University of Wollongong
Research Online

University of Wollongong Thesis Collection

University of Wollongong Thesis Collections

2013

Process ecosystem views to managing changes in business process repositories

Tri Astoto Kurniawan
University of Wollongong

Recommended Citation

Kurniawan, Tri Astoto, Process ecosystem views to managing changes in business process repositories, Doctor of Philosophy thesis, School of Electrical, Computer and Telecommunications Engineering - Faculty of Engineering and Information Sciences, University of Wollongong, 2013. http://ro.uow.edu.au/theses/4404

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

University of Wollongong 

PROCESS ECOSYSTEM VIEWS TO MANAGING CHANGES IN BUSINESS PROCESS REPOSITORIES

A Dissertation Submitted in Fulfilment of
the Requirements for the Award of the Degree of

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

Tri Astoto Kurniawan
BEng, MEng

School of Computer Science and Software Engineering
Faculty of Engineering and Information Sciences

2013

© Copyright 2013

by

Tri Astoto Kurniawan

ALL RIGHTS RESERVED

CERTIFICATION

I, Tri Astoto Kurniawan, declare that this dissertation, submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Computer Science and Software Engineering, Faculty of Engineering and Information Sciences, University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications at any other academic institution.

Tri Astoto Kurniawan
18 November 2013

Dedicated to

my wife Iin
and
my sons - Fa'iz and Nazhif,
for their patiences to accompany me in this very challenging journey

Table of Contents

List of Tables

List of Figures

Process Ecosystem Views to Managing Changes in Business Process Repositories

Tri Astoto Kurniawan

A Dissertation for Doctor of Philosophy

School of Computer Science and Software Engineering
University of Wollongong

ABSTRACT

Business process management has emerged as a focus of considerable industry and research interest in recent times. Organizations routinely find themselves in situations where they need to manage a very large number distinct processes. A single organization might need to manage hundreds or even thousands of business process models that are stored in a *process repository*. In addition, the complexity of individual process models (designs) has also grown. Business process designs within an enterprise process repository are often closely interrelated. Some designs represent reference process models, while others are context-specific generalizations or specializations of other process models. Processes are also interrelated via choreographies. Further, an organization may practice a continuous improvement approach such that it needs to frequently change its interrelated business processes in order to maintain its performance.

In such changes, an initial modification made to a process model may impact a number of other process models in the repository in order to preserve their interdependencies. However, it is not trivial to manage such changes in a process repository consisting of a large number of process models. Such change management imposes us to clearly define and to precisely preserve any existing inter-process relationship. Currently, there are not enough approaches dealing with how to define inter-process relationships which may occur in a process repository and how to maintain such relationships which may be perturbed due to changes made on a process model. We, however, recognize a little work on preserving any inter-process relationship due to process changes in process model collections through change propagation. Further, defining relationships among process models still requires more works.

Therefore, in this dissertation, we introduce *process ecosystems* to view large and complex business process repositories, which emphasize the identification and maintenance of *normative* inter-process relationships. We argue this framework can deliver significant value to fill the existing gap in dealing with process change management in a process repository. We leverage the Software Engineering Research Methodology (SERM) including conceptualization, formalization and development processes in order to construct the proposed framework.

During conceptualization, we identify problems in managing changes in business process repositories and ground them with the theoretical constructs. In the formalization process, we have identified and formalized three inter-process relationship types, i.e. *part-whole*, *inter-operation* and *generalization-specialization*, leveraging a machinery for semantic effect annotation of process models. In the development process, we

further develop process ecosystem framework as the solution for problems identified in dealing with process changes management in a process repository. We leverage techniques from constraint networks to define procedures to manage and propagate change in process ecosystems. We then evaluate such proposed framework to study its performance and accuracy. Our experimental results suggest that such procedures are efficient to propagating changes within medium-sized process repositories. Finally, we expect this new framework to significantly assist process analysts in dealing with process changes management in process repositories.

KEYWORDS: semantic effect annotation, inter-process relationship, process ecosystem, constraint networks, change propagation

Acknowledgements

It is with my great pleasure and honor to acknowledge all contributors who were involved, both directly and indirectly, in completing my dissertation and its all related publications. First of all, I would like to acknowledge my deepest gratitude to Prof. Aditya K. Ghose, who served as my only supervisor. Prof. Aditya had patiently supervised me since the beginning of my PhD studies. I came to him in October 2008 with no high rank publications in my hands as his new PhD student. Since then, Prof. Aditya always guided, motivated and supported me to elevating my capability in doing research as well as presenting scientific talks. To be honest, I would have to admit that my achievements so far have not yet satisfied all of his expectations. I, however, was very lucky to have an opportunity to work with him such that, at the end of my PhD studies, I had been developing my research capability including sharping my research analysis, shaping my scientific writing skills, generating new ideas. Having a reasonable number of papers sitting in some high rank publication places and getting my dissertation done were the most two valuable achievements in my current research career resulting from his deep supervision.

In addition, I was happy to work with Dr. Hoa Khanh Dam as a co-author in some of my publications. Dr. Hoa always red carefully my papers and gave me very useful feedbacks for improving the content of the papers. He also contributed in developing my research capability. Furthermore, I wish to thank Dr. Lam-Son Lê for his supports in developing my research capability. He helped me to prepare my paper which was then my first paper to get in for a publication. Getting the first paper in for a publication was an important milestone for me to mentally ease for other subsequent publications during this hard time. He also served as a co-author in some of my publications. My colleagues in decision systems lab. (DSL) research group at University of Wollongong also took very important part in my research journey. Their discussions during our weekly seminar, i.e. CafeDSL, became valuable inputs to improve my research contents. Giving many talks at CafeDSL had improved my presentation skills as well as my confidence due to their hard and spontaneous questions. As such, I would like to highly appreciate them who always 'crowded' the seminar with their brilliant ideas, especially my deer colleagues Evan Morrison, Konstantin Hoesch-Klohe, Kerry Hinge and Chee Fon Chang.

I wish to thank my home university, i.e. University of Brawijaya, Indonesia, for allowing me to pursue my PhD studies as well as Indonesian Government for substantially supporting me with their scholarship. I wish to send my special gratitude to

my wife, dr. Iin L. Ariati, for her sacrifice to temporarily terminate her career as a doctor in my country for being able to accompany me in pursuing this challenging journey. She was always available to care our sons M. Fa'iz Nashrullah and M. Nazhif Taufiqullah on many occasions to let me focus on my research. In particular, she always supported me with her very constructive discussions in developing some business process models coming from medical field used in my dissertation and publications. I would also like to thank my parents H. Abdul Madjid and Hj. Harijati as well as my mother in law Hj. Sundari for their unconditional encouragement and prayer for me to move forward towards my study completion. This achievement is also dedicated to the loving memory of my father in law H. Toekiman R. from which I learned a lot. Furthermore, I would thank my relatives, especially Arif H.S., Erni R.K. and Darmawan Y. with their respective families, for their supports in my achievements so far.

Finally and most importantly, I would like to thank God for giving me opportunities to meet very nice people along the journey in pursuing my PhD and making all things happen.

Publications

The following publications have resulted from the research presented in this dissertation:

- Tri A. Kurniawan, Aditya K. Ghose, Hoa K. Dam, and Lam-Son Lê. Relationship-preserving change propagation in process ecosystems. In Chengfei Liu, Heiko Ludwig, Farouk Toumani, and Qi Yu, editors, *Service-Oriented Computing*, volume 7636 of *Lecture Notes in Computer Science*, pages 63-78. Springer Berlin Heidelberg, November 2012. DOI 10.1007/978-3-642-34321-6_5.

- Tri A. Kurniawan, Aditya K. Ghose, and Lam-Son Lê. Resolving violations in inter-process relationships in business process ecosystems. In Aditya Ghose, Huibiao Zhu, Qi Yu, Alex Delis, Quang Z. Sheng, Olivier Perrin, Jianmin Wang, and Yan Wang, editors, *Proceedings of the 8th International Workshop on Engineering Service-Oriented Applications (WESOA'12)*, in conjunction with the 10th International Conference on Service-Oriented Computing (ICSOC 2012), pages 332-343, Shanghai, China, November 2012. DOI 10.1007/978-3-642-37804-1_34.

- Tri A. Kurniawan, Aditya K. Ghose, Lam-Son Lê, and Tiancheng Zhang. Design maintenance in process eco-systems. In Louise Moser, Manish Parashar, and Patrick Hung, editors, *Proceedings of the IEEE 9th International Conference on Services Computing (SCC 2012)*, pages 392-399. IEEE Computer Society, June 2012. DOI 10.1109/SCC.2012.104.

- Tri A. Kurniawan, Aditya K. Ghose, Lam-Son Lê, and Hoa K. Dam. On formalizing inter-process relationships. In Florian Daniel, Kamel Barkaoui, and Schahram Dustdar, editors, volume 100 of *Lecture Notes in Business Information Processing*, pages 75-86. Springer Berlin Heidelberg, August 2011. DOI 10.1007/978-3-642-28115-0_8.

- Tri A. Kurniawan, Aditya K. Ghose, and Lam-Son Lê. A framework for optimizing inter-operating business process portfolio. In Jaroslav Pokorny, Vaclav Repa, Karel Richta, Wita Wojtkowski, Henry Linger, Chris Barry, and Michael Lang, editors, *Information Systems Development*, pages 383-396. Springer New York, 2011. DOI 10.1007/978-1-4419-9790-6_31.

Chapter 1

Introduction

This chapter provides an introduction to this dissertation. We firstly present our motivation in Section 1.1. Based on such motivation, we define the central research questions in Section 1.2. In Section 1.3, we describe the research methodology used in this research. In Section 1.4, we describe the main contributions of this dissertation. Finally, we outline the structure of this dissertation in Section 1.5.

1.1 Motivation

Business process management has emerged over the past few decades as an important approach to the effective management of organizations. A recent business process management (BPM) study [47] indicates that BPM investments will reach $7 billion by 2018. An earlier business process modeling survey [23] reported that 93% of the business respondents (i.e. coming from various industries, organization's sizes, functions and geographical areas) were engaged in business process modeling of some form. Many medium to large organizations have collections of hundreds or even thousands of business process models (e.g. 6,000+ process models in Suncorp's insurance process model repository [37]). In addition, a study [25] suggests that continuous process

improvement is the top benefit of using BPM. All of these observations suggest that business process modeling and management on a very large-scale, and involving considerable complexity, are regarded as critical for present-day firms.

Organizations often routinely manage large numbers of process models of considerable complexity stored in enterprise *business process repositories*. The management of such repositories presents major challenges that a large body of BPM researches (for instance, [24, 27, 28, 30, 56, 61, 64]) has only partially addressed. The first challenge involves the identification and validation of *normative* inter-process relationships between process designs in enterprise process repositories. The notion of a normative relationship deserves special attention. Process repositories contain a mix of reference process models (i.e. models that describe general processes), variants of these models that apply to specialized contexts and so on. Process repositories also contain process designs that inter-operate through process choreographies. Such a complex web of inter-process relationships needs to be carefully identified. When analysts, process modelers or process repository managers suggest that a certain relationship ought to hold, it is also important to verify that such a relationship can, in fact, hold given the specific process designs in question. The second challenge involves change management. While change propagation among business process models has been considered as a mechanism for enforcing their relations in process model collections [15], important questions remain. Specifically, the challenge is to manage the trade-off between the competing pulls of minimizing changes to process models (to preserve investments in the process infrastructure associated with the existing process designs) and preserving the normative inter-process relationships discussed above. For instance, if process P2 is a specialization of process P1, and P1 needs to be modified (for instance, to deal with potential compliance violations), it may be important to modify P2 as well, in order to preserve the specialization relationship (and consequently ensure that the

compliance concerns are reflected in the modified design of P2). Change management in a process repository consisting of a complex network of interrelated process designs can be a daunting challenge.

To handle the complexity of these challenges, we view a collection of interrelated process models as an ecosystem [19]. In such an ecosystem, process models play a role analogous to that of biological entities in a biological ecosystem. They are created (or discovered, using automated toolkits [20]), continually changed during their lifetimes and eventually discarded. They undergo constant change driven by changing requirements or changing operating context. The need to preserve existing designs and the need to preserve inter-process relationships also leads to the competing pulls that characterize natural ecosystems. This view of process repositories supports the development of machinery to identify normative inter-process relationships and propagate change through these networks of process designs.

1.2 Research questions

Providing change propagation capability into BPM fundamentally requires identification and validation of any possible relationship among business processes within a process repository, which will rely on the formalization of such relationship. We, then, establish a normative relationship between a pair of processes once it is validated. Violation in any established normative inter-process relationship will be considered as a case to performing process change propagation. As we use ecosystem metaphor in such change propagation, we will refer to violation in any inter-process relationship as a perturbation of the ecosystem equilibrium. In this context, propagating the changes to resolve such violation refers to finding a new equilibrium of the ecosystem in question. This change propagation will also include redesigning processes involved in the violated inter-process relationship. We, however, need to perform effectively

such process redesign by leveraging capability library, which consists of all activities used in any process models in the repository, and inter-process relationship resolution patterns.

As such, we formulate our research questions which are classified into two main ideas as follows.

1. **Inter-process relationship fundamentals**

   (a) What type of inter-process relationship can be identified between business process models within a process repository?

   (b) How can we formally define each possible relationship which may exist between process designs?

   (c) How can we allow process analysts to be able to establish any normative inter-process relationship within a process repository?

2. **Process change propagation mechanism**

   (a) How can we define a process ecosystem in a process repository with respect to propagating process changes?

   (b) What kind of inter-process relationship resolution patterns can be constructed in dealing with such change propagation?

   (c) What kind of algorithms should we develop in order to propagating process changes in a semi-automatic manner?

   (d) How can we evaluate such algorithms with regard their performance and correctness?

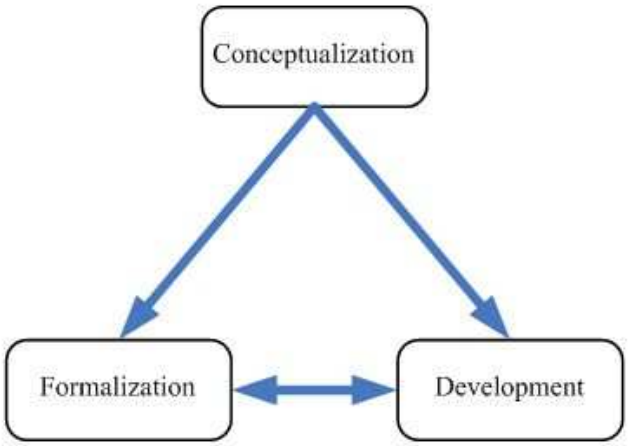


Figure 1.1: The SERM framework, adopted from [21]

1.3 Research methodology

This dissertation contributes to Software Engineering research through accommodating the Software Engineering Research Methodology (SERM) described in [21]. It involves three aspects of SERM: conceptual, formal and developmental. Figure 1.1 depicts the SERM framework which is used in this research. According to SERM, it is noted that either the formal and developmental aspects can be attempted once the idea of a research has been properly constructed. Based on a research viewpoint in SERM, it is not sufficient to perform formalization and/or development without any conceptualization. As such, in order to qualify a research to be rigorous in SERM, such a research must address issues in at least two of the three aspects, e.g. conceptual and formal, conceptual and developmental.

Conceptualization is the fundamental process performed in SERM. During this process, the theoretical grounding for the needs and requirements of the research effort are defined [21]. The following factors determine the success of this conceptualization [21]: (i) the clarity with which the researcher details the problem and grounds it with the theoretical constructs and (ii) the understandability and translatability of the concepts. In this dissertation, we identify problems in managing changes in busi-

ness process repositories and ground them with the theoretical constructs. These problems include identification of inter-process relationships, construction of semantically effect annotated process models, identification of inter-process relationships resolution patterns, formulation of process redesign and process changes propagation procedures. Based on the identified problems, we analyze a potential approach, i.e. process ecosystem, which can be constructed in order to solve problems in process changes management.

Formalization addresses the needs identified in the conceptualization through mathematical or logic-based description. Within this process, the research ideas can also be shaped and generalized [21]. Research formalization can also use application of formal techniques (e.g. optimization algorithms), mathematical modeling and evaluation, math/logic proofs, analytical modeling, computational analysis. In this dissertation, we formalize process ecosystem approach, i.e. using mathematical modeling, including inter-process relationships, process ecosystem, capability library, resolution patterns of the inter-process relationship violations and process redesign and process changes propagation algorithms. Using such formalization, we clearly describe our process ecosystem approach in dealing with process changes in a process repository.

Development focuses on developing a system to demonstrate the validity of the proposed solution [21]. Prototyping is the principal activity in this process such that we can study the performance of the developed system within a controlled environment. In this dissertation, we develop process ecosystem framework as the solution for problems identified in dealing with process changes management in a process repository. We then evaluate such process ecosystem framework to study its performance and accuracy.

Figure 1.2 illustrates the overlapping research areas of the three aspects in SERM [21]. According to the intensity of each aspect involving in such overlapping areas, we can categorize six research types in SERM, denoted in A, B, C, D, E and F types, in

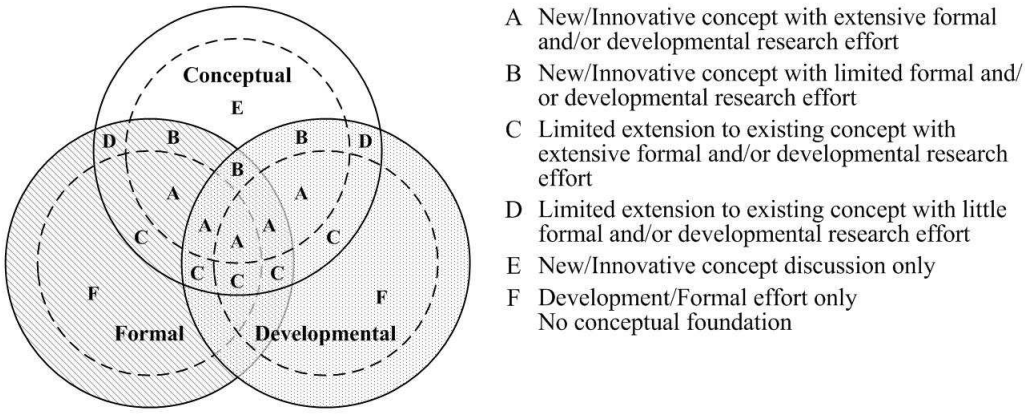


Figure 1.2: The SERM research types, adopted from [21]

which the highest and the lowest ranks are A and F, respectively. In this dissertation, we perform a research using these three aspects where our conceptualization becomes the fundamental aspect for doing the remaining aspects. As extensively described later, our process ecosystem concept obviously becomes a new approach in the BPM research area. In addition, the formalization and development of such process ecosystem extend the existing framework in managing business process changes in a process repository. As such, we argue that our research can be positioned in A type research in SERM.

1.4 Research contributions

Our research presented in this dissertation has made a number of reasonable contributions. We briefly outline such contributions compared to the existing similar contributions, if any, resulted from related work as follows.

- **Inter-process relationship formalization and establishment**

  We have considered and formalized three relationships which may exist between processes in a process repository, i.e. *part-whole*, *inter-operation* and *generalization-specialization*, based upon semantic effect annotation of process models. This formalization allows us to leverage a tool to identify any relation-

ship between processes. This contribution corresponds to our research questions 1(a) and (b). Further, we have implemented a tool to suggest process analysts with any relationship which may exist between processes based on the aforementioned types. This last contribution corresponds to our research question 1(c). Process analysts can further validate any suggested relationship type in order to establish a normative relationship between process designs in a process repository. Such establishment will take an important part in our process changes propagation approach.

To the best of our knowledge, the classification and formalization of inter-process relationships took less attention in the past, while, a machinery identification of such relationships has not yet been proposed. We, however, found similar ideas with some researches in classifying such relationships. Our part-whole relationship has the same idea with the uses-parts relationship described in [40]. While, our inter-operation relationship can be considerably compared to the interoperability between processes or workflows concepts described in [32, 59]. Finally, our generalization-specialization refers to the similar idea described in [40, 60]. In these related researches, we realize that only workflows inheritance has been formally defined, as described in [60]. Section 3.1.3 presents a detailed discussion of our inter-process relationships and their respective related work.

- **Process ecosystem formalization and establishment**

  We have proposed our process ecosystem concept which is formally defined with respect to the changes management in business process repositories, which relates to our research question 2(a). This contribution becomes a key concept in propagating process changes within a process repository.

- **Process changes taxonomy identification and resolution patterns spec-**

ification

With respect to our inter-process relationship constraints, we have identified a taxonomy of process changes that can violate such constraints, i.e. 11 types of process changes. In order to resolve such violations, we also have specified 7 resolution patterns based on different relationship types. These contributions deals with research question 2(b).

To the best of our understanding, there exists a similar work in identifying process changes taxonomy described in [67]. They propose a number of change patterns and change support features to foster the systematic comparison of existing process management technology with respect to process change support. We propose our process changes taxonomy and the corresponding resolution patterns which are inspired by their work. Our patterns are directed by a set of basic change operations using semantic effect analysis according to three common relationship types, i.e. part-whole, inter-operation and generalization-specialization. We believe that these resolution patterns become the only one proposed approach to resolve any relationship violation occurred among semantically effect annotated process models.

- **Process redesign algorithm development**

  We have developed process redesign framework to typically implement our resolution patterns, which corresponds to our research question 2(c). As such, our process changes propagation can be performed in a semi-automatic fashion. Such framework deals with the resolution patterns required in resolving any violation in part-whole and generalization-specialization relationships. As process analyst involvement is intensively required to resolve any violation in inter-operation relationship, our process redesign framework, however, does not concern such resolution.

- **Process change propagation algorithm development and evaluation**

  We have developed our process change propagation framework by leveraging constraint satisfaction problem (CSP) approach, which relates to the research question 2(c). This requires transformation of our process ecosystems into constraint networks. Further, we have evaluated such propagation framework using medium-sized process repositories, which deals with the research question 2(d).

  There exist approaches dealing with change propagation used in service computing and BPM (see, e.g. [66, 69]). They, however, only concern a pair of business artifacts. To the best of our knowledge, there exists little work on change propagation in process model collections, as can be seen in [16]. This work is a closely related work to our proposed framework, specifically in dealing with the specialization-generalization relationship. Section 2.3 provides a detailed discussion of the related work in change propagation.

1.5 Dissertation structure

The remainder of this dissertation is organized as follows.

- **Chapter 2: Background**

  This chapter provides a background of our discussions appear in this dissertation including an introduction to business process management, brief discussions on business process repository, change propagation frameworks and constraint networks.

- **Chapter 3: Process ecosystem**

  This chapter introduces our process ecosystem as a view of a collection of interrelated process models. This includes introduction of our inter-process relationship

approach including its taxonomy and respective formalizations, establishment and maintenance. We also present the notions of process ecosystem including its definition and formalization as well as its representation. Further, we overview the architecture of the process ecosystem framework as the consolidated framework of business process changes management concepts introduced across this dissertation.

- **Chapter 4: Managing change in process ecosystem**

  This chapter proposes our approach in managing change in process ecosystem due to process change requirement. Our discussion will include formalization, establishment and maintenance of capability library. We also describe process changes taxonomy and resolution patterns. In addition, we propose our process redesign mechanism and constraint network, which are leveraged in our process changes propagation.

- **Chapter 5: Implementation**

  This chapter describes the implementation of our proposed framework in dealing with process changes propagation within a collection of interrelated processes.

- **Chapter 6: Evaluation**

  This chapter presents the evaluation of our implemented framework including the pre-evaluation state, experiment set up and execution procedure, experiment results, post-evaluation state and analysis.

- **Chapter 7: Conclusion and future work**

  This chapter draws conclusion with respect to our research questions as well as our future work.

Chapter 2

Background

This chapter provides a background of discussions appear in this dissertation. This includes an introduction to BPM as well as brief discussions on business process repository, change propagation frameworks and constraint networks. We draw a summary of our discussions at the end of this chapter.

More specifically, in Section 2.1 we discuss the importance of BPM in dealing with a collection of business process models within an organization including the coverage area of BPM with respect to the process life cycle. We are interested in highlighting business process modeling as an important part of BPM in which we introduce semantic effect annotation on process models. Such annotation plays a fundamental part in our framework discussed in this dissertation.

We also discuss management of business process repository in Section 2.2 for maintaining process models. We surveyed existing process repositories as well as available corresponding requirements. In regard these requirements, we also introduce our own requirements of process repository for which our framework to be formulated and developed.

In addition, in Section 2.3 we overview change propagation frameworks which are investigated in many research areas. Finally, we introduce notions of constraint net-

works in Section 2.4 from which we will develop our process ecosystem concepts.

2.1 Business process management

2.1.1 Basic notions of business process management

There exist many different definitions of BPM being used in either business practices or business researches. For example, one may define it includes concepts, methods and techniques to support the design, administration, configuration, enactment and analysis of business processes [70]. While, other defines BPM as supporting business processes using methods, techniques and software to design, enact, control and analyze operational processes involving humans, organizations, applications, documents and other sources of information [71]. Despite differences in the definition, all of BPM definitions definitely focus on business processes. We consider a *business process* consists of a set of activities that are performed in coordination in an organizational and technical environment [70]. As such, explicit representations of business processes with their own set of activities and related execution constraints become fundamental in BPM. Furthermore, as business processes are operational processes [71], these processes represent the operations an organization performs. In order to represent a business process, we model it graphically into a *business process model* which consists of a set of activity models and their respective execution constraints. Once a business process has been explicitly defined, it can be subject to enact, analyze and improve with respect to its life cycle.

We may consider a business process life cycle depicted in Figure 2.1 [70]. This life cycle describes phases to support the operational of business processes: design and analysis; configuration; enactment; evaluation; administration and stakeholders. These phases are related to each other. Design and analysis phase is the phase where

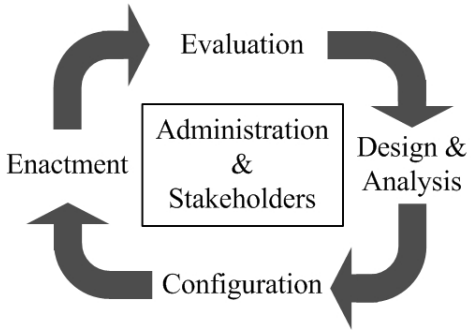


Figure 2.1: Business process life cycle, adopted from [70]

we conduct survey on the business processes and their related organizational and environment information. We, then, identify and represent them into business process models based on the survey results as well as validate them before passing into the next phase. In the configuration phase, we implement business process model through either using or not using a dedicated business process management software (BPMS). The former configuration imposes such software system to be configured with respect to the business processes to be controlled and their related organizational environments, e.g. resources, roles. While, the latter configuration requires a set of policies and procedures, from which business process executors in the organization must refer to, for dealing with the enactment phase. The enactment phase is the phase where business process instances can be enacted. This phase comprehends the run time of business processes and actively controls their instances. In the evaluation phase, we evaluate business processes with respect to their design and implementation for further improvements. The execution logs are the important information for evaluating business processes in their run time. Finally, the administration and stakeholder phase deals with organizing and maintaining any artifacts resulted across phases which previously described. This also includes maintaining business process stakeholders such as process designer, business engineer and process owner.

Business process modeling, in the design and analysis phase, plays an important

part in BPM since business process life cycle will rely on it. The accurate and valid business process models will be valuable assets for the organization. However, there exist a number of business process modeling techniques which can be used by an organization which may differ from other organizations. These techniques will be presented in Section 2.1.2.

In fact, however, the terms 'business process management', which is abbreviated to BPM, and 'business process management software', which is abbreviated to BPMS, can be confusing [74]. The first is a management discipline, whereas the second refers to a technology to implement such discipline. Some authors may use 'business process management system' term to refer to BPMS. In addition, implementation of such BPM disciplines results a variety of BPMS with their own approaches such as workflow management (WFM), case handling (CH), enterprise application integration (EAI), enterprise resource planning (ERP) and customer relationship management (CRM) [71]. For example, one may leverage CH [61] for supporting flexible and knowledge intensive business processes, which relies on what can be done in achieving business goals. Different to WFM, this approach allows a respective role or actor involved in a certain process execution to take additional activities which are not specified in the process definition due to a typical data occurs in the current context. For instance, a physician cannot request a blood test if the medical protocol does not specify such a test [61] even it is strongly required with respect to his or her expertise based on medical record of the current patient.

2.1.2 Business process modeling

As BPM relies on the representation of business processes, business process modeling and its resulting process models become critical parts in dealing with BPM. A model, in general, is an object created to represent something else for better understanding,

e.g. a globe is a model for better understanding of our planet Earth. In this context, the purpose of creating a model of something is to better understand about something using (often) a visual or graphical representation. In a similar vein, process models are specifically created to allow business process stakeholders to communicate about the structure of such processes in a more efficient and effective way. In addition, these process models become business artifacts to be further analyzed and improved in order to maintain the competitiveness of an organization.

Business process model, however, should not be confused with business process instance. The former stays at the business process conceptual level, while, the latter corresponds to the execution of a process at the business process implementation level. In this context, a business process model represents a blue print of a set of business process instances with similar structure [70]. It is composed of two main components, i.e. activity models and execution constraints. Analogous to business process model, an activity model also represents a set of activity instances. We need to further look at an activity definition, i.e. an activity represents the work performed within a process. Activities are either atomic (called a *task*, i.e. they are at the lowest level of detail presented in the diagram and can not be further broken down) or compound (called a *sub-process*, i.e. they can be broken down to see another level of process below) [72]. In addition, execution constraints describe how such activities should be performed in a process model. These constraints can specify the ordering and conditional execution of the activities within a business process.

