GSDLAB TECHNICAL REPORT

An Empirical Study on Consistency Management of Business and IT Process Models

Moisés Castelo Branco, Yingfei Xiong, Krzysztof Czarnecki, Jochen Küster, Hagen Völzer

GSDLAB–TR 2012–03–22 March 2012

Generative Software Development Laboratory
University of Waterloo
200 University Avenue West, Waterloo, Ontario, Canada N2L 3G1

WWW page: http://gsd.uwaterloo.ca/

Technical Report GSDLAB–TR 2012–03–22

An Empirical Study on Consistency Management of Business and IT Process Models

Moisés Castelo Branco[1], **Yingfei Xiong**[1], **Krzysztof Czarnecki**[1], **Jochen Küster**[2], **Hagen Völzer**[2]

[1] Generative Software Development Laboratory, University of Waterloo, Canada
[2] IBM Research Zurich, Switzerland

March 22nd, 2012

Abstract Process models support the transition from business requirements to IT implementations. 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 contributions are 1) an account of how business process models co-evolve and how their consistency is maintained in a concrete industrial setting; 2) a set of recurrent patterns used to refine business-level process specifications into IT-level models, and 3) a set of findings that confirm or contradict conventional wisdom on process model consistency management found in the literature.

Key words Business processes – consistency management – 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 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 very 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 or not satisfiable 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 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. 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 IT department of the company has more than 300 employees and is responsible for more than 200 information systems. More specifically, we applied three research methods. We first analyzed 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 interviewed 9 professionals ranging from Business Analysts to IT Developers in order to understand how they collaborated to create and maintain the models. Finally, we conducted a survey with 23 professionals to further understand the interesting issues revealed by the artifacts and interviews. Our main contributions are the following:

– *An account of how business process models co-evolve and how their consistency is maintained in a concrete industrial setting*: We provide a characterization of the models in terms of their sizes, language constructs used, the changes they undergo, and their purpose in the development process and how the stakeholders establish and maintain consistency among the models as part of the development process.
– *A set of findings about how these models co-evolve and how the consistency is maintained in the studied organization and what the stakeholders expect from tools*: These findings, some of which confirm or contradict conventional wisdom on process model consistency management found in the literature, are the following:
  1. process models are created and maintained at several levels of abstraction;
  2. business and IT process models are related by hierarchical and non-hierarchical refinement patterns;
  3. models undergo parallel maintenance;
  4. the majority of the questioned stakeholders would prefer a single model for Business and IT;
  5. differences in coverage and behavior affect consistency most;
  6. stakeholders have a subjective notion of consistency;
  7. inconsistencies can cause severe problems; and
  8. inconsistencies should be detected and communicated at the time they occur, along with proposed fixes.
– *A catalog of 11 refinement patterns*: These recurrent patterns are 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.

The remainder of the paper is structured as follows. Section 2 provides background on BPM and describes the running example, three 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 discusses the salient findings from the study, including the refinement patterns catalog. Section 6 analyzes the threats to the validity of the work. Finally, Section 7 summarizes the 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 a workflow, i.e., a flow of activities. Typical examples of business processes are *Purchasing*, *Manufacturing*, *Marketing*, and *Sales*. A business process begins with a mission objective and ends with achievement of the business 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-oriented 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 slightly simplified versions of real process models from one of the studied projects (project *P4*, Section 4.3), as our running example. 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. 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., *Costumer 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 an user or the termination of the process because of an error or a user decision (e.g., *Cancel Transaction*).

