3G4, Canada
{alexei,jm}@cs.toronto.edu
[2] Computing Department, The Open University,
Milton Keynes, MK7 6AA, U.K.
y.yu@open.ac.uk

Abstract. The success of a business process (BP) depends on whether it meets its business goal as well as non-functional requirements associated with it. BP specifications frequently need to accommodate changing business priorities, varying client preferences, etc. However, since business process goals and preferences are rarely captured explicitly in the dominant BP modeling approaches, adapting business processes proves difficult. We propose a systematic requirements-driven approach for BP design and configuration management that uses requirements goal models to capture alternative process configurations and provides the ability to tailor deployed processes to changing business priorities or customer preferences (i.e., non-functional constraints) by configuring their corresponding goal models at the goal level. A set of design time and runtime tools for configuring business processes implemented using WS-BPEL is provided, allowing to easily change the behaviour of deployed BP instances at a high level, based on business priorities and stakeholder preferences.

1 Introduction

At present, process orientation is a dominant paradigm for businesses. There are many definitions of what a business process is, but in general a BP is seen as a collection of activities that achieves some business *purpose* or *objective* aiming to create value for customers. So, business processes specify ways to achieve business *goals*. Thus, it seems to be natural for business process modeling methods to include facilities for modeling these goals. However, relatively few approaches explicitly capture, refine and analyze business goals (e.g., [9, 6]). Most leading BP modeling approaches capture processes at a workflow level, in terms of activities, flows, etc. (e.g., [20]).

Due to the need to accommodate changing business priorities as well as business cases with varying characteristics (e.g., customers with different preferences), business process specifications need to be flexible as well as capable of being configured and reconfigured appropriately. Currently, techniques as diverse as business rules and late modeling are used for changing BPs. However, these approaches are usually quite low-level and the possible configurations are not explicitly evaluated with respect to business goals and priorities. Thus, it is hard to select process alternatives with de-

sired non-functional characteristics. Additionally, most of these methods require extensive knowledge of the process and, possibly, the modeling notation to be effectively applied thus making it difficult for non-technical users to configure BPs.

To alleviate the above difficulties, we are proposing a systematic business requirements-driven method for configuration of *high-variability* business processes at a high level, in terms of business priorities. In our approach, we start by employing goal models to capture and refine business goals as well as to explore and analyze the variability (the various ways these goals can be attained) in the business domain. Quality attributes such as customer satisfaction serve as the selection criteria for choosing among BP alternatives induced by the goal models. These high-variability goal models are then used in a semi-automatic variability-preserving transformation to generate customizable executable business processes (in our case study we use the Business Process Execution Language (BPEL) [16]). Through the preserved traceability links to goal models, the executable processes can be configured based on qualitative preferences of stakeholders. Automated analysis of the models is used at design time or at runtime to identify process alternatives that best match these preferences. A GUI tool for capturing user preferences and a prototype runtime infrastructure are also provided.

The rest of this paper is structured as follows. Section 2 provides some background on goal models, and on how they can be used for software configuration and to capture and analyze variability. Section 3 describes our approach in detail. Discussion and future work section follows, while Section 5 concludes the paper.

2 Goal Models and Preferences

In this section, we introduce goal models and the relevant work on using them for software configuration.

A major breakthrough of the past decade in (software) Requirements Engineering is the development of a framework for capturing and analyzing stakeholder intentions to generate functional and non-functional (quality) requirements – Goal-Oriented RE (GORE) [2, 3]. The main concept in GORE is the *goal*. For example, a stakeholder goal for a library information system may be Fulfill Every Book Request. This goal may be decomposed in different ways. One might consist of ensuring book availability by limiting the borrowing period and also by notifying users who requested a book that the book is available. This decomposition may lead (through intermediate steps) to functional requirements such as Remind Borrower and Notify User. A different decomposition of the initial goal, however, may involve buying a book whenever a request cannot be fulfilled[1]. Obviously, there are in general many ways to fulfill a goal. Analyzing the space of alternatives makes the process of generating functional and quality requirements more systematic in the sense that the designer is exploring an *explicitly represented* space of alternatives. It also makes it more rational in that the designer can point to an explicit evaluation of these alternatives in terms of stake-

[1] This is not, however, a very practical alternative.

holder criteria to justify his choice. An authoritative account of GORE can be found in [21].

At the very heart of this new phase of Software Engineering are goal models that represent stakeholder intentions and their refinements using formally defined relationships. Functional goals are modeled in terms of hard goals (or simply goals, when there is no ambiguity). For example, Supply Customer and Fulfill Every Book Request are functional goals that are either fulfilled (satisfied) or not fulfilled (denied). Other stakeholder goals are qualitative and are hard to define formally. For instance, Customer Satisfaction and Have a Productive Meeting are qualitative goals and they are modeled in terms of *softgoals*. A softgoal by its very nature doesn't have a clear-cut criterion for its fulfillment, and may be fully or partially satisfied or denied. Softgoals can be *satisficed* – met to an acceptable degree.

