Non-Functional Requirements in Business Process Modeling

Christopher J. Pavlovski and Joe Zou
IBM Corporation
55 Pyrmont Street, Sydney
NSW Australia
chris_pav @ au1.ibm.com, joezou @ au1.ibm.com

Abstract

Business process modeling entails the capture of a set of tasks that invariably model the functional behavior of a system. Another aspect of business process modeling involves the accurate capture of operational behavior and the associated process constraints. Whether the process is automated or manual, such operational constraints and behavior exist. This may include a variety of properties including performance expectations, policy constraints, and security controls. These characteristics later manifest as the non-functional requirements of an intended system, and often such information is generally identified at some point after the business process modeling exercise. The non-functional characteristics of the business are arguably more difficult to capture in business process modeling, since the focus of such methods is the modeling of functional behavior. We propose how two new artifacts may be applied to model the constraints associated with a business process. This is the *operating condition* to denote a business process constraint and the *control case* to define controlling criteria to mitigate risk associated with an operational condition. Modeling constraints in this way provides an opportunity to capture these characteristics of business process early in the systems development life-cycle. This contributes to a model that provides a more complete representation of the overall business process. The methods will assist in mitigating risk and facilitate the early discovery of non-functional requirements during systems development.

Keywords: Business Process Modeling, Conceptual Control Case, Non-Functional Requirements, NFR.

1 Introduction

Conventional business process modeling includes the capture of functional tasks and steps that form discrete processes and sub-processes. While there is comprehensive coverage of these functional characteristics of the business, the non-functional requirements (NFRs) of a particular business task are not generally identified or captured. This may lead to key information being overlooked or deferred. This is particularly important when, from a business perspective, the need to address non-functional characteristics is business critical for certain tasks. This usually includes completion time, security privileges, the availability of a business process, and the regulatory or organization constraints that apply.

Non-functional requirements are also referred to as constraints, softgoals, and the quality attributes of a system (Mylopoulos, Chung, and Nixon 1992). In the context of business process modelling, the identification of high-level business constraints or softgoals is the focus for business users. Once the broad requirements are captured within the business process, then subsequent requirements analysis techniques will refine and specify further the individual requirements (both functional and non-functional). Given that business process modeling is an initial step in the requirements engineering process, the opportunity to capture some detail regarding the non-functional requirements is available to the analyst.

It is generally understood that the Business Process Modeling Notation (BPMN) does not support the expression of non-functional business requirements (Gorton and Reiff-Marganiec 2006). There is however, some reference to performance related requirements (OMG 2006). For instance the *AdHocOrdering* attribute denotes whether a task is to be performed sequentially or in parallel. This inherently addresses an operational constraint to ensure that a particular service level, or performance characteristic, is achieved with reference to finite shared resources. The specifications also suggest that groups may be used to highlight sections of a diagram without adding constraints for performance, which a sub-process would do. This implies that sub-processes may add such information; however no standardized approach for addressing the broader cast of non-functional requirements is outlined.

At the business process modeling stage the constraints, system qualities, and softgoals contribute to the definition of the non-functional requirements. At this initial level of requirements analysis, it would appear that a consistent approach with functional requirements discovery would necessitate the identification (or classification) of only the high-level non-functional requirements. The idea of using an *operating condition* and *control case* has been introduced to model NFRs and have been applied to use case modeling (Zou and Pavlovski 2006). The operating condition and a high-level control case may also be used at the business process modeling stage. More specifically, we propose the use of these two constructs as an extension to BPMN to model the non-functional business requirements. Business process modeling fundamentally involves the capture of dependency flow

within an ordered sequence of activities. However, supporting information may be often overlooked. Using the operating condition and control case to capture key business constraints and operational qualities, will facilitate the early detection of non-functional requirements, laying the foundation for refining these requirements. This notion of early requirements engineering is also argued by Yu (1997).

In the next section, we provide background to the related work and outline the motivations and contributions of our work. Section 3.0 introduces the proposed constructs, *operating condition* and *control case*, describing the BPMN notation used. In section 4.0 a detailed example is provided, illustrating how these artifacts may be used to capture NFRs during business process modeling. Section 5.0 describes how the capture of NFRS during business process modeling fits within an overall software development life-cycle. Finally, in section 6.0 a summary of the methods proposed here is given, we also discuss areas of further work.

2 Related Work and Motivation

While there is considerable work on modeling non-functional requirements and constraints in general process models (Lu, Sadiq, *et al*. 2006, Mylopoulos, Chung, and Nixon 1992, Bresciani and Giorgini 2002, Chung and Nixon 1995), the different approaches vary considerably in complexity and scope and may not be ideally suited for the business end users. The theme of modeling the non-functional properties of a system using BPMN is more limited, although there are several related works (Hepp and Roman 2007, Cysneiros and Yu 2004, Gorton and Reiff-Marganiec 2006, Demirors, Gencel, and Tarhan 2003). Even though non-functional requirements are not explicitly dealt with by BPMN, the straightforward approach would be to associate *annotated text* artifacts to activities where such constraints apply. This may be considered an unstructured approach to initiating the capture of these business constraints, as arbitrary information may be captured at the discretion of the analyst, and this may vary considerably or be excluded. Furthermore, annotated text does not naturally support machine interpretation.