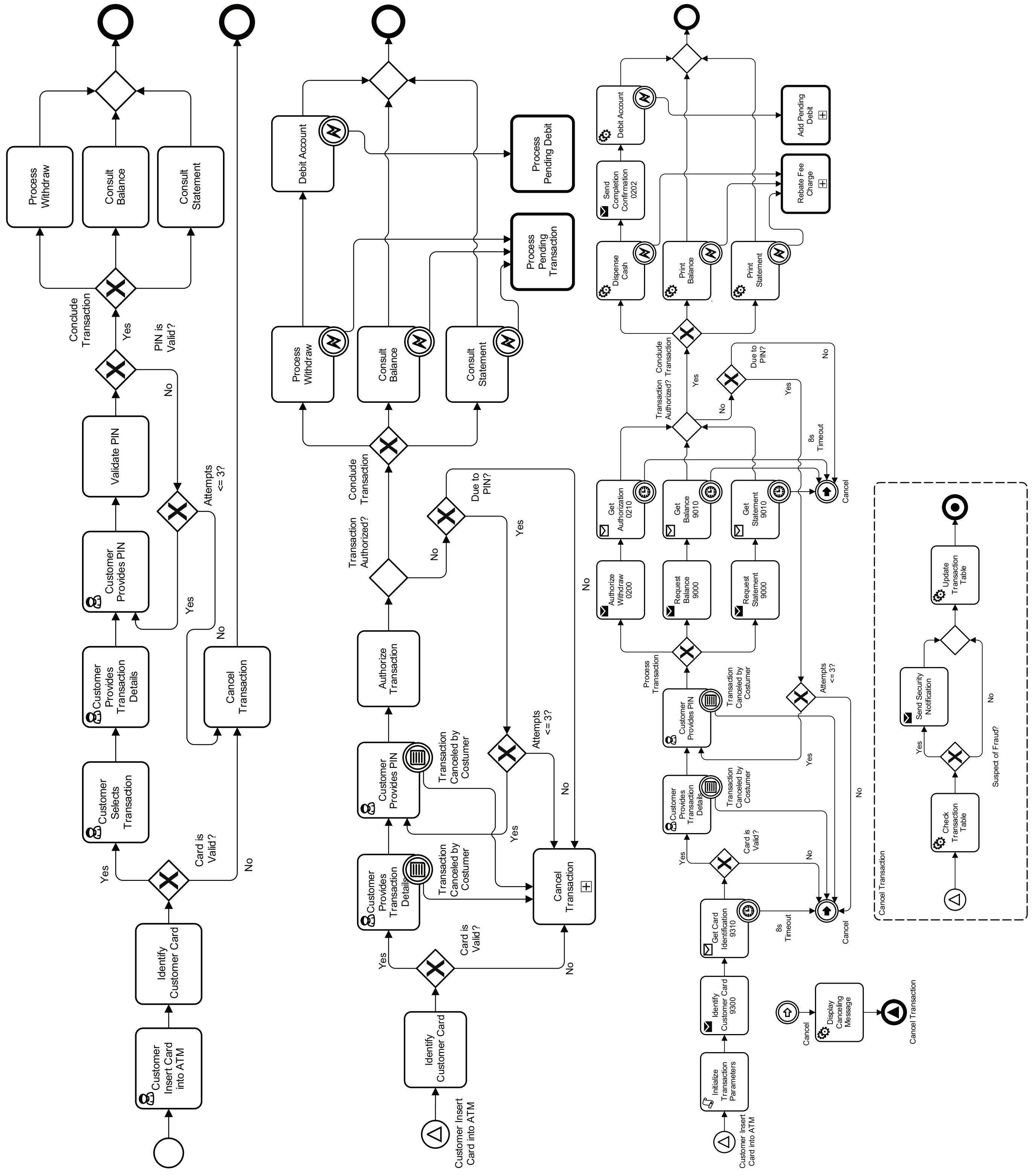
3 Related Work

We discuss related work in three groups. First, we introduce the general area of consistency management and discuss related work addressing specific consistency management activities. Then, we turn to work on consistency management of business process models. Finally, we review empirical work related to our study.

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 [17, 18]. Existing works divide consistency management into a set of activities [17,19,20]. The remainder of this subsection introduces these activities 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 [21, 22]. Such a property is typically defined as a consistency 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 [21]. For example, a reasonable policy is to require that 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.
– ***Aligning the models:*** This activity 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. Model alignment 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 usually 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 generic, graph-based matching [23] and document-to-code traceability recovery [24]. A related area is schema integration, and in particular, schema matching, which deals with establishing correspondences among database schemas (see surveys on this topic [25, 26]).



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

Fig. 1: ATM Process Models

– ***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 [18].

The adopted consistency management policy specifies the circumstances that will trigger the checks.
- ***Diagnosing causes of inconsistencies:*** This activity identifies the source, the cause, and the impact of an inconsistency [18]. The source of an inconsistency is the set of elements of software models that violate a consistency rule [20]. 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 papery [18].
- ***Fixing inconsistencies:*** The final activity is to fix inconsistencies. Several approaches exist to propose automatically one or more possible fixes to the user. For example, Nentwich et al. [27] 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. [28] present an approach that generates concrete fixes for UML models, based on predefined inconsistency rules. Ameluxen et al. [29] 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 [30].

Consistency management of process models

We summarize work 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 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.

  Behavioral consistency typically involves some notion of behavioral equivalence, such as trace equivalence or bisimulation. For example, Küster [31] 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, 32]. An example of such notion are *behavioral profiles* [33]; 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 1.0 (consistent models).
- ***Aligning the models:*** Several publications showed how matching techniques can be applied to business process models [34–36].
- ***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., [33]). Two special representations of process models are used in model comparison: process structure trees [37] and process model terms [38]. 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.
- ***Diagnosing causes of inconsistencies:*** The 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 of any work investigating how diagnosis of inconsistencies among process models is done in practice.
- ***Fixing inconsistencies:*** Hegedüs et al. recently proposed an approach to fix model inconsistencies based on state-space exploration and evaluated it on BPMN models. Küster et al. also discuss the change management and inconsistency resolution in BPM [39, 40].

Empirical Research

We are not aware of any empirical research on consistency management in BPM.