We may consider Figure 2.2 to illustrate an example of business process model. This process model represents a *Management of patients on arrival* process in a neurosurgical ward of an hospital, which was originally represented in event-driven process chains (EPC) [5]. We transformed such an original model into a model represented in business process model and notation (BPMN). As can be seen, the neurosurgeron

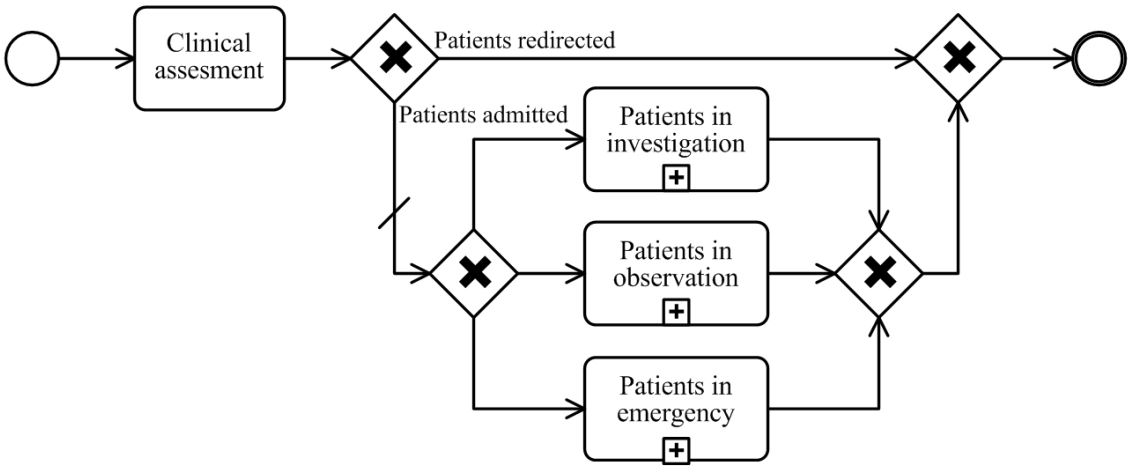


Figure 2.2: *Management of patients on arrival* process, transformed from [5]

makes a preliminary assessment of the patient's clinical condition and relies on such assessment result to recommend one of the following actions: keeping patients in observation (i.e. denoted by sub-process *Patients in observation*), keeping patients in further investigation (i.e. denoted by sub-process *Patients in investigation*), keeping patients in emergency (i.e. denoted by sub-process *Patients in emergency*) or redirecting patients to other destinations.

Representing a business process structure requires a systematic business process modeling framework. We, however, realize that there exist various distinct business process modeling frameworks recognized by both business practitioners and business researchers, i.e. petri nets, workflow nets, yet another workflow language (YAWL)[1], graph-based workflow language, EPC, BPMN[2], web service business process execution language (WS-BPEL). A recent business process modeling survey, however, reported that the most prominent business process modeling framework among the others is BPMN [23]. As such, BPMN becomes the de facto standard for business process modeling. Therefore, we are interested in describing BPMN in more detail below. In

[1]YAWL homepage `http://yawlfoundation.org/`
[2]BPMN homepage `http://www.bpmn.org/`

addition, such description becomes more important since our proposed approach will rely on business process models represented in BPMN.

BPMN

BPMN was originally developed by business process management initiative (BPMI) which was then folded into object management group (OMG)[3] [72]. It was developed by a large number of vendors to consolidate the underlying principles of business process modeling. BPMN 1.0 specification was released in May 2004 and then adopted as an OMG standard in February 2006. Since then, BPMN 1.1 and 1.2 specifications were released into public accordingly for improvements. Recently, as of March 2011, the current version of BPMN is 2.0.

BPMN defines a business process diagram (BPD) which is based on a flowcharting technique tailored for creating graphical models of operational business processes. A business process model, then, is a network of graphical objects, i.e. activities and flow controls that define their order of performance [73]. A BPD is made up of a set of graphical elements. These elements enable the easy development of simple diagrams that will look familiar to the most business analysts (e.g. flowchart diagram). The elements are chosen to be distinguishable from each other and to utilize shapes that are familiar to the most modelers, e.g. activities are rectangles and decisions are diamonds. Basically, BPMN has four categories of elements, i.e. flow objects, connecting objects, swimlanes and artifacts, as shown in Figure 2.3.

Flow objects are the main parts in building process model in BPMN which consist of three core elements, i.e. event, activity and gateway. An event is represented by a circle denoting something that happens. It can be start, intermediate or end events. An activity represents a kind of work that must be done within a business process. It can be task or sub-process. The sub-process is distinguished from the task by a small

[3]OMG homepage `http://www.omg.org/`

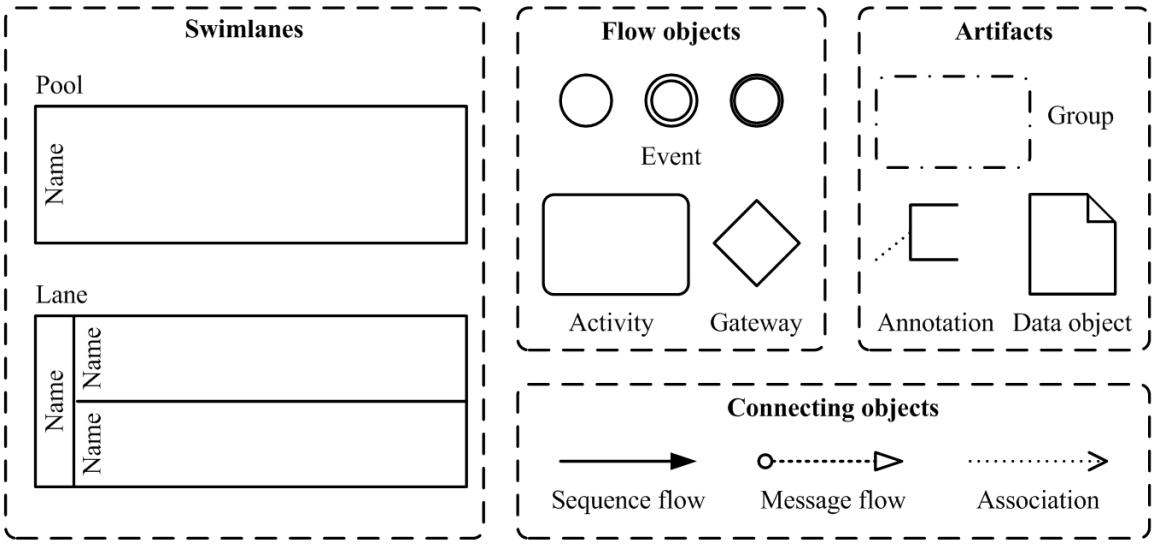


Figure 2.3: BPMN categories of elements

plus sign at the bottom center of the shape. Gateway is used to determine forking and merging behavior of the sequence flow in the process model which is specified by its internal marker, e.g. parallel, exclusive, inclusive.

Connecting objects are used to connect flow objects to create basic structure of a business process model. These include sequence flow, message flow and association. Sequence flow is used to specify the execution order of flow objects. A diagonal slash at a sequence flow indicates that such sequence flow is the default flow of a decision gateway. Message flow is used to represent message exchanges between organization boundaries (i.e. between pools). Association is used to associate an artifact to flow objects.

Swimlanes are used to organize activities into different visual categories with respect to separate functional capabilities and responsibilities within a business process model. BPMN has two types of swimlanes, i.e. pool and lane. Pool is used to represent major participants in a business process. These participants can be different organizations with different responsibilities. Lane is used to organize activities within a pool with respect to different roles. One pool may have many lanes.

Artifacts provide basic features to allow process designer to enrich process model with additional information. These include data object, group and annotation. Data object represents data which is required or produced by an activity in which it is connected to such activity using association. Group is used to group elements within a business process from which process analysis can be performed. It does not affect to the flow of the business process. Annotation is used to give a textual description associated to any object in the process model in order to make it more understandable.

In spite of its popularity over the other process modeling frameworks, BPMN, however, still has some deficiencies as reported in a recent study [46]. As such, we need to be aware of using BPMN. First, it does not adequately support for specifying business rules. Second, BPMN lacks in supporting business process decomposition. Third, ambiguity occurs when using pools and lanes for modeling organizations. Fourth, many available symbols are useless, e.g. group, off-page connector. Finally, abundance of different event constructs is not helpful to the process designer. In addition, to the best of our knowledge, BPMN and other available process modeling frameworks do not provide any facility to describe the semantics of a business process in term of its effects or outcomes. One may ask a question *what would the effects of the process be if it were to be executed up to this point?* These lacks open many opportunities for developing new approaches in order to resolve such problems. In this context, our research also aims to make significant contributions to such resolution by leveraging our concept in annotating business process models with semantic effect as well as specifying relationships among them in which process decomposition becomes feasible.

2.1.3 Semantically effect annotated process model

An effect annotation relates to a particular result or outcome to an activity in a process model [32]. An activity represents the work performed within a process. Activities are

either atomic (called a *task*) or compound (called a *sub-process*) [72]. In an annotated BPMN process model, as our approach relies on, we annotate each activity with its (immediate) effects. As an activity can be either task or sub-process, we realize that the immediate effects of an activity (i.e. sub-process) may be non-deterministic, since there may exist many alternative paths in such immediate effects produced by such activity execution. As such, we generally define the immediate effects of an activity t, i.e. denoted as e_t, as the context-independent immediate results or outcomes of executing t (in any process) consisting a set of alternative *effect scenarios* $\{es_1, \ldots, es_n\}$ based on the $1, \ldots, n$ alternative paths in such execution. We represent such outcomes in each effect scenario within the immediate effects of an activity in a conjunctive normal form (CNF) allowing us to describe such effects as a set of outcome clauses. In this context, the outcomes characteristics of task and sub-process may be different. A task has deterministic outcomes during its execution since its immediate effects have only one effect scenario. While, a sub-process may result non-deterministic outcomes in its execution since its immediate effects may have multiple effect scenarios. We shall leverage the ProcessSEER [26] tool to annotate each activity in a process model with its semantic effects.

Such annotation allows us to determine, at the design time, the effects of the process if it were to be executed up to a certain point in the model. These effects are necessarily non-deterministic, since a process might have taken one of many possible alternative paths through a process design to get to that point. The non-determinism also arises from the fact that the effects of certain process steps might undo the effects of prior steps - the inconsistencies that result in the snapshot of the domain that we seek to maintain might be resolved in multiple alternative ways (a large body of work in the reasoning about action community addresses this problem). The answer to the question is therefore provided via a set of effect scenarios, any one of which might

eventuate in a process instance.

We construct a *pair-wise effect accumulation* procedure in order to specify the cumulative effect of a given two contiguous activities after executing them in sequence. This procedure allows us to contextualize these effects by propagating them through a process model (specified in BPMN in the current instance) to determine the cumulative effect scenarios at the end of each activity. We use formal machinery (theorem-provers) to compute such cumulative effects. In such semantic effect annotation, we, however, only deal with a restricted subset of BPMN modeling framework, i.e. start or end empty events, XOR and AND gateways, task, sub-process and message flow.

Let t_i and t_j be an ordered pair of activities connected by a sequence flow such that t_i precedes t_j. Let $e_i = \{c_{i1}, \ldots, c_{im}\}$ and $e_j = \{c_{j1}, \ldots, c_{jn}\}$ be the effect annotations of t_i and t_j, respectively. If $e_i \cup e_j$ is consistent, then the resulting cumulative effect is $e_i \cup e_j$. Otherwise, we define $e_i' = \{c_k\}$ where $c_k \in e_i$ and $\{c_k\} \cup e_j$ *is consistent*, and the resulting cumulative effect to be $e_i' \cup e_j$. We shall use $ACC\,(e_p, e_q)$ to denote the result of pair-wise effect accumulation of two contiguous activities t_p and t_q with the immediate effects e_p and e_q, respectively. We denote $CE\,(P, t)$ as the cumulative effects of execution of process P at activity t. $CE\,(P, t)$ is defined as a set of alternative *effect scenarios* $\{es_{t1}, \ldots, es_{tm}\}$ based on the $1, \ldots, m$ alternative paths reaching t. For a complete process execution, we use $acc\,(P)$ to denote the end cumulative effects. Note that each of $acc\,(P)$ or $CE\,(P, t)$ is a set of effect scenarios. Each effect scenario is represented as a set of clauses and will be viewed, implicitly, as their conjunction.

We apply the pair-wise effect accumulation procedure to accumulate the effects of contiguous pairs of activities connected via control flow links. In such procedure, we traverse all activities from the left to the right within a participant lane. As splits occur, there is no change on such procedure. We perform differently such procedure due to joins. We recognize that alternative effect scenarios are introduced by AND-joins

or XOR-joins or OR-joins. We briefly explain a procedure which must be executed in order to accumulate effects due to joins by using 2-way joins as an example. We can generalize such procedure to handle n-way joins. Note that we do not consider the possibility of a pair of effect scenarios in AND-joins being inconsistent, since this would only happen in the case of intrinsically and obviously erroneously constructed process models.

In the following, let t_p and t_q be two activities immediately preceding a join. Let their cumulative effect annotations be $E_p = \{es_{p1}, \ldots, es_{pm}\}$ and $E_q = \{es_{q1}, \ldots, es_{qn}\}$, respectively. Let e be immediate effect and E be cumulative effect of an activity t immediately following the join.

For **AND-joins**, we define $E = \{ACC(es_{pi}, e) \cup ACC(es_{qj}, e)\}$ where $es_{pi} \in E_p$ and $es_{qj} \in E_q$. The result of effect accumulation in the setting described here is denoted by $ANDacc(E_p, E_q, e)$. For **XOR-joins**, we define $E = \{ACC(es_r, e)\}$ where $es_r \in E_p$ or $es_r \in E_2$. The result of effect accumulation in the setting described here is denoted by $XORacc(E_p, E_q, e)$. For **OR-joins**, the result of effect accumulation in such setting is denoted by $ORacc(E_p, E_q, e) = ANDacc(E_p, E_q, e) \cup XORacc(E_p, E_q, e)$.

Figure 2.4 exemplifies a semantically effect annotated BPMN process model leveraged from the same business process illustrated in Figure 2.2. This model illustrates the immediate effect e_i of each activity t_i. Let p be patient to be observed and treated. For example, activity t_{13} has an immediate effect $e_{13} = observed(p)$ which depicts the outcomes of executing such activity. The cumulative effects of execution the process until t_{13} can be computed by accumulating the effects starting from t_{11} until t_{13}, i.e. $assessed(p) \wedge observed(p)$. Similarly, we can also compute for the other activities.

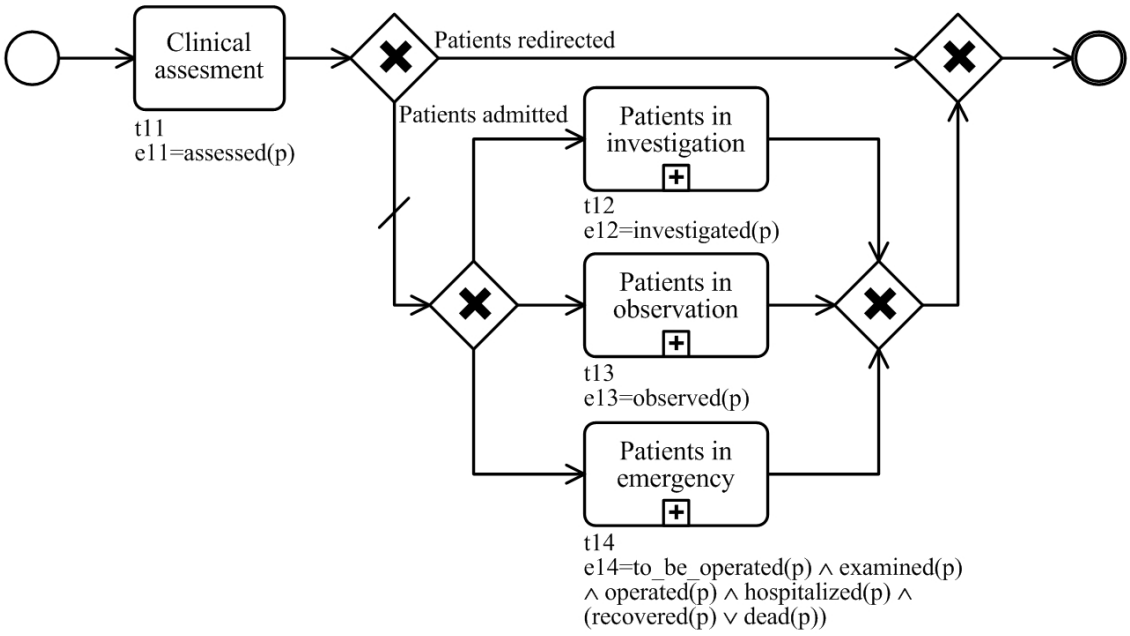


Figure 2.4: *Management of patients on arrival* process with semantic effect annotation

2.2 Business process repository management

This section presents a survey of existing elicited requirements for developing a business process repository as well as a survey of process repository frameworks which are developed and implemented in business practices. Our methodology in performing these surveys is as follows. We, firstly, analyze the available requirements of business process repository in Section 2.2.1 which mainly rely on studies described in [52, 53]. In regard to these requirements, we also propose our own process repository requirements to permit such repository to deal with process changes. We survey current business process repository frameworks in Section 2.2.2 to clearly understand their architectures and features. Finally, in Section 2.2.3 we present our survey results in a summary table as well as a discussion with respect to our proposed process repository requirements. In addition, we can further suggest any research opportunity to deal with the remaining gaps between the expected process repository capabilities and the existing ones.

2.2.1 Process repository requirements

In general, a repository is defined as a shared database of information about engineered artifacts produced or used by an enterprise [4]. We can consider such engineered artifacts include java programs, electrical circuit diagrams, business process diagrams. There must exist a repository manager providing basic services to the users dealing with such artifacts such as creating, searching, viewing, modifying and removing. Additional capabilities can be added values such as check out or check in, version control, configuration control, notification, context management and workflow control [4].

As process repository becomes a critical part in BPM due to increasing number of process models to manage, there exist increasing needs to stipulate how it should be developed and maintained in an organization with respect to its functional and non-functional aspects. Instead of viewing this repository only as a database to simply store our process definitions, we need to specify it more complex to allow us to deal with BPM. For example, we can consider process dependencies (both functional and non-functional) management, process searching and process change management to adequately support BPM. There, however, has been little work on specifying requirements for business process repository which may not be applicable to other repositories (see, e.g. [52, 53]). Specifying such requirements would drive development and maintenance of process repository such that it can adequately support BPM. To the best of our understanding, one approach specifies general process repository requirements [52], while another approach typically captures such requirements with respect to process reusability [53]. We, however, summarize these requirements as follows:

- **RQ01 - Extensibility**

  We should be able to add new process models and to modify existing process models in the repository.

- **RQ02 - Flexibility**

  We should be able to manage any variant of existing process models in the repository.

- **RQ03 - Openness**

  It should be able for any user, without prior legal permission, to deal with any process model in the repository.

- **RQ04 - Acceptance**

  We should be able to manage process models in the repository based on various business classification schemes.

- **RQ05 - Domain free**

  It should be able to store process models in the repository regardless of their domain.

- **RQ06 - Modeling language**

  It should be able to store process models in the repository in at least one process modeling language.

- **RQ07 - Representation**

  It should be able to represent process models in the repository in both graphical and textual forms.

- **RQ08 - Business model inclusion**

  It should be able to store both business and process models in the repository.

- **RQ09 - Multi level abstraction**

  We should be able to represent process models in the repository in different levels of abstraction.

- **RQ10 - Annotation**

  We should be able to annotate process models in the repository with information for further search, navigation and interpretation.

- **RQ11 - Navigation**

  It should be able to navigate, i.e. searching and locating, any process model in the repository.

These requirements would be acceptable for developing a process repository since some of them come from a scientific survey [53]. We, however, would argue that requirement RQ03 should not be the case since creating and maintaining a process model in the repository should be performed by an authorized user, who has adequate knowledge and skill, with respect to particular models. Further, viewing a process model with respect to a certain level of its abstraction relies on user's management level in an enterprise [45]. Hence, we argue that controlled access to process models in the repository should be taken into account.

In addition, these requirements may have been adopted, either partially or fully, in some of existing process repositories described later. We, however, are interested in specifying our own requirements with respect to process changes in order to contribute to developing process model repository requirements. As process changes become increasingly considered to achieve process improvement [22, 29], such changes should be properly managed in a collection of process models stored in process repository. Managing such changes is not obvious since these process models are often interdependent to each other such that changing one single process model may violate its normative relationships with the others. Preserving such relationship requires propagating such changes to the related processes. Obviously, this imposes the repository to have a process change propagation feature. Since this feature relies on the inter-process relationship descriptions, we would highlight process annotation requirement described

in [53]. However, rather than encoding such relationship descriptions manually by process analyst, we leverage semantic effect annotation of process models previously described in Section 2.1.3 to allow a machinery inference engine for establishing any relationship which may exist among processes. More specifically, we outline these additional requirements as follows:

- **RQ12 - Dependency representation**

  We should be able to represent all relationships which may exist among process models in the repository in either graphical or textual forms.

- **RQ13 - Dependency maintenance**

  It should be able to maintain all relationships which have been established between process models in the repository.

- **RQ14 - Change propagation**

  Due to process changes, it should be able to propagate such changes across entire process models in the repository with respect to their respective inter-process relationships.

- **RQ15 - Minimal change**

  We should be able to consider minimal changes in order to maintain inter-process relationships within process repository due to initial process changes.

2.2.2 Existing frameworks

We overview the existing frameworks of business process repository including their architectures and features below. In their architectures, we study any element and its respective roles from which the framework is constructed. While, in their features, we identify any functionality they offer to support BPM.

MIT process handbook[4] **[3, 40]**

This is an online process repository to facilitate users for efficiently retrieving and exploiting the relevant process knowledge. All process descriptions are represented in textual form and are internally stored in a relational database. In organizing processes, it classifies processes into two dimensions, i.e. *specialization*, which is used to differentiating a process into its different types, and *decomposition*, which is used to breaking a process into its different parts. It uses so-called *process compass* to navigate processes to the two dimensions. In process compass, decomposition refers to *uses-parts* relationship, while, specialization relates to *generalizations-specializations* relationship. For example, if we select *parts* in process compass, it will list all related processes which compose the current process being viewed. Further, its specialization falls into categories known as *bundles*, i.e. groups of related specializations which are classified based on *what* (refers to the thing) and *how* (refers to the way). For example, a process *Sell* can be specialized into bundle *Sell what?*, which may include processes *Sell service* and *Sell product*. In addition, it also supports text-based process search.

Oryx[5] **[13, 51]**

It provides accessible web-based modeling business processes which are represented in graphical form. Its architecture consists of front end and back end components. The former provides visible elements to the user in the web browser, while, the latter consists of a set of Java servlets providing functionality of Oryx. In particular, the Oryx Core, which is a part in the front end component, allows users to create, edit and view visual models within a browser with the help of stencil sets and plugins. The Oryx Core provides generic handling for such visualization. Each stencil set contains any kind of elements as well as design constraints specified for each modeling language. Hence,

[4]MIT process handbook homepage `http://process.mit.edu/`

[5]Oryx homepage `http://bpt.hpi.uni-potsdam.de/Oryx/WebHome`

it can support various process modeling languages including BPMN, EPC, workflow nets and petri nets. Process model is stored in its structure using resource description framework (RDF)[6] format. Oryx visualizes such process' structure according to the specified stencil set. In this context, process model transformation across different modeling languages can be performed. Further, Oryx also provides checking function for such created process model with respect to design constraints described in a stencil set. For querying a process, Oryx integrates BPMN-Q [50].

Advanced process model repository (APROMORE)[7] [36]

This is a service-oriented architecture (SOA)-based process repository and is deployed over the internet as an open source software-as-a-service (SaaS). Such deployment allows other developers or researchers to easily tap their new features into APROMORE in the service environment. It is built up on three layers, i.e. enterprise, intermediary and basic layers. The basic layer is the fundamental layer since it maintains business logic-centric and data-centric services. The business logic-centric services provide all related algorithms required to perform features offered by APROMORE. While, data-centric services deal with any underlying persistent data.

Basically, APROMORE is developed based on a large set of existing researches which shapes its features. These features can generally be classified into 4 distinct areas, i.e. evaluation, filtering, design and presentation. Evaluation concerns with analyzing process models to various aspects, i.e. correctness, performance and usability. Filtering provides functionalities to rank process models with respect to certain criteria including process similarity search, clone detection, pattern-based analysis and conformance analysis. Design focuses on creating and modifying process models including merge-driven creation, pattern-based design, individualization and extension

[6]RDF homepage `http://www.w3.org/RDF/`
[7]APROMORE homepage `http://apromore.org/`

control. Presentation refers to describing process models in certain formats in which process analyst prefer to, i.e. process abstraction, secondary notation and reporting. Each functionality leverages techniques from the existing researches. For example, correctness in the evaluation area refers to its properties like liveness, soundness or boundedness, which mainly are based on petri nets concepts discussed in [43, 58, 62]. Similarly, other ranges of researches contribute to the rest of functionalities.

It provides five data entities maintained in its basic layer, i.e. models, canonical models, annotations, patterns and relations archives, to support its advanced features. For example, relations archive maintains the relations between *canonical process formats* of different process models. Canonical format depicts the general structures of a process model appear in the most process modeling languages. As such, this format allows APROMORE to support different process modeling languages such as BPMN, EPC, YAWL, Protos, workflow nets and WS-BPEL.

Semantic business process repository (SBPR) [39]

SBPR is developed in the area of semantic business process management (SBPM) [24] for managing process models based on process ontologies and other useful ontologies, e.g. organizational ontology, semantic web service ontology. In this context, a process model is an instance of a process ontology, which is formalized in web service modeling language (WSML)-Flight[8]. Managing such ontological descriptions becomes fundamental part in SBPR since all related process model managements rely on them, e.g. storing and retrieving models. In particular, such ontological descriptions allow SBPR to further provide efficient querying and reasoning capabilities. SBPR classifies such query into *general query*, which can simply be performed based on the artifacts explicitly stored in the repository, and *semantic query*, which can only be performed by considering the available ontological descriptions of a process model. To support

[8]WSML homepage`http://www.w3.org/Submission/WSML/`

semantic query, it uses relational database management system (RDBMS) with an integrated rule inference system (IRIS)[9] inference engine for storage mechanism. To allow process modification by multiple users, SBPR provides locking mechanism, i.e. check in and check out functions, to guarantee a single process model is exclusively being updated by a single user at a time. This locking mechanism is only applied to process model, while its respective process ontology is not locked simultaneously. SBPR also maintains various versions of processes due to process modifications.

Integrated process management (IPM) [7]

This repository manages business process models systematically throughout all stages of their life cycles, i.e. process modeling, process pre-analysis, process enactment, process post-analysis and process evolution stages, which occur accordingly. As such, it is equipped with of a number of components which are classified into groups, i.e. process modeling and integration (PMI), process analysis and optimization (PAO), process automation and control (PAC), process-oriented integration (POI) and process knowledge management (PKM), with their own functionalities. For example, all components in PMI together perform integration between process definitions and related data using extensible markup language (XML)[10]. This integrated information will be transformed into a colored petri nets. It also consists of five distinct repositories to maintain all process-related information, i.e. process, process instance, process knowledge, process rule and process resource repositories. Each repository manages distinct information, e.g. process repository maintains information related to process definitions and analysis. Further, each stage in the life cycle may involve various different repositories and components, e.g. process modeling stage requires information from process, resource and rule repositories.

[9]IRIS homepage `http://www.iris-reasoner.org/`
[10]XML homepage `http://www.w3.org/XML/`

IPM supports its advanced features, i.e. process retrieval; process version, configuration and state managements. Its process retrieval allows user to search a process using IPM process query language (IPM-PQL)[11] and also to navigate such process using process classifications. Process state management function ensures that certain operations can only be performed on a process according to its state. In this context, IPM maintains three states: (i) *working*, i.e. process is being edited; (ii) *released*, i.e. process is completely edited; and (iii) *validated*, i.e. process is error-free guaranteed.

Integrated process repository (IPR) [65]

This repository is developed to be an integrator of multiple process reference models, e.g. MIT Process Handbook, SAP Process Reference Model and Oracle Best Practice Processes, which is based on Web 2.0 technologies. This integration is intended to better support process design in e-business by leveraging knowledge management theory, from which process design can be viewed as a process of creating process knowledge, applied to such multiple process reference models. It uses Nonaka's knowledge creation model [44] which consists of four intertwining modes, i.e. internalization, externalization, socialization and combination. As such, IPR starts its integration approach from multiple process reference sources with different process models. Experts perform the initial schema mapping and merging, and create a baseline integrated schema of process models. This baseline schema will evolve through contributions from user community, e.g. refinement, reclassification, modification. These all are performed in manual fashion. It utilizes a tool to enriching the schema. IPR stores process models in XML format. It also supports process classification.

BPEL repository [63]

It is built as an Eclipse plugin to store BPEL business processes and other related

[11] An XML-based query language for IPM

XML data. It supports the common Web service standards, e.g. BPEL, web services description language (WSDL)[12], XML schema. It uses XML format for storing business processes. It offers flexibility by using Eclipse modeling framework (EMF)[13] which can be used to automatically generate support for new XML schemas. By using EMF model, which is Java model, user will only deal with such model in which data serialization and deserialization are hidden from the user. For querying a process, it uses object-oriented query language OCL to query XML data represented in EMF objects. In this context, user can navigate through the data model and follow all associated information of the model using OCL. Currently, it implements two OCL engine, i.e. IBM[14] and Kent[15] OCL engines, with possible extension to use other OCL engines.

Business process characterizing model (BPCM) repository [18]

This repository is built in a three-layer architecture consisting presentation, repository management and storage layers. It is developed for enabling users to get benefit from knowledge extracted from business process models. As such, this repository does not maintain process models, rather, it organizes process knowledge represented in BPCM language using XML format. BPCM language consists of a set of elements, i.e. process, resource, actor, context, business domain, goal, process type and version. It allows users to include business goals in a process description. Basically, it stores all process knowledge in XML such that its interoperability can be leveraged. Further, it provides integration feature to connect with external tools. For example, a user can open an external process modeling tool (e.g. BPMN modeling tools) from its presentation layer in order to visualize any business process characterized in the BPCM repository.

[12]WSDL homepage `http://www.w3.org/TR/wsdl`
[13]EMF homepage `http://www.eclipse.org/modeling/emf/`
[14]IBM OCL homepage `http://www.ibm.com/software/ad/ocl`
[15]Kent OCL homepage `https://www.cs.kent.ac.uk/projects/ocl/`

Standard features of process repository, i.e. create, retrieve, update, delete (CRUD), are provided in the repository as well as process search.

RepoX [55]

It is an XML-based process repository for managing XML-based metadata containing the definition of workflow processes. It maintains these metadata in XML documents. It uses a client-server architecture with an object-relational database at the back end. Java remote method invocation (RMI)[16] is used to provide communication between client and server sides. It stores the XML documents in the database using character large object (CLOB) supported by the object-relational database. It allows multiple users to modify worfklow process by check in and check out mechanism. Further, it supports version and configuration management of workflow processes as well as process search.

Distributed repository [38]

This is built using SOA architecture which consists of three layers, i.e. presentation, service and data layers. Such architecture enables organizations to access a global process repository for managing collaborative business processes (CBPs). These CBPs depicts cross-organizational collaboration involving multiple entities and are not directly executable (i.e. representing conceptual processes). Implementation of a CBP requires definition of integrated business process (IBP) in each organization. Therefore, it consists of two repositories, i.e. global and local repositories. The former is public access to maintain all CBP models which can be shared to related organizations. The latter is private to each organization to maintain all IBP models. This repository maintains the consistency (i.e. consistent to respective CBPs) and interop-

[16]RMI homepage `http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html`

erability (i.e. among IBPs) of all IBP models. It provides basic services for dealing with CBPs and IBPs, i.e. CRUD, integration used for integrating external tools, access management, check in and check out, notification, life cycle management, version and configuration managements. It, however, does not support process modeling, i.e. create and edit. We have to use external process modeling tools to do such functions to be further uploaded to or downloaded from the repository. Further, it provides specific services to the global repository, e.g. organization management for managing organizations registered in the repository, as well as to the local repository, e.g. IBP model consistency checking for checking consistency between a IBP and its respective CBP.