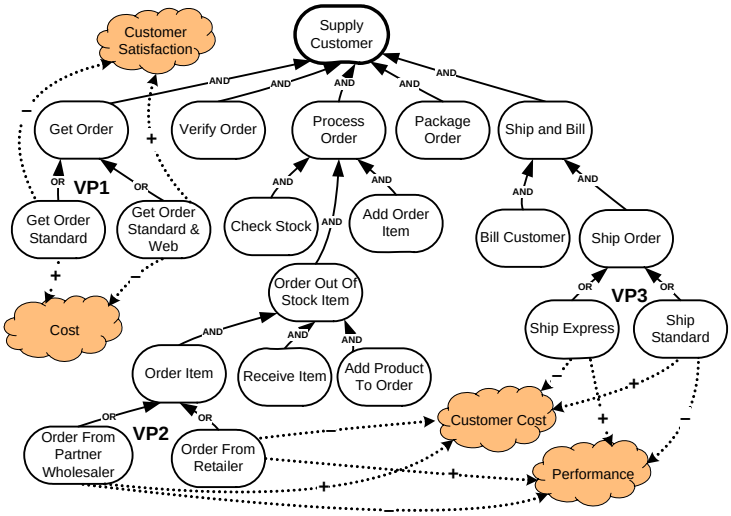


Fig. 1. A goal model showing interdependencies among goals and qualities

Goals and/or softgoals may be related through AND/OR relationships that have the obvious semantics that the AND-decomposed subgoals must all be attained for their parent goal to be achieved and at least one OR-decomposed subgoal needs to be achieved for achieving its parent goal. In addition, goals/softgoals can be related to softgoals through help (+), hurt (–), make (++), or break (--) relationships (represented with the dotted line arrows in Fig. 1). These *contribution links* allow us to *qualitatively* specify that there is evidence that certain goals/softgoals contribute positively or negatively to the satisficing of softgoals. Then, a softgoal is satisficed if there is sufficient positive and little negative evidence for this claim. This simple language is sufficient for modeling and analyzing goals during early requirements, covering both functional and quality requirements, which in this framework are treated as first-class citizens.

To illustrate what goal models are, let us look at a distribution company selling goods to customers. We will use this example throughout the remainder of the paper to demonstrate our approach. The company gets its products from wholesalers and

sells the goods to customers (see Fig. 1). It does not have any retail stores, so it receives orders though phone, fax, and, possibly, a web site and ships products using a shipping company. The top-level goal here is Supply Customer, which is AND-decomposed into a number of goals including Get Order, Process Order, and Ship and Bill [order]. Some of the subgoals have alternative solutions. For example, to ship an order, one can achieve either the Ship Express goal or the Ship Standard goal.

Quality attributes are represented as softgoals (cloudy shapes in the figure). In our example, the four top-level desired qualities are Customer Satisfaction, [Minimize distributor] Cost, [Minimize] Customer Cost, and Performance. Clearly, express shipping is fast, but expensive, thus it helps the softgoal Performance while hurting Customer Cost. Similarly, providing a web site for order submission (Get Order Standard & Web) may be more expensive for the distributor (thus the negative link to Cost), but contributes positively to Customer Satisfaction. As shown in Fig. 1, such partial contributions are explicitly expressed in the goal model. In all, the goal model in Fig. 1 shows eight alternative ways for fulfilling the goal Supply Customer. It is easy to verify that generally the number of alternatives represented by a typical goal model depends exponentially on the number of OR decompositions (labelled as variation points "VP1" through "VP3" in Fig. 1) present in the goal model (assuming a "normalized" goal model where AND and OR decompositions are interleaved). As such, goal models make it possible to capture during requirements analysis – in stakeholder-oriented terms – all the different ways of fulfilling top-level goals. A systematic approach for thoroughly analyzing the variability in the problem domain with the help of high-variability goal models is discussed in [14]. The paper proposes a taxonomy of *variability concerns* as well as the method for making sure these concerns are properly addressed during the goal model elicitation process. Now, if one were designing a flexible, customizable implementation for a process, it would make sense to ensure that the implementation is designed to accommodate most or all ways of fulfilling top-level goals (i.e., delivering the desired functionality), rather than just some.

Another feature of goal models is that alternatives can be ranked with respect to the qualities modeled in the figure by comparing their overall contributions to respective softgoals. So, the model of Fig. 1 represents a space of alternative behaviours that can lead to the fulfillment of top-level business goals, and also captures how these alternatives stack up with respect to qualities desired by stakeholders .

Goal Model Enrichments. While the goal models as described above are a useful tool in requirements elicitation and analysis, they lack precision and the level of detail for a more thorough analysis of the problem domain that is required for the subsequent design phases. For example, it might be important to model data/resource dependencies and the precedence constraints among subgoals in the problem domain. Similarly, specifying inputs and outputs for the subgoals in the goal model (i.e., what information and/or resources are required for the attainment of each goal and what resources and/or information are produced when the goal is achieved) is necessary for deriving precise system requirements. In general, a variety of enrichments can be used with goal models. The choice for enrichments depends on the types of analyses or model transformations that one would like to carry out on goal models.

We use textual *annotations* to add the necessary details to goal models. Most of the annotations specify the details of control flow among the subgoals. For example, the sequence annotation ("**;**") can be added to AND goal decomposition to indicate that

all the subgoals are to be achieved in sequence from left to right. Sequence annotations are useful to model data dependencies or precedence constraints among subgoals. The absence of any dependency among subgoals in an AND decomposition can be indicated by the concurrency ("||") annotation. Conditional annotations can also be added to specify that certain goals are to be achieved only under some specific circumstances. Lapouchnian and Lespérance [10] discuss various annotations, including loops, interrupts, etc.

It is important to note that the above-mentioned annotations capture properties of the problem domain in more detail and are not used to capture design choices, so they are requirements-level annotations.