Hutchinson et al. [41] 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 driving lessons learned from MDE practices adopted by three companies. Compared to their work which focuses on MDE in general, our work focuses particularly on BPM and consistency management.

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 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 a *process*, as we aimed to unveil how people resolve their main concern, by discovering stages and phases on doing this activity.

To answer this question, we adopted a structured combination of three research methods: 1) comprehensive artifact study, 2) semi-structured interviews and 3) electronic survey. The combination allowed 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 them.

First, we analyzed business-level and IT-level models to understand the correspondences between them. We were interested in discover 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 (IT) 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 2006, 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), modified 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, containg more than 70 models in total (see Table 1). 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 1 lists the number of models of each type. Table 2 gives the model sizes in number of elements of different types.

Table 1: 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	5
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 [42]) 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.

Table 2: Model Sizes

		Number of Model Elements				
		Pools	Tasks	Gateways	Events	Flows
P1	Business Spec.	11	59	39	25	149
	Technical Spec.	11	78	47	36	164
	Implementation	11	123	56	43	186
P2	Business Spec.	6	47	45	18	128
	Technical Spec.	6	95	47	23	142
	Implementation	6	107	52	31	154
P3	Business Spec.	4	17	4	4	19
	Technical Spec.	4	19	5	4	21
	Implementation	4	22	7	6	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	6	11	31
	Technical Spec.	8	18	9	15	43
	Implementation	8	25	11	17	57

Table 3: Change Requests

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

We made a correlation between the change request descriptions and the artifacts that changed by each request. Requests are made by business stakeholders using an organizational workflow controlled by the product *IBM ClearQuest*. Once a request is made by business, it follows a sequence of steps, such as package grouping and priority definition, until it reaches the IT department. Every request has a textual attribute describing the business objectives of the change. IT personnel need to sign up for a business request (with managerial authorization) before changing artifacts in the repository (*IBM ClearCase*).

Our objective was to find the reasons for changing the artifacts of each project we analyzed. The correlation consisted of matching the textual description of each change request, extracted from the ClearQuest database, with the change log obtained from the ClearCase. The aim of this process was to discover when inconsistencies were introduced by regular maintenance. For example, by finding a particular change in August/2009 that had affected only the business model of project *P1*, we realized from the description of the request that this change had reestablished 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 3.

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 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 showing the roles involved, their experiences with BPM, and the interview durations are shown in Table 4. 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 are inconsistencies 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/maintaining business and IT level models?

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

Are there any architecture or IT infrastructure specific constraints that impact the models?

What collaboration exists between the different roles?

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

Are there examples that inconsistencies were detected? Do you have any examples where this has had undesirable consequences?

Table 4: Interviews Conducted

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

4.5 Survey

We created a questionnaire 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 five answers: *Always necessary*, *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. The complete report of the survey and the comments made by the respondents are available online at our web site[1].

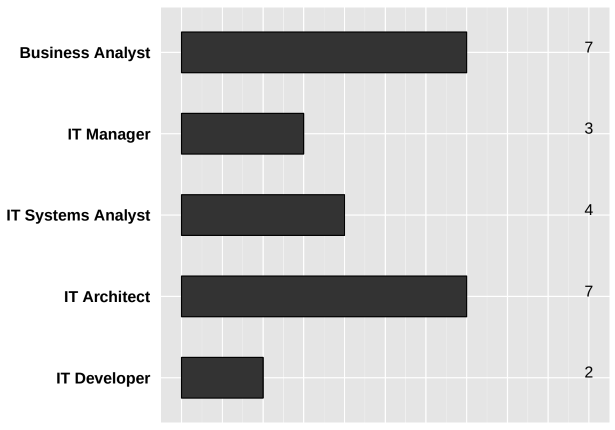


Fig. 2: Survey Answers per Professional Role

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

From the survey in the context of the data we collected, we found that our main research question can be divided in the following sub-questions:

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

For each of this sub-questions, we distill our findings in the next section.

5 Empirical Findings

5.1 Processes are developed and maintained in several levels of abstraction

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 approximate 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. 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.

– *IT Manager:* Produce contracts for meeting the business requests. Assign IT personnel to projects, contract outsourcers.
– *IT Systems Analyst:* Provide technical support for Business Analysts, correct/adjust the BPMN model, clarify business items and rules, detail tasks and flows, specify *Use Cases* for each task, gateway, conditional flows and events.
– *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:

– *Change propagation*: By applying a development process based on *RUP*, requirements are created or updated 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 need to be updated.
– *Validation*: Audits often require checking production processes against high-level specifications and control points of legal reference models like Basel II and Sarbanes-Oxley.

However, the stakeholders 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 about correspondence links among process models is that they are domain-specific and user-defined. For example, we found correspondences that can be understood only by having knowledge on existing systems used in BNB. Then, it seems to be naive to assume that automatic techniques for deriving correspondences can approach the quality achieved by humans with specific domain knowledge.