Recker, Indulska, Rosemann and Green (2006) comprehensively analyse BPMN using the Bunge-Wand-Weber ontology for theoretical analysis and complement this with an empirical survey. Their theoretical model identifies nine different limits and shortcomings of BPMN. Some of these limits relate to *construct deficit*, suggesting that users are not able to apply existing BPMN notations to fully describe certain real-world phenomena. The authors also note that survey participants, with an IT background, are in favour of more BPMN symbols with extended expressiveness so they can add sufficient rigor for making their models fit for use in software implementation projects. We also suggest that a construct deficit exists for modelling NFRs.

Rosemann, Recker, *et al.* (2006) propose the concept of *context-awareness process design* in order to draw attention to flexibility and adaptation in process modeling. The authors reason that contextual changes, (for example) such as increased incoming phone calls during storm season require integration into process design. Such context aware information clearly entails non-functional requirements in process design.

Hepp and Roman (2007) point out that constraints are relevant for modeling business processes, noting that the process space is further influenced by constraints from legal, regulatory, or managerial rules. The authors further suggest that such sequencing in a workflow-centric model may meet the constraint, but does not actually capture the constraint. They observe that workflow-centric modeling does not distinguish between the constraints of a process and the execution of the process. It is suggested that while some constraints are stored within the actual process, in order to remove redundancy and improve maintenance of such processes, such information is to be stored separately. They discuss the relationship between the workflow-centric representations and enterprise ontology, proposing an 'enterprise rules and constraints' ontology to assist in identifying constraints.

Yu (1997) argues that a different approach for requirements modeling is necessary in order to model requirements during early phases of systems development. He proposes the i* framework modelling technique, with emphasis on "why" this is done from an organisational perspective, rather than the "what" aspect during requirement modelling. Cysneiros and Yu (2004) also point out the need to model dependency, freedoms, and constraints. They discuss agent autonomy and show how the i* framework is used in a health care business process scenario to illustrate how constraints such as goal, task, and resource dependencies may be modeled. In particular, they focus their attention on how softgoals, such as response time, are addressed, observing that these are qualitative in nature. They further discuss ideas on how to map the i* model to BPMN, in order to capture these constraints and behaviour. These techniques help to elicit the business constraints. However, the mapping of specific constraints and softgoals to BPMN is preliminary and not well defined.

Since BPMN does not readily support the expression of non-functional business requirements, a graphical notation that builds upon BPMN for supporting constraints such as policies has been proposed (Gorton and Reiff-Marganiec 2006). This notation extends work on policy enforcement languages for telecommunications call control (Turner, Reiff-Marganiec, *et al*. 2006). The authors suggest their notation is more specific to modeling business processes to be used as web services. Tasks are modeled as business activities, with input and output controls added. Composite tasks are termed task sub-maps and are subject to policies that alter the flow of behaviour within the task map. Both data and control flows are modeled, with several functions defined that operate on control flows. The approach is comprehensive, though this adds complexity to the business process model, which may be more difficult to understand by business users.

Other practical studies in eliciting both functional and non-functional requirements from business processes have also suggested that the non-functional requirements

may be determined by analysing implicit requirements of the business process models (Demirors, Gencel, and Tarhan 2003). Given that some literature is dedicated to the constraints and operational performance requirements that may be applied, it seems that a construct that can be used with BPMN will be useful to system modelers and analysts.

There is general acceptance that the key aspects of process constraints, which manifest later as non-functional requirements, are not properly addressed in business process modeling. The previous works have sought to address aspects of modeling constraints; however, comprehensive coverage of all operational and non-operational NFRs appears not to be addressed. Pesic and Van Der Aalst (2006) point out that when constraints are modeled by sequence of activities this leads to over specification, which also leads to redundancy. Given that a key objective of BPMN is to model business process in a way that is easily understood by the business end users and analysts, this motivates the need to model non-functional business requirements whilst ensuring the model remains uncomplicated.

In this paper, we suggest an extension to BPMN that allows the business constraints and operational qualities, (NFRs; i.e. *context awareness*), to be identified and modeled during the early requirements engineering phase. The proposed extensions build upon the previous work by outlining artifacts that model both the business constraints and operational behaviour, illustrating how these artifacts are applied within a development life cycle. Our work does not alter the semantic flow or syntactic approach to business process modeling, but rather complements the existing notation. The main idea is to apply the *operating condition* and *control case* to business process models. These constructs have been proposed in use case modeling and the applicability of this notation to business process modeling was also suggested as further work. Using these artifacts in this way provides an opportunity to identify, at a business level, the non-functional properties of a business process. Moreover, we view the main contribution as follows.

1. Illustrate how the *operating condition* and *control case* are able to complement the existing BPMN method to commence high-level capture of NFRs.
2. Show how this initial step fits naturally within a software development life-cycle as a pre-cursor to the subsequent requirements gathering techniques and machine interpretation.
3. Formally define new BPMN artifacts for modeling non-functional requirements that are associated to business flow objects.

3 Modeling Business Process Constraints

At the business process phase of systems development the functional requirements are captured using a business process model. We wish to take advantage of the opportunity to commence discovery of the non-functional requirements during this process modeling phase. In order to maintain the principle of being readable by business users, the requirements gathering activity needs to be conducted at a suitable level of detail. Hence the intention is to bootstrap or initiate later phases of systems development that specify and capture detailed non-functional requirements.

In more specific terms, we purpose artifacts to identify the set of constraints applicable to a business process. This may include security policies, organisational policies, regulatory constraints, and operational performance characteristics. In order to model the non-functional requirement that is associated with a process task, the *operating condition* is used to denote that such a constraint is associated to a flow object. The *control case* is further used to define the business controls to be put in place to manage the risk associated with the identified operating condition. The level of requirements detail identified during business process gathering will mandate that only the high-level, or a conceptual, control case is discovered at this stage.