Reasoning with Goal Models. While goal models are used for modeling and communicating requirements, we are also interested in the automated analysis of these models. To this end, Sebastiani et al. [18] devised a sound and complete goal satisfaction label propagation algorithm that given a goal model with a number of alternative ways to satisfy its goals and a number of softgoals, can be used to find the alternative that achieves the chosen subset of goals in the model while best addressing these quality constraints (in order of their priority).

Goal Model-based Customization and Configuration. There has been interest in applying goal models in practice to configure and customize complex software systems. In [4], goal models were used in the context of "personal software" (e.g., an email system) specifically to capture alternative ways of achieving user goals as a basis for creating highly customizable systems that can be fine-tuned for each particular user. The Goals-Skills-Preferences approach for ranking alternatives is also proposed in [4]. The approach takes into consideration the user's preferences (the desired quality attributes) as well as the user's physical and mental *skills* to find the best option for achieving the user's goals. This is done by comparing the skills profile of the user to the skills requirements of various system configuration choices. For example, for the user who has difficulty using the computer keyboard, the configurator system will reject the alternatives that require typing in favour of voice input.

Goal models can also be used for configuring complex software systems based on high-level user goals and quality concerns. Liaskos et al. [13] propose a systematic way of eliciting goal models that appropriately explain the intentions behind existing systems. In [23], Yu et al. show how goal models can be used to automatically configure relevant aspects of a complex system without accessing its source code.

3 The Approach

In this section, we describe our technique for business process modeling and configuration. It is requirements-driven and is motivated by the lack of support in most current BP modeling approaches for high-level, intentional configuration of business processes. The approach involves the modeling and analysis (using quality criteria) of alternative ways of achieving business goals with subsequent generation of executable business processes that preserve the variability captured at a goal level. The assumption behind this approach is that in the business domain where it is applied, the characteristics of business cases demand tailored business process variants. Below, we

briefly outline the steps of the process and highlight the responsibilities of various actors while the subsequent sections describe the process in detail:

Table 1. Overview of the process steps

	Responsible Role	Description	Artefact Produced
1	Business Analyst (BA), Business Users	Capture and refine the goals of the business process with emphasis on variability	High-Variability (HV) Goal Model
2	BA, Requirements Engineer	Enrich the model with control flow and I/O annotations	Annotated HV Goal Model
3	BA	Analyze BP alternatives, remove infeasible ones	Annotated HV Goal Model
4	Automated	Generate High-Variability BPEL specification from HV Goal Model	Initial HV BPEL process
5	BPEL/Integration Developer	Complete the HV BPEL process, select partner Web Services, deploy process	Executable HV BPEL process
6	Business Users	Select prioritizations among available quality criteria	BP Preferences, Configured Goal Model
7	Automated	Select the best BP configuration matching user preferences	BP Configuration
8	Automated	Create BP instance with the selected configuration, execute it	Configured BPEL process

3.1 Business Process Design with Goal Models

Using goals for business process modeling is not a new idea. A number of different goal modeling notations have been used for this [6, 9]. In addition, requirements goal models have shown to be a convenient notation for the elicitation, modeling, and analysis of variability in the context of software development, configuration, and customization [13, 23]. In our approach, we use high-variability goal models to capture *why* a business process is needed – its purpose or goal – and the many different ways *how* this goal can be attained. Business process alternatives implied by the models are then evaluated with respect to their quality (non-functional) attributes.

We continue to use the Supply Customer process from Fig. 1 in this section. We have added some more details to it in Fig. 2 (note that the annotations are described in Section 3.2). To model a BP in our approach we first identify its business goal (e.g., Supply Customer). This goal becomes the root of the goal model. It is then refined using AND/OR decompositions until the resultant subgoals can be delegated to either human actors or software services.

Let us walk through the Supply Customer process once again. First, customer orders are received either through phone, fax, or the web. After verifying an order, the distributor processes the order by checking if it has all the ordered goods in stock. If so, each product is added to the order. If some item is not in stock, it is ordered from either a wholesaler or a retailer. Ordering out of stock goods through the usual channel from a wholesaler is cheaper (positive contribution to Customer Cost), but requires more time (negative contribution to Performance), while ordering the goods

from a nearby retailer to complete the order has the opposite contributions to these softgoals. After an order is packaged, it is shipped (using either the express or the standard shipping method) while the customer is sent a bill.