We thus confirm the conclusions regarding the challenges of establishing correspondences among process models presented in [12]; by stating that more research is necessary to understand the trade-off between automatic and manual efforts to establish and manage correspondence links, integrated into the development process.

The following quote is an additional summary of the development process obtained in an interview:

"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. The main problem with our BPM development is maintaining traceability among such models and artifacts - this often requires considerable rework and is specially critical when outsourcers are involved. I say that we could have much better tool support for managing this. Additionally, I suggest you look at the BNB-UP (development process website) for more information."

5.2 Business and IT process models are related by hierarchical and non-hierarchical refinement patterns

Based on the ATM case study (*P4*)(see Section 5.1), 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; The executable process is implemented on top of an ISO8583 service infrastructure [43] and the codes that appear in the names of some tasks, such as 0200 and 9010, are types of messages of this protocol. Although the executable model is implemented in BPEL, for simplicity, we remodeled here a simpler version in BPMN 2.0 [3] that preserves 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 5 shows statistics of the pattern occurrences across the models.

Table 5: Refinement Occurrences

	Occurrences				
Refinement Pattern	P1	P2	P3	P4	P5
Add properties	15	7	12	8	6
Add script task	6	3	4	1	1
Add protocol task	1	-	2	1	5
Add boundary event	3	5	9	6	6
Add technical exception flow	5	4	3	2	4
Change activity name	-	-	2	1	1
Change activity type	-	1	1	-	1
Refactor gateway	-	1	-	2	1
Splitting task into block	2	4	2	1	2
Splitting workflow	20	3	4	5	1
Suppress specification activity -	-	1	-	1	1

Add properties

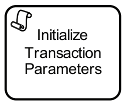
Several properties of tasks, gateways, flows, events etc. are added to the implementation-level model, such as application or service URLs, type of protocol (http/https), transactional behavior (e.g. commit before, commit after, participates etc.). Such properties do not change the workflow and may be tool/platform-specific.

Add script task

Script tasks are frequently 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. This type of task is usually better in terms of performance than calling external services. Figure 3 shows a task created in the ATM application for initializing several parameters of a *transaction* object, that controls user actions across the workflow. Such kind of task in the IT model does not have any correspondence in the business model.

Add protocol task

It is common to implement a business task by using an asynchronous service. In such cases, the protocol needs to compose and send a menssage and, after that, wait for a response. Figure 4 shows an example: the business



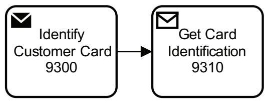
(a) Executable

Fig. 3: Add Script Task

task *Identify Customer Card* is implemented on top of the ISO8583 protocol by sending a 9300 message and waiting for a validation message (9310).

Identify
Customer Card

(a) Business and Technical Specifications



(b) Executable

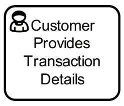
Fig. 4: Add Protocol Task

Add boundary event

Boundary events are commonly used to divert the normal flow under special conditions, for example because of a particular action performed by the operator on a human task. Sometimes such conditions need not to be represented in the business model, and are actually part of the requirements and use cases that describe a human task in detail. Figure 5 depicts an example of boundary event added to human tasks to capture the costumer's decision for canceling 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 Card Identification 9310) to cancel the transaction in the case of a timeout of 8s.

Add technical exception flow

As a special case of adding boundary events, technical exception flows are included to divert the flow in case of technical exceptions, such as an unavailable service or a permission denied. Such conditions are not expected to be represented in the business model, because they implement non-functional technical requirements elicited during the *elaboration* phase of the development process. Figure 6 shows examples of technical exceptions



(a) Business Specification

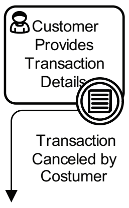
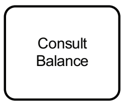
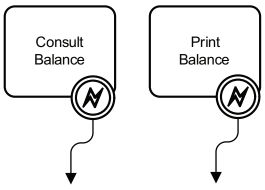

Customer Provides Transaction Details
Transaction Canceled by Costumer
(b) Technical and Executable

Fig. 5: Add Boundary Event

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

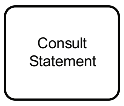
Fig. 6: Add Technical Exception Flow

Change activity name

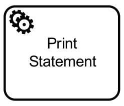
The name of an business activity can be changed to facilitate the identification of an IT service that has a similar but different name - the IT specialist can decide to keep the technical name for facilitating future maintenance. Figure 7 shows an example.

Change activity type

The type of a business activity can be changed because of implementation decisions. For example, a human task represented in the business model can be implemented as an event - this case is shown in Figure 8.

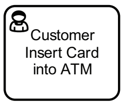


(a) Business and Technical Specifications

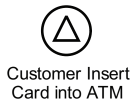


(b) Executable

Fig. 7: Change Activity Name



(a) Business Specification



(b) Technical and Executable

Fig. 8: Change Activity Type

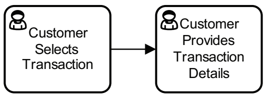
Suppress specification activity

Some elements of the business specification may be considered redundant or subsumed by a particular task in the implementation level. Figure 9 shows a case where the two human tasks described in the business were collapsed into a single human task on 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.

