SoSyM Special Issue on Enterprise Modeling manuscript No.
(will be inserted by the editor)

A Case Study on Consistency Management of Business and IT Process Models in Banking

Moisés Castelo Branco[1](✉), **Yingfei Xiong**[2,3](✉), **Krzysztof Czarnecki**[1], **Jochen Küster**[4], **Hagen Völzer**[4]

[1] Generative Software Development Laboratory, University of Waterloo, Canada
[2] Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education, China
[3] Institute of Software, School of Electronics Engineering and Computer Science, Peking University, China
[4] IBM Research-Zurich, Switzerland

The date of receipt and acceptance will be inserted by the editor

Abstract Organizations that adopt process modeling often maintain several co-existing models of the same business process. These models target different abstraction levels and stakeholder perspectives. Maintaining consistency among these models has become a major challenge for such organizations. Although several academic works have discussed this challenge, little empirical investigation exists on how people perform process model consistency management in practice. This paper aims to address this lack by presenting an in-depth empirical study of a business-driven engineering process deployed at a large company in the banking sector. We analyzed more than 70 business process models developed by the company, including their change history, with over 1000 change requests. We also interviewed 9 business and IT practitioners and surveyed 23 such practitioners to understand concrete difficulties in consistency management, the rationales for the specification-to-implementation refinements found in the models, strategies that the practitioners use to detect and fix inconsistencies, and how tools could help with these tasks. Our contribution is a set of eight empirical findings, some of which confirm or contradict previous works on process model consistency management found in the literature. The findings provide empirical evidence of 1) how business process models are created and maintained, including a set of recurrent patterns used to refine business-level process specifications into IT-level models; 2) what types of inconsistencies occur; how they are introduced; and what problems they cause; and 3) what stakeholders expect from tools to support consistency management.

Key words Business processes – consistency management – process refinement patterns – empirical study

1 Introduction

Business Process Modeling (BPM) is increasingly used by enterprises to improve their agility and operational performance by better aligning their IT infrastructure with their business needs. Typically, a BPM-driven system development process involves the participation and collaboration of many stakeholders (e.g. Business Analysts, Systems Analysts, IT Architects and Developers). These roles and responsibilities may be organisationally defined, be the result of the adopted development process, or simply reflect the different competencies and capabilities of the people involved. The distribution of responsibilities and roles usually results in the creation of different models of the same business process. These models vary from business-oriented ones, which are technology-independent and easily understandable by business people, to IT-oriented ones, which are constructed by taking into consideration technicalities of existing systems. Specialized modeling languages have been developed to represent such models, including *Business Process Modeling Notation* (BPMN) [1] for business-level models and *Web Services Business Process Execution Language* (BPEL) [2] for IT-level executable models. Since its 2.0 version, BPMN can also express executable models [3].

The multitude and heterogeneity of models created to describe a business process at different levels of abstraction and from different stakeholder perspectives lead often to inconsistencies among the models. Inconsistencies arise because the models overlap—for example, they contain elements that refer to common aspects of systems and other enterprise resources, such as organizational structure and flow of communication, and make assertions about these aspects that may be contradictory under certain conditions. On the positive side, inconsistencies highlight different perceptions and goals of the

stakeholders involved in the development process and they can be intentionally introduced to indicate aspects of a process which deserve additional information elicitation and further development. On the negative side, inconsistencies can cause development delays, increased costs, and operational and audit failures.

To manage consistency of multiple business process models, researchers have proposed different approaches [4–9], each targeting a sub-problem of consistency management. In practice, companies also employ their own processes to manage the consistency among multiple models. It is not clear to what extent the academic approaches are adopted by the companies and what remaining challenges are still faced by practitioners. Conversely, many academic approaches are based on assumptions of how models are handled in practice, and some assumptions are even contradictory. For example, Zerguini [10] and Soffer [11] assume that models at different levels of abstraction are related in a strict top-down fashion via hierarchical refinements, whereas Weidlich et al. [12] propose that non-hierarchical refinements should also be considered. It is not clear which of these assumptions are true in practice. Thus, we need empirical evidence to support such claims about consistency management of business process models.

This paper presents an in-depth empirical study on consistency management of business process models in a large company in the banking sector. The study is designed to answer an initially broadly-scoped research question: *how do people manage consistency of related business- and IT-level process models in practice?*

The IT department of the company has more than 300 employees and is responsible for more than 200 information systems, including BPM-supported ones. More specifically, we apply three research methods. We first analyse more than 70 models in five BPM projects and their change history involving more than 1000 change requests, in order to understand the relation between the models and how they evolve over time. The analyzed models include business-level ones written in BPMN and IT-level ones written in BPEL. Second, we interview 9 professionals ranging from Business Analysts to IT Developers in order to understand how they collaborated to create and maintain the models. Finally, we conduct a survey with 23 professionals to further understand the relevant issues revealed by the artifacts and interviews. As a result, we elicit several research questions that were not asked up front, but emerged during the study as we analysed the artifacts, interviews and the survey (see Section 4).

Our main contribution is a set of eight empirical findings (**F**). These findings emerged from investigating and distilling eight research questions (**RQ**) derived during the study from our initial goal. The following list summarizes the research questions and the corresponding findings, divided in three categories.

A. **Modeling Methods**

RQ1: *What development process is used to create business- and IT-level process models?* We describe the software development process used at the studied organization, including the types of process models created and their purpose and stakeholders. We also characterize the models in terms of their sizes, language constructs used, and the type of evolution they undergo.

F1: Process models are created and maintained at several levels of abstraction (Section 5.1).

RQ2: *How are business- and IT-level process models related and how do they differ?* We identify a set recurrent patterns used by the developers to refine abstract, business-level models into more concrete, IT-level models. We found instances of these patterns in the studied models and developers confirmed them. The patterns reflect the relationships between the business- and IT-level models and provide evidence of both hierarchical and non-hierarchical refinements.

F2: Business and IT process models are related by both hierarchical and non-hierarchical refinement patterns (Section 5.2).

B. **Consistency Issues**

RQ3: *How do business- and IT-level process models evolve over time?* We provide evidence on how the models coevolve. We analyze cases where the business-level process models lag behind the IT-level models and vice versa, and provide the main reasons behind these inconsistencies.

F3: Process models undergo parallel maintenance (Section 6.1).

RQ4: *How do differences between business- and IT-level process models affect consistency?* We investigate different types of differences between business- and IT-level, such as as coverage, behavior, and level of detail, and determine which of these are most likely to cause inconsistencies.

F4: Coverage and behavioral differences affect consistency most (Section 6.2).

RQ5: *How do BPM stakeholders define consistency between business- and IT-level process models?* We summarize how *different* BPM stakeholders define consistency between business- and IT-level models.

F5: Stakeholders have a subjective notion of consistency (Section 6.3).

RQ6: *Can inconsistencies cause problems in practice?* We give two concrete examples of serious incidents caused

by inconsistencies between business- and IT-level process models.

F6: Inconsistencies can cause severe problems (Section 6.4).

C. **Tool Support**

RQ7: *Are the BPM stakeholders satisfied with the tool support for the development process they currently employ?* We discuss how stakeholders evaluate the current tool support for developing process models, and poll their opinion on different approaches to align business- and IT-level models.

F7: The majority of the surveyed stakeholders would prefer a single model for Business and IT (Section 7.1).

RQ8: *How are inconsistencies dealt with?* We provide evidence on how the stakeholders deal with inconsistencies in the studied company and what tool support they expect in helping them to better perform their tasks.

F8: Inconsistencies should be detected and communicated at the time they occur, along with proposed fixes (Section 7.2).

The remainder of the paper is structured as follows. Section 2 provides background on BPM and describes the running example, the models of an *Automated Teller Machine* (ATM), which we use throughout the paper. Section 3 discusses related work on model consistency management. Section 4 describes the empirical study design, presenting details about the organization, the analyzed projects and artifacts, and the conducted interviews and survey. Section 5 presents the findings regarding the process and modeling methods used by the studied company, including the refinement patterns catalog. Section 6 presents the findings related to the issues caused by inconsistencies. Section 7 presents the stakeholder expectations regarding the tools for consistency management. Section 8 analyzes the threats to the validity of the work. Finally, Section 9 summarizes the key results, lessons learned, and concludes the paper.

2 Business Process Modeling

A business process is a collection of related, structured or ad-hoc activities (tasks) that produce a specific output, such as service or product, for a particular customer or market [13]. Structured processes, which our study focuses on, are usually modeled as workflows, i.e., flows of activities. Typical examples of business processes are *Purchasing*, *Manufacturing*, *Marketing*, and *Sales*. A business process begins with a mission objective and normally ends with achievement of the objective. The activities of a process interact with IT assets to capture, transform, or report business data. As with processes, the data may be structured, such as a new order conforming to some well-defined schema, or ad hoc (unstructured) data, such as an e-mail [14].

In practice, a range of business to IT-oriented stakeholders create and use business process models for specific purposes, including requirements elicitation, documentation, simulation, and execution [15]. Each model must be appropriate for its target audience and purpose—having adequate level of detail, focusing on relevant aspects, and neglecting irrelevant ones [8]. This goal can be achieved by creating either several separate models, each focused on particular set of stakeholders and purposes, or a single model with multiple views [16].

Figure 1 shows three models, each representing the process of using an *Automated Teller Machine* (ATM) system at different level of abstraction. We will use these models, which are versions of real process models from one of the studied projects (project *P4*, Section 4.3), as our running example. The two specifications are the same as the original models, except that it has its labels translated from Portuguese to English. Also, the IT model is translated from BPEL to BPMN. For checking consistency, we focus on the control flow of the process models. BPMN and BPEL control flow constructs are similar in the sense that each can be mapped into the other, according to the OMG specification of BPMN 2.0 [3]. The control flow of the original models was entirely preserved in these examples. Note that the original models, as represented in their respective modeling tools, also have detailed information as attributes of nodes and flows, such as the communication protocols and the addresses of the services used.

The first model (Figure 1.a) represents a business-level process specification, which is created and maintained by Business Analysts. The second one (Figure 1.b) is a refinement of the first one, created and maintained by IT Systems Analysts. These stakeholders use such models to align the modeled process with the existing service infrastructure; specify how the process interacts with IT assets; and ensure that the process is sound and free of design flaws, such as incomplete data objects and deadlocks. The third model (Figure 1.c), created by IT Architects and Developers, refines the second model and represents the executable process implementation that goes into production. The executable process is implemented on top of an ISO8583 service infrastructure [17] and the codes that appear in the names of some tasks, such as 0200 and 9010, are types of messages of this protocol. Note that the final refinement (Figure 1.c) consists of multiple, modularized executable models. These models orchestrate the actual services provided by the IT service infrastructure.

The models in Figure 1 are expressed in BPMN. The notation represents activities by rounded rectangles, events by circles, gateways by diamonds (rhombi),

and sequence flows by arrows (see *Appendix* for legend of BPMN symbols used in the example). Each model has a start, usually modeled by an start event (e.g., *Customer insert Card into ATM*), and a flow of activities that is governed by decisions (e.g., *Card is Valid?*) and exceptions (e.g., *8s Timeout*). Each model also has an end point, which represents the achievement of the process: either a value delivered to a user or the termination of the process because of an error or a user decision (e.g., *Cancel Transaction*).

3 Related Work

We discuss related work in four groups. First, we discuss the general problem of Business-IT alignment in BPM. Second, we introduce the general area of consistency management and discuss related work addressing specific consistency management tasks. Third, we turn to work on consistency management of business process models. Finally, we review empirical work related to our study. Throughout the discussion, we point out cases where our findings confirm or contradict some of the assumptions found in the related works, as summarized in Table 1.

Business-IT Alignment in BPM

Bridging the gap between business and IT abstraction levels is a standard topic in enterprise systems engineering [27, 28]. Bieberstein et al. define business-IT alignment as "a dynamic state in which a business organization is able to use information technology (IT) effectively to achieve business objectives—typically improved financial performance or marketplace competitiveness." [15]

A key aspect of the business-IT alignment is establishing a correlation between business specifications and IT implementations. We discuss some of the existing approaches below. All approaches are based on assumptions on how specifications and implementations are created and how traceability between them is established and managed. Some approaches deal with issues of transforming business processes into executable models.