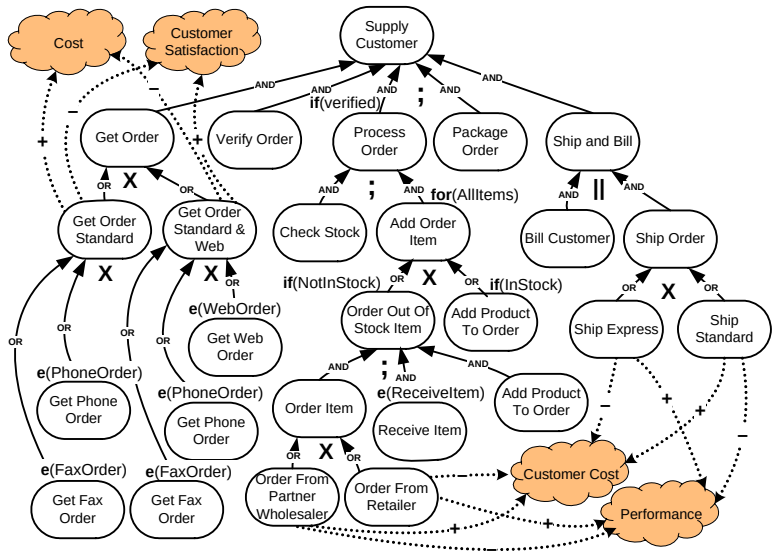


Fig. 2. A goal model for the "Supply Customer" business process

We are placing a special emphasis on business process variability since explicitly representing the space of alternatives using goal models allows for a systematic analysis and comparison of the various ways of achieving high-level business goals (i.e., the various BP alternatives). Whenever there is a number of different ways to achieve some business goal, the modeler uses OR decompositions to capture that fact. Some of these alternatives contribute differently to the non-functional business concerns such as Customer Satisfaction, [Minimize] Cost, etc. represented as softgoals in the model. We describe how these quality criteria are used in selecting the appropriate process configurations in Section 3.3.

3.2 Enriching Goal Models for BP Modeling

Since we are interested in the automated execution of business processes, we need to capture more information about BPs than the basic goal models allow. A few annotations are introduced for this purpose. Note that the annotations presented here are not required to be formal. We use the following control flow annotations when employing goal models to represent business processes:

- Parallel ("||") and sequence ("**;**") annotations can be used with AND-decomposed goals to specify whether or not their subgoals are to be achieved in a temporal order. For example, billing customers and shipping goods is done concurrently in the process.

- By default, in goal models, OR decompositions are inclusive. Exclusive OR decompositions are marked with the "X" annotation. All of the OR decompositions in our example in Fig. 2 are exclusive.
- Conditions ("if(condition)") indicate the necessary conditions for achieving subgoals. For example, in Fig. 2 the goal Order Out Of Stock Product is achieved only if the item is not already in stock.
- Loops ("while(condition)" or "for(setOfItems)"). For instance, the goal Add Order Item must be achieved for all items in the order.
- Event handlers or interrupts ("**e**(Event)"). In Fig. 2, the arrival of customer orders through fax, phone, or web is modeled by the events (e.g., **e**(PhoneOrder)) that trigger the achievement of the appropriate goals.

In addition to the above annotations, modeling of input/output parameters of goals is also important for BP modeling. Identifying inputs and outputs during the analysis of a business domain helps in determining resource requirements for achieving goals as well as for the sequencing of the goals. The types of inputs and outputs can also be specified. While optional, the input/output types can be used to generate detailed specifications for messages and service interfaces in a BP implementation. For example, Fig. 3 shows a parameterized fragment of the Supply Customer goal model. The parameters are specified inside goal nodes and the output parameters are identified with the star ("*") symbol. Deliberating about which resources/data are required for the attainment of a goal and which are produced when the goal is achieved can frequently help to identify important process details that are easy to miss otherwise. For instance, Fig. 3 adds the subgoal Pick Order Bin, which picks a location where ordered items are physically stored before being packaged.

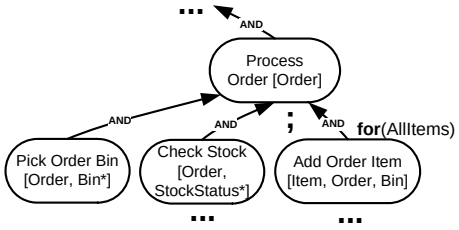


Fig. 3. Adding goal parameters

3.3 Specifying Goal Model Configurations

In goal models, there exist OR decompositions where the selection of alternatives is driven by data or events. For example, in Fig. 2 the OR decomposition of the goal Get Order Standard is event-driven as the choice depends on the way the customer submits an order. Similarly, the choice for achieving the Add Order Item goal depends on whether the item is in stock. However, there are other OR decompositions with alternatives, whose selection is not dependent on data/events. We call them *preference-driven* OR decompositions, or variation points (the data-/event-driven OR decomposition are not considered VPs as they cannot be used to configure processes). In the example in Fig. 2, these variation points are: Get Order, Order Item, and Ship Order.

From the point of view of the functionality of a business process, the achievement of any of the alternative subgoals of these VPs is exactly the same. The difference is in the way these choices contribute to the quality attributes of the process. These VPs play a central role in our business process configuration approach as they allow the selection of the best way to meet quality constraints of the stakeholders while delivering the required functionality of business processes. Thus, softgoals act as (possibly *conflicting*, as seen in our example) selection criteria for choosing the right BP alternative based on the priorities (among softgoals) of process owners, customers, etc.