Splitting task into block

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 intended specification functionality. Figure 10 illustrates such scenario, where a technical specification task, *Autorize 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.

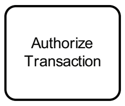

Customer Selects Transaction
Customer Provides Transaction Details

(a) Business Specification

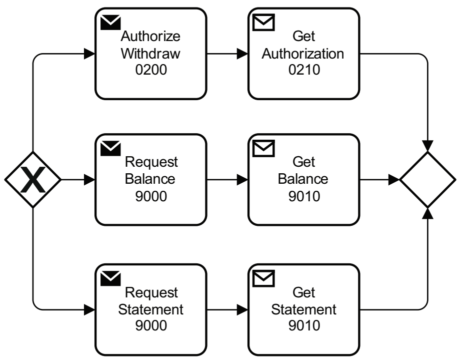
Customer Provides Transaction Details

(b) Technical and Executable

Fig. 9: Suppress Specification Activity



(a) Technical Specification



(b) Executable

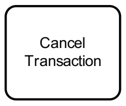
Fig. 10: Splitting Task into Block

Splitting workflow

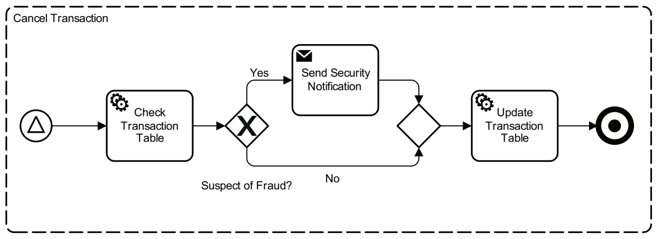
The specification workflow can be split into smaller workflows that should be orchestrated by a main flow. The typical reason for this is the creation specialized and reusable workflows. In Figure 11 the task *Cancel Transaction* was implemented by a specialized subflow that includes fraud control and is reused by other projects. It is common to use web service interfaces or event triggering for calling the subflows.

Refactor gateway

A business level gateway can be refined to take into account the technical behavior of the services involved. Figure 12 shows an example where the business specification has a business rule for checking the maximum number of times that a costumer can enter a wrong PIN. In the actual implementation, checking the validity of the PIN is a particular result of the transaction authorization.



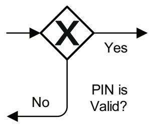
(a) Business and Technical Specifications



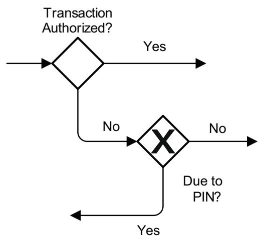
(b) Executable

Fig. 11: Splitting Workflow

The business analysts did not consider it relevant to include this level of detail in the business model, so that the workflows then became different at this point.



(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 *Splitting 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. Interestingly, several approaches for aligning business and IT perspectives are based on the assumption that hierarchical refinements are sufficient in practice, such as [10,11,44]. We also gave examples and surveyed the professionals on this and most of them support the need for both types of refinements, as shown in Table 6.

Table 6: 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%

5.3 Models undergo parallel maintenance

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. Figure 13 shows the distribution of 388 maintenance requests made on project *P1* in its three first years. We classify each request according to the model that has been affected, for example, *Only Business* means that a request has changed solely the business model, whereas *None* means that the models were not affected (but other resources, such as databases, services and application components). Around 2% of the maintenance requests affected only the business model, 10% affected both business and IT, 24% affected only the IT model, and 64% affected none of the models. Figure 14 shows the first-year stacked distribution of maintenance requests for the other projects.

By tracking the patterns of maintenance when inconsistencies were introduced, we have identified that the amount of difference that the models display does not really matter when it comes to consistency loss. The main cause of consistency loss is the parallel maintenance without mechanisms to promptly identify and report potential (subjective) inconsistencies. This means that solving the problem requires providing tool-integrated methods for checking consistency and providing fixes as soon as the potential inconsistencies are detected - preferably during the editing the models. We summarize the following cases where inconsistencies were detected and fixed when opportune:

– *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 and has been made only in the IT model. This is considered *undesirable inconsistency* by the stakeholders, and the business model is being updated now. Audits often motivate this type of maintenance.
– *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.
– *Case 3*: Update both the business and the IT model, taking the advantage of the effort to regain consistency between the models if necessary.

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. All the *Case 1* changes could be avoided if the inconsistencies had been reported when the changes on the IT model were made.

Some stakeholders complained 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 complain is the lack of tool features for easily comparing, differencing and merging process models as there are widely available for textual source code.