Buchwald et al. [20] present an approach which allows for a transfer of business requirements into executable processes. Their approach provides a three-level modeling method that automatically maintains an intermediate model called *Business-IT-Mapping Model* (BIMM) to describe how activities from the business process are transferred into activities of the system process. A BIMM manages correspondences between model activities by means of transformation operations such as *rename, insert, remove, merge,* and *split*. The technique tackles the problem of synchronizing parallel maintenance of different perspectives on the same business process (confirming **F3**). A limitation of the approach is that it only considers consistency in terms of coverage, i.e., whether or not corresponding business-relevant activities are correctly mapped between the models. In practice, consistency involves other aspects, such as control flow (see Section 6.2).

Tran et at. [29] present a modeling framework realized as a view-based reverse engineering tool-chain. The framework maps process descriptions onto appropriate high-level or low-level views. The framework can be extended with support for different modeling languages, including BPMN and BPEL. Although the approach supports representing process structures at different levels of abstraction, it does not support consistency management among these views when they are independently edited (see Section 6.1).

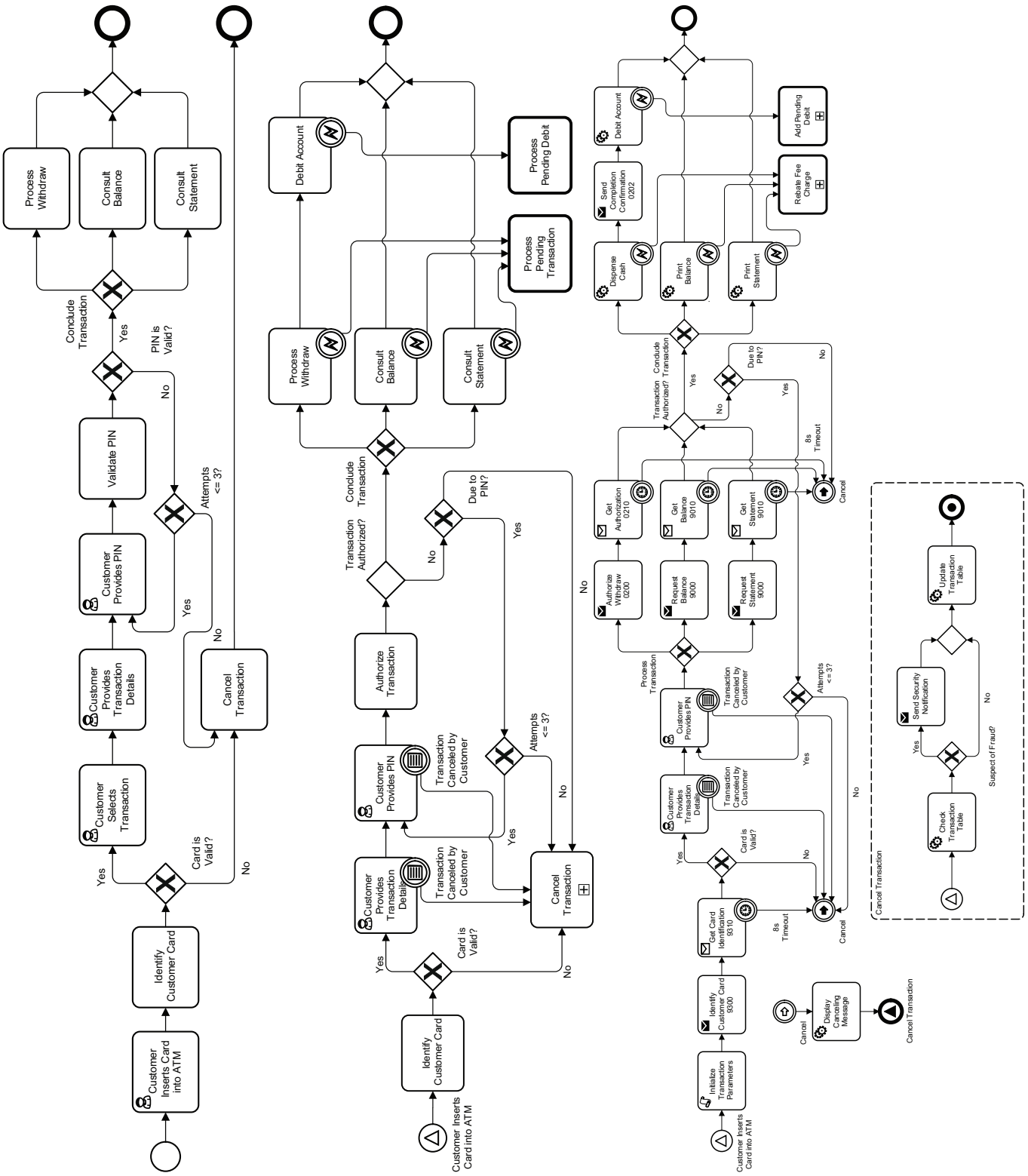
Delgado et al. [18] provide a methodology for incremental development of business processes, based on the joint application of Model Driven Development and Service Oriented Computing paradigms. Their proposed methodology recognizes the need of integrating business and IT people into the development life-cycle and conveying the right level of detail as output of each development stage. Our work *confirms* such a need (**F1**), by providing evidence on how business and IT people collaborate to create process models throughout the development process (see Section 5.1).

Decker [25] proposes patterns for introducing a process support layer that solves incompatibilities between business- and IT-level process models. The work assumes that a single process model for business and IT is inherently undesirable (contradicting **F7**) and that both perspectives are hierarchically related to each other.

Consistency management

Consistency management is a set of methods and tools for establishing and maintaining consistency among software artifacts, such as models, code, documentation, and test cases, which are usually created and used by multiple stakeholders [21, 30]. Existing works divide consistency management into a set of tasks [24, 30, 31]. The remainder of this subsection introduces these tasks and the corresponding related work in general; the next subsection discusses the related work specific to BPM.

- ***Defining consistency properties:*** Assuming a set of software models and a set of correspondence relations among their elements, consistency is a property of these models and their correspondences [32, 33]. Such a property is typically defined as a consistency



(a) Business Specification (b) Technical Specification (c) Executable Process

Fig. 1: ATM Process Models

rule, expressed in some logic and interpreted in a knowledge domain. Knowledge domains range from well-formedness of language constructs to industry- and organization-specific policies, such as legal regulations and organization-specific IT standards [32]. For example, a reasonable policy is to require that

Table 1: Summary of Findings

		Confirms	Contradicts	Section
Modeling Methods	**F1**—Processes are developed and maintained in several levels of abstraction	Delgado et al. [18], Koehler et al. [9]	—	Section 5.1
	F2—Business and IT process models are related by both hierarchical and non-hierarchical refinement patterns	Weidlich et al. [12]	Zerguini [10], Soffer [11], Dijkman et al. [19]	Section 5.2
Consistency Issues	**F3**—Process models undergo parallel maintenance	Buchwald et al. [20], Weidlich et al. [12]	—	Section 6.1
	F4—Coverage and behavioral differences affect consistency most	Weidlich et al. [12]	—	Section 6.2
	F5—Stakeholders have a subjective notion of consistency	Spanoudakis et al. [21]	Weidlich et al. [12, 22, 23]	Section 6.3
	F6—Inconsistencies can cause severe problems	Spanoudakis et al. [21], Nuseibeh et al. [24]	—	Section 6.4
Tool Support	**F7**—The majority of the surveyed stakeholders would prefer a single model for Business and IT	—	Decker [25]	Section 7.1
	F8—Inconsistencies and fixes should be presented as they occur	Hegedüs et al. [26]	Weidlich et al. [23]	Section 7.2

every business-relevant task in an executable model (e.g., *Identify Customer Card 9300* in Figure 1.c) is reflected in its business-level specification (*Identify Customer Card* in Figure 1.a); conversely, a purely technical task (*Initialize Transaction Parameters* in Figure 1.c) should not be reflected in the specification.

– ***Matching the models:*** This task deals with finding correspondence relations among elements of different models. For example, *Identify Customer Card* in Figure 1.a corresponds to *Identify Customer Card* in Figure 1.b, and to both *Identify Customer Card 9300* and *Get Card Identification 9310* in Figure 1.c. As we discuss in Section 6.1, process model matching is often challenging because identifying correspondences may require uncovering tacit knowledge, which may be only in the heads of the original creators of the models or may be lost entirely. Unless the correspondences have been recorded (e.g., via unique IDs), model alignment requires matching the models using domain- or organization-specific heuristics (e.g., by name and model structure). Examples of approaches that match different types of artifacts include document to code traceability recovery [34] and generic, graph-based matching [35]. A related area is schema integration, and in particular, schema matching, which deals with establishing correspondences among database schemas (see surveys on this topic [36, 37]). It is not clear how the existing techniques can be tailored to the problem of aligning process models. Our work presents evidence on how business and IT process models are related and how the maintenance process is done. This evidence will help developing appropriate alignment techniques.

– ***Checking consistency:*** Once the models are aligned, consistency is checked by evaluating the consistency rules. Spanoudakis and Zisman distinguish four types of approaches to consistency checking: logic-based approaches, model checking, specialized model analyses, and human-centered collaborative exploration [21]. The adopted consistency management policy is thus subjective (confirming **F5**) and specifies the circumstances that will trigger the checks.

– ***Diagnosing causes of inconsistencies:*** This task identifies the source, the cause, and the impact of an inconsistency [21]. The source of an inconsistency is the set of elements of software models that violate a consistency rule [24]. The cause of an inconsistency could be conflicting stakeholder goals or just a mistake in one or more of the conflicting models. The impact of an inconsistency are the consequences that the inconsistency has on the modeled system. Spanoudakis and Zisman include a survey of diagnosis approaches in their paper [21] and discuss negative consequences of undetected inconsistencies in a model-based system (confirming **F6**).

- ***Fixing inconsistencies:*** The final task is to fix inconsistencies. Ideally one or more fixes should be automatically proposed to the user. For example, Nentwich et al. [38] give approach that generates abstract fixes from first-order logic rules. An abstract fix specifies only the locations to be changed and the user needs to complete the edits. Egyed et al. [39] present an approach that generates concrete fixes for UML models, based on predefined inconsistency rules. Ameluxen et al. [40] propose an approach in which models are checked and corrected using graph transformation rules. Pinna et al. propose using an automated planning system, which does not require defining operations manually [41]. Similarly, Xiong et al. use constraint solving techniques to generate a list of fixes, and ensure them to satisfy a set of properties [42].

Consistency management of process models

We summarize work on consistency management in the context of BPM.

- ***Defining consistency properties:*** Weidlich et al. categorize differences among related process models that can cause inconsistencies into the following types [12]:
  - *Model coverage differences* are differences of what the related models describe in terms of functionality. For example, a particular task can exist in one model, but may be missing in the other.
  - *Behavioral differences* are differences in how a particular functionality is implemented in each of the models. For instance, the execution sequence of corresponding tasks might differ.
  - *Information density differences* are differences in the level of detail. For example, one model might have two or more tasks that decompose a single corresponding task from another model.