3.1 Operating Condition and Control Case

There are two broad categories of non-functional requirements, the operational and non-operational NFRs. The operational NFRs typically include system performance, reliability, security, response time, quality of service, and system availability. The non-operational NFRs generally include prescribed technology and development standards, portability characteristics, maintainability, and architectural constraints to be applied. At the business process level, implementation detail is not a concern, but rather the business related constraints are to be understood. As such, the set of business constraints (or operational conditions) that may be discussed during business process modeling include:

- performance of a task;
- security policies that apply;
- availability of an activity or process;
- activity response time,
- organisational standards that apply;
- regulatory constraints; and
- quality of user interaction with activity.

This list is not intended to be exhaustive, rather it provides context to the level of detail that is the focus at the business process level. Using this context, we now describe the two proposed artifacts, *operating condition* and *control case*, for capturing high-level non-functional requirements during business process modeling.

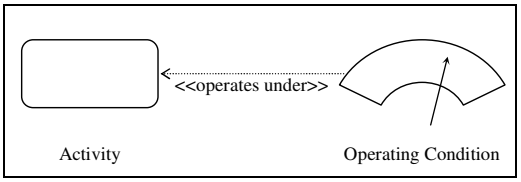

Figure 1: Operating Condition and Business Activity

The first step in defining non-functional requirements is to identify the operating conditions. The operating condition serves as a classification or grouping of constraints, hence it is a high-level view of potential non-functional requirements to be defined. At the business

process modeling stage, defining the operating conditions blends with an evolutionary approach to identifying requirements by establishing the groups and types of constraints that are applicable to a business process. The constraints listed above are applicable to an activity, and we use the operating condition notation to denote the business constraints that are associated with an activity, see Figure 1.

The operating condition used in this way declares additional semantic constraints for the process activity. It is not intended to introduce or disrupt the sequenced flow of activities, nor does it introduce or receive message flows. The operating condition signifies an applicable constraint and is an associated artifact to a business activity.

When an activity is identified as having some associated operating condition, it may be necessary to further define control mechanisms that are to be put in place to control the operating condition. This is in order to mitigate or reduce the business risk; the 'why' aspect of requirements engineering (Yu 1997). The additional control mechanisms are defined within the control case. A control case is modeled using both notation and text that captures the additional properties of the business constraint and the controls to be applied. Once again, at the business process level only the high-level business controls, that are relevant to the business process analysed, need to be captured. As such, a conceptual (or high-level) control case seems appropriate to specify the control information. The control case may also be considered an optional artifact to model, since it also refines further, by providing additional information, the identified operating condition. The diagram below illustrates the notation for the control case, see Figure 2.

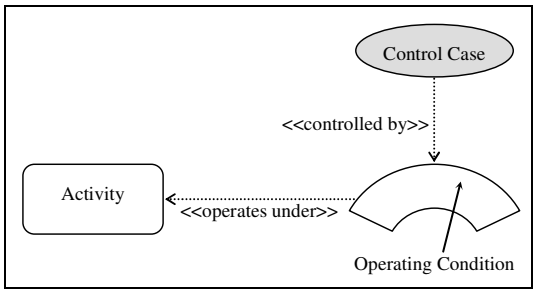

Figure 2: Control Case and Operating Condition

The control case is associated with the operating condition and is denoted as a shaded ellipse. The textual description of the control is catalogued further as text. This is shown in Figure 3. The control case captures the business risk associated with the operating condition and the business controls to be put in place to mitigate the risk. For instance, non-compliance to a regulatory constraint will result in financial penalty. A control may be put in place to ensure conformance to additional quality assurance standards is met, such as quality review, employee training, or external audits/inspections. Such controls will mitigate the risk of non-compliances and reduce the business exposure. A further example is availability of a business activity. For instance, the ability to authorise and process payment card transactions is 24 × 7 for many institutions. Loss of such availability may directly impact the revenue of the business, and hence business controls may be established to ensure that the business process has suitable redundancy applied. At the business process level we make no assumption if the process will be automated (implemented as an IT system) or manual (with human resources). The operating conditions and control cases identified and defined in this way are equally applicable to implementation outcome.

CONTROL CASE: Name of the Control Case
Operating condition: Name of associated operating condition.
Description: Description of the risk due to the operating condition and the focus area in mitigating the risk.
Business Constraint: Description of the constraint to the business process.
Business Risk: Explanation of the business risk or threat.
Business Controls: Controls to be applied to mitigate risk (e.g. ISO90001).

Figure 3: Control Case

When modeling business constraints with operating conditions, it is not necessary to name the associations in practice. In addition, the control case may be optionally used to capture additional information regarding the controls to be applied, and is anticipated to appear in sub-process diagrams rather than the top level business process model. Taking the opportunity to model these aspects of the business will provide critical input to subsequent detail requirements analysis and brings attention to business risk that is otherwise often overlooked.

3.2 Artifact Definition and Formal Notation

In a formal sense we now describe the notation for the modeling tools outlined in the previous section. Using the BPMN as a well defined notation for business process modeling, we define the artifacts *operating condition* and *control case* as follows.

ArtifactType (DataObject | Group | Annotation | Operating Condition | Control Case)

Attributes	**Description**
ConstraintType (organisational | policy | security | quality | regulatory | availability | other)**:** String	Name and type of the operating condition. The business constraints may be further sub-divided or extended where appropriate. For example quality may indicate a user interaction or quality of service constraint. Security may be designated as confidentiality, integrity, or authorisation. Performance related constraints such as response time may be added.