5.4 The majority of the stakeholders prefer a single model for Business and IT

We asked the respondents to answer which development method they consider more effective for keeping Business and IT perspectives consistent. We wanted find out how happy they are with the current development process and tool support for consistency management. The results are shown in Figure 15.

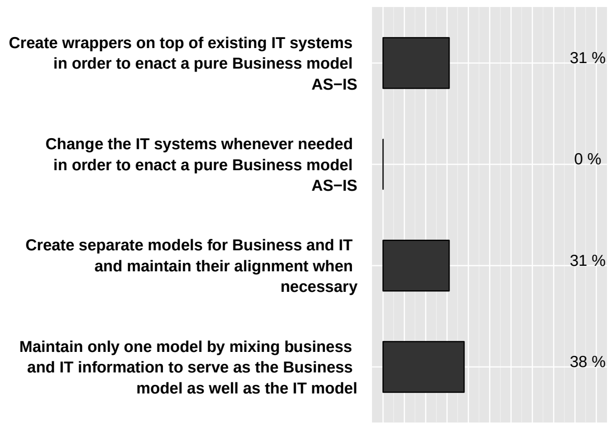


Fig. 15: Preferred Approach to Enforce Consistency

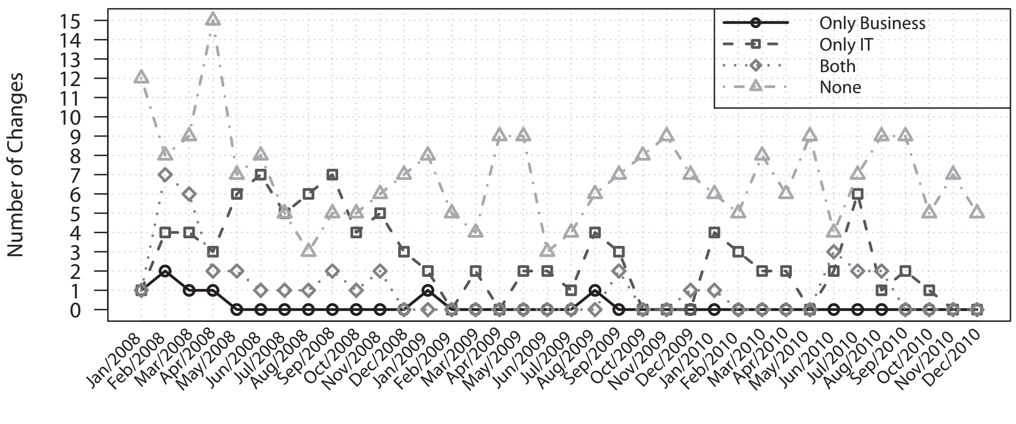
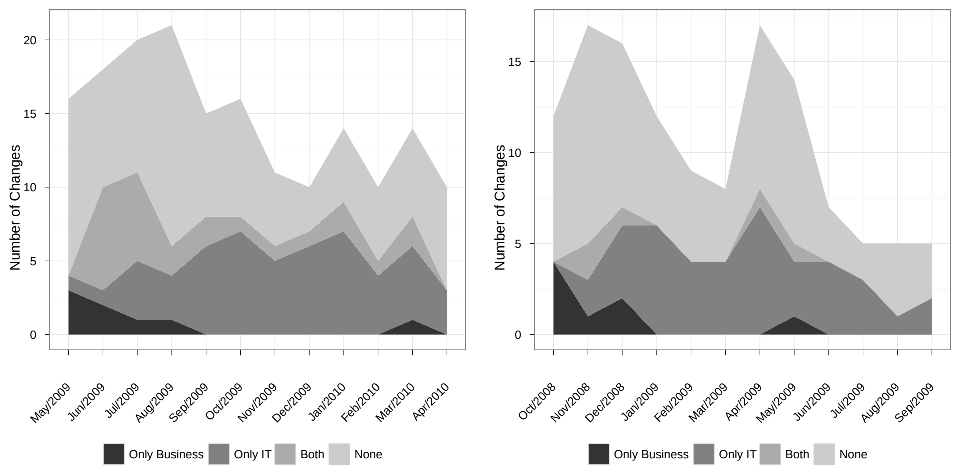
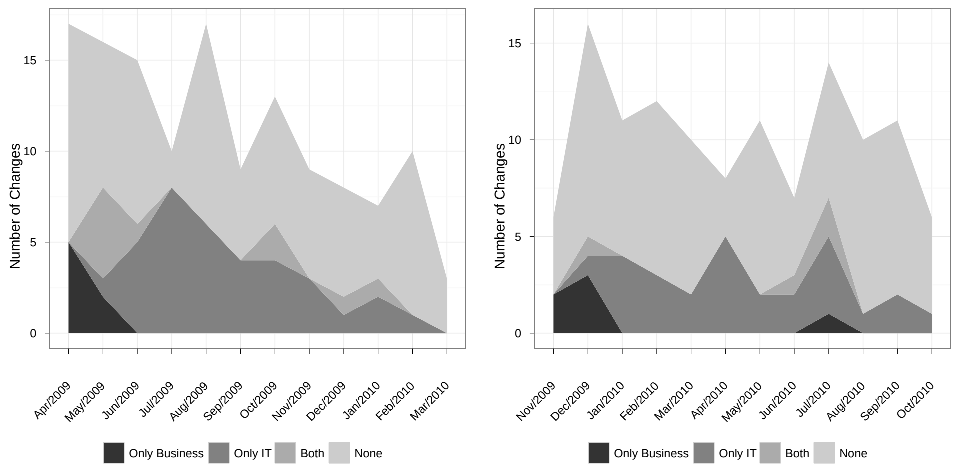