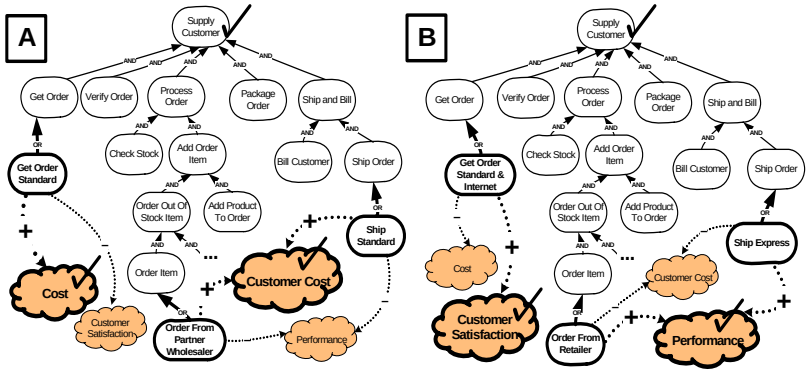


Fig. 4. Two alternative goal model configurations

To illustrate the above discussion, Fig. 4 shows two alternative configurations of the process Supply Customer. These configurations are the result of applying the top-down goal reasoning algorithm of [18] to the model in Fig. 2. The checkmarks indicate the highlighted (soft)goals, whose achievement we are interested in – the input to the algorithm (another input is the relative ranking of the softgoals, which we assume to be the same here). The first configuration (Fig. 4A) is where the Cost of running the process for the distributor and Customer Cost are the top priorities. The highlighted VP decisions contribute positively to the selected softgoals. This configuration includes, for instance, Ship Standard as it is cheaper. If, however, Customer Satisfaction and process Performance are the top priorities, then the configuration changes to the one in Fig. 4B. Thus, high-variability goal models provide a high-level view of processes with the ability to (automatically) generate BP configurations based on preferences of stakeholders expressed as prioritizations among quality criteria. These features greatly simplify the task of configuring business processes by non-technical users as these individuals can configure processes in terms of user-oriented abstract qualitative notions such as customer satisfaction, etc.

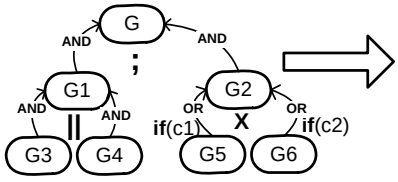
It is easy to notice that in our example, the goal model can be configured by multiple stakeholders, both from the point of view of the process owner (the distributor) by prioritizing among the Cost and the Customer Satisfaction softgoals and from the point of view of the customer by prioritizing among Customer Cost and Performance. This allows the stakeholder that owns the process to partially configure it based on that stakeholder's own preferences (i.e., the *binding* some of the variation points) while leaving other VPs unbound for the customers, partners, etc.

Note that the alternatives deemed not acceptable by the process owner (e.g., due to being too costly) can be removed from goal models, thus reducing the BP variability before the generation of executable BP models.

3.4 Generating Flexible Executable Business Processes

As we have just shown, goal models can be a useful tool for high-level configuration of business processes based on stakeholder prioritization among quality criteria. The above techniques can be used to develop, analyze, and configure BP models at design time. However, we would also like to be able to use the high-variability goal models as a starting point for the development of executable business processes that preserve the variability found in the source goal models as well as for configuring these BPs though the appropriate traceability links.

To this end, we have devised a method for using goal models to assist with the development and configuration of high-variability (flexible) BPEL processes. Unlike some workflow-level notations such as BPMN [20], our goal modeling notation is highly structured, with goals organized in refinement hierarchies. This makes it possible to generate BPEL processes (albeit lacking some low-level details) that are easily readable by humans and are structured after the respective goal models. The BPEL code generation is *semi-automatic* and the generated code, while not immediately executable and thus needing to be completed (mainly due to the fact that we do not require conditions in annotations to be formalized), nevertheless provides valuable help in producing an executable BP based on the source goal model. The code is to be further developed by integration developers, who will also be selecting/designing Web services to be used by the process.




```
<sequence name="G">
    <flow name="G1" ... >
        <invoke name="G3" ... />
        <invoke name="G4" ... />
    </flow>
    <if name="G2">
        <condition> [c1] </condition>
        <invoke name="G5" ... />
        <elseif>
            <condition> [c2] </condition>
            <invoke name="G6" ... />
        </elseif>
    </if>
</sequence>
```

Fig. 5. Example of WS-BPEL 2.0 code generation

Since BPEL is a workflow-level language, only activities that are executed by human actors or software systems (Web services) are represented in BPEL specifications. On the other hand, using goal models, we start modeling from abstract high-level goals and refine them into goals that can be assigned to humans or software. Thus, leaf-level goals correspond to the actual work that is done within a BP, while higher-level ones provide the rationale for why this work has to be done and how it relates to the ultimate purpose of a process. Thus, non-leaf goals do not create basic BPEL activities, but since they are used to group lower-level goals based on their decomposition

types (AND/OR) and control flow annotations, they help in generating the corresponding BPEL control flow constructs. We start BPEL generation from the root goal and recursively traverse the goal tree until we reach leaf goals.

We now present some of the goal model to BPEL 1.1 or 2.0[2] mappings through the example in Fig. 5, which shows a generic annotated goal model fragment. The root goal G has a sequential AND refinement, so it corresponds to the `sequence` operator in BPEL. G1 has a parallel AND refinement, so it maps to the `flow` construct. G2 has a data-driven XOR refinement (note the annotations), so it generates the `if-elseif` (BPEL 2.0) or the `switch` (BPEL 1.1) operator. Note that the conditions `c1` and `c2`, which are informal descriptions in the goal model, will be replaced with the appropriate conditions by a BPEL developer. The leaf goals correspond to Web service invocations. This is how enriched goal models are used to generate the overall structure of a BPEL process.