  We used and confirmed (**F4**) the above categories to investigate how they affect consistency (see Section 6.1). Behavioral consistency typically involves some notion of behavioral equivalence, such as trace equivalence or bisimulation. For example, Küster [43] provides a behavioral consistency notion for object-oriented behavioral models. In contrast, Weidlich et al. view the consistency of two process models as a degree of consistency rather than a strict binary criterion [12, 22]. An example of such notion are *behavioral profiles* [23]; they replace strict criteria such as trace equivalence with less strict degree of trace similarity. They build on properties of free-choice Petri nets and give a numeric degree of consistency ranging from 0 (inconsistent models) to 1 (consistent models).
- ***Matching the models:*** Effective matching techniques applied to business process models require heuristics that are notation and application specific [44–46]. Discovery of effective heuristics usually requires studying the differences among such models. In this context, for example, Dijkman [7] presents another classification of frequently occurring differences between similar business processes in general, such changing names and types of activities and modifying the flow structure. Zerguini [10], Soffer [11] and Dijkman [19] present solutions for matching hierarchically related process models. Our study provides an in-depth analysis of differences between process models targeting different levels of abstraction and shows that non-hierarchical correspondences need to be taken into account (**F2**) (see Section 5.2. Based on our findings, we have recently presented an algorithm to automatically detect correspondences between BMPN process models across levels of abstraction [47]. The algorithm has two phases and combines lexical and structural correspondences over Process Structure Trees (PSTs) [48] of the input models. The first phase matches the PST nodes using region and model element matching criteria adapted from previous work on matching ASTs [49]. The second phase establishes additional correspondences based on the position of the nodes in the PSTs.
- ***Checking consistency:*** Checking consistency of business process models may involve checking simple structural rules, such as that each business relevant task in the executable models is reflected in the business level specification, or analyzing behavioral properties using model checking or specialized algorithms (e.g., [23]). Two special representations of process models are used in model comparison: process structure trees [50] and process model terms [51]. The first representation represents the essential structure of processes as trees, allowing their easy matching and structural comparison. The second representation gives a canonical representation of process models and allows efficiently checking for a particular relaxed form of behavioural equivalence. Weidlich et al. [12, 22, 23] propose generic frameworks for checking consistency of process models, based on task ordering. Our findings reveal that consistency checking should actually take into account subjective project- and domain-specific differences among the models (**F5**) (see Section 6.3).
- ***Diagnosing causes of inconsistencies:*** The process model differences classified by Weidlich et al. [12] represent potential causes of inconsistencies. Establishing the actual root causes of the inconsistencies, such as the conflicting goals of stakeholders, usually requires additional knowledge that is not present in the models. We are not aware of any work investigating how diagnosis of inconsistencies among process models is done in practice.

– ***Fixing inconsistencies:*** Our findings show that stakeholders prefer immediate notification and editing quick-fixes (**F8**), integrated to the modeling tools, instead of an offline approach (see Section 7.2). Küster et al. present an approach to synchronize changes and fix inconsistencies on multiple process views via a *shared process model* [52]. The approach supports views on different abstraction levels, i.e., business- and IT-oriented ones. Hegedüs et al. recently proposed an approach to fix model inconsistencies based on state-space exploration and evaluated it on BPMN models [26]. Küster et al. also discuss the change management and inconsistency resolution in BPM [53, 54].

Empirical Research

We are not aware of any empirical research on consistency management in BPM, yet empirical studies exist in related areas.

Hutchinson et al. [55] address the relative absence of empirical studies of industrial model driven engineering (MDE) practices by describing lessons learned from three case studies. They applied a combination of research methods, such as interviews and questionnaire surveys for collecting data and deriving lessons learned from MDE practices adopted by three companies. Compared to their work which focuses on MDE in general, our work focuses on BPM and consistency management.

Zapf and Heinzl [56] present an empirical study of process refinement patterns in the call center domain. They compare different process partitioning strategies as typical design patterns in call centers. The analysis provides insight to the question under which circumstances a specific pattern is used. Our study provides empirical evidence of how process refinement patterns are applied in the domain of banking applications.

4 Study Design

4.1 Methodology

The study was designed to answer the following, broadly-scoped research question:

How do people manage consistency of related business- and IT-level process models in practice?

We initially left our problem statement open so that we could discover which facts about this subject really matter to the practice of BPM. We also decided to first focus on understanding the emergent *consistency management process* used at BNB, both in terms of the prescribed procedures and how the participants actually perform the tasks, in the context of the overall development process.

To answer this question, we adopted a structured combination of three research methods: 1) artifact study, 2) semi-structured interviews and 3) electronic survey. The combination allowed us to gradually refine our understanding of how consistency is managed and to triangulate multiple sources to improve confidence in our findings. We now briefly summarize each of the methods.

First, we analyzed business-level and IT-level models to understand the correspondences between them. We were interested in discovering the degree to which these models differ, the refinement patterns applied, and the type of information represented in each model.

Second, we interviewed relevant stakeholders at the studied organization to understand details about the development process, collaboration patterns among the professionals involved, reasons for applying the refinements we found, when and how the consistency among the models is maintained, and the challenges faced during consistency maintenance.

Third, based on the artifact analysis and the interviews, we created an electronic survey with questions to disambiguate unclear points and to solidify our initial findings. We collected responses to this survey from a larger set of stakeholder than those interviewed.

The following sections give more details about the studied organization and the applied methods.

4.2 The Organization

The Bank of Northeast of Brazil (BNB) is a major financial institution in Brazil. It is controlled by the federal government and oriented towards regional development. The Information Technology Area of the Bank contains over 300 professionals, responsible for maintaining more than 200 information systems in operation. Joining to these numbers are five outsourced software development companies, adding up to a virtual workforce of 1500 professionals responsible for development and maintenance of these systems. The systems are developed using a broad range of technologies, including conventional mainframe transactions and web-based services. Since 2007, BNB has used *Business Process Management* based on the WebSphere family of products from IBM, including Business Modeler, Integration Developer, Business Monitor, and Process Server. The development process is based on the Rational Unified Process (RUP), extended to include business process modeling. The first version of the development process was customized by BNB with consulting provided by IBM.

4.3 Artifact analysis

We analyzed five BPM projects, containing more than 70 models in total (see Table 2). The development process at BNB entails iterative and multi-staged model refinement, resulting in three types of models: business specifications, technical specifications, and executable implementations (cf. Figure 1). Table 2 lists the number of models of each type. It is important to mention that the project *P1* was the first one developed at BNB (pilot project), and its initial development was conducted with IBM consultancy. BNB took advantage of the pilot project to create 23 generic and reusable IT level processes (services), e.g., for logging and auditing. As they belong to *P1*, they count as implementation models in this project. That explains the large number of implementation models as part of this project (29). Table 3 gives the model sizes in number of elements of different types.

Table 2: BPM Projects

		Number of Models		
Project	Domain	Business	Technical	Implementation
P1	Customer Registration	2	2	29
P2	Credit Backoffice	6	6	6
P3	Credit Risk Assessment	2	2	4
P4	ATM	1	1	3
P5	Procurement	3	3	4

We analyzed the models by manually inspecting and identifying corresponding elements and model fragments (typically single-entry and single-exit regions [57]) based on names and structural similarity. The analysis relied on the domain knowledge of the first author; we clarified any unclear cases with the creators of the models. As a last step, we classified the correspondences into recurring refinement patterns presented in Section 5.2.

BNB manages the change of software artifacts using two IBM products—*ClearQuest* (workflow of change requests) and *ClearCase* (artifact repository). Business employees open change requests to the IT department using ClearQuest. Every request has a unique ID, a textual description and several parameters, such as priority and nature of the change (e.g., legal, evolution). Requests follow a sequence of steps, for example to group them into projects (when applicable) before they arrive to IT. IT Managers assign IT professionals (Project Managers, Architects, Developers) to every request. IT technicians only can change artifacts in ClearCase by having an assigned change request. When artifacts are changed, ClearCase stores the change request ID in the change log.

Table 3: Model Sizes

		Number of Model Elements				
		Pools	Tasks	Gateways	Events	Flows
P1	Business Spec.	11	59	38	25	149
	Technical Spec.	11	78	46	36	164
	Implementation	11	123	56	43	186
P2	Business Spec.	6	47	46	18	128
	Technical Spec.	6	95	48	23	142
	Implementation	6	107	52	31	154
P3	Business Spec.	4	17	8	6	19
	Technical Spec.	4	19	10	8	21
	Implementation	4	22	6	9	23
P4	Business Spec.	1	10	5	3	21
	Technical Spec.	1	11	6	8	27
	Implementation	1	18	9	14	51
P5	Business Spec.	8	13	10	11	31
	Technical Spec.	8	18	12	15	43
	Implementation	8	25	14	17	57

We recovered the change log of all projects and also the textual descriptions associated with every change request (from the ClearQuest database). Our objective was to find the reasons for changing the artifacts in each project we analyzed. Our first step was matching the textual description of each change request with the actual artifacts changed. The aim of this process was to discover how inconsistencies were introduced by regular maintenance. For example, by finding a particular change in August 2009 that had affected only the business model of the project *P1*, we realized from the description of the request that this change had *re-established* the consistency between the business specification and the production process (implementation). A new project was being started on the business side requiring an updated specification to build on. Then, we recorded any such cases to clarify with the people involved. In total, we manually inspected more than 1000 change requests, as shown in Table 4.

Ultimately, the artifact analysis gave us a rich evidence on how business and IT process models are related in BNB and what kinds of differences they display. This knowledge was used to further inquire the practitioners and understand the underlying reasons behind the differences.

4.4 Interviews

We used semi-structured in-depth interviews. The durations ranged from one to three hours, and the interviews were informal: although organized around a number of

Table 4: Change Requests

Project	Change Requests Analyzed
P1	388
P2	234
P3	176
P4	78
P5	207
	Total 1083

Table 5: Interviews

Interview	Role	Num. Projects	Duration (h)
1	IT Systems Analyst	2	1:45
2	IT Systems Analyst	2	1:32
3	IT Systems Analyst	3	1:40
4	IT Manager	6	1:10
5	IT Architect	4	3:01
6	IT Developer	2	2:34
7	Business Analyst	4	1:25
8	IT Architect	12	2:10
9	IT Architect	8	1:52
			Total 17:09

themes, we allowed each respondent to follow her own interest. The themes ranged from respondent's background, current role and experience, to practical working scenarios with BPM and personal feelings on how the tools should be improved.

The interviewees' roles were selected from those having personal responsibility in editing BPM models. An IT Manager was also interviewed because of his experience in several projects. These roles served as a representative sample of a larger population of professionals who later answered the survey. Statistics about the roles involved, their experiences with BPM, and the interview durations are shown in Table 5. Section 5.1 provides more details about the responsibilities of each role and the artifacts they produce.

We created transcripts of each interview and submitted them for approval of the respondents. Subsequently, we classified and categorized recurrent facts mentioned in the interviews, such as what consistency aspects are relevant; when and how inconsistencies are detected and fixed; and which tool support would help to perform these tasks. Sample questions asked are the following:

What is your current role? What types of tasks do you perform? How much experience do you have with BPM?

What are the roles involved in creating and maintaining business- and IT-level models?

What tools and architecture- and company-specific guidelines and methodologies impact the content and form of these models?

What collaborations exist between the different roles?

How do different roles coordinate and communicate when they make changes?

Are there examples where inconsistencies were detected?

Are there examples where inconsistencies had undesirable consequences?

4.5 Survey

We created a questionnaire to strengthen our initial analysis and also to disambiguate conflicting and overlapping facts from the interviews. For example, during the interviews some respondents mentioned that *task ordering* affects consistency, whereas others mentioned that it may not be important. Then we included the following question in the survey: *Corresponding tasks must obey exactly the same relative order*, and the respondents could chose between four answers: *Necessary all the times*; *Important, but not always*; *May be important sometimes*; and *Irrelevant*. We also added open fields, so that the respondents could provide comments and examples supporting their answers. The questionnaire was divided into six groups of questions: *Alignment of Business and IT Models, Tool Customization, Refinement, Change Management, Consistency Checking, and Fixing Actions*. In total, 23 professionals answered it as a web survey. Figure 2 shows the distribution of answers per professional role. Please refer to Section 5.1 for details about the roles and the work that they perform.

It is important to mention that BPM is a relatively recently adopted technology in BNB. At the time of this study, there were few BPM projects in production—they include the ones we used in this study—plus 5 new projects in early phases of development (i.e., the business models were under discussion). Around 30 professionals — including business and IT oriented ones — had been conducting these projects. Our survey collected 23 answers from this population, yielding a participation of 76%. The other professionals in the bank work with other technologies and programming languages, ranging from traditional mainframe to web-based platforms.