Fig. 13: P1 Change History



(a) P2

(b) P3



(c) P4

(d) P5

Fig. 14: First Year Change History

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. On the other hand, all business stakeholders (31% of the answers) think 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.

Surprisingly, having a single model for Business and IT is generally considered undesirable by existing works [45]. When maintaining a single model, the company might run into the problem that the business analysts and managers could no longer understand the resulting model. They might not recognize how their business individuality 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.

We actually interpret this result as a consequence of the dissatisfaction of the users with the current tool support for managing consistency of multiple process models. More research is definitely necessary to understand in practice whether a single model could fly as solution. Possibly the use of specialized views on a single shared model is a way to preserve the right level of information needed by each stakeholder.

In addition, the use of a single model may not be technically possible in all cases, as some respondents pointed out:

"I sympathise with the idea of having a single process model, and thus we 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 this would work in practice? By hiding/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 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."

"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."

"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."

"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."

5.5 Differences in coverage and behavior are what most affects 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/ignored when checking consistency between Business and IT models

The answers for these two questions are shown in Table 7 and Table 8, 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 9. 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 on these types of differences.

From this we discovered the notion of *Business Relevance* whenever the stakeholders check consistency. If a mismatch is considered relevant to the business it should be fixed, otherwise it is basically 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 *Dentisy* differences that are also not considered business relevant: both are considered details of implementation.

We then confirm the postulates presented in [12] on how differences affect consistency, 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.

Table 7: 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 8: 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 9: Consistency Aspects Mentioned in the Interviews

Consistency Aspect	Necessary All Times	Important but not Always	May Be Irrelevant	Always 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%

We believe that more investigation is necessary to understand how one can allow the stakeholders to define and manage these types of differences one by one 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 for these previous two questions were subjective, consistently with results from the interviews, as we discuss in the next subsection.

5.6 Stakeholders have a subjective notion of consistency

We asked specific questions in the interviews aiming to understand how the BPM practitioners would definitely define consistency among process models, how they re-align 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:

"This task is somehow subjective and the criteria of consistency may vary from people to people, 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 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 continual 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 [18].

Interestingly, it is generally accepted that it is hard for people to agree on the meaning of consistency between business and IT models [12,32,33].

As we started discussing in the last subsection, 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.

5.7 Inconsistencies can cause severe problems

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, see Section 5.5). 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 medium-size maintenance project. When updating the IT model, a developer inadvertently removed a functionality like the one 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 as a technical implementation aspect 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 was 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 do not have direct access to 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, it can still happens, there are no specified tests to capture every possible issue.

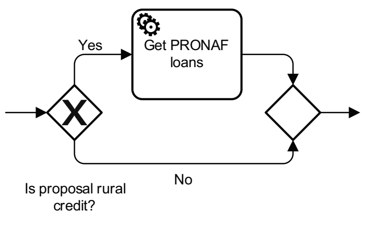


Fig. 16: Functionality Inadvertently Removed

The second case was pretty similar, but this time it was discovered by a regular audit procedure, where projects and its artifacts are inspected in terms of consistency. Unclear points are 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.

One of the business analysts added a 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, what 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 costumers and negatively impact the image of the company."

5.8 Inconsistencies and fixes should be presented as they occur

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 17 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 [33].

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 10.

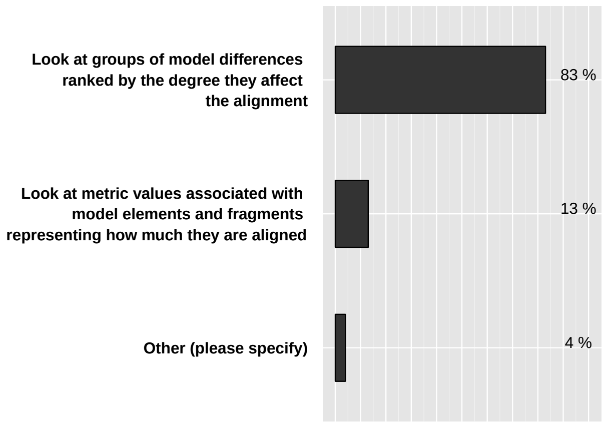


Fig. 17: Preferred Method for Aligning Models

Table 10: 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%

An IT architect has made this comment in the survey:

"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 on 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 accumulate some inconsistencies not considered 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. 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 be aware of keeping the models always consistent to an acceptable level. We can also manage the inconsistencies by organizing when they will be resolved in future projects."