Table 1: Operating Condition Attributes

This extension is consistent with the standard notation, as it is pointed out that the ArtifactType list may be extended to include new types (OMG 2006). The corresponding attribute definitions for the operating condition and control case are shown in Table 1 and Table 2 respectively.

Attributes	Description
Name: String	Name is an attribute that is a text description of the object.
Description: String	Description of the control case and the focus area in mitigating the risk.
Constraint: String	Description of business constraint identified by operating condition: i.e. organisational, policy, security, regulatory, etc.
Risk: String	Description of the business risk.
Controls: String	Description of controls to be applied to mitigate business risk.

Table 2: Control Case Attributes

Since only a high-level (conceptual) control case requires definition during business process development, a subset of the attributes of a control case (Zou and Pavlovski 2006) require definition. The following table describes the attributes that require attention.

The symbols used for operating condition and control case are defined artifacts in the process diagram; the core BPMN notation from OMG (2006) is extended to include the operating condition and control case as artifacts. The additional information shown in Table 2 for the control case would be catalogued elsewhere, and would supplement the process diagrams.

3.3 BPMN Criteria for Artifact Definition

In terms of BPMN artifact definition, we assess whether the principles for sequence flow and message flow connection are preserved. OMG's BPMN notation specification allows the modeler to add new artifact types to provide additional information on the process (OMG 2006). The operating condition provides additional information regarding the environment state where the process or activities operate. On the other hand, the control case allows the process stakeholders to appreciate the NFRs that mitigate the risk to the process or activity, and the control mechanism to achieve the NFRs. The BPMN principles for artifact sequence and message flow connections are given below (OMG 2006). Moreover, an artefact:

- must not be a target for sequence flow.
- must not be a source for sequence flow.
- must not be a target for message flow.
- must not be a source for message flow.

The proposed operating condition and control case do not invalidate these principles. The proposed artifacts are neither a target nor a source for message or sequence flows. Rather the operating condition is associated with a flow object to identify an operational constraint that applies. In addition, the control case is associated with an operating condition, depicting the controls to be placed upon the process activity, via the operating condition, in order to mitigate risk. It follows that the BPMN sequence flow rules and message flow rules are satisfied.

4 Scenarios in Business Process Modeling

In this section we use a banking ATM business process as an example to demonstrate how the operating condition and control case can help business process analysts to identify non-functional requirements to mitigate risk.

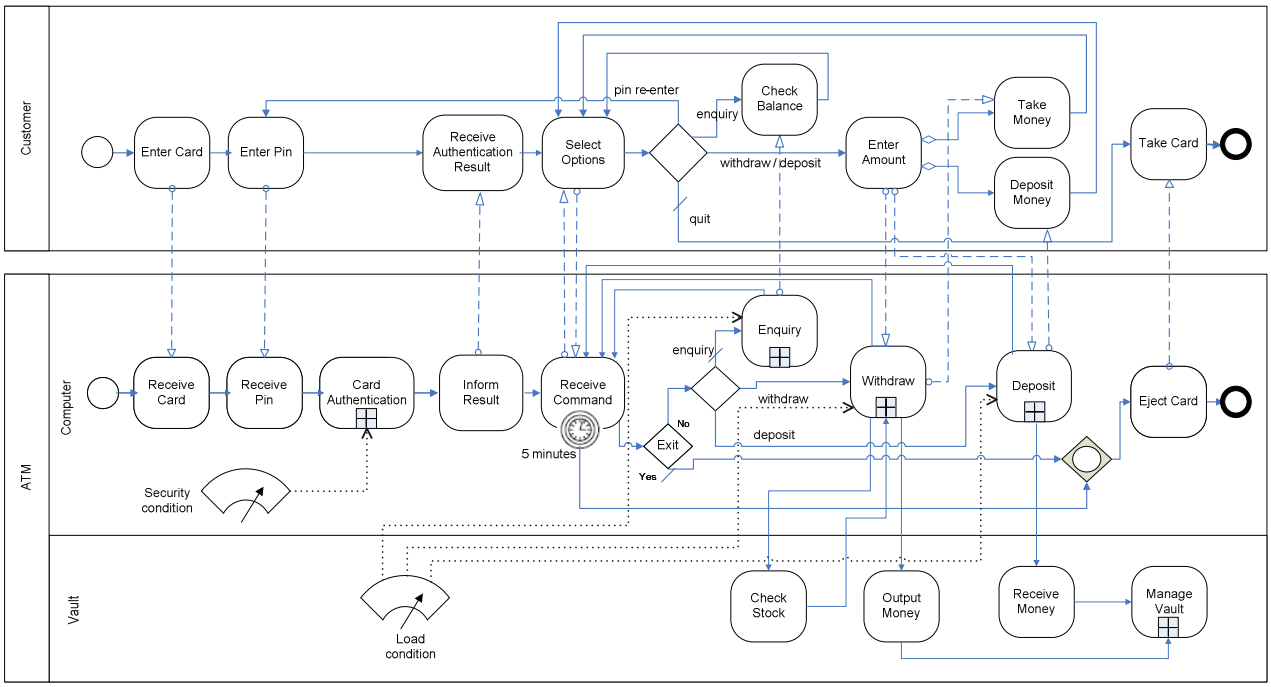

Figure 4: High Level ATM Business Process with Operating Condition

In practice, the analyst may model a business process that comprises a single start node and trigger in one map. In our example we use a combination of collaboration processes, swimlanes, and message flows between pool boundaries to provide context of complexity when introducing the operating condition and control case.