2.2.3 Discussion

We summarize our process repository survey results in Table 2.1 with respect to the process repository requirements previously described in Section 2.2.1. According to the results, all of the existing process repository frameworks do not include business models, e.g. business goals, either graphical or textual representation in their approaches. This becomes difficult for an organization to validate whether its business processes are in line with its business goals or business strategies. As business process alignment becomes important in BPM [48], such business inclusion should be taken into account in developing process repository. Additionally, all process repository requirements related to inter-process relationships and process changes management (i.e. RQ12-RQ15) have not been considered by all existing frameworks. Only MIT process handbook implements classification of inter-process relationships (i.e. RQ12) using its process compass. These gaps open very challenging researches in the area of process repository management.

We, however, realize that the existing process repository frameworks have their

Table 2.1: Business process repository frameworks survey with respect to the repository requirements

Frameworks	**RQ01**	**RQ02**	**RQ03**	**RQ04**	**RQ05**	**RQ06**	**RQ07**	**RQ08**	**RQ09**	**RQ10**	**RQ11**	**RQ12**	**RQ13**	**RQ14**	**RQ15**
MIT proc. handbook	Y	N	Y	Y	Y	N	T	N	N	Y	Y	Y	N	N	N
Oryx	Y	N	Y	N	Y	Y	G	N	N	N	Y	N	N	N	N
APROMORE	Y	Y	Y	Y	Y	Y	G	N	Y	N	Y	N	N	N	N
SBPR	Y	Y	Y	Y	Y	Y	G	N	N	N	Y	N	N	N	N
IPM	Y	Y	Y	Y	Y	Y	GT	N	N	N	Y	N	N	N	N
IPR	Y	N	Y	Y	Y	Y	?	N	N	N	Y	N	N	N	N
BPEL repository	Y	N	N	N	Y	N	T	N	N	Y	Y	N	N	N	N
BPCM repository	Y	Y	N	N	Y	N	T	Y	N	Y	Y	N	N	N	N
RepoX	Y	Y	N	Y	Y	Y	G	N	N	N	Y	N	N	N	N
Distributed rep.	Y	Y	N	N	Y	Y	GT	N	Y	Y	Y	N	N	N	N

Remarks:
Y - satisfied, N - not satisfied, G - graphical, T - textual, ? - not specified

RQ01 - extensibility *RQ02 - flexibility* *RQ03 - openness* *RQ04 - acceptance*
RQ05 - domain free *RQ06 - modeling language* *RQ07 - representation* *RQ08 - business model inclusion*
RQ09 - multi level abstraction *RQ10 - annotation* *RQ11 - navigation* *RQ12 - dependency representation*
RQ13 - dependency maintenance *RQ14 - change propagation* *RQ15 - minimal change*

own typical advantages. MIT process handbook, which maintains over 5000 processes ranging from very specific domain (e.g. for university purchasing department) to very general process (e.g. for resource allocation and multi-criteria decision making) [31], provides good examples to any users from different organizations to learn how to organize processes which might be similar to their own. Oryx becomes an extensible process repository framework due to its plugin mechanism and stencil technology to allow us to extend its capabilities and to add modeling languages. The canonical format and SaaS deployment allow APROMORE to get progressive improvement in the future due to more possible contributors to take part in open source environment [68]. SBPR allows faster modeling by a fine-grained locking mechanism, i.e. permitting a user to lock elements of a process model (e.g. sub-process) for modification rather than process as a whole, such that many users can work together at various distinct elements of a complex process. IPM stores all information related to a process in XML format such that we can easily exchange them to external tools. Moreover, by maintaining process knowledge, rule and resource in the repository, IPM provides integrated environment to the user for defining a process.

In addition, IPR provides a rich baseline schema of process reference models due to its internalization coming from many process reference sources. BPEL repository provides flexibility to user through its EMF models for dealing with XML-based process representation in the repository. BPCM repository characterizes process models to ease their reutilization during process modeling. Finally, Distributed repository allows managing business processes across organizations which are registered in the repository to maintain consistency and interoperability between processes.

2.3 Change propagation frameworks

Change propagation approaches have been intensively investigated in many research areas. In software evolution or maintenance, Aryani *et al.* [1] propose an approach for analyzing impact of a change to a software system without technical knowledge of software engineering and access to the source code. They leverage conceptual coupling between software components as a measurement of commonality of domain information between them. In engineering management, Chua and Hossain [8] propose an integrated model comprising change propagation model and scheduling model for managing changes on a design project. This integrated model allows project managers to ascertain the overall impact on design completion and the additional effort required for redesign due to external changes. In software modeling, Dam and Winikoff [9] present an approach in dealing with change propagation of unified modeling language (UML) design models during their maintenance and evolution. They leverage object constraint language (OCL)[17] for specifying consistency constraints to allow automatically generating inconsistency resolutions for such models.

Recently, this change propagation approach has also been applied to service computing and BPM (see, e.g. [66, 69]). However, there exist a little work on change propagation in process model collections [15], as can be seen in [16]. Wang *et al.* [66] present analysis of dependencies between services and their supporting business processes. On the top of this analysis, they define change types and impact patterns which are used to examine the necessary change propagation occurring in business processes and services. Weidlich *et al.* [69] attempt to determine a change region in another model by exploiting the behavioral profile of corresponding activities due to a model change. Their behavioral profile relies on three relations, which are based on the notion of weak order, between nodes in a process graph. A process graph consists

[17]OCL Specification `http://www.omg.org/spec/OCL/2.3.1/`

of a set of nodes, which represent any activity, event and gateway in a process model, as well as a set of respective edges, which represent any sequence flow in the same process model. To the best of our understanding, these researches are only dealing with a pair of business artifacts. The closely related work to our proposed framework is done by Ekanayake *et al.* [16], which deals with processes in a collection. They propose change propagation based on the shared fragments between process models. To propagate changes, they develop a special data structure for storing these fragments and process models. Once changes are made to a fragment, we consider to change all processes which consist of such fragment. Specifically, this fragment-based approach would be closely related to one of our research interests, namely change propagation for the specialization-generalization relationship.

2.4 Constraint networks

It is obvious that many problems in the computation approach involve constraints which need to be satisfied. For example, in a course scheduling, we need to allocate a time slot in a week for every course while simultaneously satisfying the lecturing constraints. Such constraints impose a single lecture is not allowed to teach more than one course at the same time slot as well as each time slot must be adequate with respect to each course credit. We can also leverage a well-known constraint problem example, i.e. map-coloring problem illustrated in Figure 2.5 [12]. In this problem, we have to color the map with only four colors whereas no two adjacent regions may share the same color. In order to ease solving such problems, they are usually modeled in so-called *constraint networks*. Constraint networks was initially introduced in [42] to represent and handle problem in picture processing involving some constraints [12]. In such initial form, only binary constraints, i.e. defined on pairs of variables, were treated which were represented in full generality as binary relations. In this context,

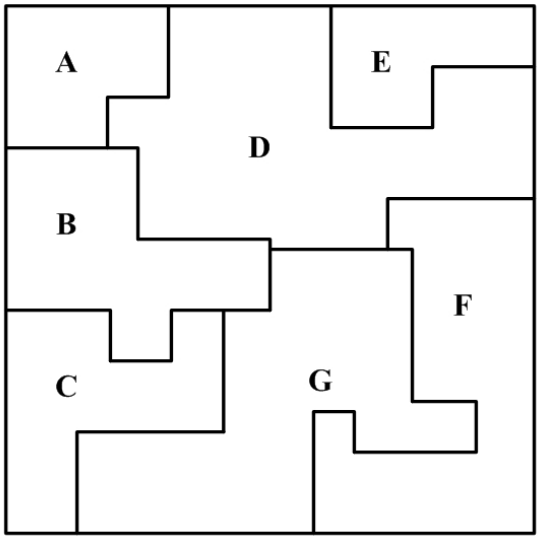


Figure 2.5: Map-coloring problem, adopted from [12]

networks of simultaneous binary relations depict constraints among more than two variables. As such, every set of constraints can be mapped into the binary case [12].

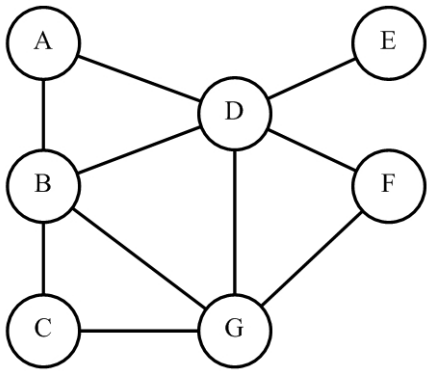


Figure 2.6: Constraint graph of the map-coloring problem

A constraint network CN consists of a finite set of *variables* $X = \{X_1, \ldots, Xn\}$, each associated with a *domain* of discrete values $D_1, \ldots, D_n$ and a set of *constraints* $C = \{C_1, \ldots, Ct\}$ [11]. Each domain D_i consists of a set of acceptable values $v_1, \ldots, v_k$ for variable $X_i \in X$. In this context, a constraint network can be viewed as a triple (X, D, C). Further, each constraint $C_i \in C$ consists of a pair $\langle scope, rel \rangle$, where *scope* is a tuple of variables that participate in the constraint and *rel* is a relation that

defines the values that these variables can take on [49]. For example, if two variables X_1 and X_2 have their own domains, then a constraint between the two such that the former has to be greater than the latter can be represented as $\langle (X_1, X_2), X_1 > X_2 \rangle$.

In regard map-coloring problem, we can capture this problem into a constraint network as follows. Regions can be mapped into variables, $X = \{A, B, C, D, E, F, G\}$. Domain of each variable is a set of colors $D_i = \{red, blue, green, white\}$. Based on the specified constraint, we have a set of constraints of adjacent regions $C = \{A \neq B, A \neq D, B \neq D, B \neq G, B \neq C, C \neq G, D \neq E, D \neq F, D \neq G, F \neq G\}$. In this context, we simply represent a constraint $\langle (X_i, X_j), X_i \neq X_j \rangle$ into $X_i \neq X_j$.

In order to adequately capture the structure of a constraint problem and its solution, we use so-called a *primal constraint graph* or just a *constraint graph* [12]. As a graph, a constraint graph consists of a set of nodes and a set of edges whereas each node represents a variable $X_i \in X$ and each edge connecting two nodes represents a constraint $C_i \in C$ whose variables are included in such constraint scope. As such, we can capture map-coloring problem illustrated in Figure 2.5 as a constraint graph depicted in Figure 2.6. The absence of an edge between two nodes represents that there is no constraint between the corresponding variables.

The constraint networks construct and its respective constraint graph become relevant to capture our problem in process changes propagation using process ecosystem views. Such views involve a number of process models which are related to each other through relationship constraints. We, however, will discuss it in more detail in Chapter 3.

2.5 Summary

In this chapter, we have provided our background of related knowledge including business process management in which a concept of semantic effect annotation on process

models has been discussed. Such annotation approach becomes fundamental in our inter-process relationships identification from which our process ecosystem can be constructed. We also overviewed current business process repository frameworks as well as the available process repository requirements. We have introduced our own process repository requirements with respect to process changes management in process repository from which we develop our process ecosystem approach. Further, existing change propagation frameworks have also been surveyed which are implemented in many areas. Finally, we discussed constraint networks for dealing with many constraint problems in computing approach. We believe that such constraint networks can be used to capture our problem in propagating changes in the process repository.

Chapter 3

Process ecosystem

This chapter introduces our process ecosystem concept for viewing a collection of process models in a business process repository. This includes our inter-process relationships management as well as the notions of process ecosystem itself. Our discussion starts with the introduction of our inter-process relationships management since we build our process ecosystem concept on the top of any relationship that can be identified between processes. We also introduce the architecture of the process ecosystem framework. At the end, we summarize the introduction to our process ecosystem.

In particular, we specify the taxonomy of our inter-process relationships in Section 3.1. We also describe the establishment and maintenance procedures for such relationships in this section which are fundamental in our process ecosystem discussion. Further, we discusses some issues with regard our inter-process relationships. In Section 3.2, we define, formalize and represent our process ecosystem. The architectural overview of the process ecoystem framework is presented in Section 3.3.

3.1 Inter-process relationship

3.1.1 Taxonomy

We now propose a taxonomy of relationships which may exist between business process models, which are classified into two categories: *functional dependencies* and *consistency links*. A functional dependency exists between a pair of processes when one process depends on the other process for realizing some of its functionalities. In other words, a process will not be able to achieve its goals without supports given by the others. In contrast, a consistency link exists between a pair of processes when both of them have intersecting parts represent the same functionality in which the outcomes (i.e. effects) of these parts are exactly the same. They, however, are functionally independent, i.e. one process does not require any support from the other.

In such categories, we now define three different types of inter-process relationship, namely *part-whole*, *inter-operation* and *generalization-specialization*. The first two fall in the functional dependencies category whereas the third is regarded to be in consistency links category. We formally define each of these relationship types using the semantic effect analysis on process models. We shall use $acc(P)$, $CE(P,t)$ and es in the following discussions, which have been previously introduced in Section 2.1.3.

Part-whole

A part-whole relationship exists between two processes when one process is required by the other to fulfill some of its functionalities. More specifically, there must be an activity in the 'whole' process representing the functionalities of the 'part' process, which is commonly referred to as a sub-process. Logically, there is an insertion of the functionalities of the 'part' into the 'whole'. For the sake of clarity, we, however, shall use the term 'main' process to refer to the 'whole' process of which the 'part' process

represents a sub-process expansion. Therefore, we first formally define the insertion of a process in another process described in Definition 1.

Definition 1. (Insertion point) *The insertion of process $P2$ in process $P1$ at activity t, $P1 \uparrow^{t} P2$, is a process design obtained by viewing $P2$ as the sub-process expansion of activity t in $P1$.*

Literally, the insertion of $P2$ at an activity t in $P1$ simply involves connecting the path entering t with the starting event of $P2$ and connecting the path leaving t with the end event of $P2$. Semantic effects can be applied to in this situation as follows. Let $T1 = \{t_{11}, t_{12}, \ldots, t_{1i}\}$ and $T2 = \{t_{21}, t_{22}, \ldots, t_{2j}\}$ be the set of consecutive activities of process models $P1$ and $P2$, respectively. Let $CE\,(P1, t_{1s})$ be the cumulative effects of process model $P1$ at the point of activity t_{1s} where $1 \leq s \leq i$. The cumulative effects computation involves a left-to-right pass of evaluating the activities within a process until the defined point of activity t_{1s}. Then, $CE\,(P1 \uparrow^{t_{1s}} P2, t_{1s})$ can be computed by replacing activity $t_{1s} \in T1$ with a set of activities within $P2$ through the following procedures:

1. accumulate the effects from activity t_{11} until activities t_{1s-1} within $P1$, where t_{1s-1} denotes all activities immediately precede activity t_{1s}, might be in parallel;

2. continue the effects accumulation involving all activities within $P2$ through passing from the most left activity t_{21} to the most right one t_{2j};

3. continue the accumulation through t_{1s+1} until t_{1i} within $P1$, where t_{1s+1} denotes all activities immediately succeed activity t_{1s}.

Using the definition of process insertion, we formally define the part-whole relationship described in Definition 2. This is a *context-dependent* part-whole relationship definition since it relies on the cumulative effects across all activities preceding t_i.

Definition 2. (Context-dependent part-whole) *Given process models* $P1$ *and* $P2$*,* $P2$ *is a direct part of* $P1$ *iff there exists an activity* t *in* $P1$ *such that* $CE\,(P1, t) = CE\,(P1 \uparrow^{t} P2, t)$.

Further, we also define a *context-independent* part-whole relationship based on the effect annotation of the insertion point activity which is independent from its process context, as described in Definition 3.

Definition 3. (Context-independent part-whole) *Given process models* $P1$ *and* $P2$*,* $P2$ *is a direct part of* $P1$ *iff there exists an activity* t *in* $P1$ *with immediate effects* $e_t = \{es_{t1}, \ldots, es_{tm}\}$ *such that* $\forall es_q \in acc(P2)$, $\exists es_{tp}$ *where* $es_{tp} = es_q$ *and* $\forall es_{tp}$, $\exists es_q \in acc(P2)$ *where* $es_q = es_{tp}$*, and* $1 \leq p \leq m$

Let us consider an example of part-whole relationship illustrated in Figures 3.1 and 3.2, which are denoted as $P1$ and $P2$, respectively. In this setting, the latter becomes a part process of the former with respect to both context-dependent and context-independent part-whole definitions. Such relationship is reflected by activity *Patients in emergency* (t_{14}) in $P1$ which is the abstract activity representing process $P2$. The immediate effect of such activity, i.e. $e_{14} = assessed\,(p) \wedge to\_be\_operated\,(p) \wedge examined\,(p) \wedge operated\,(p) \wedge hospitalized\,(p) \wedge (recovered\,(p) \vee dead\,(p))$, involves multiple effect scenarios represented by its disjunction symbol connecting the two last effect literals in which only one of them will happen at the same time. This yields the result of executing activity t_{14} in the main process is completely the result of executing the part process, and vise versa, as can be seen in our analysis below. As such, the insertion point here is at activity t_{14} in the main process $P1$.

We compute the cumulative effects of the main process at such point $CE\,(P1, t_{14}) = \{es_{141}, es_{142}\}$. In this setting, $es_{141} = assessed\,(p) \wedge to\_be\_operated\,(p) \wedge examined\,(p) \wedge operated\,(p) \wedge hospitalized\,(p) \wedge recovered\,(p)$; and $es_{142} = assessed\,(p) \wedge to\_be\_operated\,(p) \wedge examined\,(p) \wedge operated\,(p) \wedge hospitalized\,(p) \wedge dead\,(p)$. Then, let us compute the

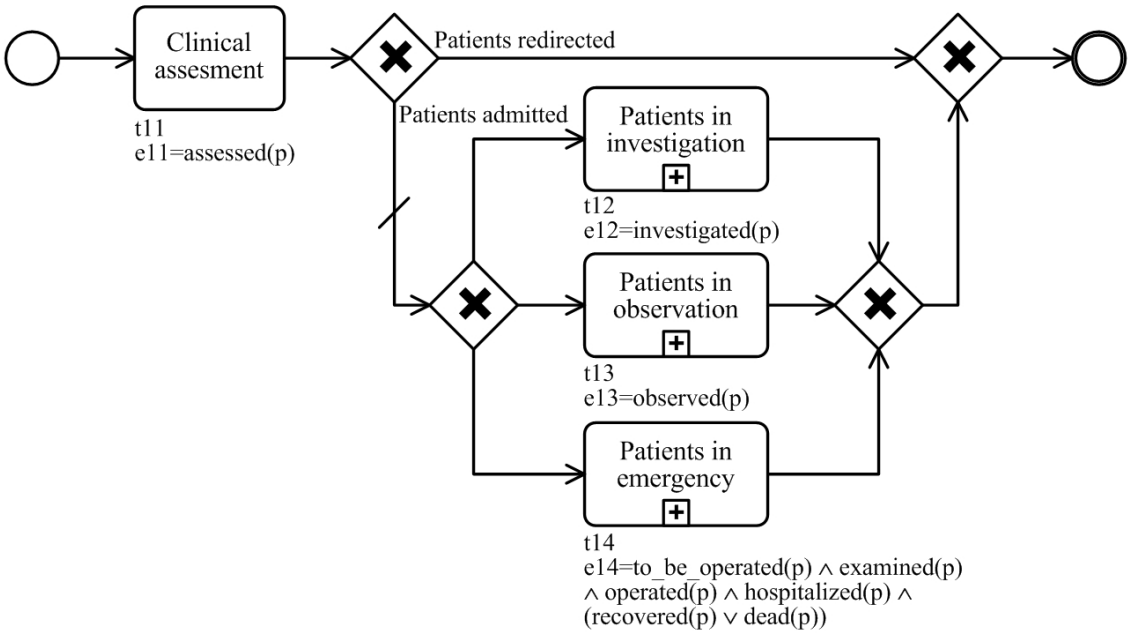


Figure 3.1: *Management of patients on arrival* process plays the 'main' process.

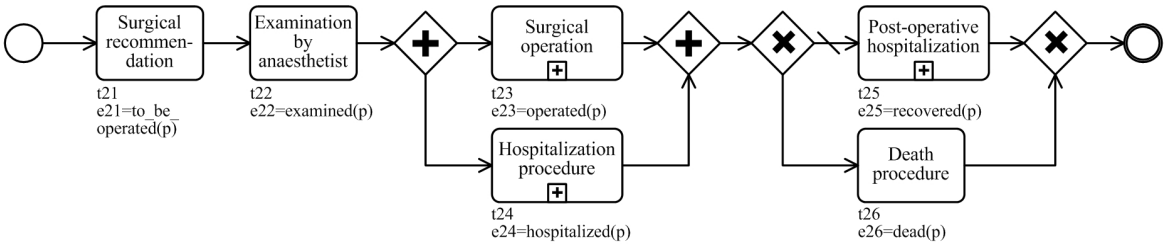


Figure 3.2: *Patients in emergency* process becomes the sub-process expansion of activity t_{14} of the process in Figure 3.1.

cumulative effects of the main process at t_{14} by inserting the part process into the main process $CE\,(P1 \uparrow^{t_{14}} P2, t_{14}) = \{es'_{141}, es'_{142}\}$. Here, $es'_{141} = assessed\,(p) \wedge to\_be\_operated\,(p) \wedge examined\,(p) \wedge operated\,(p) \wedge hospitalized\,(p) \wedge recovered\,(p)$; and $es'_{142} = assessed\,(p) \wedge to\_be\_operated\,(p) \wedge examined\,(p) \wedge operated\,(p) \wedge hospitalized\,(p) \wedge dead\,(p)$. As such, we can infer that $P2$ is the part process of $P1$ at the insertion point activity t_{14}, since $CE\,(P1, t_{14}) = CE\,(P1 \uparrow^{t_{14}} P2, t_{14})$ with respect to the context-dependent part-whole definition.

Further, the immediate effects of t_{14} involves two effect scenarios representing two possible different outcomes of its execution, i.e. $es_1 = to\_be\_operated\,(p) \wedge examined\,(p) \wedge operated\,(p) \wedge hospitalized\,(p) \wedge recovered\,(p)$; and $es_2 = to\_be\_operated\,(p) \wedge examined\,(p) \wedge operated\,(p) \wedge hospitalized\,(p) \wedge dead\,(p)$. While, the end cumulative effect of the part process is $acc(P2) = \{es_{21}, es_{22}\}$, where $es_{21} = to\_be\_operated\,(p) \wedge examined\,(p) \wedge operated\,(p) \wedge hospitalized\,(p) \wedge recovered\,(p)$; and $es_{22} = to\_be\_operated\,(p) \wedge examined\,(p) \wedge operated\,(p) \wedge hospitalized\,(p) \wedge dead\,(p)$. Since $es_1 \models es_{21}$ and $es_2 \models es_{22}$, we can infer that $P2$ is the part process of $P1$ at the insertion point activity t_{14} with respect to the context-independent part-whole definition.

Inter-operation

An inter-operation relationship exists between two processes when there is at least one message exchanged between them and there is no cumulative effects contradiction between activities involved in such exchanging messages. We formalize the definition of inter-operation relationship in Definition 4.

Definition 4. (Inter-operation) *Given process models $P1$ and $P2$, an inter-operation relationship exists between these processes including activities t_i and t_j iff the following two conditions hold:*

- *$\exists t_i$ in $P1$ $\exists t_j$ in $P2$ such that $t_i \rightharpoonup t_j$ denotes t_i sends a message to t_j, or in the reverse direction $t_j \rightharpoonup t_i$;*
- *Let $E_i = \{es_{i1}, es_{i2}, \ldots, es_{im}\}$ be cumulative effects of process $P1$ at activity t_i, i.e. $CE\,(P1, t_i)$, and $E_j = \{es_{j1}, es_{j2}, \ldots, es_{jn}\}$ be cumulative effects of process $P2$ at activity t_j, i.e. $CE\,(P2, t_j)$. Then, there is no contradiction between E_i and E_j for all $es_{ip} \in E_i$ and $es_{jq} \in E_j$ such that $es_{ip} \cup es_{jq} \vdash \perp$ does not hold, where $1 \leq p \leq m$ and $1 \leq q \leq n$.*

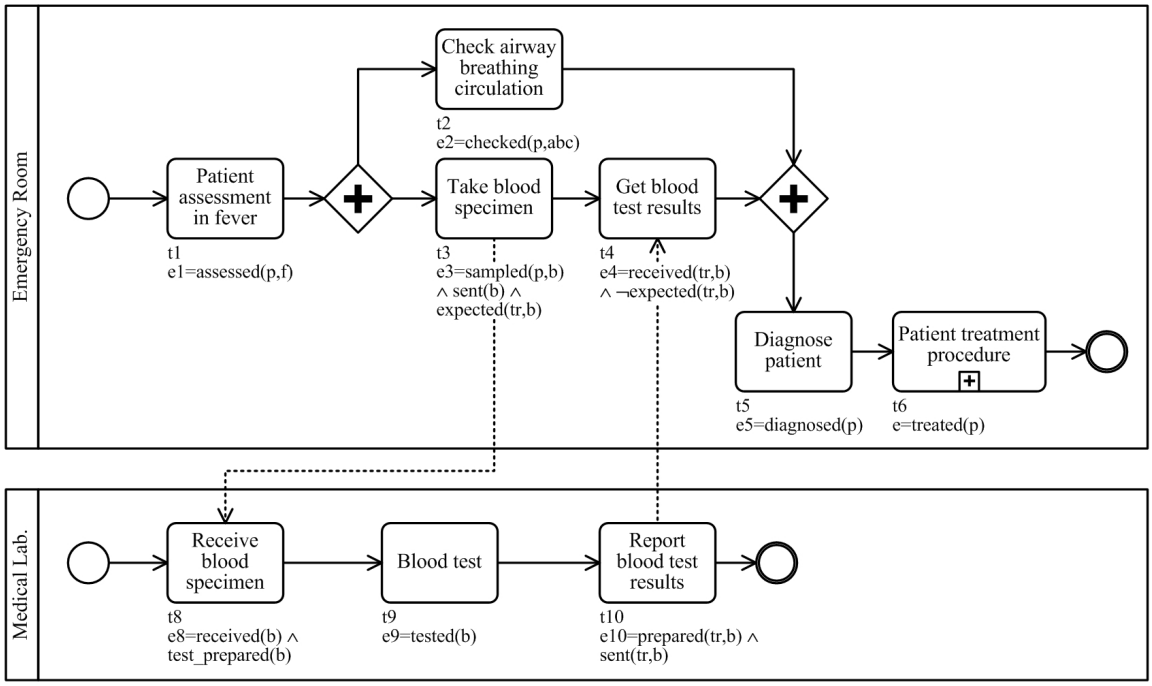


Figure 3.3: *Handling of patient in fever in emergency room* inter-operation processes

We shall refer to an activity sending a message a *sender* activity, and an activity receiving a message a *receiver* activity. Effect contradiction exists if the expected effects (i.e. computed at the receiver) differ from the given effects (i.e. computed at the sender). If this occurs, we consider that an inter-operation does not exist between a given pair of processes.

Figure 3.3 exemplifies an inter-operation relationship between processes of *Handling of patient in fever in emergency room*. Let the upper and the lower processes be denoted as $P1$ and $P2$, respectively. This setting involves two messages exchanged between both processes with their corresponding sender and receiver activities, respectively, i.e. activities t_3 and t_8; activities t_{10} and t_4. Activity *Take blood specimen* t_3 in $P1$ sends a message to activity *Receive blood specimen* t_8 in $P2$; on the contrary, activity *Report blood test results* t_{10} in $P2$ to activity *Get blood test results* t_4 in $P1$ in order to fulfill the functionalities of such processes. Semantically, we can compute $CE\,(P1, t_3) = \{es_{13}\}$ where $es_{13} = assessed\,(p, f) \wedge sampled\,(p, b) \wedge$

$sent\,(b) \wedge expected\,(tr, b)$. Similarly, $CE\,(P2, t_8) = \{es_{28}\}$ where $es_{28} = received\,(b) \wedge test\_prepared\,(b)$. We can observe that $es_{13} \cup es_{28} \vdash \perp$ does not hold. Dually, we can also compute $CE\,(P2, t_{10}) = \{es_{210}\}$ where $es_{210} = received\,(b) \wedge test\_prepared\,(b) \wedge tested\,(b) \wedge prepared\,(tr, b) \wedge sent\,(tr, b)$. While, $CE\,(P1, t_4) = \{es_{14}\}$ where $es_{14} = assessed\,(p, f) \wedge sampled\,(p, b) \wedge sent\,(b) \wedge received\,(tr, b) \wedge \neg expected\,(tr, b)$. It is also obvious that $es_{210} \cup es_{14} \vdash \perp$ does not hold. Further, we can illustrated effect contradiction between processes described in Figure 3.3 as follows. For example, if we include $labeled(b)$ as the expected effect in immediate effect $e8$ and $\neg labeled(b)$ as the given effect in the immediate effect $e3$, then we fall into this contradiction. Such contradiction occurs since at t_8 we expect that the blood specimen has been labeled at the point of t_3.

Generalization-specialization

A generalization-specialization relationship exists between two processes when one process becomes the functional extension of the other. More specifically, the *specialized* process has the same functionalities as in the *generalized* one and also extends it with some additional functionalities. Our interpretation of such relationship was inspired by the notion of subtyping that was first made popular in programming language theory and later extended to conceptual modeling. We do not directly link this interpretation to the definition of object-oriented inheritance or subclass, which is in fact a mechanism to achieve subtyping. In essence, we may not apply a pairwise comparison of activities to the two process models in question. Instead, we compare their cumulative effects to see if the *specialized* process can safely be used in a context where the *generalized* one is expected, as described below.

Using semantic effect analysis, the functionalities are represented as the immediate effects (of an individual activity) and the cumulative effects (of the complete process).

One way to extend the functionalities is adding some additional activities such that the intended cumulative effects of the process are consequently extended. Another way involves expanding the immediate effects of the existing activities. In this case, the number of activities remains the same for both processes but the capabilities of the *specialized* are extended. Note that the *specialized* process inherits all functionalities of the *generalized* process. We originally introduced the formal definition of such generalization-specialization relationship in our previous publication [35]. For the sake of improvement, we, however, have revisited such formal definition due to two reasons, i.e. correctness and extensiveness.