The complete report of the survey and the comments made by the respondents are available online at our web site.[1]

From the survey and the data we collected in the previous two phases, we found that our main research question can be divided in the following sub-questions:

[1] `http://gsd.uwaterloo.ca/empiricalstudybpm`

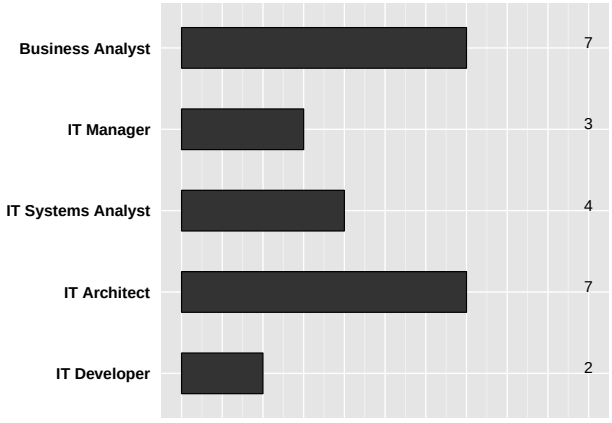


Fig. 2: Survey Answers per Professional Role

1. *What development process is used for creating business- and IT-level process models?*
2. *How business- and IT-level process models are related and how do they differ?*
3. *How do business- and IT-level process models evolve over time?*
4. *How do differences between business- and IT-level process models affect consistency?*
5. *Can inconsistencies cause problems in practice?*
6. *How do BPM stakeholders define consistency between business- and IT-level process models?*
7. *Are the BPM stakeholders satisfied with the development process they currently employ?*
8. *How are inconsistencies dealt with?*

For each of this sub-questions (see Table 1), we distill our findings in the next sections.

5 Empirical Findings on Modeling Methods

5.1 ***F1****—Processes are developed and maintained in several levels of abstraction*

Summary The state of the art recognizes the need for specialized models (or specialized views) in business process modeling (e.g., [9, 18]), such that specific needs of the stakeholders are respected in terms of concepts, modeling notation, and level of detail. Our work confirms such a need in the analyzed case study, by providing evidence on how process models are created, ranging from business- to IT-oriented ones.

The development process adopted by BNB starts by *Business Analysts* producing a *Business Specification* (Figure 1.a) which focuses on the concepts and rules relevant to the business level. The business specification is refined by *IT Systems Analysts* to create a *Technical Specification* (Figure 1.b). The technical specification has two objectives: a) to ensure that the process is sound and free of design flaws, such as incomplete data objects, deadlocks, and lack of synchronization; and b) to adapt the specification to the existing service infrastructure, making it clear and understandable to developers and outsourcers. The business and technical specifications are written in BPMN. The technical specification is subsequently refined by *IT Architects* and *IT Developers* to implement the executable process (Figure 1.c). Executable processes are written in BPEL. Naturally, several other artifacts are part of the development process, for instance, glossaries, requirement documents, use cases, architecture documents, business rules descriptions, and test cases. IT Managers are also involved in negotiating deadlines, assigning IT professionals, and contracting external services and manhours. Below are descriptions of the main roles involved in developing a BPM project in BNB:

- *Business Analyst:* Define and simulate the business process in terms of organizational structure (lanes, pools), business items (information to flow), resources (e.g., people who interact with the process), tasks (human and automated), business rules and *Key Performance Indicators* (time, costs, etc.). The business process is created in BPMN. Business rules and use cases to each business task are specified in natural language in associated specification documents.
- *IT Manager:* Produce contracts for meeting the business requests. Assign IT personnel to projects and contract outsourcers.
- *IT Systems Analyst:* Provide technical support for Business Analysts; correct and adjust the BPMN model; clarify business items and rules; detail tasks and flows; specify *Use Cases* for each task, gateway, conditional flow, and event.
- *IT Architect:* Create a BPEL model out of the BPMN. Refine the BPEL. Describe service interfaces, integration methods (queue manager, message broker, service bus), design human tasks, produce an *Architecture Document*, *Technical Use Cases*, *Design Models*, and *Deployment Plan*.
- *IT Developer:* Produce code (BPEL, Java, other languages). Create testable builds.

The consequence of this development process is that three different process models for each project are created and maintained. This is considered suitable by BNB to effectively separate concerns and to convey the right information to diverse stakeholders. The common use cases for consistency management throughout the development process are the following:

- *Change propagation*: By applying a development process based on *RUP*, requirements are created or up-

dated by carrying out the business modeling discipline. Business-level process specifications are updated and the changes should be propagated across related IT-level models. Similarly, due to incident resolution or time constraints it is possible that a process running into production is modified before updating its specification. Later the specification needs to be updated.
– *Validation*: Audits often require checking production processes against high-level specifications and control points of legal reference models, such as Basel II and Sarbanes-Oxley.

It is important to note that the actual workflow is defined by the business-level model *together* with business rules. In particular, detailed cases, such as when withdrawals are authorized, are specified in the business rules document, and might not be visible in the workflow of the process model. These points are more effectively captured as rules, and adding them to the diagram would result in visual clutter. The business-level model is intended to give an overall, high-level flow, and the stakeholders know that they also need to review the business rules document in addition to the process models, in order to cover all the business-relevant details.

The following quote provides a summary of the development process obtained in an interview from an *IT Systems Analyst*:

"The development is done in several iterations for accomplishing the project milestones. This is managed by the project manager following the same methodology used for any other software project. The objective of the inception phase is to clarify what should be done, then all the requirements should be clear at the end of this phase. Most of the collaboration is performed by business analysts and system analysts, although the architect is also involved in some meetings to anticipate possible integration issues, such as data replications and unavailable services or application components. The artifacts discussed in the inception phase are mainly BPMN models, use cases for tasks, and business rules. In the elaboration phase the objective is to eliminate all the architectural risks and know how the project should be implemented. The main artifacts are the integration model (BPEL), the architecture document and the technical use cases. Most of the collaboration is done by the architect and the developers. Systems analysts still collaborate with architects and developers in the elaboration and construction phases when a business rule or a use case is not well understood."

In the rest we will focus only on the business specification and the IT implementation and refer them as business (level) model and IT (level) model, respectively.

5.2 ***F2****—Business and IT process models are related by both hierarchical and non-hierarchical refinement patterns*

Summary Existing works argue the need for *non-hierarchical* refinements when deriving IT process models from their business specifications (e.g., [12]), while other works propose the transition from business to IT process models in a strictly hierarchical fashion (e.g., [10,11,19]). Our study provides evidence of the need for both types of refinements (hierarchical and non-hierarchical).

Using the ATM case study (*P4*)(see Section 2) as running-example, we now present the refinement patterns we identified. We chose *P4* as illustration because it is the smallest one and it also contains concrete instances of all the patterns we found in the other projects. Although the executable model is implemented in BPEL, for simplicity, we remodeled it in BPMN 2.0 [3], preserving the salient refinements patterns applied in the real project. Naturally, mismatches that stem from using different languages pose further complications; however, the problem of managing consistency of related process models is generic and independent of any specific language [12].

Table 6 shows statistics of the pattern occurrences across the models.

Table 6: Refinement Occurrences

	Occurrences				
Refinement Pattern	P1	P2	P3	P4	P5
Add properties	27	32	12	8	6
Add script task	21	13	4	1	4
Add protocol task	31	16	2	5	4
Add boundary event	34	9	9	6	6
Add technical exception flow	15	14	3	2	4
Change activity name	14	5	2	3	2
Change activity type	12	3	11	4	2
Refactor gateway	6	8	-	1	3
Split task into block	28	24	4	1	2
Split workflow	25	3	4	5	4
Suppress specification activity	11	7	5	1	6

Add properties

Description Parameters for grounding the executable model on top of the underlying IT infrastructure are added during the implementation.

Motivation Several properties of tasks, gateways, flows, events, etc., are added to the implementation-level model, such as application or service URLs, protocol types

(e.g., http or https), transactional behavior (e.g., commit before, commit after, participates, etc.). Such properties do not change the workflow and may be tool or platform-specific.

Example Each ISO8583 sending or receiving task shown in Figure 1 (e.g., *Identify Customer Card 9300* and *Get Card Identification 9310*) has parameters that include the message queue, authentication method, security protocol, and message encoding.

Add script task

Description Script tasks are used to initialize variables and implement business rules and non-functional requirements that access or transform business objects data, e.g., logging steps of the workflow.

Motivation This type of task is frequently used because it has significantly better performance than calling external services.

Example Figure 3 shows a task created in the ATM application for initializing several parameters of a *transaction* object, which controls user actions across the workflow. Such kind of task in the IT model does not have any correspondence in the business model.

(a) Executable

Fig. 3: Add Script Task

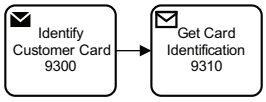
Add protocol task

Description An asynchronous service can be implemented by a connection-less request or reply protocol.

Motivation It is common to implement a business task by using an asynchronous connection-less service. In such cases, the protocol needs to compose and send a message and, after that, wait for a response.

Example Figure 4 shows an example where the business task *Identify Customer Card* is implemented on top of the ISO8583 protocol by sending a identification request message (9300) and waiting for a validation message (9310).

(a) Business and Technical Specifications



(b) Executable

Fig. 4: Add Protocol Task

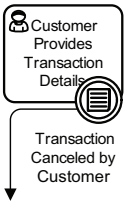
Add boundary event

Description Boundary events are used to divert the normal flow under special conditions, for example, because of a particular action performed by the operator on a human task.

Motivation The reason to divert the flow can be merely technical or too low-level to be represented in the business model. Such conditions can be implemented as result of requirements and use cases that describe a human task in detail.

Example Figure 5 depicts an example of boundary event added to human tasks to capture the customer's decision to cancel the transaction at any time. Another example can be seen in Figure 1, where boundary events were added to asynchronous receiving tasks (e.g. *Get Statement 9010*) to cancel the transaction in the case of a timeout of 8s.

(a) Business Specification



(b) Technical and Executable

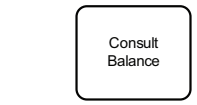
Fig. 5: Add Boundary Event

Add technical exception flow

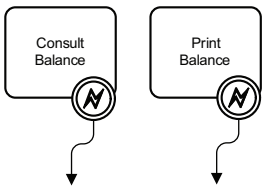
Description Technical exception flows are included to divert the flow in case of technical exceptions, such as an unavailable service or a permission denied.

Motivation Technical exceptions are not expected to be represented in the business model, because they implement non-functional requirements elicited during the *elaboration* phase of the development process.

Example Figure 6 shows examples of technical exceptions flows added for dealing with service errors, in which the transaction parameters are saved and the system administrator is notified to complete the transaction later.



(a) Business Specification



(b) Technical and Executable

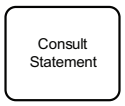
Fig. 6: Add Technical Exception Flow

Change activity name

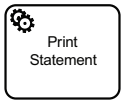
Description The name of a business activity can be changed to facilitate the identification of an IT service that has a similar but different name.

Motivation IT specialists can decide to use technical names in model elements for facilitating maintenance.

Example Figure 7 shows an example.



(a) Business and Technical Specifications



(b) Executable

Fig. 7: Change Activity Name

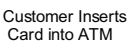
Change activity type

Description The type of a model element can be changed because of an implementation decision.

Motivation It is easier for business people to stick with basic modeling constructs (such as plain tasks and gateways), while other types of model elements are more suitable to implement the business intent.

Example Figure 8 shows an example were a human task represented in the business model was implemented by an event.

(a) Business Specification



(b) Technical and Executable

Fig. 8: Change Activity Type

Suppress specification activity

Description Business elements can be suppressed during the implementation.

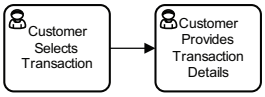
Motivation Some elements of the business specification may be considered redundant, not subject to automation, or subsumed by a particular task at the implementation level. Typical examples for applying this refinement pattern are:

- Combine several business tasks into a single service call (the service provided is coarser than the business steps described),
- Combine human tasks into a single human task, with the separate steps of the human task being described elsewhere as a screenflow, for example.
- Ignore manual business tasks, for example, "Send contract to the post office."

Example Figure 9 shows a case where the two human tasks described in the business model were collapsed into a single human task in the technical and implementation levels.

Split task into block

Description A single business task can be implemented by a combination of services.



(a) Business Specification

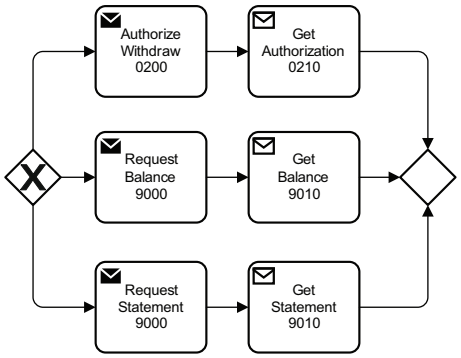
(b) Technical and Executable

Fig. 9: Suppress Specification Activity

Motivation To implement a specification task, it may be necessary to combine several existing services, including additional control flow logic to organize the way the services should be called to achieve the specified functionality.