While we abstract from some of the low-level BPEL details such as correlations, with the information captured in the annotated goal models, we also generate the following aspects of BPEL/WSDL specifications (we do not show the complete mapping due to the lack of space):

- We do an initial setup by defining the appropriate interface (`portType`), etc. for the process. A special `portType` for invoking the process and providing it with the (initial) configuration is also defined.
- An event-driven OR decomposition (e.g., Get Order Standard in Fig. 2) maps into the `pick` activity with each alternative subgoal corresponding to an `onMessage` event. Since each such event must match an operation exposed by the process, an operation with the name of each subgoal is added to the `portType` of the process. A message type for the received event is also added to the process interface. A BPEL developer must define the message as the event annotations specified at the requirements level usually lack the required message details. The activities that are executed for each `onMessage` event are the BPEL mappings of the subtrees rooted at the subgoals in the decomposition.
- A conditional/loop annotation for a goal G is mapped to the appropriate BPEL construct (e.g., `if-elseif` or `switch`, `while`, etc.) with the activity to be executed being the result of mapping the goal model subtree rooted at G into BPEL. The formal conditions currently have to be specified manually.
- Leaf-level goals map into Web service invocations. The information in the goal model helps in defining the interface for the Web services invoked by the BP. We define appropriate WSDL messages based on input/output parameters of these goals. If data types are omitted from the goal model, they have to be supplied by a developer.
- Softgoals are used as the evaluation criteria in the configuration process and thus do not map into the resulting BPEL specification.

The main idea behind the generation of high-variability BPEL processes is the preservation of BP variability captured in goal models. As we have shown above, data- and

[2] While in our case study we generated BPEL 1.1 processes, WS-BPEL 2.0 [16] allows for simpler, more natural mapping from annotated goal models.

event-driven variability is directly preserved through the appropriate mapping to BPEL. Additionally, we need to preserve the preference-driven VPs in the executable BPs since they are the main vehicle for process configuration based on stakeholder preferences. In our approach, for each preference-driven VP we generate a BPEL `switch` construct (or `if-elseif` if using BPEL 2.0) where each case (branch) corresponds to an alternative subgoal (e.g., Ship Order in Fig. 2 will produce the cases for Ship Express and Ship Standard). The condition in each case checks to see if the case is the current choice for the VP by comparing the name of the alternative subgoal it corresponds to (e.g., "Ship Express") to the string extracted from the current BP configuration (see the next section for details), thus ensuring the correct branch is taken. The activities executed in each case are automatically generated and represent the BPEL mapping of the alternative subgoals of the VP. A VP also gets a name from the corresponding goal node (we assume that VP names are unique).

Fig. 6 shows our Eclipse-based goal modeling and analysis tool OpenOME [17] being used to design business processes with both the goal model (right pane) and the BPEL (left pane) visualizations.

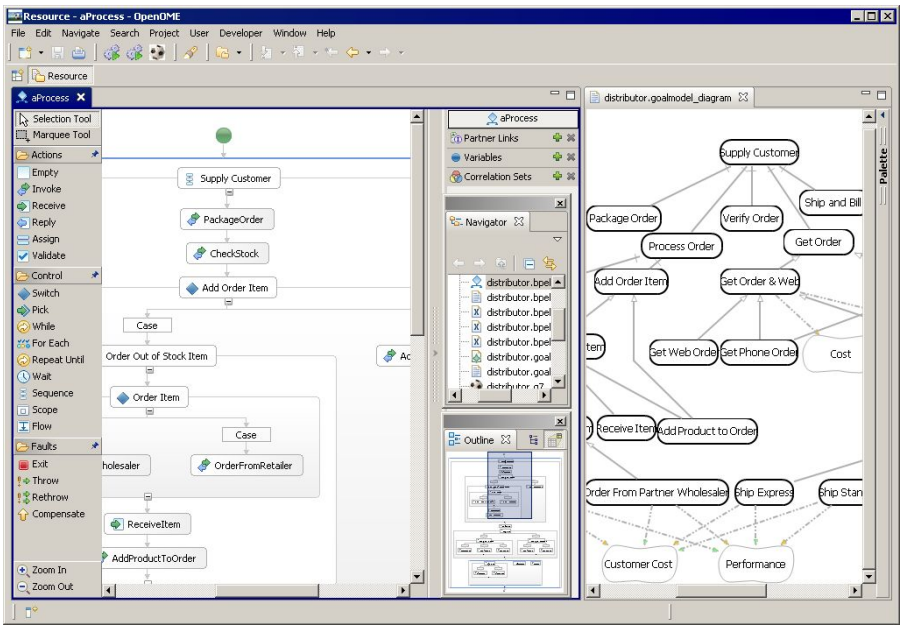


Fig. 6. OpenOME being used to design a business process

3.5 Quality-Based Business Process Configuration

Once a High-Variability BPEL process is fully developed and deployed, its instances can be configured by users through the prioritization among its associated quality criteria. This task has two elements. First, we elicit user preferences and generate the corresponding process configuration. Second, an instance of a BP has to be provided with this configuration. Let us look at these two subtasks in more detail.

There are several ways to specify user preferences in our approach. First, users can use OpenOME to specify which softgoals they want satisficed in a process and run a top-down analysis algorithm (similar to what we did in Fig. 4). The result will be a particular BP configuration that best suits the user. Another possibility is to use the GUI tool (see Fig. 7) that simplifies the task even more by only exposing quality attributes of a process and by allowing users to specify the partial ordering of the attributes in terms of their importance (Rank) as well as their expected satisficing level (with convenient sliders). Multiple profiles can be created for a particular BP model – for varying market conditions, customers, etc. Behind the scenes, a preference profile is converted into a goal model configuration (using the same goal reasoning algorithm of [18]). The tool can then create instances of processes with the desired configuration.