Definition 5. (Strict generalization-specialization) *Given process models* $P1$ *and* $P2$*,* $P2$ *is a specialization of* $P1$ *if the following holds:*

- $\forall es_i \in acc\,(P1)$*,* $\exists es_j \in acc\,(P2)$ *such that* $es_j \models es_i$
- $\forall es_j \in acc\,(P2)$*,* $\exists es_i \in acc\,(P1)$ *such that* $es_j \models es_i$*,*

In regard correctness, we formally redefine such relationship described in Definition 5. This correction should replace its original definition introduced in [35] since in such original definition we may mistakenly infer that the two processes are the same processes. We further assume that this formal definition is applicable for both design and instance levels of process views. So that every effect scenario which exists in both process design and its corresponding instance satisfies such formal definition. We believe that such definition, to some extent, is similar to process specialization defined in [40], i.e. considering process specialization relies on a constraint that every instance of the specialized process is also an instance of the generalized process. We differ to that in [40] in which their process specialization relies on the maximal execution set semantic, i.e. the behavior of the specialized process is subset of the behavior of the generalized one.

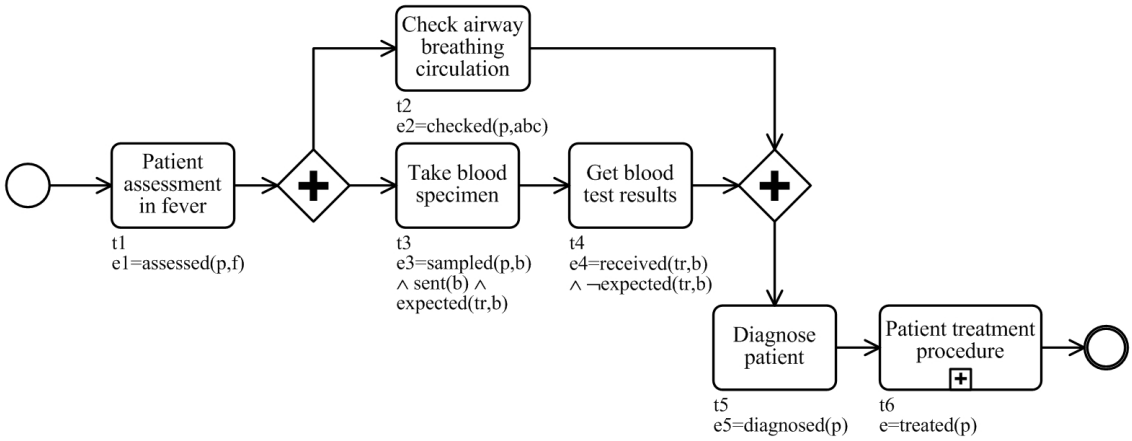


Figure 3.4: *Handling of patient in fever in emergency room* process plays the 'generalized' process.

Figures 3.4 and 3.5, which are respectively denoted as $P1$ and $P2$, illustrate a generalization-specialization relationship between two processes. These figures illustrate how a patient in fever should be handled in an emergency room. In this context, $P2$ becomes the specialization of $P1$. Obviously, their design and instance levels views satisfy constraints specified in Definition 5. As can be seen, $P2$ has exactly the same functionalities with $P1$. The former, however, has some specific functionalities on activities *Patient assessment in fever and twitch*, which is the extension of activity *Patient assessment in fever*, and *Take skull x-ray and CT-scan*, which is the additional activity. Both activities together extend the functionalities of $P1$.

Furthermore, we can semantically observe such relationship overs Definition 5. Let us compute $acc(P1) = \{es_{11}\}$, where $es_{11} = assessed(p, f) \wedge checked(p, abc) \wedge sampled(p, b) \wedge sent(b) \wedge received(tr, b) \wedge \neg expected(tr, b) \wedge diagnosed(p) \wedge treated(p)$. While, computing the end cumulative effects of $P2$ yields $acc(P2) = \{es_{21}\}$, where $es_{21} = assessed(p, f) \wedge checked(p, abc) \wedge sampled(p, b) \wedge sent(b) \wedge received(tr, b) \wedge \neg expected(tr, b) \wedge taken(x) \wedge taken(y) \wedge diagnosed(p) \wedge treated(p)$. We may observe that $es_{21} \models es_{11}$ for every effect scenario occurs in every process design and process instance of $P1$ and $P2$. As such, their design and instance levels views satisfy constraints

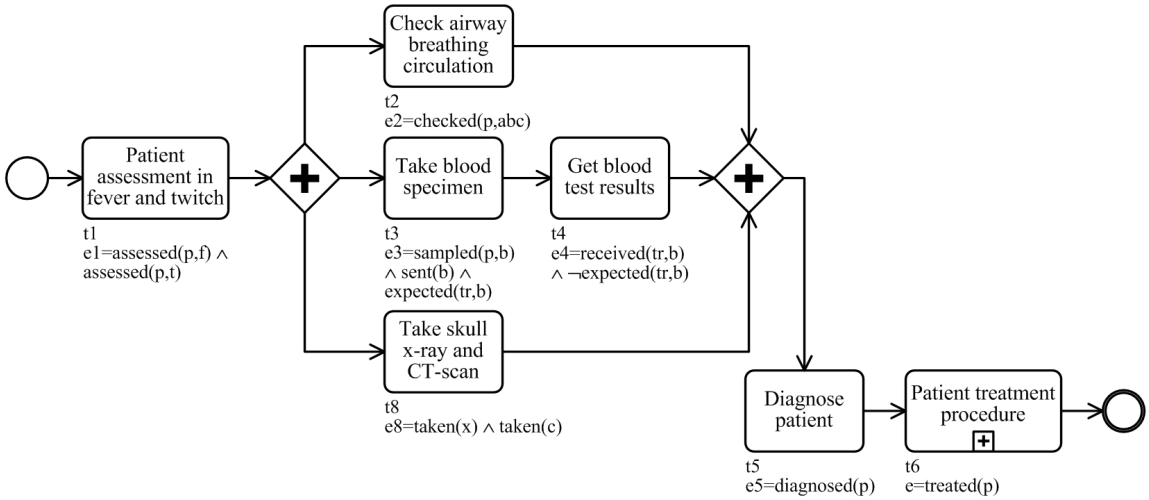


Figure 3.5: *Handling of patient in fever and twitch in emergency room* process becomes a 'specialized' process of process in Figure 3.4.

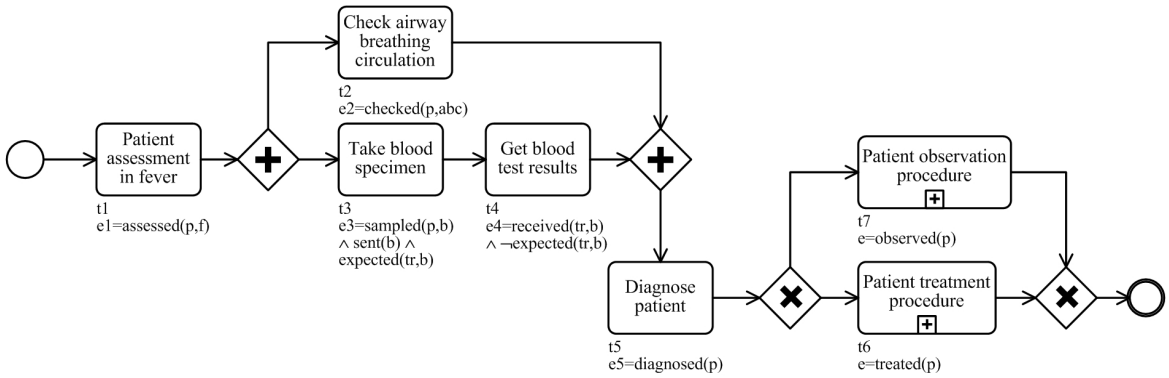


Figure 3.6: *Handling of patient in fever in emergency room* process with alternative *Patient observation procedure* activity

specified in the definition.

Definition 6. (Relax generalization-specialization) *Given process models* $P1$ *and* $P2$*,* $P2$ *is a specialization of* $P1$ *iff* $\forall es_i \in acc(P1)$*,* $\exists es_j \in acc(P2)$ *such that* $es_j \models es_i$*.*

In regard extensiveness, we relax to identifying such relationship with respect to Definition 5. In this context, we further realize that a process analyst, in some cases of business practices, may intuitively infer two process models are in specialization-

generalization relationship even constraints described in Definition 5 can not be entirely satisfied. We may consider Figure 3.6, which is denoted as $P2'$, to illustrate such case. With respect to Definition 5, process $P2'$ can not be considered as a specialized process of $P1$ since the former has an effect scenario which does not exist in the latter due to introducing a pair of XOR gateways and a sub-process *Patient observation procedure*.

Let us compute $acc\,(P2') = \{es'_{21}, es'_{22}\}$, where $es'_{21} = assessed\,(p, f) \wedge checked\,(p, abc) \wedge sampled\,(p, b) \wedge sent\,(b) \wedge received\,(tr, b) \wedge \neg expected\,(tr, b) \wedge diagnosed\,(p) \wedge treated\,(p)$; and $es'_{22} = assessed(p, f) \wedge checked(p, abc) \wedge sampled(p, b) \wedge sent(b) \wedge received(tr, b) \wedge \neg expected(tr, b) \wedge diagnosed(p) \wedge observed(p)$. Using our previous computed $acc\,(P1)$, we may observe that $es'_{21} \models es_{11}$ and $es'_{22} \not\models es_{11}$ for every effect scenario occurs in every process design and process instance of $P1$ and $P2'$. In particular, $P2'$ may be executed at effect scenario es'_{22} which does not entail any effect scenario of $P1$. We, however, may intuitively suggest that both processes are in such relationship at design level since they refer to the idea of inheritance concept which is a powerful concept used in object-oriented approach (see, e.g. [6]), i.e. one class shares the structure and/ behavior defined in one (single inheritance) or more (multiple inheritance). In this context, we relax to have another definition by considering processes at the design level only, as described in Definition 6. Based on such relaxed definition, we can infer that $P2'$ is the specialized process of $P1$ since $es'_{21} \models es_{11}$ for every effect scenario occurs in process design of $P1$.

3.1.2 Establishment and maintenance

The formal definitions of inter-process relationships discussed in the previous section empower us to systematically specify (manually or automatically) relations between process models in process repositories. To do so, analysts who need to manage the

large and complex process repositories must effectively explore the space of all possible pairings of process designs to determine what relationships (if any) should hold in each instance. Note that this generates a space of $\binom{n}{2}$ possibilities, where n is the number of distinct process designs in the repository. Clearly, this has the potential to be an error-prone exercise. An analyst may normatively specify a relationship that does not actually hold (with regard to our definition of the relationship) between a pair of process designs. In some cases, an analyst might need help in deciding what relationship ought to hold. We therefore develop a *user-interactive* machinery to assist analysts in deciding what type of normative relationship should hold between a pair of processes.

We consider two approaches in such assistance, i.e. *checking* and *generating* modes. In the *checking* mode, we will assess whether a relationship specified by an analyst does indeed hold with regard to our formal definitions. If this is the case, the relationship between the two processes can be established. Otherwise, the tool would alert the analyst and also suggest the actual relationship that may be found between the two processes[1]. In the *generating* mode, our machinery systematically goes through all process models in the repository, generates all possible relationships between them and presents such relationships to the analyst for confirmation. Note that the space of alternative relationships can be large, specially in the case of part-whole relationships. For example, given 4 processes $P1, P2, P3$ and $P4$ where $P2$ is part of $P1$, $P3$ is part of $P2$ and $P4$ is part of $P3$. Not only direct relationships, the tool would also suggest all indirect relationships among them, e.g. $P4$ is (indirectly) part of $P1$, $P4$ is (indirectly) part of $P2$. These indirect relationships, however, would not be useful to be maintained since change propagation can still be performed effectively through the direct ones. Therefore, we realize that the decision should be made by the analyst in both approaches.

[1]If it is 'unknown' relationship (refers to our formal definitions), it is not maintained.

Once a relationship is established, a *relationship descriptor* is created. Such a descriptor contains details that are relevant to its associated relationship including identities of each pair of processes and their established relationship type. A descriptor, however, may also be enriched with any additional information relevant to the existing relationship types, e.g. the insertion point activity in a part-whole relationship. A set of relationship descriptors, as defined in Definition 7, must be properly maintained (i.e. created, updated, and removed) during the establishment and maintenance of the relationships in order to keep it consistence with its underlying process models in the process repository.

Definition 7. (Relationship descriptors) *A Relationship descriptors RD is a set of relationship descriptors, where each relationship descriptor* rd *is a tuple* $\langle pa, pb, rel, t_{ip} \rangle$ *with:*

- pa *is the first process of a pair of processes;*
- pb *is the second process of a pair of processes;*
- $rel \in REL$ *is the established relationship instance such that* pa rel pb*, where* REL *is a set of relationship instances* $\{part\_of, inter\_op, specialization\_of\}$*;*
- t_{ip} *is the insertion point activity - might be empty.*

The relationship-establishing algorithms for both approaches require transformation of each process model into a graph, i.e. transforming each activity, gateway and start or end events into a node and each flow into an edge. These algorithms may involve two runs in evaluating a given pair of processes for each relationship type excluding the inter-operation. On the first run, we evaluate the first process with respect to a normative relationship constraint to the second one. If the constraint does hold, we establish the relationship between the two processes. Otherwise, in the second run

we evaluate the second process with respect to the constraint to the first one. For example, the part-whole establishment algorithm can be described as follows. The inputs are two process graphs, i.e. denoted as *pa* and *pb*, and the outputs are either an instance of relationship descriptor or *null*. The algorithm will assess whether *pa* is part of *pb* by computing $CE\,(pb, n)$ and $CE\,(pb \uparrow^n pa, n)$ of each sub-process node $n \in pb$. If $CE\,(pb, n) = CE\,(pb \uparrow^n pa, n)$ then it returns an instance of relationship descriptor of such relationship. Otherwise, it returns *null*. On the first run, we evaluate the first process to the second one. If it is not satisfied, we then evaluate the second process to the first one on the second run. If a given relationship type cannot be established, we continue the evaluation with the other relationships between the processes in question. Note that such a part-whole relationship evaluation must be performed before a generalization-specialization relationship evaluation, since the former is a special case of the latter. Inappropriate evaluation ordering of these two relationships might yield unexpected result, i.e. every part-whole will be suggested as a generalization-specialization. In addition, there is no evaluation ordering constraint for inter-operation relationship type.

3.1.3 Discussion

In regard inter-process relationship formalizations, we, however, also consider other relationship settings between a pair of processes in which they are not directly related to each other. For example, let process $P3$ be a detailed process (not described in the diagram) of activity *Surgical operation* in Figure 3.2, which is a sub-process of $P2$. In this context, we logically consider process $P3$ also be a part of process $P1$ though there is no activity in $P1$ which is completely represented by the functionalities of $P3$. Obviously, there exist an activity in $P1$ which entails the functionalities of $P3$. As such, we can consider an *indirect* part-whole relationship exists between $P3$ and $P1$.

Similarly, we can identify such indirect inter-process relationship between processes for other types, i.e. inter-operation and generalization-specialization. However, we are only interested to consider the direct one since our process changes propagation approach in a process ecosystem will rely on such relationship. Further, this indirect relationship can be considered in performing process changes impact analysis in a process ecosystem such that all processes involved in such indirect relationships are subject to be impacted.

In addition, we also consider other inter-process relationship types besides the aforementioned three types which may exist in a complex business process repository. These may include abstraction-refinement (i.e. presenting a process in different representations based on their granularity levels), resource-sharing (i.e. two processes are related to each other due to dependencies on the same resource) and informational inter-operation (i.e. two processes are related to each other with respect to the conformance on the message exchanged between them). We, however, leave them to be in our future investigation.

Further, we realize that identifying relationships between business processes has been previously performed by other researchers with respect to their own concepts (see, e.g. [32, 40, 57, 60]). Our part-whole relationship is similar to that described in [40] in which they use *uses-parts* process classification. They use decomposition approach to organize their process knowledge in such classification. We extend and formalize inter-operation relationship described in [32]. Further, our specialization-generalization relationship is also similar to that introduced in [40]. They use *specializations-generalizations* process classification. We believe that our strict specialization-generalization relationship definition, to some extent, is similar to process specialization defined in [40], i.e. process specialization relies on a constraint that every instance of the specialized process is also an instance of the generalized process. Our specialization differs from that

described in [40], whereas their process specialization relies on maximal execution set semantics, i.e. the behavior of the specialized process is subset of the behavior of the generalized one. To the best of our knowledge, our interpretation on specialization-generalization relationship is close to the *projection inheritance* defined in [60]. In our inter-process relationship concepts, we leverage our semantic effect annotation to allow us to manage any relationship which may exist between process models in the process repository in a semi-automatic manner.

3.2 Notions of process ecosystem

3.2.1 Definition and formalization

We use ecosystem metaphor to view a collection of interrelated process models within a process repository. In biological ecosystem, all entities (i.e. organisms) are connected to each other. One population interacts with one another in a complex structure of relationships including oppositional (e.g. one becomes predator for the others) and symbiotic (e.g. each entity benefits to each other) relationships. In process ecosystem, however, we further restrict the definition of this ecosystem with respect to our change propagation context such that any process model in a process ecosystem must be traceable to any other process model in the ecosystem, as formally described in Definitions 8 and 9. Note that we refer to two processes are traceable to as a special notion of traceability which involves change justification. This traceability may involve many relationships with regard to the relationships we formally defined earlier. A process model may have more than one relationship with the others in the ecosystem. In addition, there may be more than one process ecosystem within a process repository. Furthermore, since a change made to a process would only affect other processes in the same ecosystem, we will only propagate the changes in a process ecosystem based

on the description of any relationships resides in the relationship descriptors.

We consider a process ecosystem is in *equilibrium* if and only if every inter-process relationship in the ecosystem is consistent with our earlier definitions. We shall refer to a process ecosystem that violates the consistency equilibrium condition a *disequilibrium* ecosystem. Consistency perturbation is often the outcome of change to one or more process models in an ecosystem. In our approach, restoring the equilibrium involves making further changes to other process models in the ecosystem and so on, which is, in fact, change propagation. We shall refer to an ecosystem resulted from such restoration a *restored-equilibrium* ecosystem. We will discuss this in more detail in Chapter 4.

Definition 8. (Process traceability) *Two processes p_i and p_j, within a process repository with a relationship descriptors RD, are traceable to each other, $p_i \overset{trace}{\leftrightarrow} p_j$, iff there exists at least a sequence path of processes $\langle p_1, p_2, \ldots, p_n \rangle$ such that $\forall k$, $1 \leq k \leq n$, $(p_k, p_{k+1}) \in RD$, and $p_i = p_1$, $p_j = p_n$.*

Definition 9. (Process ecosystem) *A tuple $PE = (P, F_{rel})$ is a process ecosystem within a given process repository, where: (i) P is a nonempty finite set of process models in the repository; (ii) $F_{rel} \subseteq P \times P \subseteq RD$ is a relationship function mapping each processes in P described in RD such that $\forall p_i, p_j \in P$, $p_i \overset{trace}{\leftrightarrow} p_j$.*

3.2.2 Representation

In order to represent a process ecosystem, we leverage constraint networks described in Section 2.4 from which a constraint graph can be derived. Some transformations, however, have to be performed for such representation. We map every single process model in the process repository as a variable in the constraint networks. Regarded as such variable in the constraint networks, a process model has its own domain values, i.e. any process model variant. We argue that domain values of a process model is finite

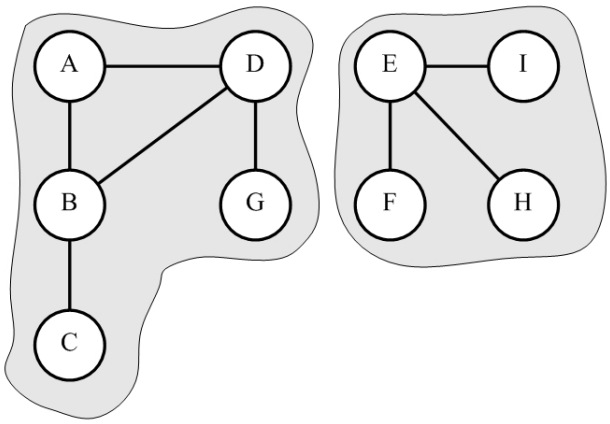
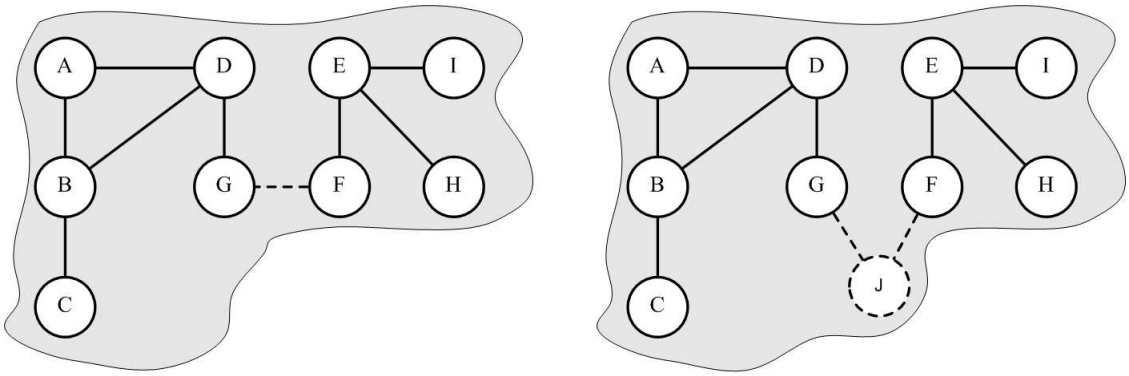


Figure 3.7: A process repository with two separate process ecosystems

since we consider that each process variant has to satisfy the end cumulative effects of such process model. Generating any variant of a process model in such a way while simultaneously considering any design constraints (e.g. activities ordering, restricted available activities) is limited. As such, it is reasonable to capture each process model as a variable in constraint networks and to represent it as a node in a constraint graph. Furthermore, we capture every inter-process relationship described in the relationship descriptors as a constraint in the constraint networks. In this context, this binary constraint involves a pair of process models. In a constraint graph, we represent such constraint as an edge connecting two nodes which represent a pair of process models.

Once such transformation is complete, we can represent our process ecosystem in a constraint graph. Figure 3.7 exemplifies process ecosystems representation that may exist in a process repository involving nine process models $A, \ldots, I$ with their corresponding relationships. On the one hand, process models A, B, C, D and G are traceable to each other due to their relationships. On the other hand, process models E, F, H and I are traceable to each other. In this setting, any process model in the former group is not traceable to any process model in the latter group since there does not exist any relationship between them. As such, with respect to Definition 9, there exist two separate process ecosystems which are graphically grouped in two



(a) Introducing a new relationship (dashed line) to the existing process models

(b) Introducing a process model (dashed circle) with respective new relationships to the existing process models in separate process ecosystems

Figure 3.8: Dynamicity of process ecosystems structure originally shown in Figure 3.7

separate boundaries, i.e. left and right groups. We intend to leverage the illustration in Figure 3.7 to clearly describe our process ecosystem boundaries.

Such representation is useful in dealing with process changes propagation such that we are only interested in all processes involved in the same ecosystem with the process getting the initial changes. This representation, however, can be dynamic which means the structure of process ecosystems within a process repository may change due to any change on the relationship descriptors triggered by process changes. These may include two basic changes: (i) introducing a new relationship between the existing process models, see Figure 3.8(a) for a new dashed line; and (ii) introducing a new process model with respective new relationships to the existing process models in the separate process ecosystems, see Figure 3.8(b) for a new dashed circle and new dashed lines. In such changes, the two process ecosystems, which are originally separated in Figure 3.7, become a single process ecosystem involving all process models reside in the process repository. Consequently, the number of processes in our interest becomes larger in order to propagating changes in such a merged process ecosystem. We also may consider another possible change on the process ecosystems structure, i.e. introducing a new process ecosystem due to either introducing a set of new interrelated process

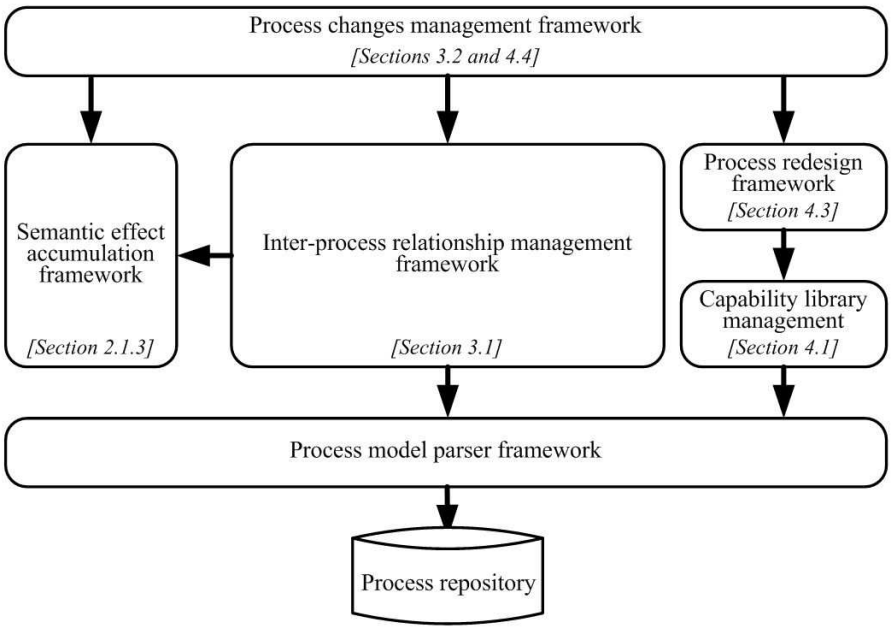


Figure 3.9: The process ecosystem framework

models or removing (for any reason) some existing inter-process relationships.

3.3 Architectural overview of the process ecosystem framework

In this section, we describe the architecture of the process ecosystem as a consolidated framework in dealing with process changes management in a process repository. The framework is composed of a number of key components depicted in Figure 3.9, i.e. process model parser framework, inter-process relationship management framework, semantic effect accumulation framework, capability library management, process redesign framework and process changes management framework. The dependencies between components are represented by arrows, e.g. process changes management framework depends on semantic effect accumulation, inter-process relationship management and process redesign frameworks in order to perform its functionalities.

Process model parser framework focuses on transforming any business process model created by using a certain business process modeling language (i.e. BPMN) into a graph representation, and vise versa. This component is fundamental in our process ecosystem framework since the remaining components rely on its parsing results. *Inter-process relationship management framework* deals with the management of any relationship which may exist between business process models in the process repository. Such management includes establishment and maintenance of any inter-process relationship, as conceptually describe in Section 3.1. It is also involved in assessing the constraint satisfaction of any such relationship during process changes management. This inter-process relationship management plays a vital role in our process ecosystem framework as our process changes management relies on the dependencies between process models in order to propagate the changes. In the process ecosystem framework, it depends on the process model parser framework and semantic effect accumulation framework components.

Capability library management relates to management of a library which consists of the specification of all activities used in any process model in the process repository. Each activity will be specified once in the library, which is annotated with its relevant effects. Such library will be important in redesigning a process model during process changes management as the effects of their activities will contribute to achieve the end cumulative effect of the resulting process model. We discuss the underlying concept of this capability library in detail in Section 4.1 including its formalization, establishment and maintenance. In the process ecosystem framework, its functionalities depend on the process model parser framework component. *Semantic effect accumulation framework* takes responsibility for accumulating effects at any point in a process model based on the underlying concept described in Section 2.1.3. Such accumulated effects provide important information required in determining an inter-process relationship

type and assessing any inter-process relationship violation which may take place.

Process redesign framework concerns with reorganizing activities in a process model in order to resolve any inter-process relationship violation while propagating process changes in a process repository. It will generate an alternative design of a process model which is involved in such violation by leveraging any activity specified in the capability library. It relies on process redesign concept used in semantically effect annotated process models described in Section 4.3 . As such, it will use capability library management component to perform its functionalities within the process ecosystem framework. *Proces changes management framework* focuses on providing procedures in reestablishing the equilibrium of a process ecosystem which may be perturbed by introducing process changes. Such equilibrium can be achieved once the constraints of each inter-process relationship are satisfied. In order to do so, it requires supports given by semantic effect accumulation framework, inter-process relationship management framework and process redesign framework components. We construct this process changes management framework according to the underlying concepts of process ecosystem and process changes propagation, described in Section 3.2 and 4.4, respectively.

3.4 Summary

So far, we have proposed our process ecosystem concepts including identification and formalization of relationships which may exist among business processes in such process ecosystem. We focused on three relationship types, i.e. part-whole, inter-operation and generalization-specialization. Formalizing these relationships becomes a fundamental concept in our approach since our process ecosystem and process changes propagation concepts will be built on the top of it. We also introduced our procedures in establishing and maintaining such relationships such that they can be done in a ma-

chinery fashion. These procedures, however, still require process analyst involvement for judging the relevant inter-process relationships that should be maintained in the process repository for further usages. In addition, a discussion has been arranged to have further observation on our inter-process relationship concepts including indirect relationships and related work in the area. We also have introduced our definition and formalization of process ecosystem itself as well as its representation using constraint networks. Finally, a consolidated architectural overview of the process ecosystem framework has also been presented to show all involved components in order to build the framework. Therefore, we argue that our discussion in the next chapter, i.e. managing change in process ecosystem, can be adequately performed based on our introduction in this chapter.

Chapter 4

Managing change in process ecosystem

In this chapter, we propose our approach for managing process changes in process ecosystem. In the beginning of such management, it requires taxonomy identification of process changes that can violate inter-process relationship constraints. This taxonomy guides us to construct resolution patterns which are required to resolve any inter-process relationship violation. Based on these patterns, we develop process redesign approach to allow us to redesign a process involving in a violated inter-process relationship such that the relationship constraints can be resatisfied. We further intend to redesign such process in an effective and efficient manner by leveraging any activities used in any process models in the repository which are maintained in the capability library. Since the changes should be propagated across the process ecosystem, we also develop our own procedures of such propagation inspired by a well-known approach in constraint problems, i.e. CSP.

We, however, discuss these concepts in different order. More precisely, we introduce our capability library idea including its formalization, establishment and maintenance in Section 4.1. We also specify the taxonomy of process changes as well as its respective

resolution patterns in Section 4.2. In Section 4.3, we propose our process redesign approach. Furthermore, we present our changes propagation procedures in process ecosystem in Section 4.4. Finally, we summarize our discussion at the end of this chapter.