Figure 4 shows a high level, simplified ATM business process. From a goal oriented requirement engineering perspective, the focus is on the enterprise goal and the process goal (Kavakli and Loucopoulos 2005). A reasonable enterprise goal in this case is attaining customer satisfaction while reducing banking operational cost. The process goal here is to satisfy the customer's ATM transaction requests in an efficient manner. When a financial institute faces the task of implementing this generic process, it will need to first analyse the specific operating conditions around the process and understand the risks involved with those conditions. Then it may need to add control mechanisms to mitigate the business risk for specific processes.

The physical location of the ATM and also the banks automated IT systems determine the operating conditions of the ATM process. For example, the operating condition for an ATM inside the branch is very different to one in a shopping mall; and it is quite different again with one on a street as well. For instance, a shopping mall ATM would expect a higher daily transaction volume, therefore requiring more frequent stocking on notes. Conversely, being situated on the street may expose higher risk of theft and fraud which may require a stronger degree of physical security controls.

Identifying the operating conditions that are associated with the activity is a first step to collect the non-functional requirements for the business process. We now discuss how the operating condition is applied in Figure 4 below. In this example, two operating conditions are added to the original BPMN diagram as follows.

Security Condition: This is the security policy associated with the card authentication activity. This means that additional authentication requirements apply which may include pin strength policy such as a minimum number of digits.

Load Condition: The transaction load condition is associated with the three sub-processes: withdrawal, enquiry and deposit. This has implications on the sub-process response time to ensure that these are conducted in a timely manner.

The two operating conditions illustrated may also be associated with sub-processes that will typically require further refinement. In general, at the highest level of the business process model, it is anticipated that only the operating condition would need to be illustrated. In subsequent diagrams that decompose sub-processes, the control cases may be added. The decision whether to add a control case to the diagram is best decided by considering whether the operating condition is sufficient to draw attention to the non-functional requirement. During business process modelling, in many instances the type of business constraint will be well understood, such as the security policy or authentication requirement. In other cases, where variants exist or where the operating condition may apply differently to a range of activities, the control case will help to elicit and document these variations in constraint, risk, and business controls.

Returning to the ATM example, we refine further the NFRs by identifying additional operating conditions and control cases while decomposing sub-processes. We choose the cash withdraw sub-process as an example. In Figure 5 below, the load condition (defined in Figure 4) is further refined and associated with two activities: check account balance and credit/debit account. A control case is also associated with this operating condition, declaring controls over response time for the load condition. A security policy condition and control case is also defined for these two activities to manage confidentiality. For each control case defined, a textual description of the control is catalogued further. The control case description for 'control confidentiality' is shown in Figure 6. This describes the constraints, risk to the business, and business controls to be applied to mitigate that risk.

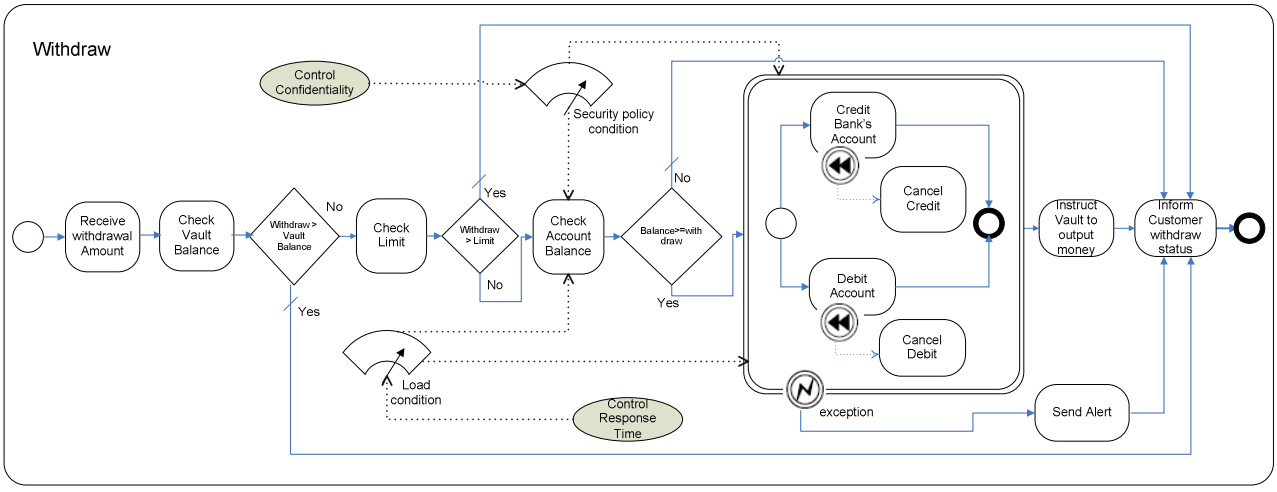

Figure 5: Operating Conditions and Control Cases in the Withdraw Sub Process

CONTROL CASE: Control Confidentiality
Operating Condition: Security Policy Condition.
Description: Confidentiality and integrity of sensitive customer information is to be maintained during the performance of the business process.
Business Constraint: Australia Payment Clearance Association (APCA) has mandated 3DES encryption for handling of sensitive personal data in ATM transactions.
Business Risk: Transmit sensitive data in plain text or weak encryption such as DES will expose the business at risk of non-compliance and potential litigation.
Business Controls: To mitigate the legal risk and interoperability risk, adopt the mandated encryption standards and supporting technology. • For existing ATM in non-compliance, upgrade to 3DES. • For new ATM, purchase ATM from the APCA approved list. • Adopt Remote Key Loading techniques.