Example Figure 10 illustrates such scenario, where a technical specification task, *Authorize Transaction*, is split into a block of ISO8583 service calls, organized as an exclusive gateway that controls the type of authorization required for each transaction type.

(a) Technical Specification



(b) Executable

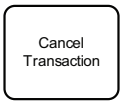
Fig. 10: Split Task into Block

Split workflow

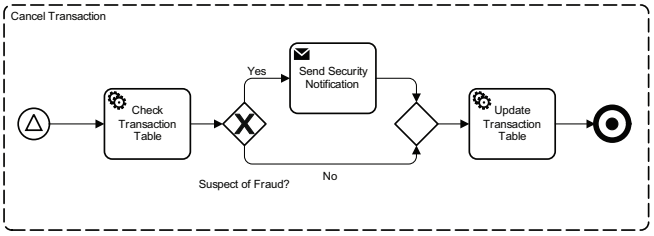
Description The specification workflow can be split into smaller workflows that should be orchestrated by a main flow.

Motivation The typical reason for this pattern is the creation of specialized and reusable workflows, such as for logging and auditing purposes.

Example In Figure 11, the task *Cancel Transaction* was implemented by a specialized subflow that includes fraud control and is reused by other projects. It is common to use web service interfaces or event triggering for calling the subflows.



(a) Business and Technical Specifications



(b) Executable

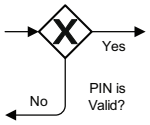
Fig. 11: Split Workflow

Refactor gateway

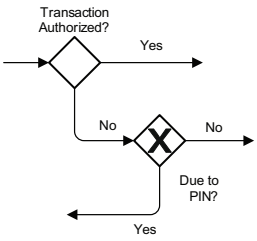
Description A business level gateway may need to be refined to take into account the technical behavior of the services involved.

Motivation IT services may impose constraints on the control flow. For example, the business model may specify tasks executing in parallel; however, in the implementation the corresponding IT services are called in sequence to avoid deadlocks.

Example Figure 12 shows an example where the business specification has a rule for checking the maximum number of times that a customer can enter a wrong PIN. In the actual implementation, checking the validity of the PIN is a particular result of the transaction authorization. In this particular project, some of the other cases where the transaction is not authorized are also relevant to the business (e.g., insufficient funding). However, since the business analysts did not know how the systems were implemented, they specified such cases as part of business rules of three business tasks: *Process Withdraw*, *Consult Balance* and *Consult Statement*. Business rules documents are produced together with business process models (see Section 5.1). The business analysts did not consider necessary to change the business model to approximate it to the actual system, at which point the workflows became different.



(a) Business Specification



(b) Technical and Executable

Fig. 12: Refactor Gateway

Hierarchical and non-hierarchical refinements

A refinement pattern is hierarchical if it is possible to fit the refined model elements into a collapsed subprocess that preserves the original number of incoming and outgoing sequence flows, otherwise it is non-hierarchical. The pattern *Split task into block* (see Figure 10) is an example of hierarchical refinement whereas *Refactor gateway* (see Figure 12) is an example of non-hierarchical one. Other examples of non-hierarchical refinements can be seen in Figure 1, where flows were added to divert the main workflow in case of *timeouts*. It is important to note that this specific example of refactoring a gateway is non-hierarchical, but not all instances of the pattern are such—for example, refining a parallel flow into a sequence would be hierarchical.

Interestingly, several approaches for aligning business and IT perspectives are based on the assumption that hierarchical refinements are sufficient in practice [10, 11, 19]. Our case study clearly shows that non-hierarchical refinements occur and are relevant in practice. One could argue that non-hierarchical refinements are present in the case study since the tools used there have not enforced hierarchical refinements, as in, for example, the approach by Dijkman et al. [19]. However, the majority of surveyed stakeholders express the need to support both hierarchical and non-hierarchical refinements in practice, as shown in Table 7.

We believe though that settling the question of whether hierarchical transformation can be expressive enough to create models for different perspectives requires further research.

Table 7: Refinement Patterns Needed by Stakeholder

	Strictly hierarchical	Any type of refinement	Role does not need to apply refinements
Business Analyst	13%	87%	0%
IT Systems Analyst	13%	87%	0%
IT Architect	9%	91%	0%
IT Developer	9%	78%	13%

6 Empirical Findings on Consistency Issues

6.1 ***F3****—Process models undergo parallel maintenance*

Summary Existing works recognize that process modeling needs support for parallel maintenance of models or views that target different levels of abstraction and stakeholder perspectives (e.g., [20]). Our study provides evidence on such a need, by analyzing the change history of five real-world projects. The histories cover both the development prior production and also changes after the projects went into production. Their analysis shows that, while the majority of changes affect only artifacts other than the related business- and IT-level models (i.e., use case descriptions, databases, services, and application components), about 10% of changes affect both models simultaneously, about 20% affect only IT-level models, and about 5% affect only business-level models. The number of changes varies according to the projects' life-cycles, with the majority of business-model-only changes occurring at project start. The analysis also revealed (i) cases where the business models were changed in response to earlier IT model changes, in order to eliminate inconsistencies considered as undesired by the stakeholders; and (ii) cases where changes first specified in the business models where later implemented in the IT models; these cases were viewed as acceptable "controlled inconsistencies" by the stakeholders.

As announced in Section 4, we report on a substantial dataset featuring in total 5 BPM projects, 74 models (business and IT ones) and more than 1000 changes made on these models throughout their life-cycles. A project at BNB contains the versions and baselines of all the artifacts of a system, including models, textual documents—such as use cases and business rules—and source code. We inspected the change history of each project to identify when inconsistencies were introduced by day-to-day maintenance and when they were found and fixed.

We classified each change request with respect to the model types being affected. For example, *Only Business* means that a request has changed solely the business model but not the IT model, whereas *None* means that the request has changed neither the business model nor the IT model (but other resources, such as databases, services, and application components). The changes to

the artifacts ranged from adding or modifying a single model element (e.g., a task or flow) to applying multiple patterns in multiple places.

Figure 13 shows the distribution of the changes per type in each project and Table 8 shows the corresponding percentage numbers.

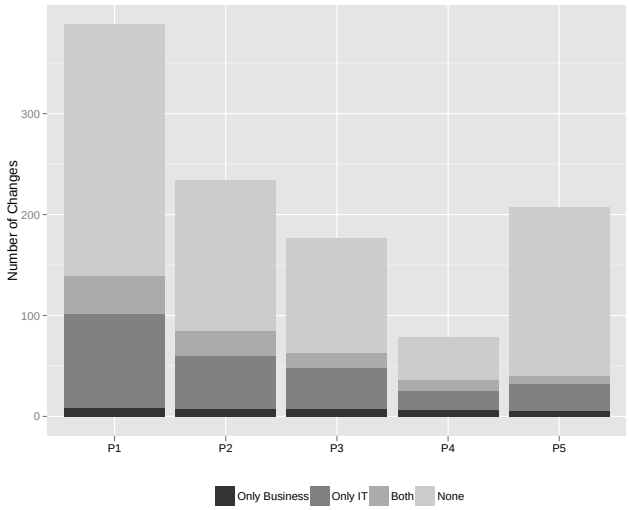


Fig. 13: Distribution of Changes per Type

Table 8: Percentage of Changes per Project

	Change Type			
Project	Only Business	Only IT	Both	None
P1	2%	24%	10%	64%
P2	3%	22%	11%	64%
P3	4%	23%	9%	64%
P4	9%	24%	13%	54%
P5	3%	13%	3%	81%

Figure 14 shows the temporal distribution of 388 changes made on project *P1* throughout its three first years. Figure 15 shows the first-year *stacked* distribution of changes for the other projects.

The analysis of the change history revealed that inconsistencies were introduced mainly due to parallel maintenance. The following two cases summarize situations where the inconsistencies were detected and fixed:

- *Case 1*: Update only the business model because at least one previous maintenance request that should have affected both the business and the IT model has been made only in the IT model. This is considered *undesirable inconsistency* by the stakeholders, and the business model is being updated to address it. Audits often motivate this type of maintenance. Another reason is when a new project requires an accurate business-level model for AS-IS analysis.
- *Case 2*: Update only the IT model to reflect, e.g., a planned process optimization that has previously been made only in the business model. This is considered a *controlled inconsistency* by the stakeholders.

For project *P1* we identified changes made only in the business model because of *Case 1* in January 2009 and August 2009. Also, we identified requests because of *Case 1* in projects *P2* and *P5* in March 2010 and July 2010, respectively. In project *P3*, we identified a process optimization made initially in the business model because of *Case 2* in May 2009.

Some stakeholders complain that the current tool support to plan and manage future changes (*controlled inconsistencies*) is deficient, although they mitigate the issue by using resources of the artifact repository. The main complaint is the lack of tool features for easily comparing, differencing, and merging process models, as they are widely available for textual source code.

The stakeholders also complain about the tool support for managing traceability and correspondence links among this multiplicity of models. This is particularly critical when specifications are updated and given to outsourcers. From time to time, the correspondences need to be reestablished and described using textual artifacts and model annotations, which is time-consuming when maintained manually. Another important aspect of correspondence links among process models is that they are domain- and project-specific. For example, we found correspondences that can be understood only by having knowledge of existing systems used in BNB. Then, automatic techniques for deriving correspondences should have means to include specific domain knowledge as part of the matching methods.

Our analysis confirms the challenges of establishing correspondences among process models identified in [12]. Since establishing correspondences may require uncovering tacit knowledge—present only in the original model creators' heads or lost entirely—, a fully automatic approach with high recall does not seem realistic. More research is necessary to understand the trade-offs between automatic and manual efforts to establish and manage correspondence links, integrated into the development process.

The following quote was made by an *IT Systems Analyst*:

"The main problem with our BPM development is maintaining traceability among such models and artifacts over time - this often requires considerable rework and