6 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 limitations such as (to the best of our knowledge) the complete 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 that kind of artifact and information extremely sensitive or even confidential. This makes it much harder having concrete data to drive research.

Our study is subject to three main limitations: 1) the small scope of subjects and the fact it 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) some of the survey answers may be wrong because they are based on subjective and 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. 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.

7 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 by refinement patterns, however maintained separately for satisfying the needs of different stakeholders, possibly in different languages.
– *Hierarchical and non-hierarchical refinements:* It seems naive to assume that refinements can 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 determines the consistency rules.
– *There is a lack of support for parallel maintenance:* Parallel maintenance benefits from differencing and merging techniques, something lacking in the major tools.
– *Rapid detection of inconsistencies:* Inconsistencies should be managed as soon as they are detected.

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.

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. [Online]. Available: http://dl.acm.org/citation.cfm?id=1882061.1882072
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. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85758-7_20
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. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-87877-3_19
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, pp. 37–. [Online]. Available: http://dl.acm.org/citation.cfm?id=1317532.1318035
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. [Online]. Available: http://doi.acm.org/10.1145/1035167.1035171

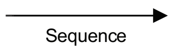
9. J. Koehler, R. Hauser, J. Küster, K. Ryndina, J. Vanhatalo, and M. Wahler, "The role of visual modeling and model transformations in business-driven development," *Electron. Notes Theor. Comput. Sci.*, vol. 211, pp. 5–15, April 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1367148.1367336
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. [Online]. Available: http://dl.acm.org/citation.cfm?id=1275852.1275858
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. [Online]. Available: http://portal.acm.org/citation.cfm?id=171556
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. [Online]. Available: http://dx.doi.org/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. J. Küster, "Consistency management of object-oriented behavioral models," Ph.D. dissertation, Universität Paderborn, 2004.
18. 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.
19. A. Finkelstein and I. Sommerville, "The Viewpoints FAQ," *Software Engineering Journal*, vol. 11, no. 1, pp. 2–4, Jan. 1996.
20. 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.
21. 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. [Online]. Available: http://dx.doi.org/10.1109/32.824413
22. 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. [Online]. Available: http://doi.acm.org/10.1145/1866272.1866279
23. 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. [Online]. Available: http://doi.acm.org/10.1145/1858996.1859020
24. 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. [Online]. Available: http://portal.acm.org/citation.cfm?id=776816.776832
25. J. Euzenat and P. Shvaiko, *Ontology matching.* Heidelberg (DE): Springer-Verlag, 2007.
26. E. Rahm and P. A. Bernstein, "A survey of approaches to automatic schema matching," *The VLDB Journal*, vol. 10, pp. 334–350, December 2001. [Online]. Available: http://dx.doi.org/10.1007/s007780100057
27. 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.
28. 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.
29. 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.
30. 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.
31. 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).
32. 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. [Online]. Available: http://bpt.hpi.uni-potsdam.de/pub/Public/MatthiasWeidlich/collection_of_mismatches.pdf
33. 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.
34. 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. [Online]. Available: http://dx.doi.org/10.1109/EDOC.2009.11
35. 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. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-69534-9_34

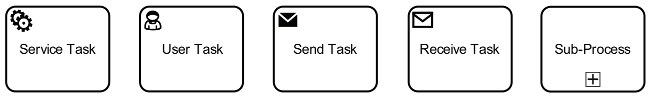
36. M. Weidlich, R. Dijkman, and J. Mendling, “The ICoP framework: identification of correspondences between process models,” in *Proceedings of the 22nd international conference on Advanced information systems engineering*, ser. CAiSE’10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 483–498. [Online]. Available: http://dl.acm.org/citation.cfm?id=1883784.1883832
37. 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. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85758-7_10
38. 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. [Online]. Available: http://portal.acm.org/citation.cfm?id=1929101.1929113
39. 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. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85758-7_19
40. J. M. Küster and K. Ryndina, “Improving inconsistency resolution with side-effect evaluation and costs,” in *MoDELS*, 2007, pp. 136–150.
41. 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. [Online]. Available: http://doi.acm.org/10.1145/1985793.1985882
42. 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. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74974-5_4
43. 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
44. 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. [Online]. Available: http://dl.acm.org/citation.cfm?id=1025120.1025696
45. 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.

Appendix

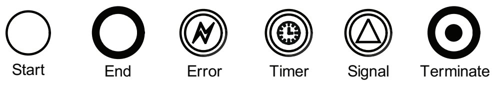
Basic BPMN Notation



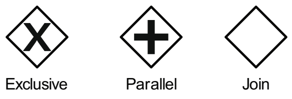
(a) Flow



(b) Tasks



(c) Events



(d) Gateways