Figure 6: Completed High Level Control Case

The approach outlined can be used to identify further operating conditions and control cases for other sub-processes as required. The control case specifies additional granularity of information in an incremental fashion. In general, a top-down process refinement will be applied in an iterative and incremental fashion to ensure all significant operating conditions and control cases are captured.

5 Software Development Life Cycle Usage

We now illustrate how the use of the control case and operating condition in business process analysis, contributes to defining non-functional requirements as part of an overall software development life-cycle. For any software development methodology, whether it is agile based or plan-driven, software development involves essential activities such as requirement analysis, design, implementation and test (Royce 1970, Boehm 1988). A BPM scenario also includes the feedback control from runtime monitoring back to design (Muehlen and Ho 2005).

In order to describe how the capture of NFRS during business process modeling fits within an overall software development life-cycle we adopt a generic method that includes activities such as business analysis, requirement definition, architecture and design, implementation, testing, operation/maintenance and run time monitoring. At each step we show how the identified business constraints evolve further into detailed non-functional requirements. In particular, business constraints identified during business process analysis are used as input and refined further to define non-functional requirements during detailed requirements definition in subsequent phases. The operating condition and control cases defining NFRs are then applied to formalise the solution architecture, may be machine interpreted to aid in software construction, and are used to define acceptance criteria for testing the solution.

Although the following describes the development activities in a sequential manner, we do not assume that the development of control case follows this strict process. In fact, iterative and incremental development within each step is likely to deliver the best outcome, as advocated by the agile software development approach.

5.1 Business Analysis

At the early requirement elicitation stage, various techniques such as interviewing, scenarios analysis, soft systems methods, prototyping and participant observation can be used to collect requirements (Kotonya and Sommerville 1998). Through those techniques, the analyst will come up with some raw requirement information such as business goals, process models, KPIs, standards and regulation constraints.

At the business analysis phase the focus is to identify the high-level functional requirements that are essential to achieve the business goals. On the other hand, it is also equally important to identify the major risks that may undermine those business goals. The risks stem from the business constraints, which manifest in various operating conditions in which the business process has context. The non-functional requirements describe the controls required to manage the risks. Thus a business model presented in BPMN notation provides a suitable starting point to identify the major operating conditions and the high-level control cases for managing the identified risks.

The high-level operating conditions and control cases can be added to the BPMN model through the proposed new artifacts, thus the business constraints are captured in the process model and communicated across various stakeholders. At this stage, only the key operating conditions and high-level control cases need to be highlighted in the process model, as the objective is not to compromise the readability of the BPMN model.

There are several benefits in using the proposed artifacts, in comparison to annotated text in BPMN. The analyst is guided to consider the constraints in a more structured and rigorous manner. This is in terms of the operating conditions associated with the process and the related risks, rather than drawing non-functional requirements based on tacit knowledge from past experience or *ad-hoc* estimates. Annotated text is unstructured meaning that arbitrary information may be captured at the discretion of the analyst, and this may vary considerably or be excluded. The formal artifacts can be machine interpreted for conversion into code fragments (see 5.4 and 6.1); such a task may not be possible with free form annotated text. In addition, the operating condition and control case artifacts provide a vehicle for business analysts to discuss and commence capture of non-functional requirements, which would otherwise be left to later phases such as the architecture, design, or construction stage.

Once again, it is important to note that only the high-level control cases are identified at this phase. This constitutes a small sub-set of the total control cases that will be

defined in the next phase, detailed requirements definition. This also ensures that an original objective of BPM is preserved; to model organisation business processes while hiding implementation detail. Additionally, the identified operating conditions serve as the primary mechanism for identifying the complete set of control cases during the detailed requirements definition. In the next phase, the analyst would review each identified operating condition and determine what are all the controls required to manage the business risk.

5.2 Detailed Requirements Definition

During the detailed requirements definition stage, both the functional and non-functional requirements are refined and captured in a precise form. In the case of functional requirements, the common practice is to use a use case model to capture the requirements. For non-functional requirements, the high-level control cases identified in the business analysis stage can be further elaborated to a detailed control case model (Zou and Pavlovski 2006). The fundamental approach involves reviewing each identified operating condition from the business process model and determining all possible associated control cases. Some of the key control cases have been already identified by the business process model, and these serve as starting points to identify the complete set of control cases required to manage the business risks. During this process new operating conditions may also be revealed.

The elaboration process adopts a risk-based decision-making (Haimes 1998) approach to set realistic targets. The detailed control case model provides a formal structure that incorporates detailed operating conditions, risk ranking, quantifiable targets for operational NFRs such as availability, response time and throughput, policy or procedure for non-operational NFRs and finally the residue risk. The control case can be associated with use case through the operating condition of the use case. It can also capture cross system non-functional requirements, which are not specific to a particular use case. Thus the combination of the use case model and the control case model provides a complete picture on the system requirements.

5.3 Architecture and Design

At the architecture and design phase, the NFRs captured in control cases will be evolved from "what" to "how". Control mechanisms and required technology for achieving the targets will be defined by the control case. For example, security access control technology will be used to achieve security requirements. Clustering, load balancing and caching technology will be used to achieve availability and performance targets. The control mechanism provides indicative costing, which allows the analyst to conduct risk/cost trade-off and re-examine the defined NFRs.