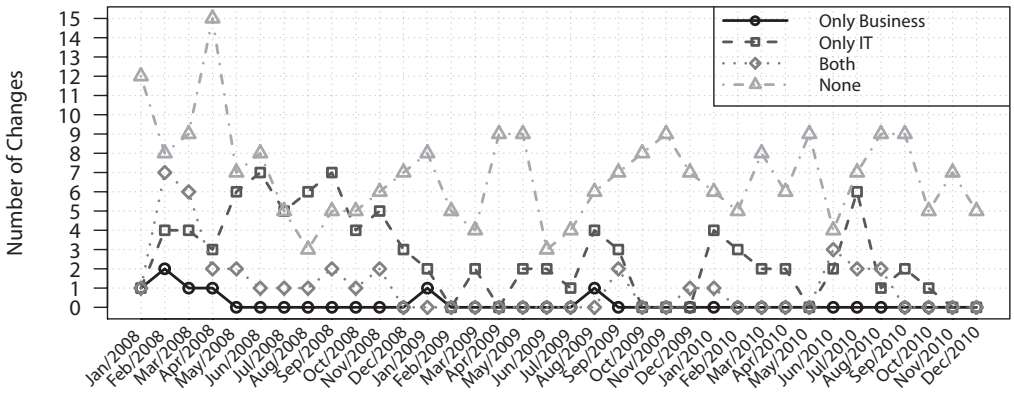
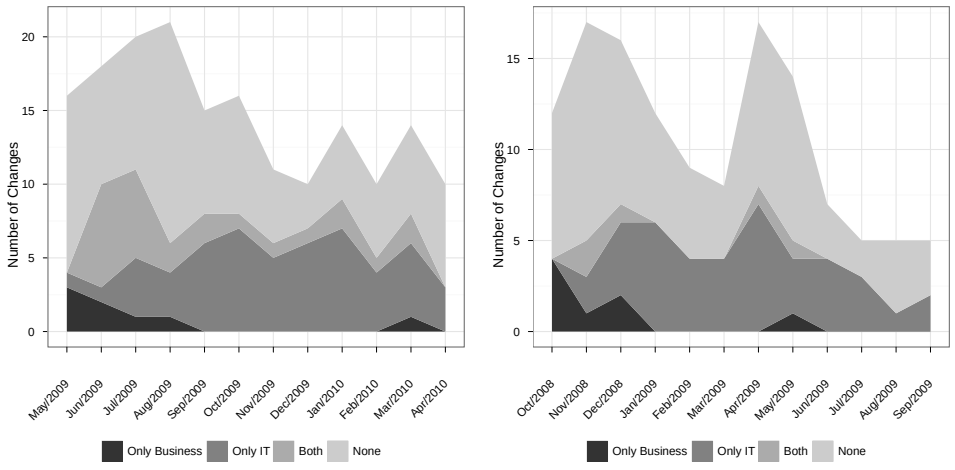
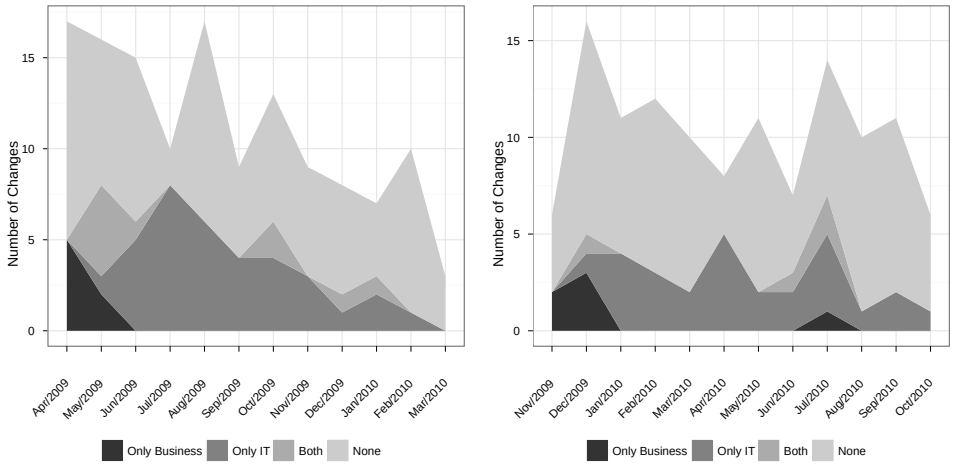


Fig. 14: P1 Change History



(a) P2

(b) P3



(c) P4

(d) P5

Fig. 15: First Year Change History

is specially critical when outsourcers are involved. I say that we could have much better tool support for managing this."

*6.2 **F4**—Coverage and behavioral differences affect consistency most*

Summary The state-of-the art discusses types of differences among process models that may affect the consistency among them [12]. Our work provides results of a survey where real practitioners were asked to evaluate to what extent such types of differences impact their notion of consistency.

From the artifact analysis, we prepared examples of the three types of model differences defined in [12] (briefly explained in Section 3) and asked the respondents to answer two questions:

Please indicate to what extent the following types of differences affect the notion of consistency between Business and IT models; and

Please indicate to what extent the following types of differences may be tolerated or ignored when checking consistency between Business and IT models.

The answers to these two questions are shown in Table 9 and Table 10, respectively. We also asked the respondents to rank a set of consistency aspects that were frequently mentioned in the interviews. The results are shown in Table 11. 86% of the respondents support that a difference in coverage always affects consistency, 68% support that a difference in behaviour sometimes affects consistency, and 68% support that a difference in density does not affect consistency. 74% support that corresponding tasks between the models must obey the same relative order. We also collected open answers from the respondents explaining their understanding of these types of differences.

Table 9: How Differences Affect Consistency

Type of Mismatch	Always Affects	Sometimes Affects	Does not Affect
Coverage (There is a difference between WHAT is modeled.)	86%	14%	0%
Behavior (There is a difference in HOW a certain scenario is implemented.)	14%	68%	18%
Density (There is a difference in the LEVEL OF DETAIL a certain scenario is implemented.)	0%	32%	68%

Table 10: How Differences are Tolerated

Type of Mismatch	Never Tolerated	Sometimes Tolerated	Always Tolerated
Coverage (There is a difference between WHAT is modeled.)	50%	50%	0%
Behaviour (There is a difference in HOW a certain scenario is implemented.)	14%	77%	9%
Density (There is a difference in the LEVEL OF DETAIL a certain scenario is implemented.)	0%	59%	41%

Table 11: Consistency Aspects Mentioned in the Interviews

Consistency Aspect	Necessary all the times	Important but not always	May be important sometimes	Irrelevant
Corresponding model elements have the same names	30%	70%	0%	0%
Corresponding tasks must obey the same relative order	74%	26%	0%	0%
Corresponding tasks have the same types (service, human etc.)	22%	61%	13%	4%
Corresponding gateways have the same number of incoming and outgoing flows	9%	52%	30%	9%
Corresponding business objects must have exactly the same fields	13%	52%	30%	4%
Every task in the business model has at least one corresponding task in the IT model	9%	70%	22%	0%
Every gateway in the business model has at least one corresponding gateway in the IT model	13%	70%	17%	0%
Every event in the business model has at least one corresponding event in the IT model	9%	70%	22%	0%

Our analysis leads us to propose the notion of *Business Relevance*, which seems to be crucial whenever stakeholders check consistency. If a mismatch is considered relevant to the business it should be fixed, otherwise it is ignored. Although this definition is subjective, we noticed that typically differences that are considered technical details of implementation are ignored. For example, Figure 3 shows a case of *Coverage* mismatch that is not business relevant: the added script task is essentially a detail of implementation and does not have any correspondence in the business-level model. Similarly, Figure 6 and Figure 10 show respectively examples of *Behaviour* and *Density* differences that are also not con-

sidered business relevant: both are details of implementation.

The results confirm the postulates presented in [12] on how differences affect consistency. We extend the postulates by adding the concept of business relevance, which can be summarized as follows:

- Difference in *Coverage* is what most affects consistency, as long as it is business relevant.
- Difference in *Behavior* is relevant when it affects task ordering.
- Difference in *Density* does not seem to affect consistency. It is generally considered an implementation detail and thus not relevant to business.

We believe that more investigation is necessary to understand how one can allow the stakeholders to define and manage these types of differences individually in each project. The same type of mismatch can be considered to affect or not to affect consistency, depending on its *business relevance*. This is why the answers to these previous two questions were subjective, consistently with results from the interviews, as we discuss in Section 6.3.

6.3 ***F5****—Stakeholders have a subjective notion of consistency*

Summary Although the surveyed practitioners are able to assess to what extent certain types of differences among process models affect a notion of consistency, they do not have a definitive agreement on its meaning. Existing works confirm the subjective nature of consistency in software modeling (e.g., [21]), while other works propose the use of consistency measures in terms of trace similarity (e.g., [23]). Our work concludes that more research is necessary to understand how the stakeholders could manage their own concepts of consistency, which may vary depending on the role and project.

We asked specific questions in the interviews aiming to understand how the BPM practitioners would define consistency among process models; how they realign inconsistent process models; and how they decide when the models are consistent *enough*. The following quotes are representative of what we obtained regarding this point:

IT Systems Analyst: "This task is somehow subjective and the criteria of consistency may vary from person to person, but I would say that it is not hard or complex to be performed manually. At the end of the day we always achieve a good understanding of possible adjustments that should be made in the models in order to regain their consistency. Anyway, I would say that we lack better tool support for doing this task: today we basically print the models (or some parts), stick them on the wall and visually inspect them together. It would be better having some online support for checking consistency, for example, whenever a potential inconsistency is detected, the tool could highlight that and the modeler could take an immediate action."

IT Architect: "It is not technically complex to reestablish consistency of business and IT models, although it is often laborious and time-consuming: we need to do it by visual inspection. It is not really complex because it is not a very strict thing; for example, we do not really need to compute all possible traces - this level of consistency is often too much. In general, some discrepancies in terms of traces may be considered important to be adjusted and others can be just ignored."

We interpret the practical evidence as confirmation that differences between the models and potential inconsistencies are the inevitable result of the need to describe complex systems from different perspectives, to distribute responsibilities to different stakeholders in the software development life cycle, and to allow them to work independently without requiring a continuous reconciliation of their models and views for, at least, certain periods of time. This is why stakeholders do not have a definitive notion of consistency and think that at the end of the day they are always capable of regaining an 'agreement' with respect to a consistency level, which is not really 'strict'. The benefit of working collaboratively with process models indicates that inconsistencies need to be "managed", that is detected, analysed, recorded and possibly resolved [21].

Interestingly, some works accept that it is hard for people to agree on the meaning of consistency between business and IT models [12,22]. A more recent work from Weidlich et al. [23] considers process model consistency as a degree measure in terms of trace similarity, rather than a binary (true or false) criterion. Although this perspective is more consistent with our finding, we did not see a clear trend towards defining consistency in terms of trace similarity in the case study. In particular, as discussed in Section 7.2, the majority of business and IT stakeholders prefer to see concrete model differences affecting consistency rather than metric values describing the degree of consistency.

As we started discussing in Section 6.2, we believe that more research is necessary to understand how the stakeholders could manage their own concepts of consistency. A potential way to investigate is to keep track of the correspondences among business-relevant model elements or fragments. Whenever a change affect such a business-relevant correspondence, the stakeholders should be notified to decide whether to fix or not potential issues. Taking into account other artifacts, like separately captured business rules, is another challenge.

*6.4 **F6**—Inconsistencies can cause severe problems*

Summary It is natural to expect that *undesired* inconsistencies in software artifacts may cause problems, for example, by delaying a new version of a system (e.g., [21, 24]). We provide examples of unnoticed inconsistencies that: (i) delayed business and impacted customers (production process instances needed to be canceled and recreated) and (ii) affected compliance regulations, subjecting the company to fines. Such problems can cause serious financial and corporate image losses. Our motivation to present such cases is reinforcing the need for better support in process modeling for dealing with multiple abstraction levels.

We identified two particular cases in which inconsistencies caused troubles. The first case was caused by an incomplete technical-level process specification (a problem of business-relevant *coverage* mismatch, Section 6.2). An (inconsistent) technical-level process specification, the corresponding IT model and several other artifacts (use cases, architecture document etc.) were sent to an outsourcer as part of a maintenance project. When updating the IT model, a developer inadvertently removed the functionality shown in Figure 16. The developer was new to the team and thought that this functionality should be deleted from the IT model: the developer did not see any *reasonable* correspondence in the specification, and also no reference to it in the architecture document. As a result, the problem passed unnoticed during the tests and the phase of user approval, and was discovered very late when the project went into production.

This was considered a severe problem, because some running instances of the process had to be canceled and recreated, delaying business. In outsourcing, the communication throughout a project usually observes a rigid schedule and the external developers cannot directly talk to business or systems analysts: double-checking the understanding of a specification is not as simple as in internal development. Although the test cases were improved after the incident, similar incidents can still happen if the business and IT-level models are incomplete, as there are no specified tests to capture every possible issue.

The second case was similar, but this time it was discovered by a regular audit procedure, where projects and their artifacts were inspected for consistency. Unclear points were marked to be explained. It turned out that the specification was outdated, and a notification was issued to correct the problem. This is also a severe problem, because business specifications are used for satisfying regulation purposes. Whenever a compliance issue is reported by an audit procedure the company is subject to fines and the managers are subject to legal responsibility.

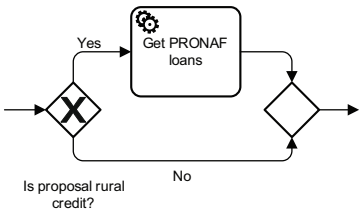


Fig. 16: Functionality Inadvertently Removed*

* *PRONAF* is a business acronym in BNB that means a credit line for *family agriculture*.

One of the *Business Analysts* made the following comment about such incidents:

"It is somehow frustrating that BPM has not solved our problem of reliably communicating with outsourcers by using process models as specifications. In practice the technology is preferable for internal development, where the communication between business and IT is straightforward. There is always a risk of something is missing in one or another model, or some correspondence not being completely understood. We have to maintain heavy textual documents describing the correspondences between specifications and implementations, which is cumbersome and time-consuming. Today the quality team is spending a huge effort to guarantee that such problems of misunderstanding the models do not require to cancel production process instances. This may affect customers and negatively impact the image of the company."