4.1 Capability library

A capability library can be viewed as a repository of the specification of all activities, which are annotated with their respective immediate effects, used in any process model contained in the process repository. Thus, every activity used in each process model should be defined in the library. This is a critical representation of *enterprise competence* - the intent is that it contains specification of all activities that an enterprise is able to execute, and thus all of the building blocks from which one might build the design of any business process supported by the enterprise. We will describe below the formalization, establishment and maintenance of such library.

4.1.1 Formalization

We capture each activity specified in the capability library as the blue print of any activity used in any process model in the process repository. We consider each activity in the capability library can be used multiple times either in a single process model or in many distinct process models. Each activity in the capability library, however, must be specified once such that it is unique compared to the other activities. Such uniqueness can be basically identified by their own immediate effects which are implicitly represented by their activity name. Hence, we formalize activity specified in the capability library defined in Definition 10 which is required to build the capability library defined in Definition 11.

Definition 10. Activity. *An activity is a tuple* $t = \langle name, e \rangle$*, where:*

- *name denotes a unique name of the activity specified in the capability library;*
- *e denotes a unique immediate effects of performing the activity represented in controlled natural language (CNL);*

Definition 11. Capability library. *A tuple* $CL = \langle T, R, F \rangle$ *is a capability library of a given process repository, where:*

- T *is a nonempty finite set of activities;*
- R *is a nonempty finite set of roles;*
- $F \subseteq T \times R$ *is the activity performer relation such that* $\forall t \in T$, $\exists r \in R$ *in which* r *executes* t*.*

We consider each activity specified in the capability library has its own unique immediate effects such that $\forall t_i \in T$ in CL there exists no $t_j \in T$ in CL where $e_i = e_j$ and $i \neq j$. We also imagine a situation where this constraint is violated by definition, as follows, $\forall t_i \in T$ in CL there exists $t_j \in T$ in CL where $i \neq j$ and: (i) $e_i = e_j$ and $name_i \neq name_j$ or (ii) $e_i \neq e_j$ and $name_i = name_j$. So, we need to resolve such violations either by rationalizing them into a single activity or differentiating their immediate effects and names. Further, we realize that there might happen an incompleteness of effect annotation on an activity due to manual annotation such that two or more activities have the same immediate effects. As such, a detection mechanism should be provided in order to prompt process analyst to resolve such incompleteness through the aforementioned approaches.

As business process automation becomes intensively engaged, enterprise requires to precisely define the resources (refer to roles representing capabilities of enterprise

groups) involved in the execution of its process activities [75]. Moreover, recent contribution in business process abstraction also consider such roles as the operators for abstracting activities in a process model [54]. Hence, we take into account enterprise's roles, which are commonly represented with swimlanes in process model, in our capability library definition. Each activity specified in the library must be uniquely performed by a particular role. A single role, however, may perform many activities.

We consider an activity specified in the capability library can be either task or sub-process (as defined in [72]). We, however, believe the functionality of such a sub-process represents the functionality of a particular complete process model, which relies on a set of activities, in the process repository. We argue that each sub-process specified in the capability library must have exactly an underlying process model in the process repository as the expansion of such a sub-process. As such, it leads to the sub-process completeness constraint defined in Definition 12. In this context, there exists an one-one relation between sub-process and its underlying process model. It, however, is not necessarily required for each process model to have a corresponding sub-process specified in the library.

Definition 12. Sub-process completeness. *Any sub-process t_i (whose immediate effects e_i has multiple effect scenarios $es_1, \ldots, es_n$) specified in the capability library is complete iff there exists a process model expansion P_i in the process repository such that $\forall es_p \in e_i$, $\exists es_q \in acc(P_i)$ such that $es_q = es_p$, where $1 \leq q \leq n$ and $t_i \in T$.*

In regard to any process model in the process repository, we assume that every activity used in such process model corresponds to a particular activity specified in the capability library. Such correspondence describes that the definition of any activity used in a process model is similar to that of its corresponding activity specified in the capability library including name and immediate effects, as well as the role for executing such activity. In this context, there must exist an one-one relation between

an activity used in any process model and its corresponding activity specified in the capability library. We realize that a single activity specified in the capability library can be used multiple times either in a single process model or in many distinct process models. We identify that each activity used in any process model can only be differentiated among the others by its unique id. As such, we can define an one-many relation between an activity specified in the capability library and its corresponding activities used in process models. There, however, might exist no process model in the process repository which uses an activity specified in the capability library.

We consider the capability library completeness in order to properly represent the enterprise competence. Such completeness requires every activity used in any process model has its corresponding activity specified in the library, as well as each sub-process specified in the library has its corresponding process model expansion in the process repository. As such, we argue that a capability library is incomplete based on the following reasons. First, there exist a process model whose activity has immediate effects which are not exactly specified by any activity in the capability library. Second, there exists a sub-process specified in the capability library, whose functionality is not supported by any underlying process model in the process repository. If so, process analyst should be prompted by a detection mechanism to resolve such incompleteness.

4.1.2 Establishment and maintenance

In this part, we describe in detail how the capability library can be established and then maintained during its life cycle. In establishing the library, we need to consider the following scenarios: (i) process repository does not exist and (ii) process repository already exists consisting a number of process models. We elaborate both scenarios in the following.

Scenario (i) will develop capability library while designing a new process model in

the process repository. Once we need to put a new activity, which is not yet described by any activity in the capability library, in a new process model, we need to add a new capability (i.e. activity) to be specified in the capability library. Otherwise, we only need to use the existing activities specified in the library for describing any activity used in a new process model.

Scenario (ii) is more complex and requires an existing process repository to be populated to create a new capability library. First, all activities used in all process models including tasks and sub-processes must be extracted based upon their names. This extraction also involves identification of process models utilizing each activity and the role executing it. Second, these extracted activities need to be clustered with respect to their similarity based upon their names[1]. Two different extracted activities with slightly different names are likely to be considered as the same activity. This clustering result suggests process analysts to judge whether or not two or more extracted activities should be described by a single activity in the capability library. Third, once all activities have been specified in the library, process analysts should complete their immediate effects annotation as well as any roles inconsistencies resolutions which might be required.

Along the life cycle of a capability library, we can imagine three distinct kinds of changes to it due to any process change as follows.

- **Capability repurposing:** In this setting, we take an existing capability t, and use it as the basis for introducing a new capability t' whose effects are (a hopefully small) variant of the effects of t. The resulting library contains both t and t'. This would allow us to leverage the repurposed version of t, e.g. in resolving a relationship violation, while leaving the definition of t unchanged in the other processes contained in the repository.

[1]We can use one of many available text similarity engines.

- **Capability redefinition:** In this setting, we redefine the semantics (i.e. effects) of an existing capability t such that the change applies to all business processes that leverage t.
- **Capability creation:** In this setting, we create new capability t where t has not yet been used by any process and its semantics (i.e. effects) differ from any existing capability.

4.2 Process changes taxonomy and resolution patterns

As process changes become the source of a relationship violation, we identify a taxonomy of all possible process changes which potentially trigger such violation. Based on this taxonomy, we construct our resolution patterns as the guidance for resolving any inter-process relationship violation that may occur. Our change taxonomy and resolution patterns are constructed based upon inter-process relationship types, i.e. part-whole, inter-operation and generalization-specialization.

4.2.1 Taxonomy of changes triggering relationship violations

Based on the inter-process relationship types, we identify a taxonomy of all possible process changes which potentially violate the relationship constraints between a pair of processes $P1$ and $P2$. By identifying such taxonomy, we become aware the risk of a certain process change to the relationship which may exist between processes.

Part-whole

Let $P1$ and $P2$ be the main and the part processes, respectively. In addition, let t_i (with immediate effects e_{t_i}) be an insertion point of $P2$ in $P1$ such that the constraints

are satisfied: (i) $CE\,(P1, t_i) = CE\,(P1 \uparrow^{t_i} P2, t_i)$ for a context-dependent part-whole or (ii) $e_{t_i} = acc\,(P2)$ for a context-independent part-whole.

(a) Changes on the main process. Changing $P1$ into $P1'$ may violate the relationship to its part process iff such changing breaks the relationship constraints. This can happen due to the following structural changes:

- **C-W1**: any structural change (which may affect the semantics) leads to t_i such that $CE\,(P1', t_i) \neq CE\,(P1' \uparrow^{t_i} P2, t_i)$ for a context-dependent part-whole or
- **C-W2**: changing t_i into t_i' such that $e_{t_i'} \neq acc\,(P2)$ for a context-independent part-whole.

Capability repurposing or capability redefinition can be considered as the sources for both structural changes through either expanding the capability (i.e. by adding some new effects) or contracting the capability (i.e. by removing some existing effects).

(b) Changes on the part process. Changing $P2$ into $P2'$ may violate the relationship to its main process iff $acc\,(P2) \neq acc\,(P2')$ with the set of effect scenarios E and E', respectively. This can happen due to the following structural changes:

- **C-P1**: adding or removing any activity,
- **C-P2**: capability repurposing on any activity,
- **C-P3**: capability redefinition on any activity,
- **C-P4**: replacing any activity by another activity whose immediate effects are completely different,
- **C-P5**: swapping activities across XOR gateways,

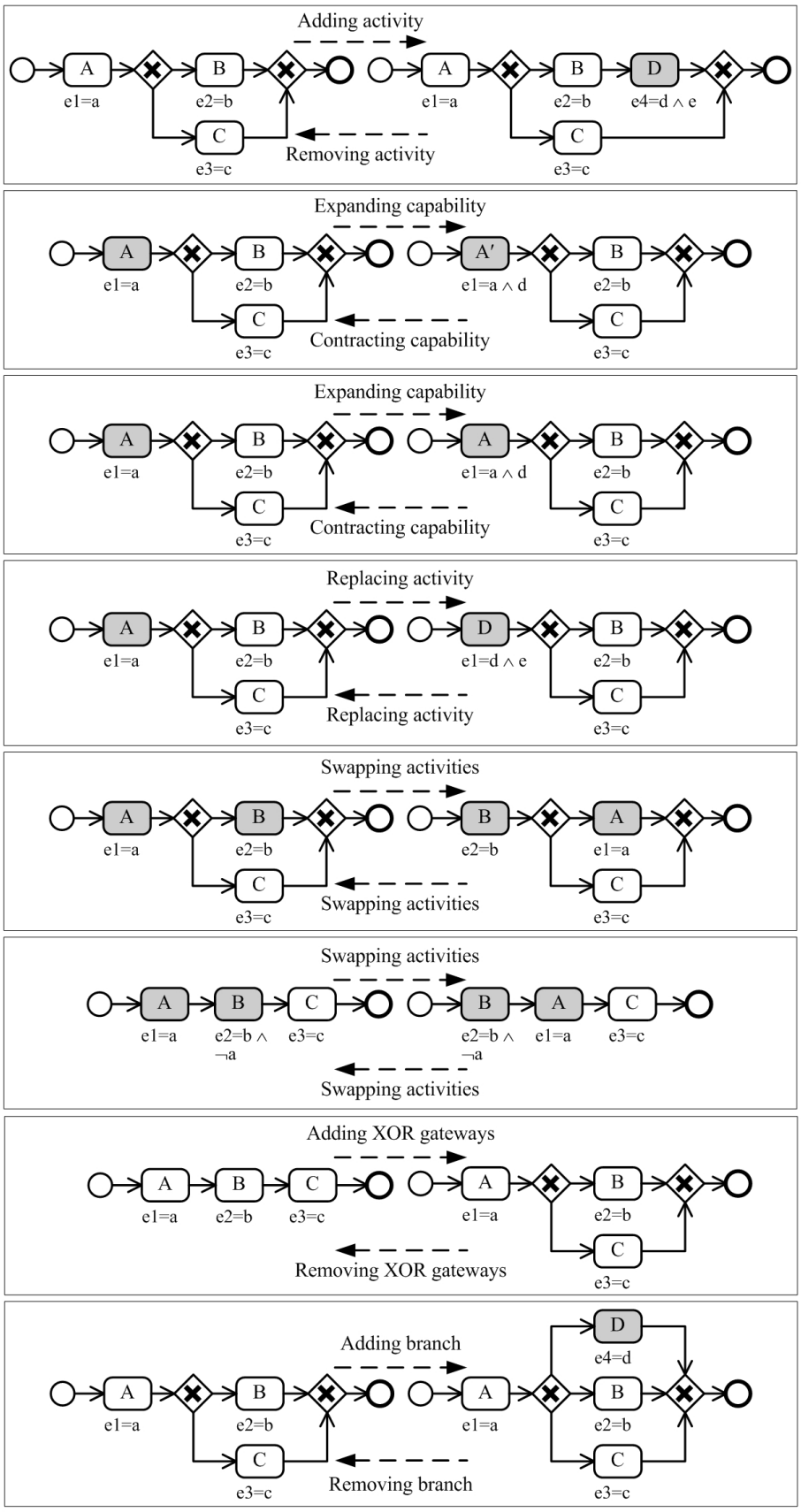


Figure 4.1: Examples of structural changes introducing end cumulative effects changes of a process

- **C-P6**: swapping activities in which their immediate effects undo to each other,
- **C-P7**: adding or removing XOR gateways,
- **C-P8**: adding or removing branch(es) of the existing XOR gateways.

Figure 4.1 illustrates these structural changes in which each process at the left side is the original process. Let Figure 4.1(a) be our example to analyze. We can compute its end cumulative effects $acc\,(P2)$ which has two effect scenarios, i.e. $es_{21} = a \wedge b$ and $es_{22} = a \wedge c$. If we change $P2$ into $P2'$ by inserting an activity D (with immediate effects $e_4 = d \wedge e$), in sequence with activity B, we would have $acc\,(P2')$ which consists of two effect scenarios, i.e $es'_{21} = a \wedge b \wedge d \wedge e$ and $es'_{22} = a \wedge c$. This change yields $acc\,(P2) \neq acc\,(P2')$. In opposite view, we can similarly analyze both processes if each process at the right side is considered as the original process. In a similar vein, we can also further analyze for the rest operations.

Note that capability repurposing (**C-P2**) and capability redefinition (**C-P3**) are special cases of activity replacement (**C-P4**). Further, we would not consider adding or removing AND gateways as operations to change the end cumulative effects of the part process since we do not consider the possibility of effect scenarios between AND branches being inconsistent, as described in Section 2.1.3. As such, we argue that the sources of these changes stem from the individual activity changes as described above.

Inter-operation

An inter-operation relationship violation occurs iff there exist a contradiction between effect scenarios of the cumulative effects at each pair of sender-receiver activities caused by changing a process (**C-I1**), as described in Section 3.1, which is exemplified[2] by Figure 4.2. Referring to this figure, the contradiction can be described as follows.

[2]Such violations can occur as the result of changing process in many different ways.

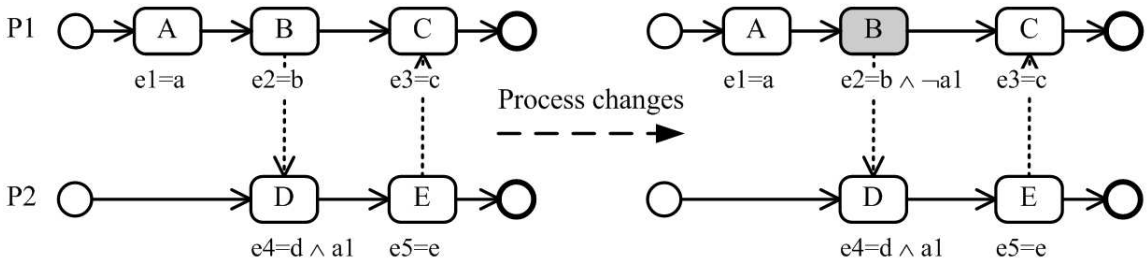


Figure 4.2: Illustration of an inter-operation relationship violation due to process changes: effects contradiction (**C-I1**)

There are two pairs of sender-receiver activities in an inter-operation relationship between processes $P1$ and $P2$, originally shown at the left side, i.e. activities $B - D$ and $E - C$. The computed cumulative effects of each process at these activities are $CE(P1, B) = \{\{a \wedge b\}\}$, $CE(P2, D) = \{\{d \wedge a1\}\}$, $CE(P2, E) = \{\{d \wedge a1 \wedge e\}\}$ and $CE(P1, C) = \{\{a \wedge b \wedge c\}\}$, respectively. Obviously, there is no contradiction between effect scenarios in $CE(P1, B)$ and $CE(P2, D)$, and in $CE(P2, E)$ and $CE(P1, C)$. Now, we would apply capability redefinition on activity B in $P1$, as shown at the right side. This change would get $CE(P1', B) = \{\{a \wedge b \wedge \neg a1\}\}$, which is, in turn, contradicting $CE(P2, D)$ such that $CE(P1', B) \cup CE(P2, D) \vdash \perp$.

Generalization-specialization

Let $P1$ and $P2$ be the generalized and the specialized processes, respectively, such that the constraints are satisfied: (i) $\forall es_i \in acc(P1)$, $\exists es_j \in acc(P2)$ such that $es_j \models es_i$; (ii) and $\forall es_j \in acc(P2)$, $\exists es_i \in acc(P1)$ such that $es_j \models es_i$. Since this relationship is a special case of the part-whole relationship, its basic structural changes will also be applicable.

(a) Changes on the generalized process. Transforming $P1$ into $P1'$ may violate the relationship iff $acc(P1) \neq acc(P1')$. The aforementioned structural changes **C-P1** to **C-P8** can be considered as the sources of such violation. We, however, argue that

changes resulted from removing activity (i.e. **C-P1**) and contracting capability (i.e. **C-P2** and **C-P3**) will not introduce any violation since any effect scenarios in $acc(P1')$ still satisfies the constraints. We can leverage illustrations shown in Figure 4.1, i.e. now viewed as the generalized process, to describe all possible violations triggered by changing $P1$. For example, in Figure 4.1(a), we have $acc(P1) \neq acc(P1')$ such that the constraints are violated. While, in the opposite view for the removing activity, we can conclude that changing the structure of the right-side original process does not violate the relationship. This is because the entailments specified in the constraints are still satisfied. Similarly, we can also analyze the rest operations.

(b) Changes on the specialized process. Changing $P2$ into $P2'$ may violate the relationship iff $acc(P2) \neq acc(P2')$. Specifically, the difference must occur at the shared capability with $P1$. This shared capability represents a section in the specialized process that produces outcomes similar to the ones resulted in the generalized process. We can determine this shared capability through effect accumulation computation. The aforementioned structural changes **C-P1** to **C-P8** at the shared capability can be considered as the sources of such violation. Hence, we can also leverage the illustrations given in Figure 4.1, i.e. now viewed as a section in $P2$ that shares capability with $P1$. Note that any change made to $P2$ beyond the shared capability will not violate the relationship.

4.2.2 Resolution patterns

The resolution of any relationship violation involves changing the structure of the processes (i.e. redesigning process models) participating in the relationships in question. Basically, resolving a relationship violation between two processes is, in fact, the propagation of changes made to one process to the others to preserve their relationships.

This resolution may involve the CL in providing any capability required for such resolution. Ideally, an enterprise would prefer to resolve such violation without changing the CL as any required capability already exists in the CL. This might be the case for fixing some violations. But in many other cases, we need to change it. Further, we assume that there exist relationship descriptors RD for all normative relationships established between processes in the process repository.

Further, we realize that redesigning a process model in our framework relies on the semantic effect unit of analysis (i.e. immediate effects, effect scenarios and cumulative effects). Such analysis is required in order to justify any change made to a process model, e.g. one or more activities should be added if the functionalities (i.e. effects) of a process expanded. As effect scenarios take an important part in such analysis, our resolution procedures require transformation of a given set of original effect scenarios E_p into a given set of target effect scenarios E_q of a process model. The former represents current effect scenarios of the process being changed, while, the latter denotes effect scenarios of the expected process yielded from such resolution. In such transformation, we need to correspond between these effect scenarios to exactly know which effect scenario in the original set to be transformed into which effect scenario in the target set. In this context, we shall refer to any effect scenario that is in correspondence *bounded effect scenario (BES)*. In addition, we may also have an effect scenario, either in E_p or E_q, that is not in correspondence. We shall refer to such effect scenario *unbounded effect scenario (UES)*. Detailed description and implementation of such effect scenarios correspondence can be found in Section 5.3.3 as a part of our semantic-guided process redesign implementation.

We classify our resolution patterns into general and specific patterns based on the inter-process relationship types. The former can be applied to resolve some violations occurred in many different inter-process relationship types, while, the latter will be

applicable to a certain inter-process relationship type. In the following, we describe separately both of them. In the general patterns part, we identify all resolution patterns which are generally applicable to many inter-process relationship types. While, in the subsequent part, we describe all resolution patterns which are specifically required in each inter-process relationship type, if any, as well as any general pattern which is applicable to such relationship type. Based on given two interrelated processes $P1$ and $P2$, we shall refer to resolution on $P1$ if we need to change $P1$ to resolve any violation stemming from $P2$ changes, and vise versa.

General patterns

Given two corresponded effect scenarios $es_i \in E_p$ and $es_j \in E_q$, we construct the following general resolution patterns in order to transform es_i to es_j:

- **(R-1: Remove activities)** A set of capabilities C_{del} should be removed from es_i iff $C_{del} \subset (es_i \cup es_j) \setminus (es_i \cap es_j)$ and $C_{del} \subset es_i$.
- **(R-2: Add activities)** A set of new capabilities C_{add} should be added into es_i iff $C_{add} \subset (es_i \cup es_j) \setminus (es_i \cap es_j)$ and $C_{add} \subset es_j$.

In order to apply pattern **R-2**, we firstly need to perform effect scenario refinement on es_i using pattern **R-1** for removing any capability which is irrelevant with respect to es_j. We use $es'_i = \phi(es_i)$ to denote a function ϕ for doing such refinement on es_i yielding es'_i. In this case, we operate our transformation procedure based on es'_i.

In regard uncorresponded effect scenarios, we construct other general resolution patterns as follows:

- **(R-3: Remove a group of activities)** A set of activities whose end cumulative effects have an effect scenario es_i should be removed iff es_i satisfies any $es_j \in E_{del}$, where E_{del} is a set of UESs and $E_{del} \subset E_p$.

- **(R-4: Arrange a group of activities)** A set of activities, which is initially empty, should be arranged such that there exists an effect scenario es_i in its end cumulative effects which satisfies any $es_j \in E_{add}$, where E_{add} is a set of UESs and $E_{add} \subset E_q$. Note that es_i is initially empty.

Patterns **R-2** and **R-4** involve searching algorithms applied in the CL in order to leverage any existing capability that can appropriately contribute to resolve any relationship violation. We reduce our search space by considering only a subset of the CL, denoted as CL', which is a set of activities whose immediate effects contribute, partly or completely, to such transformation. We construct a general formalization of such procedures defined in Definition 13. It guarantees that any capability involved in the resolution is in the capability domain, and permits us to conduct a finite search.

Definition 13. Resolution search problem. *Let es_i and es_j be initial and goal states of effect scenarios, respectively, for an effect scenario transformation. Let $CL' \subset CL$, which is a set of activities whose immediate effects can be considered to transform es_i into es_j, be a search domain. A resolution function is defined as $es_o = ACC_{res}(es_i, IE)$, where IE is a set of immediate effects of a series of activities $Acts \subseteq CL'$, for accumulating their effects such that $es_o \models es_j$.*

In our search problem, we use an heuristic search approach as we only consider any activity in the CL whose immediate effects probably contribute to achieve the goal effect scenario. Our search starts from the original effect scenario es_i which may have been refined from its irrelevant effects using pattern **R-1** with respect to es_j. Further, we proceed forward by considering each activity $act \in CL'$, whose immediate effects can be accumulated to closely achieve es_j. The search continues until es_j can be satisfied. There, however, might exist an activity $act \in CL'$ which can not be used to closely achieve es_j such that we will have a set of activities $Acts \subseteq CL'$ consisting only applicable activities in such search.

Applicable resolution patterns based on relationship types

In this part, we describe all resolution patterns which are applicable to each inter-process relationship type. It includes the general patterns, if required, and any specific resolution pattern, which can be specifically applied for resolving any inter-process relationship violation on such relationship type.

(i) Part-whole. Let $P1$ and $P2$ be the main and the part processes, respectively. Let t_i (with immediate effects e_{t_i}) be an insertion point of $P2$ in $P1$. Regardless the violation trigger, the resolution procedures will change $P1$ into $P1'$, $P2$ into $P2'$ and t_i into t_i' such that the constraints are resatisfied: (i) $CE\left(P1', t_i'\right) = CE\left(P1' \uparrow^{t_i'} P2', t_i'\right)$ for a context-dependent part-whole where t_i' may or may not be equal to t_i; or (ii) $e_{t_i'} = acc\left(P2'\right)$ for a context-independent part-whole, where $acc\left(P2'\right)$ has a set of effect scenarios E_2'.

(a) Resolution on the main process. This will transform $P1$ into $P1'$ due to changing $P2$ into $P2'$ using structural changes **C-P1** to **C-P8**. This causes $acc\left(P2\right) \neq acc\left(P2'\right)$, which, in turn, violate the relationship. Based upon the constraints, we consider two corresponding specific resolution patterns:

- **(R-5: Change at least the insertion point activity)** A set of activities T including t_i and some activities preceding t_i should be changed such that the first constraint resatisfied. This can be achieved in many different ways, e.g. capability repurposing and redefinition. This requires analyst involvement since it is not easy for a tool to deal with such resolution. The difficulties arise due to there might exist various distinct activities to be modified and many various ways to do such modifications.

- **(R-6: Replace the insertion point activity)** The insertion point t_i should

be replaced by any activity t'_i (with the immediate effects $e_{t'_i}$ and a set of effect scenarios E'_i) in the CL such that the second constraint resatisfied. This can be achieved iff the following two conditions hold: (i) all effect scenarios in both E'_i and E'_2 are BESs; and (ii) $\forall es_j \in E'_2$, $\exists es_i \in E'_i$ such that $es_j \models es_i$ and $\forall es_i \in E'_i$, $\exists es_j \in E'_2$ such that $es_i \models es_j$. If t'_i does not exist in the CL, it, however, can be introduced using capability repurposing or capability redefinition approaches.

Further, we can provide a suggestion to the analyst in determining the resolution strategy, as follows: (i) if some effect scenarios in e_{t_i} are violated by any effect scenario in E'_2, pattern R-6 should be taken; and (ii) if all effect scenarios in e_{t_i} are violated by any effect scenario in E'_2, patterns R-5 and R-6 would be the options.

(b) Resolution on the part process. This will transform $P2$ into $P2'$ due to changing $P1$ into $P1'$ using **C-W1** and **C-W2**. This causes $e_{t_i} \neq e_{t'_i}$, which, in turn, violate the relationship. We use the second constraint for the resolution based on patterns **R-1** to **R-4**. In this context, we rely on $acc\,(P2)$ and $e_{t'_i}$ which represent E_p and E_q used in such patterns, respectively.

(ii) Inter-operation. It will be more complex to resolve violations in an inter-operation relationship due to structural change **C-I1**. The resolution requires analyst involvement for a negotiation between roles participating in the relationship in order to determine an appropriate procedure for such resolution **(R-7: Analyst involvement)**. In such specific resolution pattern, once the agreement holds, the analyst can perform some structural changes (e.g. adding, removing, replacing or reordering activities and gateways).

(iii) Generalization-specialization. Let $P1$ and $P2$ be the generalized and the

specialized processes, respectively. Regardless the violation trigger, the resolution will change $P1$ into $P1'$ and $P2$ into $P2'$ such that the constraints are resatisfied: (a) $\forall es_i \in acc(P'1)$, $\exists es_j \in acc(P'2)$ such that $es_j \models es_i$; and (b) $\forall es_j \in acc(P'2)$, $\exists es_i \in acc(P'1)$ such that $es_j \models es_i$.

(a) Resolution on the generalized process. This will change $P1$ into $P1'$ due to changing $P2$ into $P2'$ using **C-P1** to **C-P8**. This causes $acc(P2) \neq acc(P2')$, which, in turn, may violate the relationship. We perform the resolution based on patterns **R-1** to **R-4**. In this context, we rely on $acc(P1)$ and $acc(P2')$ which represent E_p and E_q used in such patterns, respectively.

(b) Resolution on the specialized process. This will transform $P2$ into $P2'$ due to changing $P1$ into $P1'$ using **C-P1** to **C-P8**. This causes $acc(P1) \neq acc(P1')$, which, in turn, always violate the relationship. We perform the resolution based on patterns **R-1** to **R-4**. In this context, we rely on $acc(P2)$ and $acc(P1')$ which represent E_p and E_q used in such patterns, respectively.

4.2.3 Summary

In regard our taxonomy of process changes described earlier, we argue that such taxonomy is complete to identify all potential changes that can violate an inter-process relationship between processes. We describe this as follows.

In the part-whole relationship, we consider that if its constraints, for either a context-dependent or a context-independent described earlier, can not be satisfied due to any process change, such part-whole relationship has been violated. For changing the main process, we argue that our structural changes **C-W1** and **C-W2** are complete since any structural change made to such process beyond the insertion point activity

will not affect the cumulative effects of such insertion point, which, in turn, does not violate the relationship constraints. While, for changing the part process, we argue that our structural changes **C-P1** to **C-P8** are complete since any structural change made to such process (see, e.g. change patterns described in [67]) can be described into such identified structural changes.

In the inter-operation relationship, any contradiction occurs between effect scenarios of a pair of sender-receiver activities within processes in an inter-operation relationship will violate such relationship constraints. We argue that any structural change made to any point in such processes leads to a pair of sender-receiver activities introducing such contradiction has been covered by our identified structural changes **C-I1**. As such, any structural change made to such processes beyond any pair of sender-receiver activities will not introduce such contradiction, which, in turn, does not violate the relationship constraints.

In the specialization-generalization relationship, we argue similarly to that in the part-whole relationship. For changing the generalized process, any structural change made to such process (see, e.g. change patterns described in [67]) can be described into our changes taxonomy **C-P1** to **C-P8**. Further, this is also applicable for changing the part process at the shared activity with the generalized process. As such, we argue that any structural change made to the part process beyond such shared activity will not violate the relationship constraints.

In regard the resolution patterns, Table 4.1 summarizes all such patterns required to redesign a process model in order to resolve any relationship violation yielded from any structural change identified in the changes taxonomy. In addition, we argue that such any violation occurred in our process repository can always be resolved, as described in Theorem 1, as we consider to evolve the CL during our resolution process such that it provides any required activity.