The control case model will provide the basis for the architect to choose the technology standards, platforms and existing components. This also provides input when designing the security architecture, application deployment, and physical hardware architecture. In particular, the non-functional requirements enable the designer to address the operational aspects of the architecture such as scalability, load balancing, redundancy and fail-over.

5.4 Construction

During the construction phase, the software developer builds the components based upon the design specifications developed from use case requirements. Where available, the control cases defined from BPMN models can be machine interpreted for conversion into comments, code fragments, or XML policies for inclusion to generated code (also an area of further work; see section 6.1). This also applies to detailed control cases defined during the requirements definition phase using suitable control case modelling tools.

The associated control case will remind the developer that there are acceptance criteria to be met in terms of non-functional requirements. For example, this will clearly assist in coding decisions where key performance criteria, such as response time, must be met. Thus the non-functional requirements will not be overlooked by developers.

5.5 Testing

Often the quality of the software has a direct relationship to the quality of the test case prepared. This may be true no matter which software development approach is taken, whether it is test-driven development or traditional test cycle during a waterfall SDLC. Traditionally, the functional test case is developed based on a use case. However, there is a gap in developing the test cases such as performance and security. The gap can be addressed by the control case model developed during the requirement definition stage. The test case and acceptance criteria may be based upon the defined control cases and provide a consistent mechanism for communicating the non-functional requirements during systems testing.

5.6 Operation, Maintenance, and Monitoring

During operation and maintenance, the service level agreement is an important issue for all the stakeholders. The control case can be used as an input to various parties to negotiate the service level agreement. In order to observe and record the system behaviour in production, additional tools are deployed to monitor the launched system. The intent is to observe certain operating conditions and measure the real performance of the system. Operating conditions and control cases prepared during the requirement analysis and definition stage can be used as input when formulating the monitoring solution. For instance, the proposed artefacts help to identify the key areas that require reporting and measurement.

5.7 Summary of Expected Benefits

In summary, we note that the notion of operating condition and control case provides input to each stage of a software development lifecycle. In addition, the

artefacts may be used to contribute to project estimation, preparation of work task break downs, and assist in risk management. Muehlen and Ho (2005) identify various risks in each stage as well as during stage transition of the BPM lifecycle. The identified risks that are associated with non-functional requirements are mainly in categories of communication, information and system/technology:

- Lack of communication between stakeholders,
- Inadequate information,
- Lack of technology flexibility,
- Lack of technology compatibility, and
- Lack of technology scalability.

The control case approach can improve the communication of NFRs between stakeholders. It also captures the important NFR information both in operational and non-operational categories. As such we suggest that by adding the operating condition and control case artifacts to the BPMN model, risk can be mitigated in BPM, as well as facilitating NFR discovery during the software development lifecycle.

Furthermore, there are several advantages using the proposed artifacts over annotated text. Annotated text is free from and may not be directly useful to machine interpretation for code generation. While NFRs captured as annotated text may be converted to documentation elements, the proposed new artifacts also provide a structure that supports a more formal conversion to policies and code fragments for use during software construction. In addition, these may be used in decisions steps during detailed requirements, architecture, testing and operation and maintenance.

6 Conclusions and Discussion

At the top-level business process model it is appropriate to depict the operating condition. As the business processes are decomposed further, the control case may be introduced to increase the granularity of information. The addition of the operating condition and high-level control case to the business process model provides a framework for capturing business constraints and operational qualities. The use of an operating condition within BPMN allows business users to express and discuss these requirements. This facilitates the capture of non-functional requirements, providing early visibility to all business stakeholders.

A key preliminary step in determining non-functional requirements is to identify the type and categories of NFRs. The operating condition is the first (high-level) step to identify such business requirements. Incorporating this construct within BPMN blends well with the overall approach of iterative decomposition of requirements, providing a fundamental and preliminary step to identify and communicate these characteristics of business activity. Subsequent methods, such as requirements design by use case, are then able to extend the process models in a natural way to further refine the detailed requirements.

The control case may also be introduced at a high-level to further define the relevant business control to mitigate risk. As a pre-cursor, the high-level control case provides a framework for further definition as a more fully developed control case is developed in later life cycle phases. A key step to be able to describe adequate control cases is an identified catalogue of operating conditions.

As suggested in BPMN, additional standard artifacts may be added to the BPMN specification (OMG 2006). Taken together, the modeled business process provides early visibility of the complete set of requirements to end users, business stakeholders, business analysts and system implementers. This provides an approach to model NFRs in early phase requirement engineering as argued by Yu (1997); the control case addresses ‘why’ the requirement is captured, by defining the risks, and the operating condition expresses ‘what’ is to be modeled. It is hoped that the operating condition and control case are suitable candidates for addressing the key task of discovering business constraints during process modeling, laying the foundation for defining detailed non-functional requirements in subsequent phases.

6.1 Further Work

The mapping of the proposed artifacts to both documentation elements and source code is an area of further work. In the straightforward case, constraints and operating conditions can be converted and included as comments in generated source code as well as source code stubs to assist in ensuring performance is addressed by the developer. For example, a response time constraint associated with an activity may be transformed to a policy that controls how invocation of external partners may be achieved, forcing a synchronous only call to guarantee a certain response time.

There are also several additional areas where more intelligent conversion may be applied. A security policy constraint may be converted to a WS-SecurityPolicy (OASIS 2005) for use in runtime security policy enforcement. A further area involves using the artifacts to create QoS policy assertions, under the policy assertion guidelines (W3C 2007), which can then be used in runtime monitoring and performance control. For instance, during peak loads additional processes may be spawned to cater for increasing transaction conditions. These and other artefact conversions are suggested areas of further work.