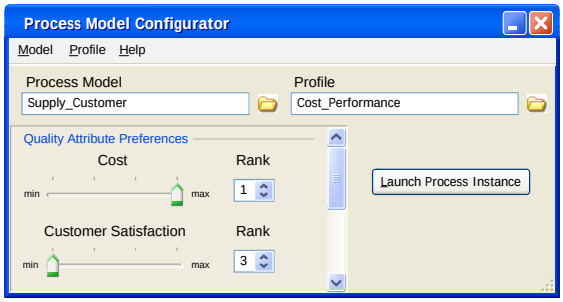


Fig. 7. BP preference configuration tool

Another part of our prototype BP configuration toolset is the Configurator Web service. This service implements a subset of the functionality of OpenOME, mainly the top-down reasoning engine and a persistent knowledge base for storing process configurations. It is designed to communicate these configurations to appropriate BP instances at runtime. The main operations of the service are as follows:

- *`registerProcess`* is used to associate a unique `processID` parameter with the endpoint of a deployed high-variability process (both are inputs).
- *`launchProcessInstance`* is be used by the GUI tool to create and run an instance of a process. Inputs are `processID` and a goal model configuration. A new instance of a BP identified by `processID` is created. It is given an `instanceID`, which uniquely identifies the process instance together with its configuration so that it is possible to evolve BP configurations independently. The configuration is stored in a knowledge base. The configuration and the `instanceID` are communicated to the BP instance.
- *`getConfiguration`*, which can be used by process instances to get their configurations from the Configurator Web service. The input is an `instanceID` and the output is the current configuration for that process instance. This operation can be used to get an updated configuration for a process instance.

The configuration provided to process instances is a list of variation points and the name of the selected subgoal in each of them. Below is an example:

```
<FullConfig>
        <VPConfig>
                <VP> ShipOrder </VP>
                <Selection> ShipStandard </Selection>
        </VPConfig>
        ...
</FullConfig>
```

Then, XPath [22] queries are used to extract the configuration. For example, the query `/FullConfig/VPConfig[VP="ShipOrder"]/Selection` extracts the configuration for the variation point `ShipOrder`. The result is matched with the appropriate case in the `switch` construct corresponding to the variation point as described in the previous section. Thus, the executing process becomes configured according to the user preferences specified in terms of priorities among quality criteria associated with the business process.

4 Discussion and Future Work

Most popular BP modeling approaches such as BPMN [20] or EPCs [8] are workflow-level notations. They do not allow the analysis of process alternatives in terms of high-level quality attributes or business goals and thus do not provide traceability of BP alternatives to requirements. There are, however, BP modeling approaches that explicitly capture and refine business goals (e.g., [6, 9]). Unlike our approach, these notations do not model process variability or the effect of alternatives on quality attributes. While some research has focused on variability in business process models [19], our approach centers on capturing and analyzing variability at the requirements level. Similarly, research on configurable BPEL processes (e.g., [5]) so far mostly concentrated on low-level configurability that may not be visible to process users.

A number of approaches based on the Tropos framework [1] applied ideas from requirements engineering and agent-oriented software engineering to BPEL process design [7] and SOA architecture design [12]. Both these approaches give heuristics on creating BPEL processes based on requirements models, but fall short from providing semi-automatic generation procedures. Likewise, BP variability is not explored. Nevertheless, we believe that agent-oriented modeling techniques of Tropos are useful for BP modeling and analysis and are currently working on integrating them into our approach. Similarly, we are looking at supporting context-based softgoal prioritization where the preferences change depending on the characteristics of business cases. For instance, in our Supply Customer example, the process owner may want to set Customer Satisfaction to be the top priority for high-valued customers.

We are collaborating with a large BPM software vendor to use our method with their workflow-level BP modeling and analysis notation and tools, thus allowing a more gradual development of BPs first with goal models, then with a workflow-level notation, and finally with BPEL while preserving the traceability among the notations, specifically, among the variation points.

This approach is part of a larger effort for developing requirements-driven adaptive business processes. To this end, we are working on implementing the support for the dynamic reconfiguration of high-variability processes based on changing require-

ments, stakeholder preferences and data captured by a BP monitoring environment. In terms of the approach presented here, dynamic process adaptation requires changes in the way BP configurations are updated and propagated to process instances. For example, one technique, which is currently implemented, is to push the updated configuration to the process instance through a special callback operation.

The drawbacks of the approach presented here include the need to explicitly model and analyze process alternatives as well as the fact that the qualitative analysis of alternatives may be too imprecise and subjective. We are working on integrating quantitative analysis of alternatives into our approach [11]. One of the elements of this addition to the method is a more precise specification of process alternatives' contributions to softgoals. Similarly, the softgoals themselves can be *opertionalized* into measurable quantities. Another extension that we are working on is the introduction of hard constraints that will play a role similar to the role of skills in [4] – helping to remove process alternatives that are incompatible with the characteristics of the process participants. We are also developing better tool support for this approach and working on improving the infrastructure and the generation of BPEL code as well as on supporting the modeling and analysis of BP exceptions.

5 Conclusion

We have presented an approach for requirements-driven design and configuration of business processes. Requirements goal models are used to capture and refine business goals with the emphasis on identifying alternative ways of attaining them while (possibly conflicting) quality constraints are used to analyze and select appropriate process alternatives. Goal model annotations for capturing process-relevant details are also introduced. Then, given an annotated high-variability goal model, a variability-preserving procedure generates a well-structured high-variability WS-BPEL specification (with programmers needing to fill in details of data handling, to define conditions and some other aspects of the process), which can be configured given high-level user preferences. A prototype system for preference profile specification and BP configuration is discussed.