Table 4.1: Resolution patterns summary

Relationship types	Change operations	Resolution patterns
Part-whole	C-W1 to C-W2	R-1 to R-4
	C-P1 to C-P8	R-5, R-6
Inter-operation	C-I1	R-7
Generalization-specialization	C-P1 to C-P8	R-1 to R-4

Theorem 1. Resolution availability. *There is always a sequence of resolution operations that will resolve any relationship violation.*

Proof 1. *We argue that for each activity t_i, with immediate effect e_{t_i}, which is used in any process model in a process repository, there exists a corresponding activity t_j, with immediate effect e_{t_j}, in CL. As such, any capability change introducing t'_i, with immediate effect $e_{t'_i}$, which may trigger any relationship violation will also evolve CL into CL' due to introducing t'_j, with immediate effect $e_{t'_j}$. Note that t'_i can be yielded from capability repurposing, capability redefinition or capability creation. Hence, such a relationship violation will always has its resolution due to the followings:*

- *resolution patterns **R-1** and **R-3** do not deal with any capability evolution in CL,*
- *resolution patterns **R-2, R-4** and **R-6** always have any capability required for the resolution since any effect $e_{t'_j}$, which is required to transform any original effect scenario es_i into any target effect scenario es_j, is already available in CL as the immediate effect of t',*
- *resolution patterns **R-5** and **R-7** always require analyst involvement in which any new capability may be introduced.*

4.3 Process redesign

We present our process redesign approach which is particularly developed in dealing with managing process changes in the process ecosystem. We develop this approach based on the resolution patterns introduced in Section 4.2, which essentially rely on the inter-process relationship types. Since this approach is not the main contribution of our research presented in this dissertation, we, however, intend to implement our process redesign approach to take a part in validating our process ecosystem approach through some experiments in dealing with process change propagation.

Basically, we consider two important aspects of a process model, i.e. semantics and structure. The semantics of a process model refer to any description of the meaning of such process model. In our conception, this semantics refer to the cumulative effects either at any point within or at the end of a process model. While, the structure of a process model relates to any description of the organization of such process model with respect to the coordination between its elements. In regard process redesign, we can perform a change to both semantics and structure of a process model. We realize that there exist a strong connection between them, i.e. changing the semantics of a process model always requires its structural change, nevertheless a structural change does not necessarily affect the semantics of such process model. For example, let two consecutive activities A and B with the immediate effects a and b, respectively, simply compose a process model. Changing the structure of such process model by reordering its activities such that B precedes A does not change the original semantics of such process model.

As our resolution patterns rely on the semantic analysis of process models, any process redesign approach built on the top of them will basically involve structural change of the processes. Based on a restricted subset of BPMN modeling framework described in Section 2.1.3, we consider four distinct types of change on such process

structure, which, in turn, can be viewed as the basic change operators, i.e. adding activity, removing activity, adding a pair of XOR gateways and removing a pair of XOR gateways. We do not take into account adding and removing any AND gateway as basic change operators since they do not essentially change the semantics (i.e. functionalities) of a process model, rather affect its non functional measures of such process, e.g. performance (i.e. execution time), compliance (i.e. activity ordering constraint). Each basic change operator imposes typically certain condition in which such operator is applicable, as follows.

- **Adding activity.** Adding an activity, which exists in the capability library, to a process model is basically introducing a new set of effects to the end cumulative effects of such process model. We consider to add an activity to a process model under the following conditions: (i) capability replacement or (ii) capability addition. Capability replacement is basically removing the old capability and adding the new capability at the former's place. These two capabilities may have either totally or slightly different effects. Such slightly different effects may be resulted from the capability repurposing or capability redefinition. While, capability addition is introducing a new capability such that it enriches the functionalities of a process model.

- **Removing activity.** Removing an activity from a process model is basically reducing a set of effects of the end cumulative effects of such process model, which, in turn, downgrading the capability of the process. We consider to remove an activity if its effects are not longer required to contribute to the functionalities of a process model.

- **Adding a pair of XOR gateways.** Adding a pair of XOR gateways is basically expanding the alternative paths of executing a process model. We consider

to implement it if we need to represent various distinct outcomes of a process execution.

- **Removing a pair of XOR gateways.** Removing a pair of XOR gateways is basically reducing the alternative paths of executing a process model. This can be considered if such process model has less alternative outcomes, i.e. compared to its original, during its execution.

In addition, in redesigning a process in our process ecosystem approach, we consider both manual and automatic techniques. We use the former technique to redesign any process involved in a violated inter-operation relationship. In such technique, instead of redesigning a process model in isolation within its *local* context, we consider its *global* context such that any constraint relates to its inter-operation with other processes can be preserved. While, the latter technique will be used in redesigning any process involved in violated part-whole and generalization-specialization relationships. In such technique, we can use any search algorithm to identify any alternative of redesigning a process model. Further, we can also use semantics of a process model to guide such redesigning process. Detailed description and implementation of this semantic-guided process redesign can be found in Section 5.3.3 from which an approximation to the process redesign approach can be leveraged in our process ecosystem experiments to deal with process change propagation.

4.4 Changes propagation in process ecosystem

As constraint networks have been leveraged into process ecosystem representation, we may have opportunities to utilize any technique used in solving constraint problems in such constraint networks. Such a well-known technique is CSP approach in which a constraint problem is solved when each variable has a value that satisfies all the

constraints on the variable [49]. Similarly, in process ecosystem, changes initially made to a process may cause disequilibrium in the ecosystem in the form of critical inter-process relationships being violated from which constraint problems occur. In this view, a process ecosystem is considered to be in an equilibrium if its all inter-process relationships are mutually consistent. This can be achieved by redesigning processes (i.e. propagating the initial changes) involved in such violated relationships across the ecosystem. Therefore, change propagation is reduced to finding an equilibrium in a process ecosystem.

A CSP consists of a set of variables X, for each variable there exists a finite set of possible values D, and a set of constraints C restricting the values that the variables can simultaneously take [2]. Each constraint defines a relation between a subset of variables and constraints the possible values for the involved variables. We consider a constraint involving only one variable as a unary constraint, two variables as a binary constraint, and so on.

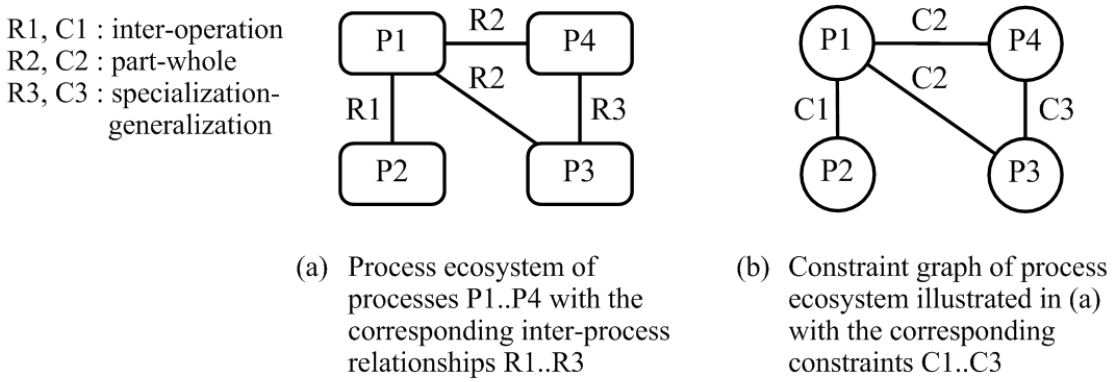
A solution to a CSP is an assignment of a value from its domain to every variable, in such a way that every constraint is satisfied. We may want to find only one solution, all solutions or an optimal solution [2]. Solutions to a CSP can be performed by searching systematically for the possible values which can be assigned to a variable. There are generally two search methods. The first method involves either traversing the space of partial solutions [2] (e.g. backtracking, backjumping and backmarking algorithms) or reducing the search space through constraint propagation (i.e. look-ahead). In variable selection, the look-ahead strategy seeks to control the size of the remaining search space. In value selection, it seeks a value that is most likely to lead to a consistent solution. Performing constraint propagation at each variable will result a smaller search space, but the overall cost can be higher since the cost for processing each variable will be more expensive. The second method involves repairing

an inconsistent complete assignment or solution (e.g. min-conflicts and repair-based algorithms [41])

Empirically, it is shown that ordering variables for value assignment can have substantial impacts on the performance of finding CSP solution [2]. The ordering variables could be either: (i) *static ordering*, the order of variables is defined before the search starts and not be changed until the search complete or (ii) *dynamic ordering*, the next variables to be assigned are dynamically defined at any point depends on the current state of the search. There are some heuristics in selecting variable ordering, i.e. variable with the smallest domain (in dynamic ordering) or variable which participates in the most constraints.

We argue that maintaining the equilibrium in a process ecosystem can be casted as a binary CSP. We can build a constraint network [12], which is represented in a constraint graph, of the process ecosystems to depict a binary CSP. In such network, each node represents a process model, all possible variants of redesigning a process denote the domain value of each node and each edge represents a relationship constraint between processes, as illustrated in Figure 4.3. Process $P1$ in Figure 4.3(a) can be mapped into a node $P1$ in Figure 4.3(b), as well as its relationship to process $P2$, i.e. $R1$, which is mapped into an edge between nodes $P1$ and $P2$, and so forth for the remaining processes and relationships. Indeed, the domain value of each node is finite since there exist constraints (at least, end cumulative effects and activity temporal constraints) that must be satisfied by process variants.

We consider constraint graph of a process ecosystem as a tuple $G_e = (V, C)$ where V and C denote a set of process nodes and a set of relationship constraints between processes, respectively. In the resulting constraint graph G_e, each process node is of the form $(id, T, E, G, start, end)$ where $id, T, E, G, start, end$ represent ID, set of activities, set of edges, set of gateways, start and end events of a process, respectively.



(a) Process ecosystem of processes P1..P4 with the corresponding inter-process relationships R1..R3

(b) Constraint graph of process ecosystem illustrated in (a) with the corresponding constraints C1..C3

Figure 4.3: Transforming a process ecosystem into a constraint graph

Further, each relationship constraint is of the form $(id, source, target, type)$ where $id, source, target, type$ represent ID, source node, target node and type of an inter-process relationship, respectively. An equilibrium in the process ecosystem then is considered as a solution in CSP once all constraints are satisfied by value assigned to each node, i.e. a variant of each process. We might not have a solution for a given perturbed-equilibrium process ecosystem since there does not exist a variant of a particular process node for resolving the violations[3].

We propose two algorithms, i.e. *repair* and *constructive*, for generating a restored-equilibrium process ecosystem, as shown in Algorithms 1 and 2, respectively. The analyst can perform either one or both of them to generate a restored-equilibrium process ecosystem. We implement dynamic ordering in searching process to be evaluated through one which participates in the most constraints, represented by ***GetNextProcess*** function. We search a process in the perturbed-equilibrium process ecosystem which is in the following conditions: (i) not yet evaluated, (ii) violates its relationship constraints with the previously evaluated processes and (iii) participates in the most constraints.

In the repair mode, inspired by Min-Conflicts algorithm [41], we search the new

[3]Finding an optimal solution would be our next investigation.

Algorithm 1: Generating a restored-equilibrium process ecosystem : *repair* mode.

Input:
p, a changed process model
CL, capability library
PE_p, graph of a perturbed-equilibrium process ecosystem
Result: a restored-equilibrium PE_x or *null*

```
1  begin
2      V_done, a set of evaluated process identifiers, initially empty
3      p_var, the selected variant of a redesigned process, initially null
4      p_m, the process to be changed, initially null
5      PE_x ← PE_p; p_var ← p;
6      V_done ← V_done ∪ {identifier of p};
7      p_m ← GetNextProcess(PE_x, V_done);
8      while p_m ≠ null and p_var ≠ null do
9          p_var ← ProcessChangeForMinConflicts(p_m, PE_x, V_done, CL);
10         if p_var ≠ null then
11             replace p_m in PE_x by p_var;
12             V_done ← V_done ∪ {identifier of p_m};
13             p_m ← GetNextProcess(PE_x, V_done);
14         else
15             PE_x ← null;
16         end
17     end
18     return PE_x;
19 end
```

equilibrium of process ecosystems by minimizing conflicts between variants of process being changed with the other processes which are not yet evaluated, and satisfying relationship constraint with the previously evaluated processes. It is represented by ***ProcessChangeForMinConflicts*** function which searches a variant of changed process by satisfying the following criteria: (i) satisfies all relationship constraints with the previously evaluated processes; and (ii) has the minimal violations with all the rest processes in the ecosystem. This function leverages process redesign function for generating a variant of a process being evaluated. Finally, we would select a variant which has the minimum conflict and continue until all constraints satisfied, shown in

Algorithm 2: Generating a restored-equilibrium process ecosystem : *constructive* mode.

Input:
p, a changed process model
CL, capability library
PE_p, graph of a perturbed-equilibrium process ecosystem
Result: a restored-equilibrium PE_x or *null*

```
1  begin
2      V_done, a set of evaluated process identifiers, initially empty
3      p_var, the selected variant of a redesigned process, initially null
4      p_m, the process to be changed, initially null
5      PE_x ← PE_p; p_var ← p;
6      V_done ← V_done ∪ {identifier of p};
7      p_m ← GetNextProcess(PE_x, V_done);
8      while p_m ≠ null and p_var ≠ null do
9          p_var ← RedesignProcess(pm, PE_x, V_done, CL);
10         if p_var ≠ null then
11             replace p_m in PE_x by p_var;
12             V_done ← V_done ∪ {identifier of p_m};
13             p_m ← GetNextProcess(PE_x, V_done);
14         else
15             I_pm ← a path running from p to p_m;
16             p_mdb ← the preceding of p_m in I_pm ;
17             if p_mdb = p then
18                 PE_x ← null;
19             else
20                 remove identifier of p_mdb from V_done;
21                 p_m ← p_mdb;
22                 p_var ← p_mdb;
23             end
24         end
25     end
26     return PE_x;
27 end
```

Algorithm 1.

In the constructive mode, inspired by Graph-based backjumping algorithm [10], we search a new equilibrium of a process ecosystem by redesigning the process being evaluated to satisfy its constraint with the previous evaluated process until all constraints

are satisfied. This is represented by ***RedesignProcess*** function, which was previously introduced in Section 4.3, for searching a variant of process being evaluated satisfies all relationship constraints with the previously evaluated processes. Once there is no variant of the process being evaluated satisfies the constraint, we would jump back to the most recent related process (with respect to the process being evaluated) which is already evaluated, as shown in Algorithm 2. Then, this recent process should be further redesigned to allow the subsequent evaluated process satisfies its relationship constraints.

4.5 Summary

In this chapter, we have presented our approach in managing change in process ecosystem including capability library, process changes taxonomy and resolution patterns, process redesign and changes propagation in process ecosystem. Our capability library contains all activities used in any process model in the process repository. Managing such library becomes critical since it represents the competencies of an organization and can be further leveraged in redesigning process models. We have introduced our taxonomy of process change that can trigger inter-process relationship violations from which resolution patterns can be constructed in order to resolve any inter-process relationship violation. Further, we have discussed our process redesign approach which is built on the top of our resolution patterns. Finally, we have proposed our approach in dealing with propagating process changes in process ecosystem by leveraging all concepts which have been previously discussed.

Chapter 5

Implementation

This chapter describes the implementation of our approach which has been introduced in previous chapters. We call our tool supporting process ecosystem concepts as P-Gamelan[1]. This chapter covers overview of its system architecture described in Section 5.1, description of the effect accumulation engine and its used respective inference engine presented in Section 5.2 and all packages which contribute in performing functionalities required to propagating process changes described in Section 5.3. Further, as a proof-of-concept, we implemented our process ecosystem approach to deal with change propagation in a collection of business processes within Eclipse[2] environment. Note that, at this preliminary development, we however do not provide any graphical user interface (GUI) as it is intended to be our future improvement.

5.1 System architecture

The architecture of P-Gamelan is illustrated in Figure 5.1 which consists of a set of functionalities and data storages. Functionalities constituting P-Gamelan are classified

[1]P-Gamelan stands for 'Process Gamelan'. Gamelan is a traditional musical ensemble from Indonesia, featuring a variety of instruments. It is composed of a number of instruments as a distinct object, arranged and tuned to stay together.

[2]Eclipse homepage `http://www.eclipse.org/`

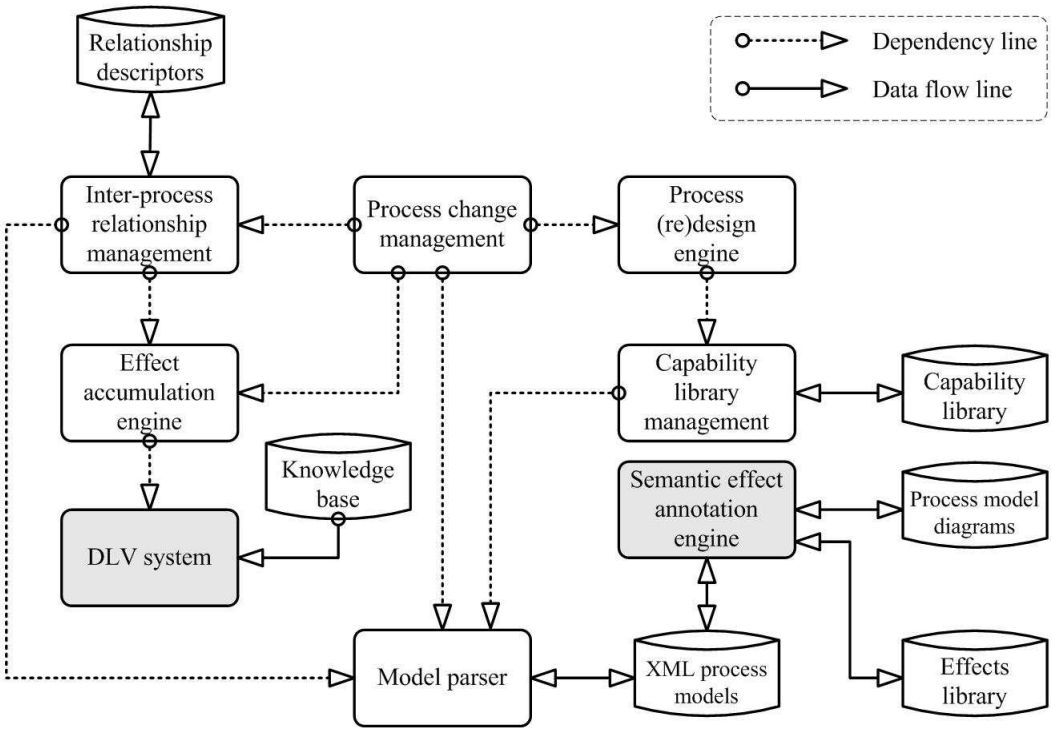


Figure 5.1: System architecture for process ecosystem tool support

into two categories, i.e. developed engines and off-the-shelf toolkits. In the developed engines, some required functionalities are provided: inter-process relationship management, effect accumulation engine, process change management, process (re)design engine and capability library management. While, in the off-the-shelf toolkits, two toolkits developed by other parties are included: DLV system[3] and semantic effect annotation engines. DLV system is a disjunctive logic programming system, which is used for effects reasoning. In the semantic effect annotation engine, ProcessSEER [26] has been used for annotating each activity in a process model with immediate effects. We differentiate the two-grouped functionalities with colored and transparent rectangles, i.e. the former refers to the off-the-shelf toolkits, while, the latter refers to the developed engines. In addition, data maintained in P-Gamelan are separated into various distinct storages including relationship descriptors, knowledge base, capability

[3]DLV system homepage `http://www.dlvsystem.com/dlv/`

library, effects library, process models represented both in XML and diagram files. In particular, we use distinct lines to represent relationship between elements in our system architecture, i.e. dashed line denotes dependency relationship among functionalities and solid line represents data flow between functionalities and respective data storages. We describe each element in more detail in the following sections.

5.2 Effect accumulation engine and model parser

Effect accumulation engine[4] has been developed separately from P-Gamelan such that it can be used in different other frameworks which deal with such effect accumulation function in any process model. This engine is used for accumulating the effects at any point in the process model if it were executed until such point. We use the pair-wise accumulation procedure introduced in Section 2.1.3. Such effect accumulation engine takes an important part in assessing any violated inter-process relationship in process change propagation in process ecosystem. We implemented our effect reasoning in such pair-wise accumulation procedure by leveraging a theorem-prover developed under answer set programming (ASP) paradigm, i.e. DLV system.

In addition, we also developed model parser to be a part of P-Gamelan, which is used to transform a process model represented in XML format into a process graph in our framework, and vise versa. On the one hand, we can construct such process graph to be further manipulated in any functionality performed by P-Gamelan. On the other hand, we may transform such process graph resulted from any process redesign into an XML-based description of process model which can be a modification to the existing description. Such XML-based description can be further transformed into respective process diagram using process modeling tool.

[4]The corresponding deliverables including downloadable program, API docs and demos can be accessed at `http://www.dsl.uow.edu.au/~triak/effect-accumulator/deliverables/effect-deliverables.html`

5.3 P-Gamelan packages

P-Gamelan has been equipped with packages that are important to support its capabilities for dealing with process change propagation in process ecosystem. These packages include inter-process relationship management, capability library management, process (re)design engine and process change management. We describe them in more detail as follows.

5.3.1 Inter-process relationship management package

This package is used to manage inter-process relationships between process models in the repository including their establishment and maintenance. It is also used for assessing inter-process relationship constraints satisfaction during propagating changes in a process ecosystem. Establishment is performed to populate relationship descriptors based on any inter-process relationship which may exist among processes in the repository. Along the life cycle of a business process, its relationships to other processes may change. As such, relationship descriptors need to be maintained to keep it consistence with the current dependencies between processes in the repository. We implement two modes in such management, i.e. checking and generating, which require process analyst involvement to make judgments to any suggested inter-process relationship. It uses model parser engine to transform the structure of a process model represented in the XML format into a process graph. It also uses effect accumulation engine to compute cumulative effects at any point in the process. Such effect computation will be used to assess any inter-process relationship constraints satisfaction which may exist between a pair of processes.

5.3.2 Capability library management package

This package is provided to manage our capability library consisting of all activities used in any process model in the process repository. This library becomes important for an organization as it represents its competences. Such management includes establishment and maintenance of the specification of such activities. Establishment concerns in populating such capability library with activities used in any process model in the repository. Along the life cycle of a business process, its structure may change with respect to its activities, i.e. a new activity can be introduced, an existing activity can be removed or modified. As such, this capability library need to be maintained such that it satisfies the library consistency constraints. This library will take an important part in redesigning a process model during propagating change in a process repository. It uses model parser engine to transform the structure of a process model represented in XML format into a process graph for further structural extraction to provide specification of activities which are involved in such process model.

5.3.3 Process (re)design engine package

A semantic-guided process redesign has been developed as a simple approximation to process redesign approach for supporting our process change propagation in process ecosystem. This engine is used to redesign any process model involved in violated part-whole and generalization-specialization relationships in such change propagation. We firstly reconstruct the semantics of our current process by leveraging our capability library with respect to the semantics of the final process and continue with constructing a new final process model based on the resulting reconstructed semantics. Finally, we can infer that any basic change operator has been applied to redesign our current process. To this end, we perform two important tasks in such process redesign, i.e. semantic effect reconstruction and process model transformation, described as follows.

Semantic effect reconstruction

It concerns transforming any relevant effect scenario in the current process into its corresponding effect scenario in the final process model. In between, we introduce an intermediate effect scenario required in such transformation. In order to do so, we have to correspond any relevant effect scenario in a given set of the original effect scenarios E_p of the current process with any effect scenario in a given set of the target effect scenarios E_q of the final process model. Such correspondence is a critical task in our approach as we can clearly understand the current and final states of such effect scenarios transformation. We refer to a relevant effect scenario in E_p as an effect scenario in E_p that has a corresponding effect scenario in E_q. Such effect scenarios correspondence relies on the effect scenarios distance measure $d\left(es_j, es_i\right) = \frac{|es_j \Delta es_i|}{|es_j \cup es_i|}$, for any $es_j \in E_q$ and any $es_i \in E_p$. We adopted such measure from *Jaccard distance* [14, 17] which is widely used in data sets. $|es_j \Delta es_i|$ represents set cardinality of symmetric differences between es_j and es_i. In this context, we will select the most minimum distance between many competing pairs of effect scenarios in order to correspond two effect scenarios. Further, we realize that the value of such distance measure will be in the range of 0 to 1. Intuitively, value 1 represents that two effect scenarios in question are absolutely different. In contrast, value 0 means that the two effect scenarios are exactly the same. So, any distance measure which falls in between these numbers represents either degree of difference, if it moves towards 1, or degree of similarity, if it moves towards 0. Note that we ignore any pair of effect scenarios whose distance measure equals 1 since they are not in correspondence. We, however, take into account all effect scenarios of the final process model in such effect scenarios transformation as they become the reference of such transformation.

Once such effect scenarios correspondence has been done, we further perform effect scenario transformation. We, however, need to clearly specify the differences between

current, intermediate and final effect scenarios terms which will be used throughout this section. In our approach, the *current effect scenario (CES)* is any effect scenario which is derived from the current process model being redesigned. This is an *as-is* effect scenario. While, the *final effect scenario (FES)* is any effect scenario created to build a new variant of the current process. This is a *to-be* effect scenario. In this context, each effect scenario is a set of effect literals (e.g. *window_closed*, *door_opened*).

Basically, our approach deals with transforming CES into the FES by leveraging the CL. We assume that there exist a number of activities in the CL including all activities used in the current and final process models. Further, in some instances, we realize that a CES may contain effects which are irrelevant with respect to the corresponding FES. All irrelevant effects must be removed from the current effect scenario since they are not required in our final redesigned process. Hence, such current effect scenario should be adjusted such that it only contains retained effects in accordance with the final effect scenario. We, then, refer our *intermediate effect scenario (IES)* to this adjusted effect scenario. In this context, we actually transform the IES into the FES. The results of this semantic reconstruction are a set of reconstructed effect scenarios represented in CNF, denoted as CE_i, and a set of all activities required to achieve CE_i, denoted as $Acts_i$. These two sets will be used in process model transformation. We, however, provide different such reconstruction procedures based on the resolution patterns, i.e. redesigning main process, redesigning part and specialized processes and redesigning generalized process.

Process model transformation

This transformation focuses on constructing a new process model based on a reconstructed effect scenarios CE_i and a set of all activities required to achieve CE_i, i.e. $Acts_i$, which are yielded from our semantic effect reconstruction. We leverage process

graph as the intermediate representation for such transformation. In this transformation, we view each effect scenario as a set of sequence effect literals without considering any undo and negation effects as our process redesign engine is intended for the purpose of process ecosystem experiments. As such, we assume that the occurrence of effects in the reconstructed effect scenarios exactly represents the execution order of activities in a process model since such effects are yielded from executing such activities accordingly. We develop such transformation through the following procedures:

1. Identify any possible effects which commonly occur in the main path and each sub paths. Common effects in the main path can be identified by observing such effects that occur in all effect scenarios in CE_i. While, common effects in the sub paths can be identified based on any effect which commonly occurs in a set of paths branching from the main, and so forth.

2. Continue to observe the rest effects which are specific to a certain effect scenario.

3. Correspond each observed effect with any capability in $Acts_i$.

4. Draw a process graph. If $|CE_i| > 1$, we consider to introduce pairs of XOR gateways to introduce multiple paths in the graph. Otherwise, there exists a single path in the graph. An XOR split gateway, from which a set of sub paths initiated, should be placed immediately after the identified common effects. While, an XOR join gateway, from which a set of sub paths terminated, should be placed immediately before the identified common effects. Each node in the process graph corresponds to any capability in $Acts_i$ or any XOR gateway. Add an additional node at each the beginning and the end of such process graph to represent empty start and empty end events, respectively.

5. Translate such constructed process graph into a process model represented in BPMN.

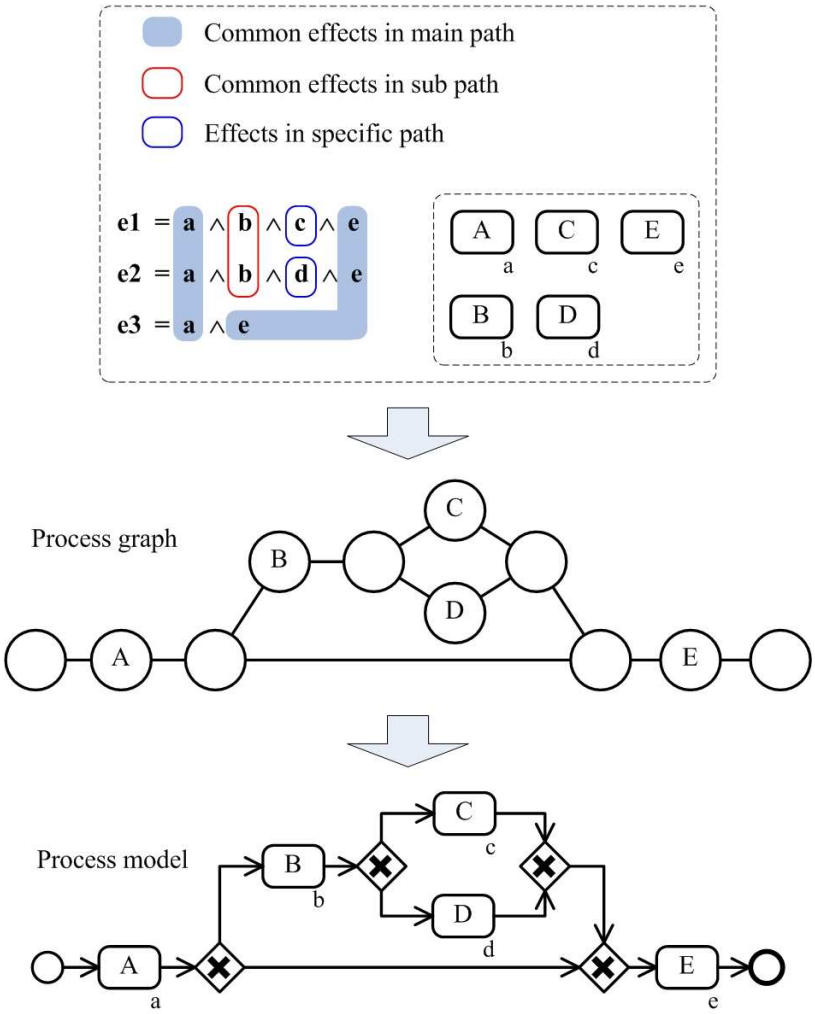


Figure 5.2: Process model transformation procedure using illustrative process

We, however, realize that inserting XOR split or join gateways should be taken carefully, since it is not always required if there exists a sub-process in $Acts_i$ whose immediate effects involve multiple effect scenarios and entail any effect being observed. Figure 5.2 illustrates our process model transformation procedures described above involving $CE_i = \{es_{i1}, es_{i2}, es_{i3}\}$ where $es_{i1} = a \wedge b \wedge c \wedge e$, $es_{i2} = a \wedge b \wedge d \wedge e$ and $es_{i3} = a \wedge e$ as well as $Acts_i = \{A, B, C, D, E\}$.