As a further area of work, the control case and operating condition may be used as a vehicle to integrate non-functional requirements as context information (Rosemann, Recker, *et al.* 2006) within BPMN. For instance, the operating condition is a category of environmental context, whereas the control case can fall into either external context or internal context. These two artifacts add clarity to the context-awareness process model as they reveal the rationale and motivation of the risk mitigation in process modeling.

Finally, empirical studies which provide evidence of the efficacy of the extensions are suggested as further work. An appropriate form may be case studies of actual IT projects. In addition, surveys may be conducted with PBMN modelers to understand their perceptions towards these proposed techniques.

6.2 Acknowledgements

We thank Kylie Skeahan for her feedback and discussion on the modeling techniques presented in this paper. We also thank the anonymous reviewers for their feedback and suggested changes to the paper, in particular the related work and motivations.

7 References

Boehm, B. (1988): A Spiral Model of Software Development and Enhancement. In *Computer*, **21**(5):61 – 72, IEEE Computer Society Press.

Bresciani, P. and Giorgini, P. (2002): The TROPOS Analysis Process as Graph Transformation System. *Proc. of the OOPSLA Workshop on Agent-Oriented Methodologies (AOM-2002)*, Seattle, USA, 1-12.

Chung, L. and Nixon, B. (1995): Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach. *Proc. of IEEE 17th International Conference on Software Engineering*, Seattle, USA, 25-37.

Cysneiros, L.M. and Yu, E. (2004): Addressing Agent Autonomy in Business Process Management - with Case Studies on the Patient Discharge Process. *Proc. of the Information Resources Management Association Conference*, New Orleans, USA.

Demirors, O., Gencel, C., and Tarhan, A. (2003): Utilizing Business Process Models for Requirements Elicitation. *Proc. of 29th Euromicro Conference: New Waves in Systems Architecture*, 409-412, IEEE Computer Society.

Gorton, S. and Reiff-Marganiec, S. (2006): Towards a Task-Oriented, Policy-Driven Business Requirements Specification for Web Services. *Proc. of 4th International Conference on Business Process Management*, Vienna, Austria, 465-470.

Haimes, Y. (1998): Risk Modeling, Assessment, and Management, John Wiley & Sons Publishing.

Hepp, M. and Roman, D. (2007): An Ontology Framework for Semantic Business Process Management. *Proc. of International Conference on Wirtschaftsinformatik (Commercial Informatics)*, WI, Karlsruhe, Germany.

Kavakli, E. and Loucopoulos, P. (2005): Goal Modeling in Requirements Engineering: Analysis and Critique of Current Methods. In *Information Modeling Methods and Methodologies: Advanced Topics of Database Research*, 102-124, IGI Publishing.

Kotonya, G. and Sommerville, I. (1998): Requirements Engineering Processes and Techniques. UK, John Wiley & Sons Publishing.

Lu, R., Sadiq, S., Padmanabhan, V., and Governatori, G. (2006): Using a Temporal Constraint Network for Business Process Execution. *Proc. of 17th Australasian Database Conference*, Hobart, Australia, **49**:157–166, ACM Conference Series.

Muehlen, M. and Ho, D.T.Y. (2005): Risk Management in the BPM Lifecycle. In *Business Process Management Workshops*, International Workshops, Berlin, **3812**:454-466, Springer.

Mylopoulos, J., Chung, L., and Nixon, B. (1992): Representing and Using Non-Functional Requirements: A Process-Oriented Approach. In *Software Engineering*, **18**(6):483-497, University of Toronto.

OASIS (2005): Web Services Security Policy Language (WS-SecurityPolicy). V 1.1, OASIS. http://www.oasis-open.org/committees/download.php/16569/.

OMG (2006): Business Process Modeling Notation Specification. OMG Final Adopted Specification, Object Management Group (OMG).

Pesic, M. and Van Der Aalst, W.M.P. (2006): A Declarative Approach for Flexible Business Processes. In *Workshop on Dynamic Process Management (DPM 2006)*, LNCS, Vienna, Austria, **4103**:169-180, Springer-Verlag.

Recker, J., Indulska, M., Rosemann, M., and Green, P. (2006): How Good is BPMN Really? Insights from Theory and Practice. *Proc. of the 14th European Conference on Information Systems*. Goeteborg, Sweden, pp. 1582-1593.

Rosemann, M., Recker, J., Flender, C., and Ansell, P. (2006): Understanding Context-Awareness in Business Process Design. *Proc. of the 17th Australasian Conference on Information Systems*. Australia Association for Information, Adelaide, Australia.

Royce, W. (1970): Managing the Development of Large Software Systems. *Proc. of IEEE WESCON*, 1-9, IEEE.

Turner, K.J., Reiff-Marganiec, S., *et al*. (2006): Policy support for call control. In *Computer Standards and Interfaces*, **28**(6):635-649, Elsevier Science.

W3C (2007): Web Services Policy 1.5 - Guidelines for Policy Assertion Authors. W3C Working Draft 30 March 2007**,** http://www.w3.org/TR/ws-policy-guidelines/#WS-SecurityPolicy.

Yu, E. (1997): Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. *Proc. of the 3rd IEEE International Symposium on Requirements Engineering*, 226-235.

Zou, J. and Pavlovski, C.J. (2006): Modeling Architectural Non Functional Requirements: From Use Case to Control Case, *Proc. of IEEE International Conference on e-Business Engineering (ICEBE '06)*, Shanghai, China, 315-322, IEEE.