The benefits of the approach include the fact that BPs can be automatically configured in terms of criteria accessible to non-technical users, thus greatly simplifying process configuration. The method helps in transitioning from business requirements analysis to BP design and implementation by allowing to gradually increase the level of detail in process models and by providing a semi-automated variability- and structure-preserving procedure for generation of executable business processes. The approach is also helping to maintain the processes' traceability to requirements.

References

1. J. Castro, M. Kolp, J. Mylopoulos. Towards Requirements-Driven Information Systems Engineering: The Tropos Project. *Information Systems*, 27(6):365-389, 2002.
2. A. Dardenne, A. van Lamsweerde and S. Fickas. Goal-Directed Requirements Acquisition, *Science of Computer Programming*, 20:3-50, 1993.
3. L. Chung, B. Nixon, E. Yu, J. Mylopoulos. Non-Functional Requirements in Software

Engineering. Kluwer, 2000.
4. B. Hui, S. Liaskos, and J. Mylopoulos. Requirements Analysis for Customizable Software: Goals-Skills-Preferences Framework. Proc. *International Requirements Engineering Conference* (*RE'03)*, Monterrey, CA, September 2003.
5. D. Karastoyanova, F. Leymann, A. Buchmann. An approach to Parameterizing Web Service Flows. Proc. *International Conference on Service-Oriented Computing 2005*, Amsterdam, The Netherlands, December 2005.
6. V. Kavakli, P. Loucopoulos. Goal-Driven Business Process Analysis Application in Electricity Deregulation. *Information Systems*, 24(3):187–207, 1999.
7. R. Kazhamiakin, M. Pistore, M. Roveri. A Framework for Integrating Business Processes and Business Requirements. Proc. *EDOC 2004*, Monterey, USA, 2004.
8. G. Keller, M. Nuttgens, A.W. Scheer. Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)". Technical Report 89, Institut fur Wirtschaftsinformatik Saarbrucken, Saarbrucken, Germany, 1992. (In German).
9. P. Kueng, P. Kawalek. Goal-Based Business Process Models: Creation and Evaluation. *Business Process Management Journal*, 3(1):17-38, 1997.
10. A. Lapouchnian, Y. Lespérance. Modeling Mental States in Agent-Oriented Requirements Engineering. Proc. *International Conference on Advanced Information Systems Engineering (CAiSE'06)*, Luxembourg, June 5-9, 2006.
11. A. Lapouchnian, Y. Yu, S. Liaskos, J. Mylopoulos. Requirements-Driven Design of Autonomic Application Software. Proc. *International Conference on Computer Science and Software Engineering CASCON 2006*, Toronto, Canada, Oct 16-19, 2006.
12. D. Lau, J. Mylopoulos. Designing Web Services with Tropos. Proc. *International Conference on Web Services (ICWS'04)*, San Diego, CA, USA, 2004.
13. S. Liaskos, A. Lapouchnian, Y. Wang, Y. Yu, S. Easterbrook. Configuring Common Personal Software: a Requirements-Driven Approach. Proc. *International Requirements Engineering Conference (RE'05)*, Paris, France, Aug 29 - Sep 2, 2005.
14. S. Liaskos, A. Lapouchnian, Y. Yu, E. Yu, J. Mylopoulos. On Goal-based Variability Acquisition and Analysis. Proc. *International Requirements Engineering Conference (RE'06)*, Minneapolis, USA, Sep 11-15, 2006.
15. J. Mylopoulos, L. Chung, and B. Nixon. Representing and Using Non-functional Requirements: a Process-oriented Approach, *IEEE Transactions on Software Engineering*, 18(6):483–497, 1992
16. OASIS: Web Services Business Process Execution Language Version 2.0 Primer (Draft). Available at: www.oasis-open.org/committees/documents.php?wg_abbrev=wsbpel, 2007.
17. OpenOME. Available at: www.cs.toronto.edu/km/openome/, 2007.
18. R. Sebastiani, P. Giorgini, J. Mylopoulos. Simple and minimum-cost satisfiability for goal models. Proc. *International Conference on Advanced Information Systems Engineering (CAiSE 2004)*, Riga, Latvia, 2004.
19. A. Schnieders, F. Puhlmann. Variability Mechanisms in E-Business Process Families. Proc. *International Conference on Business Information Systems* (*BIS 2006)*, Klagenfurt, Austria, 2006.
20. S. White. Business Process Modeling Notation (BPMN) Version 1.0. Business Process Management Initiative, BPMI.org, May 2004.
21. A. van Lamsweerde. Requirements Engineering in the Year 00: A Research Perspective. Proc. *International Conference on Software Engineering* (*ICSE'00)*, Limerick, Ireland, June, 2000.
22. World Wide Web Consortium: XML Path Language (XPath) 2.0 Recommendation. Available at: www.w3.org/TR/2007/REC-xpath20-20070123/, 2007.
23. Y. Yu, A. Lapouchnian, S. Liaskos, J. Mylopoulos. Requirements-Driven Configuration of Software Systems. Proc. *WCRE 2005 Workshop on Reverse Engineering to Requirements (RETR'05)*, Pittsburgh, PA, USA, November 7, 2005.