5.3.4 Process change management package

This package is used to manage our change propagation in a process ecosystem[5]. As an initial change made to a process model should be properly propagated across process ecosystem, we develop this package to be able to coordinate many tasks which are involved in such propagation. This includes transforming XML-based description of process model into its corresponding process graph, retrieving any respective inter-process relationship type from the relationship descriptors according to the process model being observed, computing the cumulative effect required to asses any violated inter-process relationship in the ecosystem and redesigning a process model required in resolving any violated inter-process relationship. To this end, it leverages model parser engine, inter-process relationship management package, effect accumulation engine and process (re)design engine, respectively. Further, it also uses such model parser engine to construct a new XML-based description of process model based on the redesigned process model.

We provide two modes in propagating change in process ecosystem, i.e. constructive and repair. We develop such modes inspired by CSP technologies for dealing with any constraint problem such as preserving any inter-process relationship within a process ecosystem while propagating process change in such ecosystem. We implement dynamic ordering for selecting a process model in the ecosystem to be firstly observed among competing process models which are probably affected by change made to a process model. As such, we prioritize a process model involved in most constraints to be observed if its relationship to the changed process has been violated.

[5]The corresponding deliverables including downloadable program, API docs and demos can be accessed at `http://www.dsl.uow.edu.au/~triak/pgamelan/deliverables/pgamelan-deliverables.html`

5.4 Data storage

We maintain several data in our storage to support P-Gamelan's functionalities as follows. They are stored using the file system.

- **Knowledge base.** This data represents implication relationships between effects which is required in performing effects reasoning through ASP inference engine. Such relationships are written in simple format contained in a text file system. For example, we use $door\_closed \rightarrow movie\_start \ \& \ light\_off$ to represent that effect *door_closed* implies the occurrence of effects *movie_start* and $light\_off$.

- **Effects library.** This library contains effects used in any activity performed by the organization. It provides options to the process analyst during annotating a process model with the semantic effects. As such, annotation can be done using consistent terms to refer to the same effects. This library evolves as the increasing number of activities performed by the organization.

- **Capability library.** This library consists of all activities used in any process model in the process repository. We view activity in such library as activity type from which activity objects can be derived to be used in any process model.

- **Relationship descriptors.** This data describes all inter-process relationships established in the process repository. Each pair of processes has a type of inter-process relationship which might be enriched with additional information, i.e. insertion point for a part-whole relationship.

- **XML process models.** This represents a collection of process models described in their XML-based documents. As we rely on BPMN modeling, these data are generated by BPMN modeling tool.

- **Process model diagrams.** This represents a collection of process models described in diagram documents. Similar to its XML-based documents, these data are generated by BPMN modeling tool.

5.5 Summary

In this chapter, we have described the implementation of our process ecosystem approach in dealing with process change propagation in a collection of business processes in process repository. This includes introducing the system architecture which is called as P-Gamelan, and describing all related elements which constitute the functionalities of P-Gamelan. We, however, realize that this implementation has some limitations as it is a preliminary development of P-Gamelan for being used in our experimental purposes.

Chapter 6

Evaluation

We have implemented our process ecosystem approach in dealing with process changes propagation in Chapter 5. While, this chapter presents an empirical evaluation which has been performed through some experiments in order to evaluate our process ecosystem approach and its implementation. We focus on two evaluation measures: efficiency and accuracy. Our experiments relied on different sizes of process ecosystems which were determined by their number of process models. Our efficiency measure relied on the average elapsed time of reestablishing the equilibrium of each process ecosystem due to process changes. We measured the accuracy of our approach by comparing its results with our manual analysis results with respect to any process model which should be redesigned for propagating the changes made initially on a particular process model. More specifically, in Section 6.1, we first illustrate the pre-evaluation state of a certain process repository in which a process change occurs in order to show the condition before our process ecosystem approach applies. In order to allow us perform an empirical evaluation on the implemented approach, we describe our experiment set up and execution procedure in Section 6.2. In Section 6.3, we provide experiment results. Finally, we describe the post-evaluation state and analysis in Section 6.4.

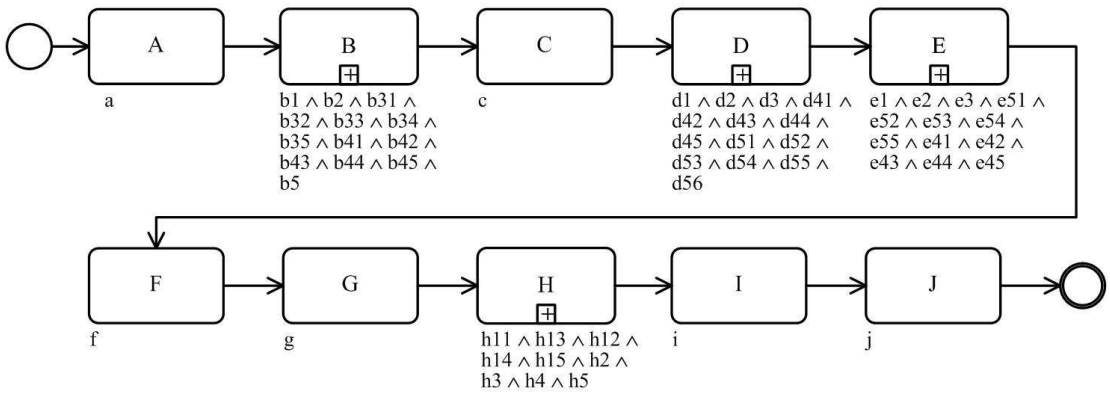


Figure 6.1: The original design of a semantically effect annotated process model $P1$

6.1 Pre-evaluation state

In this section, we illustrate the condition before performing our process ecosystem approach in dealing with process changes in a certain process repository. Such a repository is composed by a number of process models which are related to each other through any inter-process relationship type described in Section 3.1. We have a process model $P1$ in our process repository, as shown in Figure 6.1, which will be changed. It is composed by 10 activities including 4 sub-processes and 6 tasks in which each activity has been annotated with effects. We, then, apply activity repurposing on activities B and D by introducing new effects $b6$ and $d57$, respectively, such that it becomes a process model illustrated in Figure 6.2. Since we do not have any inter-process relationship descriptors as used in process ecosystem framework, we have to manually analyze the impact of changes made to $P1$ with respect to other process models in our process repository.

Due to changes made on the sub-processes, we argue that the underlying process models which represent the capability of such activities must be considered to be modified in order to preserve the part-whole relationships between the 'whole' process model $P1$ and its 'part' process models. As such, we have to identify two process

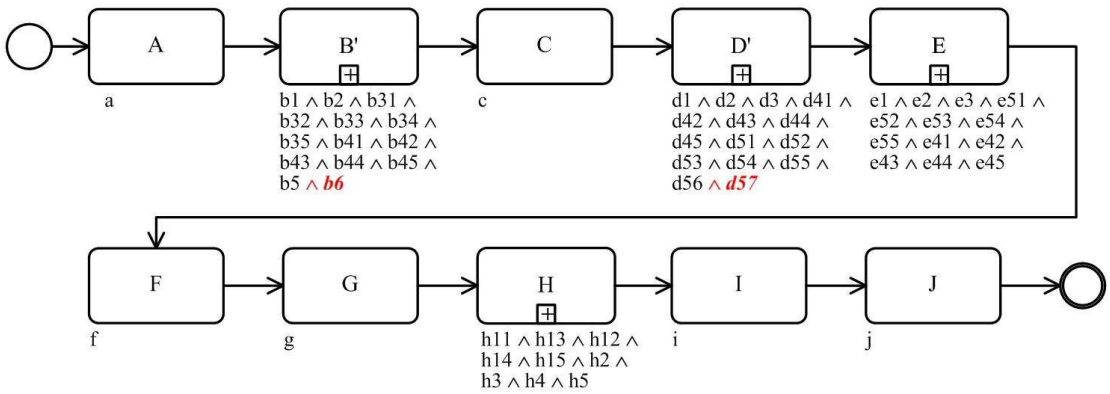


Figure 6.2: The changed design of the process model shown in Figure 6.1 using capability repurposing on activities B and D to be B' and D', respectively

models represent sub-processes B and D, respectively. Once they are found, i.e. process models $P11$ and $P12$, they have to be changed such that their end cummulative effects satisfy new effects of activities B and D, as shown in activities B' and D' in Figure 6.2. Changing such underlying process models may affect the other process models in the repository which are in relation with these processes once there exists a violation on a certain inter-process relationship. Obviously, we propagate the initial changes made to process model $P1$ to the rest of process models in order to preserve all inter-process relationship constraints. However, it will take time to do so as it is performed by the process analyst in a manual fashion. The bigger number of process models maintained in a process repository, the longer time required to manage changes in such a process repository. In the next section, we will evaluate the application of our process ecosystem framework in dealing with process changes in a process repository.

6.2 Experiment set up and execution procedure

In this section, we describe our experiment set up for supporting proof of our process ecosystem approach in dealing with process changes propagation. This includes the

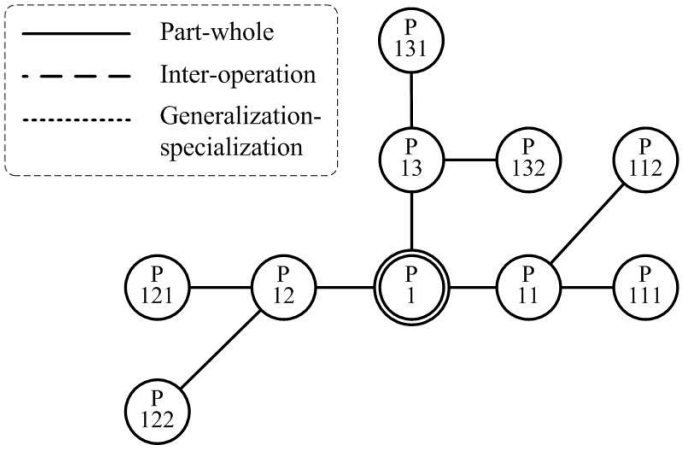


Figure 6.3: A constraint graph represents a process ecosystem consisting of 10 various processes where process $P1$ becomes the changes trigger

following set up: process ecosystems, relationship descriptors, capability library and execution procedure of the experiments. In general, we intended to have 8 distinct process ecosystems with different sizes of processes which were modeled in BPMN. We annotated each activity in each process model with its corresponding immediate effects. We leveraged our developed effect accumulation engine to compute the cumulative effects at any activity being observed. In addition, we applied both repair and constructive modes in our measures in order to assess how such approaches perform on different sizes of process ecosystems. Assessing such performance of each approach on different sizes of process ecosystems requires us to make their process models be comparable. This can be achieved by expanding the number of process models such that the smaller ecosystem is a subset of the bigger ecosystem. We describe our experiment set up as follows.

6.2.1 Process ecosystem set up

We randomly generated 8 process ecosystems with different sizes. We started with a process ecosystem having 10 interrelated processes. For generating the second process

ecosystem consisting 20 process models, we expanded the first generated ecosystem by appending 10 process models which have relationships to any existing process model. Similarly, we applied such random generation to the rest ecosystems until the last process ecosystem consisting 80 process models, as shown in Table 6.1. Further, our process models had various number of activities, i.e. between 5 to 20 activities including sub-processes and tasks. In each activity, we annotated it with 1 to 15 simple effect clauses (e.g. *h11* be one of effects resulted by an activity).

The constraint graph of any generated process ecosystem was constructed by our framework. Figure 6.3 illustrates the corresponding constraint graph of the generated process ecosystem involving 10 process models with its corresponding inter-process relationships. The experiment set up for the rest process ecosystems in different sizes, which are illustrated in their respective constraint graphs, can be found in Appendix A. Each process model is represented by a node with the name of the process, e.g. $P1, P11$. Each inter-process relationship is illustrated by an edge corresponding two different nodes. We, however, attempted to set up our process ecosystems such that all of our formalized inter-process relationships, i.e. part-whole, inter-operation, generalization-specialization, were used in the most such ecosystems. We can identify each inter-process relationship represented by different types of lines in the diagram.

6.2.2 Relationship descriptors set up

Our framework generated all possible relationships which might exist between processes in a process ecosystem and maintained them in the relationship descriptors. The analyst involvement, however, was still required for justifying the relationships suggested by the framework since multiple relationships might be unnecessarily suggested, e.g. in generalization-specialization relationship. We provided different relationship descriptors for each process ecosystem. We noticed that number of relationships gen-

erated by our framework was larger than number of relationships maintained in the relationship descriptors. It happened since our framework generated all possible relationships between process models in a process ecosystem.

6.2.3 Capability library set up

Our framework populated different capability library for each process ecosystem. Such capability library consisted of any activity description used in any process model in the ecosystem including name, immediate effects, role and insertion point activities (if applicable) of such process. We planned to set up different capability library for each generated process ecosystem. We leveraged our capability library described earlier such that any activity with conflicting roles was not allowed. In regard capability repurposing, we maintained both the original and the new activities in the library such that both of them became activity options in redesigning a process.

6.2.4 Execution procedure

In general, we similarly performed our experiments on each process ecosystem. For each process ecosystem, the experiments followed the execution procedure as follows.

1. We prepared a generated process ecosystem involving process model $P1$ as the changes trigger. Such process $P1$ was changed using capability repurposing on its two sub-processes represent the functionalities of processes $P11$ and $P12$, accordingly. Such change violated some relationships of $P1$ which were then propagated to the other related processes for maintaining the equilibrium of such a process ecosystem. Figure 6.1 depicts the original design of process $P1$ which was modified to be a process shown in Figure 6.2. The constraint graph of the first process ecosystem involving $P1$ as the changes trigger is illustrated in Figure 6.3.

2. We prepared relevant relationship descriptors and capability library for the generated process ecosystem under evaluation. Once the process ecosystem was changed, we replaced both relationship descriptors and capability library with the relevant ones.

3. We engaged both repair and constructive modes to reestablish the equilibrium of the perturbed process ecosystem under evaluation. We measured 3 times the elapsed time of such reestablishment and averaged them in each mode. Such each elapsed time included selecting a process to be redesigned in propagating the changes, redesigning such selected process and checking the relationship constraints of such a redesigned process. In addition to the elapsed time, we recorded the following information for each approach: number of processes, number of constraints, number of checked constraints, number of violated constraints, number of redesigned processes.

   Number of constraints represent the number of inter-process relationships which may exist among process models in the process ecosystem under evaluation. We refer to the number of checked constraints as the number of constraints of a process being modified excluding any constraint of such a process with any process which previously has been modified. For example, modifying $P1$ results 3 constraints to be checked since $P1$ has 3 relationships with 3 other processes, as illustrated in Figure 6.3. We consider all of these 3 relationships since $P1$ is the changes trigger. While, modifying $P11$, i.e. due to changes to $P1$, however, will only need to check 2 constraints corresponding to its relations, i.e. processes $P111$ and $P112$. Further, number of violated constraints represents the number of inter-process relationships of a certain process model being observed, which were violated. It is noted that the number of checked constraints and the number of violated constraints may not be equal as the observed constraint may not be

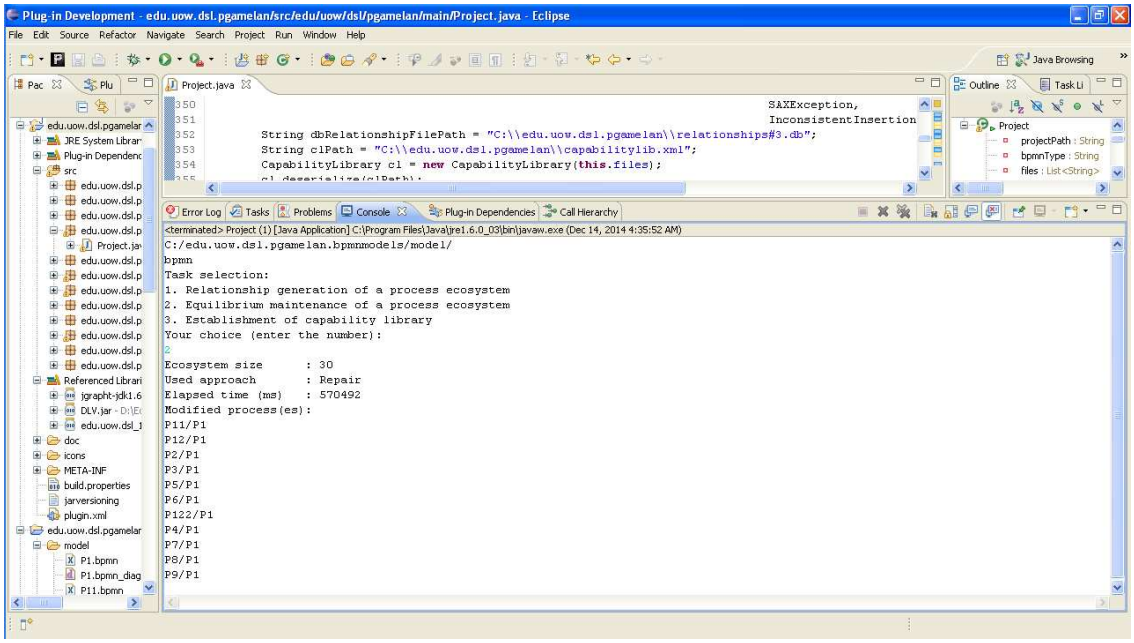


Figure 6.4: Snapshot of evaluating a process ecosystem consisting 30 processes in the repair mode

violated. Finally, number of redesigned processes refer to the number of process models in a process ecosystem that need to be modified in order to resolve any violated inter-process relationship.

6.3 Experiment results

We have executed an empirical validation[1] to assess how the repair and constructive modes perform on different sizes of the process ecosystems. Figures 6.4 and 6.5 show two execution results using repair and constructive modes, respectively, applied in a process ecosystem consisting 30 models. In such figures, we can evaluate all process models which must be redesigned (e.g. $P11/P1$, which means process model $P11$ consisting a swimlane $P1$) in order to reestablish the equilibrium of a process ecosystem under evaluation. The capability library and relationship descriptors for such execution

[1] All experiments were run on a i3 Intel Core-2.27 GHz, RAM 2.85 Gb laptop.

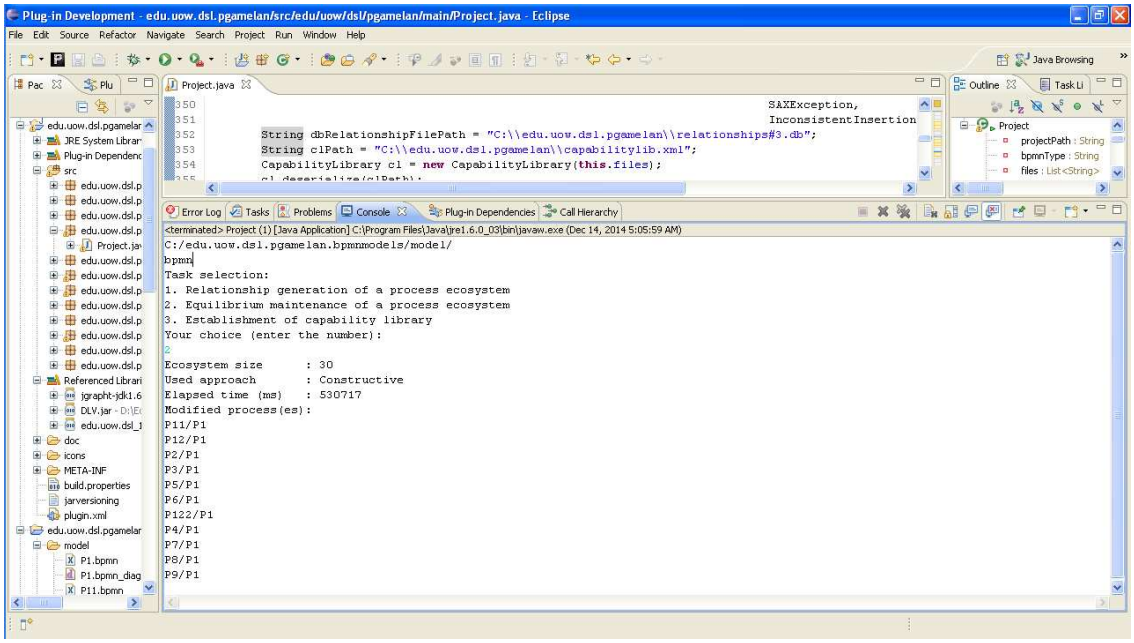


Figure 6.5: Snapshot of evaluating a process ecosystem consisting 30 processes in the constructive mode

are illustrated in Figure 6.6 and 6.7, respectively. The complete results of all executions are shown in Table 6.1 which describes how the repair and constructive modes perform in proportion to the number of checked constraints in terms of the elapsed time for establishing a new equilibrium of a process ecosystem. In this setting, we have different number of checked constraints in each process ecosystem having different number of processes.

We, however, realize that number of the checked constraints in each ecosystem depends on the nature of the initial changes to propagate. In particular, given the same setting of process ecosystems, we may have different setting of the checked constraints if we apply different initial changes.

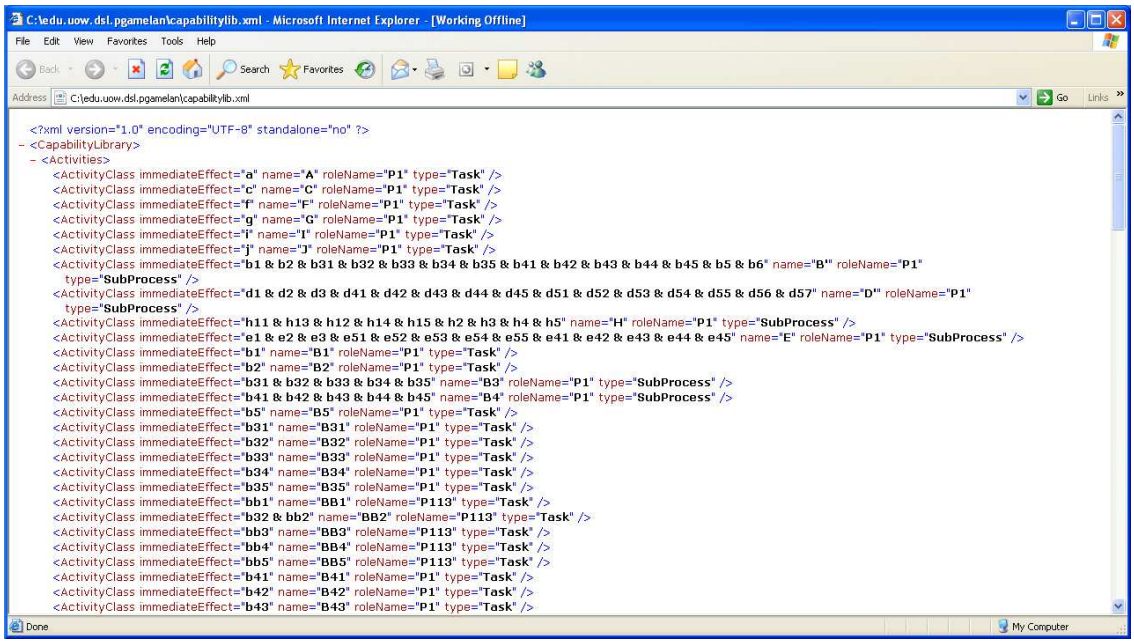


Figure 6.6: Snapshot of a capability library used in a process ecosystem consisting 30 processes

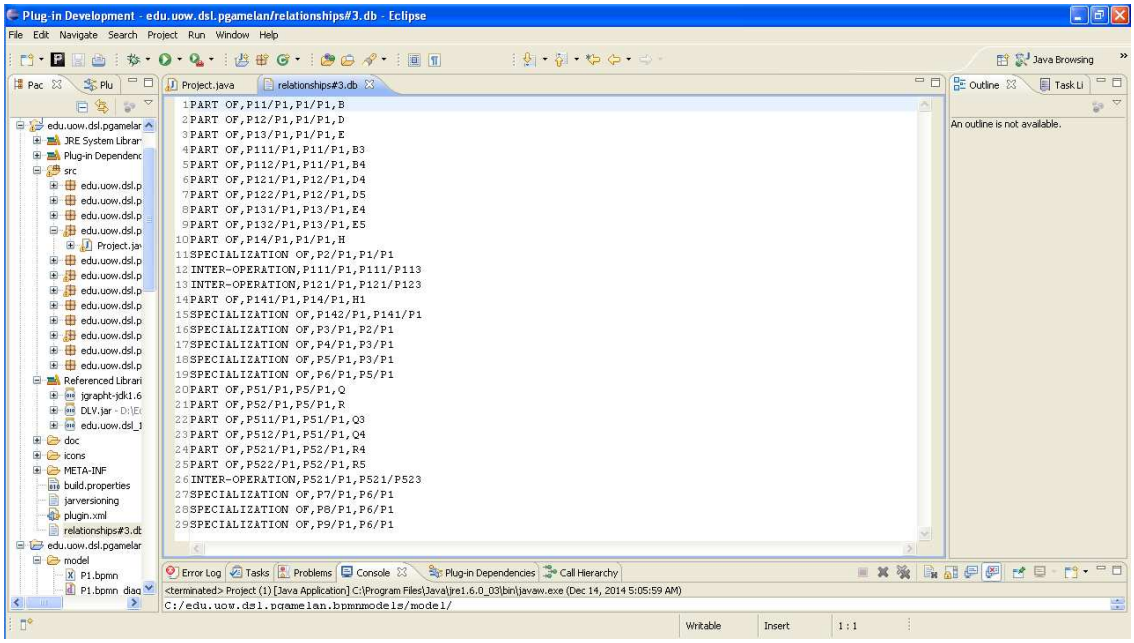


Figure 6.7: Snapshot of a relationship descriptors used in a process ecosystem consisting 30 processes

Table 6.1: Average elapsed time of reestablishing the equilibrium of various sizes of process ecosystems

					Avg. elapsed time	
No. of processes	**No. of constraints**	**No. of checked constraints**	**No. of violated constraints**	**No. of redesigned processes**	**Repair mode (sec)**	**Constructive mode (sec)**
10	9	7	3	3	59	57
20	19	13	8	8	199	193
30	29	18	11	11	571	531
40	39	21	12	12	958	867
50	50	24	12	12	1,341	1,172
60	60	27	13	13	1,908	1,703
70	70	31	14	14	2,355	1,929
80	80	40	18	18	3,069	2,690

6.4 Post-evaluation state and analysis

In this section, we describe the condition of our process ecosystems after performing the evaluation using our implemented tool. In each generated process ecosystem, we have reestablished the equilibrium of such process ecosystem involving the changes trigger process $P1$ shown in Figure 6.2. For example, a constraint graph of the first ecosystem illustrated in Figure 6.8 describes the propagation of changes triggered by process $P1$. Such propagation requires changes to be made to other related processes, i.e. $P11$, $P12$ and $P122$. By changing these three related processes, we can preserve the inter-process relationship constraints in the process ecosystem.

Furthermore, our internal validity results a strong relation between the number of checked constraints and the elapsed time, i.e. the more number of checked constraints the longer elapsed time. The more number of checked constraints will bring the more number of processes to be possibly redesigned. This consequently contributes to the elapsed time. Our experimental results also suggest that the proposed approaches are efficient (i.e. help analysts propagate changes regardless of the complexity of the inter-process relationships in the process ecosystem) and scalable in propagating changes to maintain the equilibrium of a medium-sized process ecosystem (up to 80 processes).

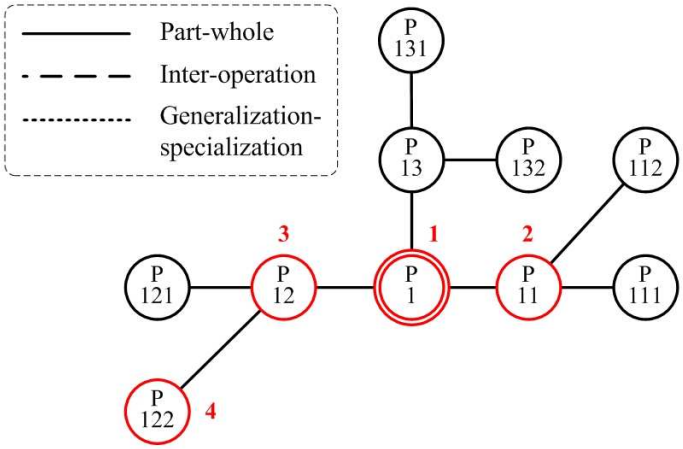


Figure 6.8: Process changes propagation of process ecosystem in Figure 6.3

Additionally, performing the repair mode to get a restored-equilibrium process ecosystem takes longer time than performing the constructive mode. This could be explained as follows. In the repair mode, we need to verify the consistency between all processes that are related to the process being modified whereas in constructive mode, we only check the consistency between the process being modified and related processes that were already modified. Further, these results become an improvement of our previous experiment results described in [33, 34] due to leveraging our process redesign tool. The current results show that they are much faster than the previous ones. This happens since we have improved our framework implementations, e.g. process graph reconstruction due to introducing a process variant, preprocessing in deleting the contents of temporary folders, such that they support for efficiency achievement. In addition, we have not analyzed the complexity of our algorithms in order to correlate the elapsed time with parameters represented by the five leftmost columns of Table 6.1.

Threats to validity

We have investigated 8 different process ecosystems which were diverse in size. The inter-process relationships existing in these process ecosystems also vary and cover all

the relationship types that we have defined. We did not discard any relationships or process ecosystems as outliers and we evaluated the impact of changes across all of them exhaustively. Since our approach performed relatively well for all these models and rules, we believe that the threats to internal validity are small. While our framework and the evaluation focused on BPMN process models, the notions and ideas can be generalized to other types of business process models. Future work also investigates how our approach scales with larger real-world process ecosystems when they become available and more importantly are *fully semantically* annotated.

Accuracy analysis

We use Figure 6.8 to illustrate our accuracy analysis of propagating the initial change made to process model $P1$. This figure represents such change propagation in process ecosystem involving 10 process models which is originally illustrated in Figure 6.3. Our change propagation framework suggests the order of such propagation, as depicted in red numbers and red nodes. Since we made initial changes to two sub-processes in $P1$ which represent the functionalities of process models $P11$ and $P12$, accordingly, such changes should be propagated to these processes. The order of redesigning both processes can be either since both processes have the same cardinality of the constraints, i.e. three. Further, since redesigning process model $P12$ affected to its sub-process representing the functionality of process model $P122$, such propagation continued to process model $P122$. In this setting, process model $P12$ becomes the main process of process model $P122$. In the same vein, we can analyze such change propagation accuracy using other process ecosystems shown in Appendix A.

6.5 Summary

In this chapter, we have presented our evaluation which was performed to support the proof of our process ecosystem in dealing with process change propagation. This includes description of the experiment set up involving setting up for process ecosystems, relationship descriptors, capability library and the execution procedure of the experiments. We, however, only present the first process ecosystem set up in this chapter. The set up of the rest of process ecosystems can be found in Appendix A. The experiment results suggest that our approach is efficient and accurate to deal with process change propagation. We, however, need to further investigate using real business process repository taken from industry to get more precise data since our current experiments were performed using limited environment.