7 Empirical Findings on Tool Support

*7.1 **F7**—The majority of the surveyed stakeholders would prefer a single model for Business and IT*

Summary It is generally accepted that a single model for business and IT is undesirable [25]. Mixing business and IT concepts may produce cluttered models that are hard to understand and maintain, specially on the business side. Curiously, most of the specialists from BNB would prefer a single model for both business and IT. We actually do not interpret this finding as strictly contradicting the state-of-the-art. We rather interpret this result as a twofold message: (i) people want a single model (i.e., a single source of truth) for the overlapping part of related business and IT models, and (ii) they are dissatisfied with the current tool support for managing consistency of multiple (independently edited) process models.

We asked the practitioners to answer which development method they consider more effective for keeping

Business and IT perspectives consistent. We ultimately wanted to find out how happy they are with the current development process and tool support for consistency management. The results are shown in Figure 17.

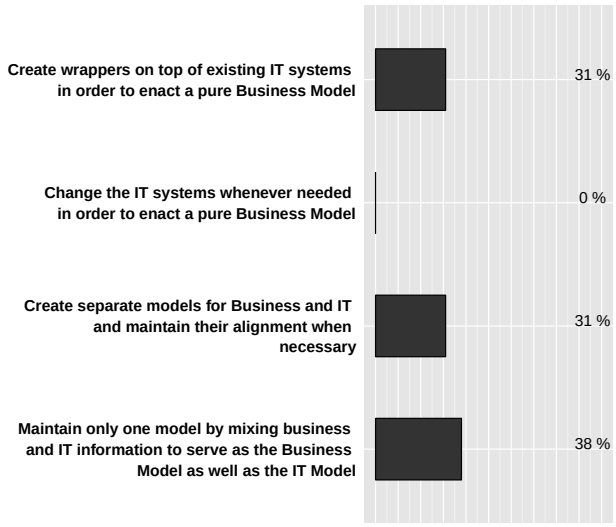


Fig. 17: Preferred Approach to Enforce Consistency

We examined the answers with respect to the roles of the respondents—the first question of the survey asked respondents to provide their roles. Curiously, we noticed that all the respondents who answered *Create wrappers on top of existing IT systems in order to enact a pure Business Model* were Business Analysts. They manifest that pure business models are the ones really needed by the company and that the IT department should do anything necessary to enact them directly. Dealing with some 'pollution' of information in a single model is even considered preferable by business people over the burden and the risk of losing consistency between different models. None of the IT specialists chose that answer. Regarding the remaining answers, most of the IT stakeholders (38% of the answers) are skeptical whether it would be actually possible to enforce consistency by maintaining different models for Business and IT.

Surprisingly, having a single model for Business and IT is generally considered undesirable by existing works [25]. When maintaining a single model, the company might run into the problem that business analysts and managers could no longer understand the resulting model. They might not recognize how their business is reflected in the resulting model. Another problem of this approach occurs when the business model is used to satisfy compliance check points. Mixing Business and IT concepts can force changing the terminology or the level of granularity of business concepts, making the model less clear and less useful for fulfilling the regulations.

Collating the answers and comments from the practitioners, given in both the interviews and the survey, we interpret this result as a twofold message:

- People want a single model for their *common* (overlapping) aspects, i.e., a single source of truth—that is the same as consistency. Nevertheless, there may be disjoint parts for business- and IT-specific concepts, which could live in separate models. For example, all the IT aspects that are not *business relevant* (see Section 6.2) could be maintained in a different place, other than the 'single' model. The same applies for business concepts that are not system-supported, such as manual tasks. This does not completely contradicts the literature, which mainly refers to the specific parts.
- There is a dissatisfaction of the users with the current tool support for managing consistency of multiple process models. More research is necessary to understand whether a single model for the common part would be feasible as solution in practice. An open problem is that the existing process metamodels do not reflect the situation of two overlapping models for business and IT. That is, alternatively to synchronizing two separate models, a new metamodel could be developed to reflect the multiple views use case better. A possible direction is to allow custom views on a shared model [52].

In addition, the use of a single (i.e., unique) model for both common and specific parts may not be technically possible, as some respondents pointed out:

IT Systems Analyst 1: "I sympathise with the idea of having a single process model, as it would eliminate this burden of synchronizing business and IT processes. However, I still have some unclear points in my mind on how this would work: 1 – If the language is the same, most probably the mechanism of having modeling perspectives is critical, since the business roles should stick with their basic building blocks, while on the IT side we have full modeling capability. How would this work in practice? By hiding or showing things, like model elements? Is it really possible to do this? What if by adding transaction scopes and controls we need to split the original process and thus drastically change the business view? It is not clear for me whether you can just hide or show things. 2 – It seems that you expect improving the collaboration between business and IT, but what exactly do you expect that tools would do for improving collaboration? For me the collaboration today is already good with the current tools, although there is a lack of automated support for change propagation and synchronization. However, I do think that the tools also lack a better integration with the development process, such as iteration planning and fine-grained change traceability."

IT Systems Analyst 2: "I believe in a single model only if we can still have specialized views for business and IT – I do not know how this would differ from having different models, since in practice we may implement the business model only partially, or split it into several pieces. On the other hand, if the tool enforces a unique model for both business and IT and does not give any freedom of changing it in parallel for particular users, I am afraid that people would create two different models anyway."

IT Architect: "For me a single model is viable and ideal when you have a highly mature IT service infrastructure, with several business services already available and aligned with the business objectives. In case you need to implement many things from scratch, it is almost inevitable having the business model only as a reference and the executable model more similar to the reality of existing systems."

Business Analyst: "The ideal solution is having only the business model, because it is in the end the consumable asset of the company. With a single model, the alignment will be enforced by the technology, which is good. In the case of technical issues preventing the enactment of a pure business model, it should be possible to solve that by other means instead of changing the model itself."

*7.2 **F8**—Inconsistencies and fixes should be presented as they occur*

Summary Existing approaches propose *quick-fixes*, generated during the editing of the models [26], other works propose off-line reports where the practitioners can assess the degree of consistency of related process models [23]. The surveyed practitioners prefer instantaneous fixing actions, integrated to the modeling tools.

We asked the respondents about their preferences on how to check whether the models are consistent and how potential inconsistencies should be automatically presented by the tools. Figure 18 shows that the respondents seem to prefer looking at concrete model differences, which may be grouped into high-level model changes, rather than metric measures associated with a degree of consistency as proposed in [23].

With respect to fixing actions, most of the respondents would prefer having quick fixes, automatically generated by the tools during the modeling task, as shown in Table 12.

An *IT Architect* has made this comment in the survey:

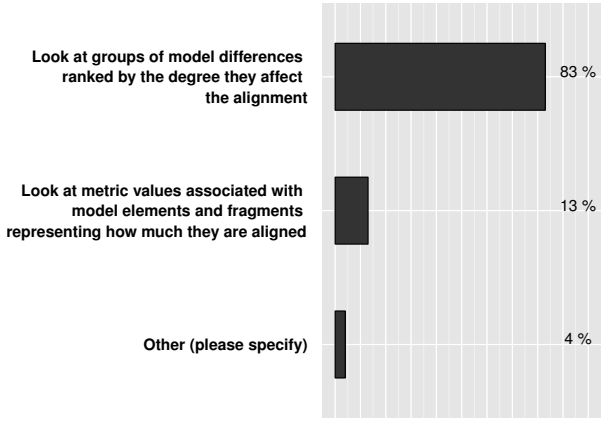


Fig. 18: Preferred Method for Aligning Models

Table 12: How Fixing Actions Should be Presented

	Instantly, during the modeling task	As an offline report, when required
For Business Stakeholders	86%	14%
For IT Stakeholders	95%	5%

"I think that one of the main reasons for the lack of alignment between business and IT is not related to how we create business and IT models or related to what contents they should have or not. I believe that the development process plays an important role in this: today we try to minimize the lack of alignment by enforcing a close relationship between the technical modeller and the business analyst. This is good for new projects, but it often fails in day-to-day for several reasons: in practice many changes are minor, which leads to accumulating some inconsistencies considered not critical until a big change is necessary. Usually most of the change requests made by a business role are described only textually and the business model is not even touched – the problem here is that the business analyst believes that only the production process should be updated and its documentation does not need updating. It is hard to enforce a policy requiring the business analyst to always update the business model, because the one who knows when the documentation should be updated is the business analyst anyway. There are long periods of maintenance that affect primarily the executable model, so during the life cycles of small projects you accumulate several small 'waterfalls' of textual requirements in the sense that the business model (as it should also be part of the requirements) is 'forgotten'. I think that the best way to address this is by showing potential inconsistencies immediately, whenever the models are changed. This would make people aware to keep the models always consistent to a sufficient level. We can also manage the inconsistencies by planning when they should be resolved in future projects."

8 Threats to Validity

Many empirical studies suffer from limitations such as the number of subjects not being representative of the entire population, the differences between development methods and tools employed across different organizations, and so on. In particular, in the domain of process modeling there are still more challenges, such as (to the best of our knowledge) the absence of available open-source projects ranging from business-level specifications to IT-level implementations, and also the fact that companies which adopt such technologies usually consider process artifacts extremely sensitive and confidential. As a result, there is little data from practice to drive BPM research.

Our study is subject to three main limitations: 1) the limited scope of participants and the fact that the study is based on a single company; 2) potential misunderstandings, as it involved translating interviews and survey responses from Portuguese to English, and talking to subjects with varied experience levels and skills; and 3) potential errors in the survey because the survey responses are based on subjective and relatively quick assessments of the respondents.

We think that these limitations do not invalidate our results. While we believe that the numbers of analyzed artifacts and interviewed and surveyed people are substantial, we do not intend to draw any general conclusions about all development processes of process models. Clearly, different development processes and organizational cultures will likely lead to different results. Furthermore, while the refinement patterns are grounded in the studied artifacts, interviews and questionnaires reveal subjective perceptions of the participating subjects. We invite the reader to focus on the overall findings and not on specific numbers. We expect that most BPM development processes that are similar to the one we investigated would face similar difficulties in maintaining consistency of related business and IT process models.

9 Conclusions

We have performed a comprehensive study of an industrial BPM-driven development process, including the analysis of more than 70 models, 17 hours of interviews with practitioners, and inspection of around 1000 change requests in 5 BPM projects. Our study covers several aspects of consistency management, including types of inconsistencies, causes, impacts, and tool preferences.

The findings detailed in our study highlight some limitations in the way that state-of-the-art BPM solutions work:

- *Development process:* Effective consistency management appears to require a progressive disclosure approach, in which models are created by a smooth progression from high-level specifications to IT-level models, preserving a chain of manageable correspondences. Today, related models are initially created using common refinement patterns, but then maintained separately for satisfying the needs of different stakeholders, possibly in different languages.
- *Hierarchical and non-hierarchical refinements:* Refinements should not be restricted to hierarchy. Accordingly, a progressive disclosure modeling approach should take that fact into consideration.
- *Stakeholders need a way to define consistency properties:* Consistency is a subjective notion. The same pair of models may or may not be considered inconsistent. The notion of business relevance influences strongly the consistency rules.
- *There is a lack of support for parallel maintenance:* Parallel maintenance requires differencing and merging techniques, something lacking in the major tools.
- *Detection of inconsistencies:* Inconsistencies should be detected and communicated as soon as they occur and then managed according to a clearly defined process.

We hope that these findings will help researchers and tool builders improve tool support for business process modeling. Such improvements would take into account the common refinement operations used by developers and include rapid detection and presentation of inconsistencies to the user, including possible fixes.

Acknowledgment

We would like to thank the Bank of the Northeast of Brazil (Banco do Nordeste – BNB) for granting us full access to people and artifacts, fundamental for conducting this study. This work was partially supported by an IBM PhD CAS Fellow Scholarship, the Ontario Research Fund's Research Excellence Project on Model-Integrated Software Service Engineering, and the National Natural Science Foundation of China under Grant No. 61202071 and No. 61121063.

References

1. D. Miers and S. A. White, *BPMN Modeling and Reference Guide Understanding and Using BPMN*. Lighthouse, Pt, FL, USA: Future Strategies Inc., 2008.
2. OASIS, "Web Services Business Process Execution Language (WSBPEL) TC," http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel.

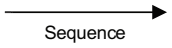
3. Object Management Group, "Business Process Model and Notation (BPMN) Version 2.0," http://www.omg.org/spec/BPMN/2.0/.
4. M. Weidlich, R. Dijkman, and M. Weske, "Deciding behaviour compatibility of complex correspondences between process models," in *Proceedings of the 8th international conference on Business process management*, ser. BPM'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 78–94.
5. R. Dijkman, "Diagnosing differences between business process models," in *Proceedings of the 6th International Conference on Business Process Management*, ser. BPM '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 261–277.
6. C. Li, M. Reichert, and A. Wombacher, "On measuring process model similarity based on high-level change operations," in *Proceedings of the 27th International Conference on Conceptual Modeling*, ser. ER '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 248–264.
7. R. Dijkman, "A classification of differences between similar business processes," in *Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference*. Washington, DC, USA: IEEE Computer Society, 2007.
8. M. Henkel, J. Zdravkovic, and P. Johannesson, "Service-based processes: design for business and technology," in *Proceedings of the 2nd international conference on Service oriented computing*, ser. ICSOC '04. New York, NY, USA: ACM, 2004, pp. 21–29.
9. J. Koehler, R. Hauser, J. Küster, K. Ryndina, J. Vanhatalo, and M. Wahler, "The role of visual modeling and model transformations in business-driven development," *Electron. Notes Theor. Comput. Sci.*, vol. 211, pp. 5–15, April 2008.
10. L. Zerguini, "A novel hierarchical method for decomposition and design of workflow models," *J. Integr. Des. Process Sci.*, vol. 8, pp. 65–74, April 2004.
11. P. Soffer, "Refinement equivalence in model-based reuse: Overcoming differences in abstraction level," *J. Database Manag.*, vol. 16, no. 3, pp. 21–39, 2005.
12. M. Weidlich, A. P. Barros, J. Mendling, and M. Weske, "Vertical alignment of process models - how can we get there?" in *CAiSE 2009 Workshop Proceedings: BPMDS*, 2009, pp. 71–84.
13. T. H. Davenport, *Process innovation: Reengineering work through information technology*. Boston, MA, USA: Harvard Business School Press, 1993.
14. C. Rolland and N. Prakash, "Bridging the gap between organisational needs and ERP functionality," *Requirements Engineering*, vol. 5, pp. 180–193, 2000, 10.1007/PL00010350.
15. N. Bieberstein, S. Bose, M. Fiammante, K. Jones, and R. Shah, *Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005.
16. R. Bobrik, M. Reichert, and T. Bauer, "View-based process visualization," in *BPM*, 2007, pp. 88–95.
17. International Organization for Standardization, *Financial transaction card originated messages – Interchange message specifications – Part 1: Messages, data elements and code values*. [Online]. Available: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=31628
18. A. Delgado, F. Ruiz, I. G.-R. de Guzman, and M. Piattini, "A model-driven and service-oriented framework for the business process improvement," *Journal of Systems Integration*, vol. 1, no. 3, pp. 45–55, 2010.
19. R. M. Dijkman, D. A. C. Quartel, L. F. Pires, and M. J. v. Sinderen, "A rigorous approach to relate enterprise and computational viewpoints," in *Proceedings of the Enterprise Distributed Object Computing Conference, Eighth IEEE International*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 187–200.
20. S. Buchwald, T. Bauer, and M. Reichert, "Bridging the gap between business process models and service composition specifications," in *Service Life Cycle Tools and Technologies: Methods, Trends and Advances*. Idea Group Reference, November 2011, pp. 124–153.
21. G. Spanoudakis and A. Zisman, "Inconsistency management in software engineering: Survey and open research issues," in *in Handbook of Software Engineering and Knowledge Engineering*. World Scientific, 2001, pp. 329–380.
22. M. Weidlich, G. Decker, M. Weske, and A. Barros, "Towards vertical alignment of process models - a collection of mismatches," Hasso Plattner Institute, Tech. Rep., 2008.
23. M. Weidlich, J. Mendling, and M. Weske, "Efficient consistency measurement based on behavioural profiles of process models," *IEEE Transactions on Software Engineering*, vol. 99, no. PrePrints, 2010.
24. B. Nuseibeh, S. Easterbrook, and A. Russo, "Leveraging inconsistency in software development," in *in Software Development", Computer, 33(4):24-29, IEEE Computer*. Society Press, 2000, pp. 1–33.
25. G. Decker, "Bridging the gap between business processes and existing IT functionality," in *Proceedings of the 1st International Workshop on Design of Service-Oriented Applications (WDSOA)*. Amsterdan, The Netherlands: ICSOC, 2005, pp. 17–24.
26. A. Hegedüs, A. Horváth, I. Ráth, M. C. Branco, and D. Varró, "Quick fix generation for DSMLs," in *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing VLHCC 2011*. IEEE, 2011.
27. H.-M. Chen, "Towards service engineering: Service orientation and Business-IT alignment," in *Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, ser. HICSS '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 114–.
28. D. Werth, K. Leyking, F. Dreifus, J. Ziemann, and A. Martin, "Managing SOA through business services: a business-oriented approach to service-oriented architectures," in *Proceedings of the 4th international conference on Service-oriented computing*, ser. ICSOC'06. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 3–13.
29. H. Tran, U. Zdun, and S. Dustdar, "View-based integration of process-driven SOA models at various abstraction levels," in *Proceedings of First International Workshop on Model-Based Software and Data Integration MBSDI 2008*. Springer, 2008, pp. 55–66.
30. J. Küster, "Consistency management of object-oriented behavioral models," Ph.D. dissertation, Universität Paderborn, 2004.

31. A. Finkelstein and I. Sommerville, "The Viewpoints FAQ," *Software Engineering Journal*, vol. 11, no. 1, pp. 2–4, Jan. 1996.
32. W. Emmerich, A. Finkelstein, C. Montangero, S. Antonelli, S. Armitage, and R. Stevens, "Managing standards compliance," *Software Engineering, IEEE Transactions on*, vol. 25, no. 6, pp. 836–851, 1999.
33. Z. Diskin, Y. Xiong, and K. Czarnecki, "Specifying overlaps of heterogeneous models for global consistency checking," in *Proceedings of the First International Workshop on Model-Driven Interoperability*, ser. MDI '10. New York, NY, USA: ACM, 2010, pp. 42–51.
34. A. Marcus and J. I. Maletic, "Recovering documentation-to-source-code traceability links using latent semantic indexing," in *Proceedings of the 25th International Conference on Software Engineering*, ser. ICSE '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 125–135.
35. Z. Xing, "Model comparison with GenericDiff," in *Proceedings of the IEEE/ACM international conference on Automated software engineering*, ser. ASE '10. New York, NY, USA: ACM, 2010, pp. 135–138.
36. J. Euzenat and P. Shvaiko, *Ontology matching*. Heidelberg (DE): Springer-Verlag, 2007.
37. E. Rahm and P. A. Bernstein, "A survey of approaches to automatic schema matching," *The VLDB Journal*, vol. 10, pp. 334–350, December 2001.
38. C. Nentwich, W. Emmerich, and A. Finkelstein, "Consistency management with repair actions," in *Software Engineering, 2003. Proceedings. 25th International Conference on*, 2003, pp. 455 – 464.
39. A. Egyed, E. Letier, and A. Finkelstein, "Generating and evaluating choices for fixing inconsistencies in uml design models," in *Automated Software Engineering, 2008. ASE 2008. 23rd IEEE/ACM International Conference on*, 2008, pp. 99 –108.
40. C. Amelunxen, E. Legros, A. Schürr, and I. Stürmer, "Checking and enforcement of modeling guidelines with graph transformations," in *Applications of Graph Transformations with Industrial Relevance*, 2008, pp. 313–328, LNCS 5088, Springer.
41. J. Pinna Puissant, T. Mens, and R. Van Der Straeten, "Resolving Model Inconsistencies with Automated Planning," in *Proceedings of the 3rd Workshop on Living with Inconsistencies in Software Development*. CEUR Workshop Proceedings, 2010, pp. 8–14.
42. Y. Xiong, A. Hubaux, S. She, and K. Czarnecki, "Generating range fixes for software configuration," in *ICSE'12: 34th International Conference on Software Engineering*, 2012, pp. 58–68.
43. J. M. Küster, "Towards inconsistency handling of object-oriented behavioral models," *Electronic Notes in Theoretical Computer Science*, vol. 109, pp. 57 – 69, 2004, proceedings of the Workshop on Graph Transformation and Visual Modelling Techniques (GT-VMT 2004).
44. R. Dijkman, M. Dumas, L. Garcia-Banuelos, and R. Kaarik, "Aligning Business Process Models," in *2009 IEEE International Enterprise Distributed Object Computing Conference*. IEEE, Sep. 2009, pp. 45–53.
45. B. van Dongen, R. Dijkman, and J. Mendling, "Measuring Similarity between Business Process Models," in *Proceedings of the 20th Int'l Conference on Advanced Information Systems Engineering (CAiSE 2008)*, ser. Lecture Notes in Computer Science, Z. Bellahsène and M. Léonard, Eds., vol. 5074. Montpellier, France: Springer Verlag, 2008, pp. 450–464.
46. M. Weidlich, R. Dijkman, and J. Mendling, "The ICoP framework: identification of correspondences between process models," in *Proceedings of the 22nd international conference on Advanced information systems engineering*, ser. CAiSE'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 483–498.
47. M. C. Branco, J. Troya, K. Czarnecki, J. Küster, and H. Völzer, "Matching Business Process Workflows Across Abstraction Levels," in *Proceedings of 15th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, ser. MODELS 2012. ACM/IEEE, 2012.
48. J. Vanhatalo, H. Völzer, and F. Leymann, "Faster and More Focused Control-Flow Analysis for Business Process Models Through SESE Decomposition," in *ICSOC 2007*, ser. LNCS. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 43–55.
49. B. Fluri, M. Wursch, M. Pinzger, and H. Gall, "Change Distilling: Tree Differencing for Fine-Grained Source Code Change Extraction," *Software Engineering, IEEE Transactions on*, vol. 33, no. 11, pp. 725 –743, nov. 2007.
50. J. Vanhatalo, H. Völzer, and J. Koehler, "The refined process structure tree," in *Proceedings of the 6th International Conference on Business Process Management*, ser. BPM '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 100–115.
51. C. Gerth, J. M. Küster, M. Luckey, and G. Engels, "Precise detection of conflicting change operations using process model terms," in *Proceedings of the 13th international conference on Model driven engineering languages and systems: Part II*, ser. MODELS'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 93–107.
52. J. Küster, H. Völzer, C. Favre, M. C. Branco, and K. Czarnecki, "Supporting different process views through a shared process model," IBM Research Zurich, Tech. Rep., 2012. [Online]. Available: http://domino.research.ibm.com/library/cyberdig.nsf/papers/FA822A5E450EB08685257A1600462337/$File/rz3823.pdf
53. J. M. Küster, C. Gerth, A. Förster, and G. Engels, "Detecting and resolving process model differences in the absence of a change log," in *Proceedings of the 6th International Conference on Business Process Management*, ser. BPM '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 244–260.
54. J. M. Küster and K. Ryndina, "Improving inconsistency resolution with side-effect evaluation and costs," in *MoDELS*, 2007, pp. 136–150.
55. J. Hutchinson, M. Rouncefield, and J. Whittle, "Model-driven engineering practices in industry," in *Proceeding of the 33rd international conference on Software engineering*, ser. ICSE '11. New York, NY, USA: ACM, 2011, pp. 633–642.
56. M. Zapf and A. Heinzl, "Evaluation of generic process design patterns: An experimental study," in *Business Process Management, Models, Techniques, and Empirical Studies*. London, UK, UK: Springer-Verlag, 2000, pp. 83–98.

57. J. Vanhatalo, H. Völzer, and F. Leymann, "Faster and more focused control-flow analysis for business process models through sese decomposition," in *Proceedings of the 5th international conference on Service-Oriented Computing*, ser. ICSOC '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 43–55.

Appendix

Basic BPMN Notation



(a) Flow

(b) Tasks

(c) Events

(d) Gateways