Chapter 7

Conclusion and future work

In this chapter, we summarize our process ecosystem approach, in Section 7.1, for managing business processes due to process changes with respect to the research questions introduced in Section 1.2. This includes discussion on limitations of our current approach that should be taken into consideration. As such, we realize that some challenging issues still remain for further investigation in our future work, as outlined in Section 7.2.

7.1 Conclusion

As we realize that there exist a little work in dealing with managing process changes in a collection of business processes, we aim to fill this gap by developing this research. We view such a collection of business processes as an ecosystem in which there exist inter-dependencies between entities composing the ecosystem. First, managing process ecosystem requires identification and formalization of inter-process relationships as it is a fundamental aspect in our approach. So far, we have identified three relationship types which may exist between business processes, i.e. part-whole, interoperation and generalization-specialization. We also have formalized them based on

semantic effect annotation of process models. Establishment of such relationships in a large and complex business process repository requires a supporting tool such that it can be performed in an efficient and effective manner. We, however, realize that such establishment still requires involvement of process analyst to make decision in regard inter-process relationships should be maintained in the repository. In regard our research question with respect to inter-process relationship management, we can conclude that our approach has its capability to deal with such management.

Second, managing process ecosystem also imposes identification of its boundary, i.e. what kind of definition of such process ecosystem can be constructed with respect to our need for managing process changes in a collection of business processes. We have properly introduced our process ecosystem definition based on the process traceability notions. As such, in propagating process changes in the repository, we only focus on all relevant process models which might be affected and can consequently leverage our process change propagation procedures in an efficient and effective way. Hence, we conclude that our approach has such proper definition. In addition, our experiment results also suggest that our process ecosystem boundaries have been adequately leveraged with respect to our research question in process ecosystem definition.

Third, managing process changes in a process ecosystem requires appropriate procedures to ensure that such ecosystem is always in its equilibrium. We have proposed two modes, i.e. repair and constructive. We develop these modes inspired by CSP technologies as we capture our process ecosystem with constraint networks. Our experiment results suggest that these modes are efficient and effective to propagate changes in process ecosystem. With respect to our research question dealing with providing algorithm for propagating process changes, we also acknowledge that our approach has appropriate procedures for such propagation.

Fourth, managing process changes propagation in a process ecosystem involves pro-

cess redesign approach to be able to resatisfy constraints of inter-process relationship being violated. We have constructed resolution patterns for any inter-process relationship violation from which our process redesign approach is built. Our resolution patters as well as process redesign approach become specific to each inter-process relationship type. Our experiment results also tell us that our process redesign approach properly works by which process changes can be efficiently propagated across process ecosystem. Therefore, we assume that our process ecosystem approach has adequately provided inter-process resolution patterns as well as process redesign approach with respect to our research question concerning such resolution patterns.

Limitations

Our current approach, however, has some limitations in term of its applicability. First, we only identified three relationship types which may exist among business processes. We realize that there may exist more than three types in the real business process repository, e.g. abstraction-refinement which depicts relationship between processes with different level of granularity to represent the same business process. Second, our process redesign approach relies on the inter-process relationships identified in this research. More specifically, it is guided by our resolution patterns for resolving any inter-operation relationship violation. As such, this approach can not be adequately applied into broader process redesign cases. Third, our experiments were performed in a limited experiment environment, i.e. process models and their respective relationships could not reflect the real business process repository. It, however, has adequately supported the proof of our approach.

7.2 Future work

According to the limitations of our current approach, we outline some future research opportunities which can further improve such approach. First, other relationship types between business processes should be taken into consideration. They have to be further investigated and formalized properly. These may include: (i) abstraction-refinement, i.e. presenting a process in different representations based on their granularity levels; (ii) resource-sharing, i.e. two processes are related to each other due to dependencies on the same resource; and (iii) informational inter-operation, i.e. two processes are related to each other with respect to the conformance on the message exchanged between them. Second, finding the optimal solution with minimal change strategy of restored-equilibrium process ecosystem. It requires a selection of any alternative of the restored-equilibrium process ecosystem to be minimally changed compared to the original process ecosystem. In this context, redesigning a process model to get minimal changes compared to its original process model does not necessarily contribute to achieve minimal changes on the restored-equilibrium process ecosystem. We, however, will not concentrate on the local process redesign, rather, we will deal with the global process ecosystem restoration. Third, development of process redesign for more general approach should be investigated. Such general approach may consider goals of business process based on the semantic effect annotation of process models. Fourth, performing experiments on our framework using case-studies taken from the industry should be considered. This will improve our approach coverage based on any case occurs in the real business process repository.

Appendix A

Process ecosystem set up and change propagation analysis

Figures A.1, A.2, A.3, A.4, A.5, A.6 and A.7 depict process ecosystems consisting of 20, 30, 40, 50, 60, 70 and 80 distinct process models, respectively, with their corresponding inter-process relationships represented in constraint graphs. They were originally in their own equilibria with respect to inter-process relationship constraints formalized earlier. We applied the same initial change treatment to these ecosystems as in process ecosystem previously described in Section 6.2. Such initial change triggered perturbation on their equilibria such that it was propagated across the respective ecosystems to restore such equilibria by resolving any violated inter-process relationship. This involved redesigning any process model involved in such violation. Our approach were able to perform such change propagation and suggested all redesigned process models. For our change propagation analysis purpose, we highlight such redesigned process models (i.e. denoted by the red nodes) in the constraint diagrams as well as the ordering of propagating the changes (i.e. denoted by the red numbers) across an ecosystem.

For supporting proof of our approach, we need to perform further analysis on such

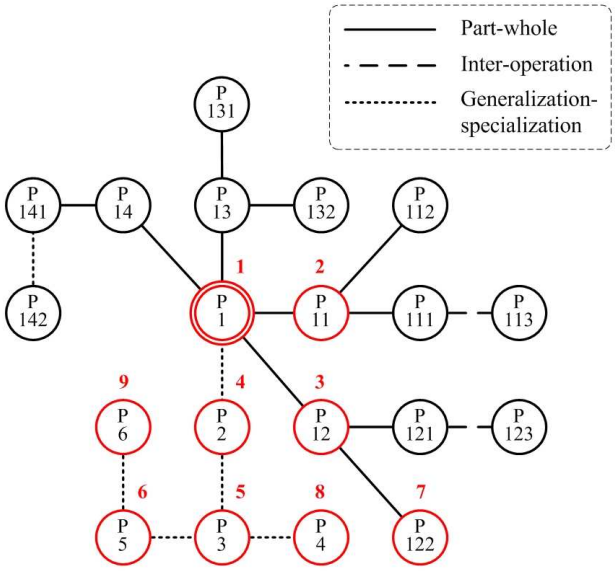


Figure A.1: Experiment set up and changes propagation of process ecosystem involving 20 process models

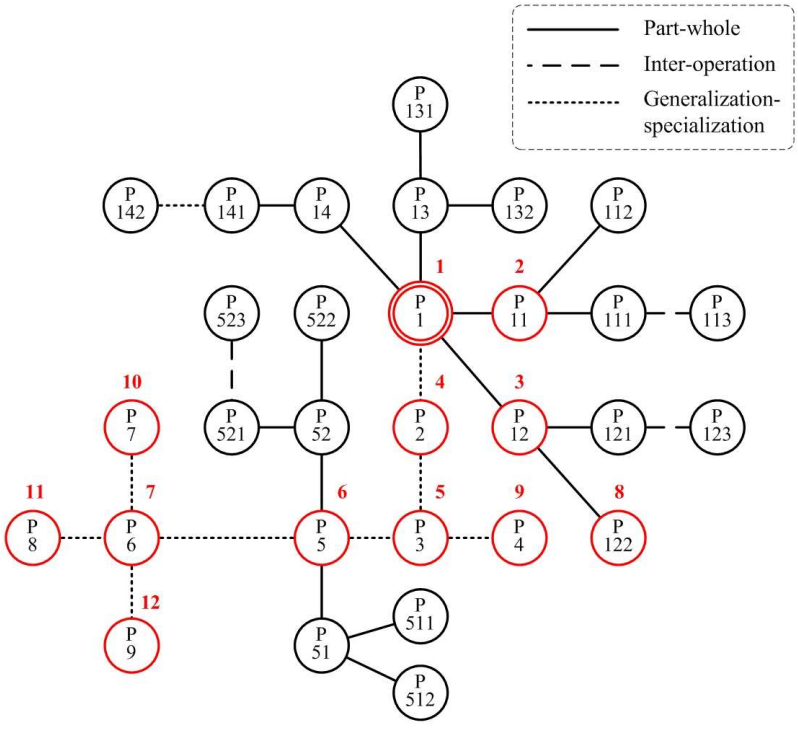


Figure A.2: Experiment set up and changes propagation of process ecosystem involving 30 process models

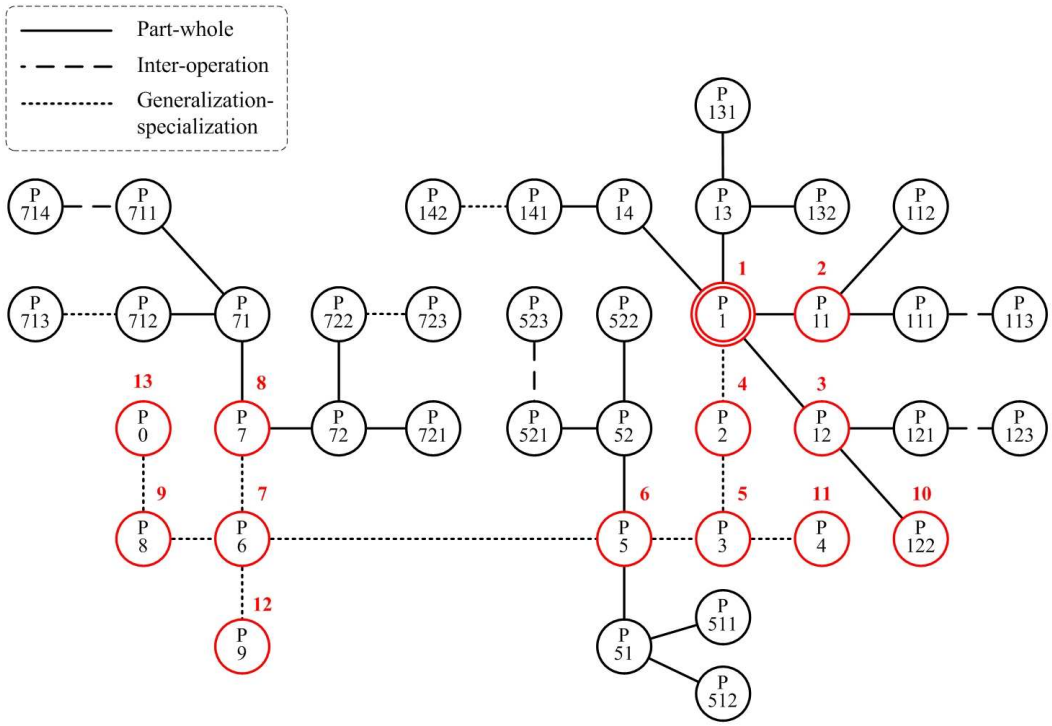


Figure A.3: Experiment set up and changes propagation of process ecosystem involving 40 process models

diagrams in term of correctness of the redesigned process models as well as correctness of the propagation order. As process model $P1$ became process having the initial change, all process models being its specialization, either directly or indirectly, should be redesigned in order to satisfy their relationship constraints. Further, all process models representing modified sub-processes in $P1$ should also be redesigned. Our experiment results suggest that these process models had been redesigned accordingly, as shown in Figures A.1 to A.7.

We implemented dynamic ordering for selecting a process model to be redesigned due to the initial process changes. As we applied the same initial process changes, this dynamicity relied on different structures of the process ecosystems, as can be seen in our diagrams. In such ordering, we prioritized to redesign a process model, which participated in the most constraints, over the other process models in which their inter-

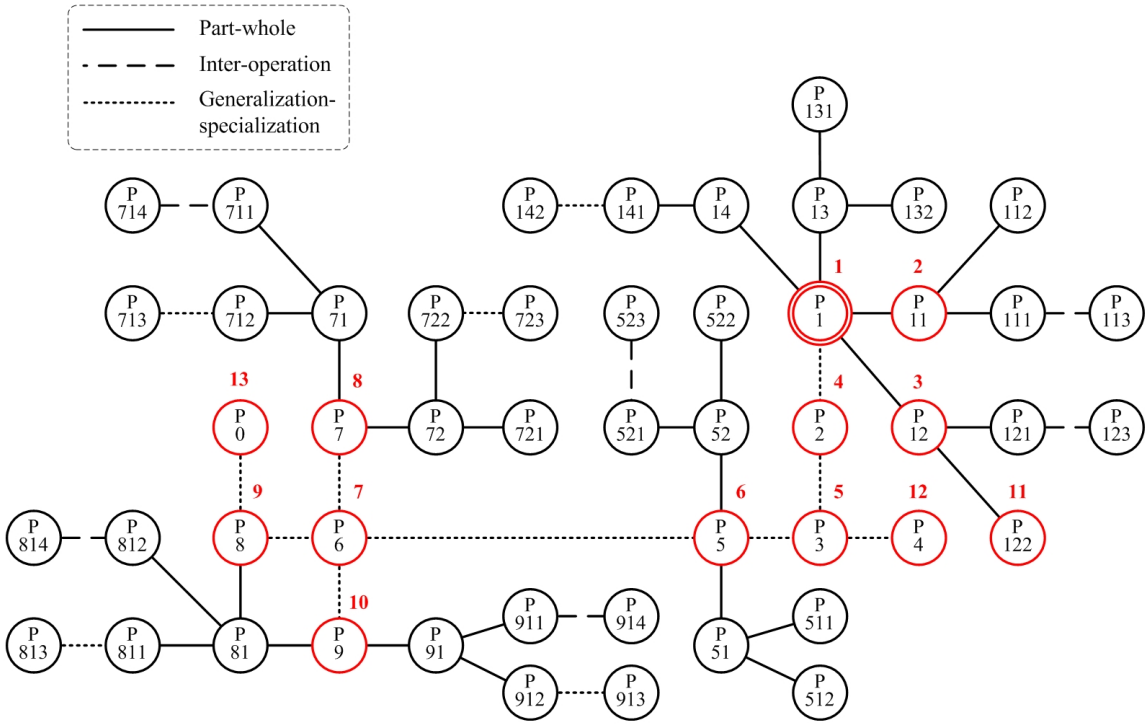


Figure A.4: Experiment set up and changes propagation of process ecosystem involving 50 process models

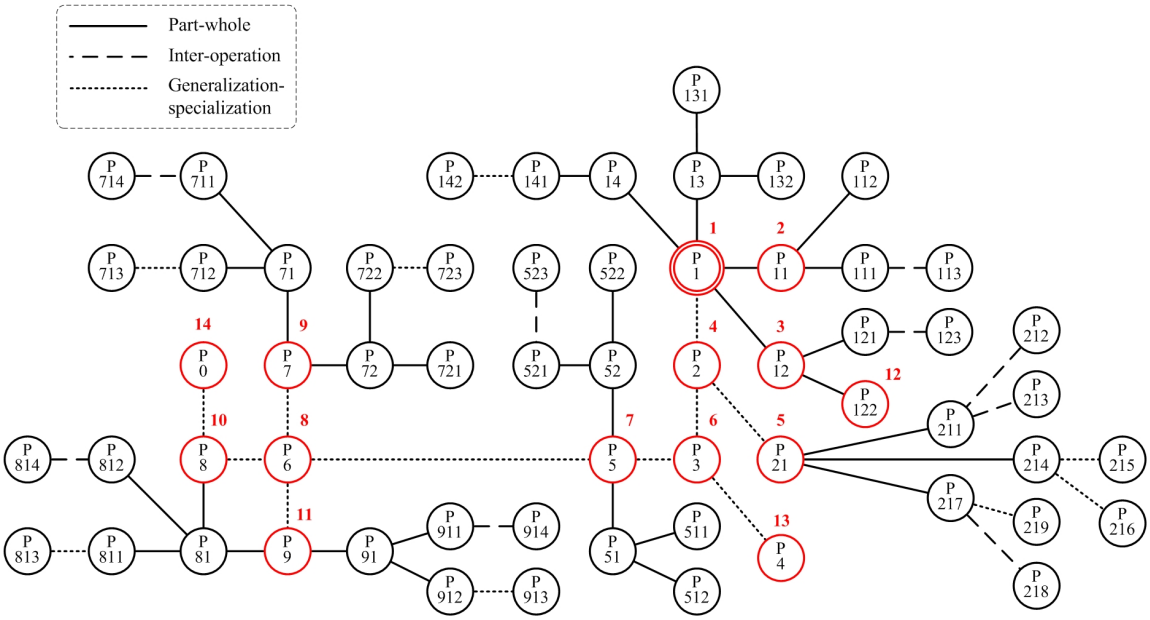


Figure A.5: Experiment set up and changes propagation of process ecosystem involving 60 process models

process relationships were violated. For example, the order of redesigning processes in ecosystem depicted in Figure A.1 is slightly different from that illustrated in Figure A.2, even though most of their process models are in common. The former provides the following order: $P1, P11, P12, P2, P3, P5, P122, P4$ and $P6$. While, the latter has the following order: $P1, P11, P12, P2, P3, P5, P6, P122, P4, P7, P8$ and $P9$. Clearly, the propagating order of the two ecosystems involving process models $P122, P4$ and $P6$ is different. This happens as, in the former ecosystem, process model $P6$ has the same priority with the other two processes in which the order can be either. While, in the latter ecosystem, process model $P6$ has the highest priority compared to $P122$ and $P4$ due to its new relationships to process models $P17, P8$ and $P9$. Similarly, we can also perform such ordering analysis to the rest of our diagrams.

According to our change propagation analysis, we can figure out that our process ecosystem approach has been adequately implemented to manage a collection of business processes in dealing with process change propagation.

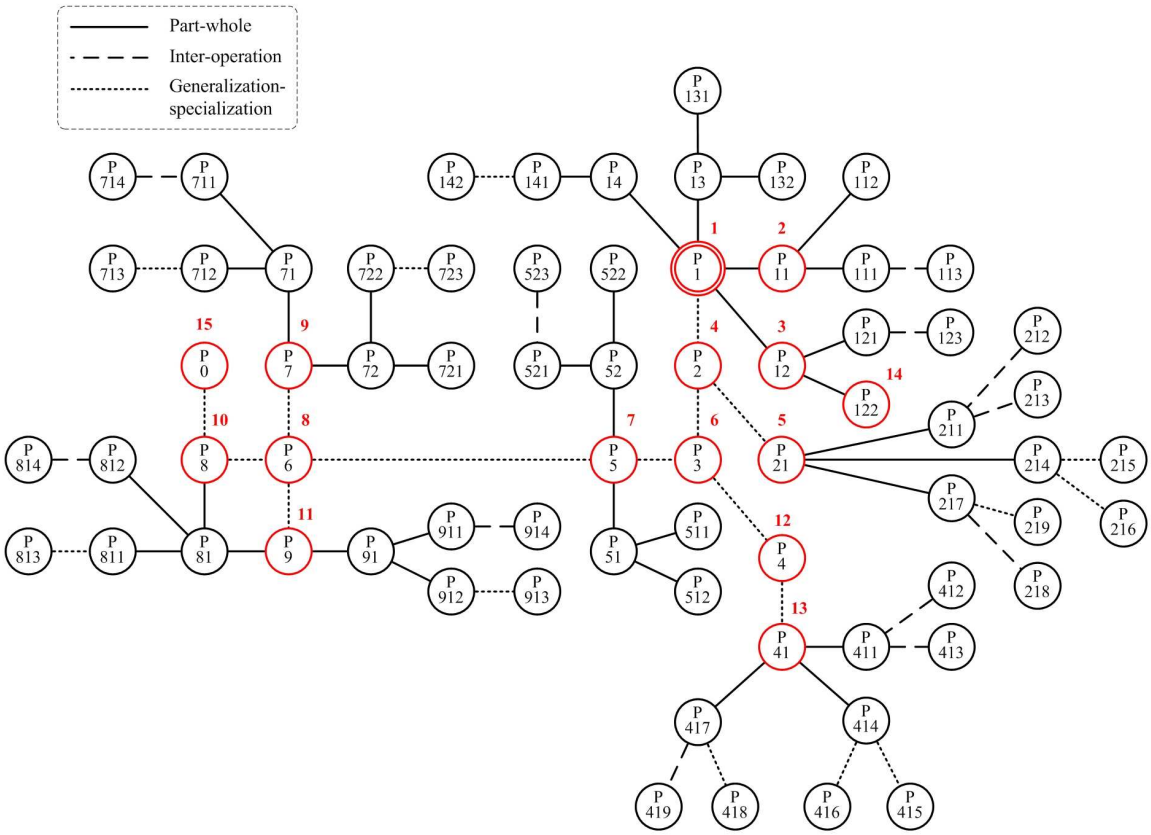


Figure A.6: Experiment set up and changes propagation of process ecosystem involving 70 process models

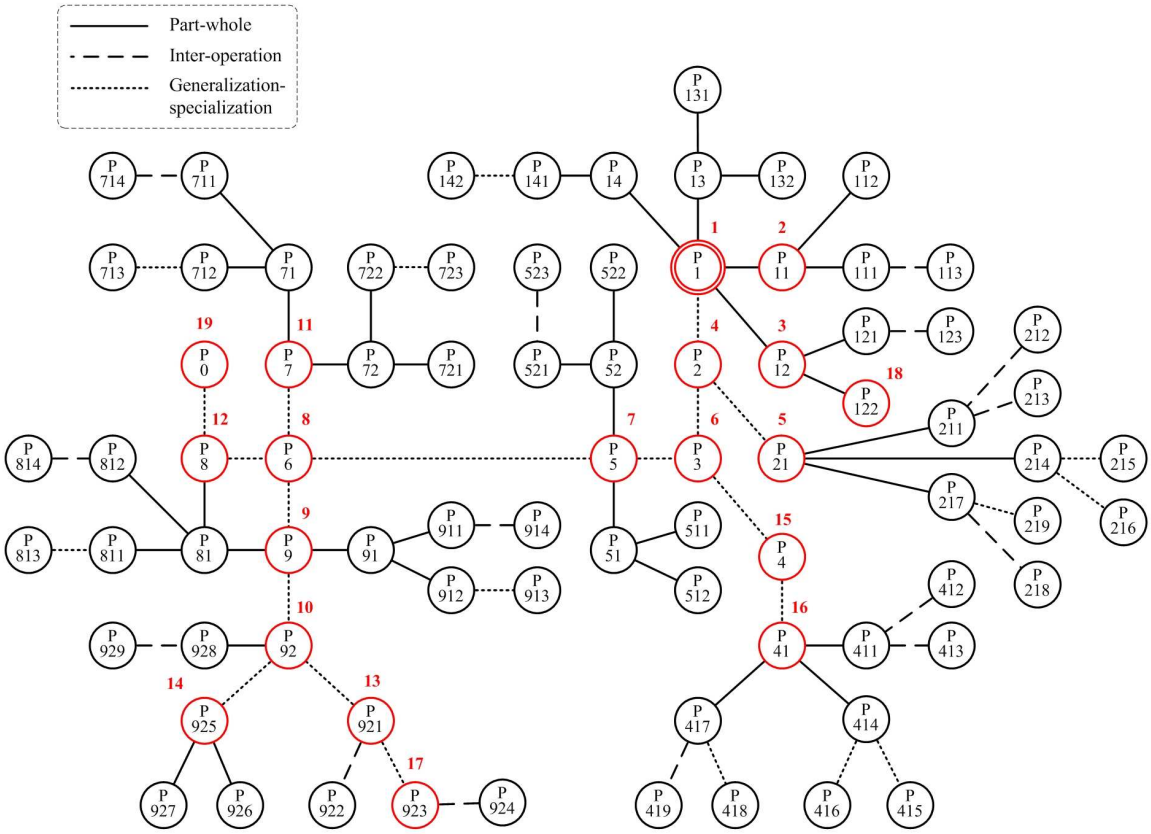


Figure A.7: Experiment set up and changes propagation of process ecosystem involving 80 process models

References

[1] A. Aryani, I.D. Peake, and M. Hamilton. Domain-based change propagation analysis: An enterprise system case study. In *2010 IEEE International Conference on Software Maintenance (ICSM)*, pages 1–9. IEEE, 2010.

[2] R. Bartak. Constraint propagation and backtracking-based search. *Charles Universität, Prag*, 2005.

[3] Abraham Bernstein, Mark Klein, and Thomas W Malone. The process recombinator: a tool for generating new business process ideas. In *Proceedings of the 20th International Conference on Information Systems*, pages 178–192. Association for Information Systems, 1999.

[4] P.A. Bernstein and U. Dayal. An overview of repository technology. In *Proceedings of the International Conference on Very Large Data Bases*, pages 705–705. Institute of Electrical & Electronics Engineers (IEEE), 1994.

[5] M. Bevilacqua, F. E. Ciarapica, and G. Giacchetta. Business process reengineering in healthcare management: a case study. *BPM Journal*, 17(1):42–66, 2011.

[6] Grady Booch. *Object Oriented Analysis & Design with Application*. The Benjamin/Cummings Publishing Company, Inc., 1994.

[7] I. Choi, K. Kim, and M. Jang. An xml-based process repository and process query language for integrated process management. *Knowledge and Process Management*, 14(4):303–316, 2007.

[8] David KH Chua and Md Aslam Hossain. Predicting change propagation and impact on design schedule due to external changes. *Engineering Management, IEEE Transactions on*, 59(3):483–493, 2012.

[9] H.K. Dam and M. Winikoff. Supporting change propagation in UML models. In *2010 IEEE International Conference on Software Maintenance (ICSM)*, pages 1–10. IEEE, 2010.

[10] R. Dechter. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intelligence*, 41:271–312, 1990.

[11] Rina Dechter. Constraint networks. *Encyclopedia of Artificial Intelligence*, pages 276–285, 1992.

[12] Rina Dechter. *Constraint Processing*. Morgan Kaufmann Publishers, 2003.

[13] G. Decker, H. Overdick, and M. Weske. Oryx–an open modeling platform for the bpm community. *Business process management*, pages 382–385, 2008.

[14] Michel Marie Deza and Elena Deza. *Encyclopedia of distances*. Springer, 2009.

[15] R. Dijkman, M.L. Rosa, and H.A. Reijers. Managing large collections of business process modelscurrent techniques and challenges. *Computers in Industry*, 63(2):91, 2012.

[16] Chathura Ekanayake, Marcello La Rosa, Arthur ter Hofstede, and Marie-Christine Fauvet. Fragment-based version management for repositories of business process

models. *On the Move to Meaningful Internet Systems: OTM 2011*, pages 20–37, 2011.

[17] Osamu Fujita. Metrics based on average distance between sets. *Japan Journal of Industrial and Applied Mathematics*, 30(1):1–19, 2013.

[18] Shang Gao and John Krogstie. A repository architecture for business process characterizing models. *The Practice of Enterprise Modeling*, pages 162–176, 2010.

[19] A. Ghose and G. Koliadis. Model eco-systems: preliminary work. In *Proceedings of the Fifth Asia-Pacific Conference on Conceptual Modelling*, volume 79, pages 19–26. Australian Computer Society, Inc., 2008.

[20] A. Ghose, G. Koliadis, and A. Chueng. Rapid business process discovery (R-BPD). In *Conceptual Modeling-ER 2007*, pages 391–406. Springer, 2007.

[21] Dawn G Gregg, Uday R Kulkarni, and Ajay S Vinzé. Understanding the philosophical underpinnings of software engineering research in information systems. *Information Systems Frontiers*, 3(2):169–183, 2001.

[22] V. Grover. From business reengineering to business process change management: a longitudinal study of trends and practices. *IEEE Transactions on Engineering Management*, 46(1):36–46, 1999.

[23] Paul Harmon and Celia Wolf. Business process modeling survey. Website, December 2011. `http://bptrends.com/surveys_landing.cfm`.

[24] M. Hepp, F. Leymann, J. Domingue, A. Wahler, and D. Fensel. Semantic business process management: A vision towards using semantic web services for business process management. In *IEEE International Conference on e-Business Engineering, ICEBE 2005*, pages 535–540. IEEE, 2005.

[25] J. B. Hill, M. Cantara, E. Deitert, and M. Kerremans. Magic quadrant for business process management suites. Technical report, Gartner Research, 2007.

[26] K. Hinge, A. Ghose, and G. Koliadis. Process SEER: A tool for semantic effect annotation of business process models. In *IEEE International Enterprise Distributed Object Computing Conference, EDOC'09*, pages 54–63. IEEE, 2009.

[27] Constantin Houy, Peter Fettke, Peter Loos, Wil van der Aalst, and John Krogstie. Bpm-in-the-large–towards a higher level of abstraction in business process management. *E-Government, E-Services and Global Processes*, pages 233–244, 2010.

[28] N.R. Jennings. Using intelligent agents to manage business processes. In *IEE Colloquium on Intelligent Agents and Their Applications, (Digest No: 1996/101)*, pages 5–1. IET, 1996.

[29] W.J. Kettinger and V. Grover. Special section: toward a theory of business process change management. *Journal of Management Information Systems*, 12(1):9–30, 1995.

[30] G. Kim and Y. Suhh. Ontology-based semantic matching for business process management. *ACM SIGMIS Database*, 41(4):98–118, 2010.

[31] Mark Klein and Abraham Bernstein. Searching for services on the semantic web using process ontologies. In *The First Semantic Web Working Symposium (SWWS-1)*, 2001.

[32] G. Koliadis and A. Ghose. Verifying semantic business process models in interoperation. *IEEE SCC*, pages 731–738, 2007.

[33] T. A. Kurniawan, A. K. Ghose, H. K. Dam, and L. -S. Lê. Relationship-preserving change propagation in process ecosystems. *Service-Oriented Computing*, pages 63–78, 2012.

[34] T.A. Kurniawan, A.K. Ghose, H.K. Dam, L.S. Lê, and T. Zhang. Design maintenance in process eco-systems. In *IEEE Ninth International Conference on Services Computing (SCC) 2012*, pages 392–399. IEEE, 2012.

[35] T.A. Kurniawan, A.K. Ghose, L.S. Lê, and H.K. Dam. On formalizing inter-process relationships. In *Business Process Management Workshops*, pages 75–86. Springer, 2012.

[36] M. La Rosa, H.A. Reijers, W.M.P. Van der Aalst, R.M. Dijkman, J. Mendling, M. Dumas, and L. García-Bañuelos. Apromore: An advanced process model repository. *Expert Systems with Applications*, 38(6):7029–7040, 2011.

[37] Marcello La Rosa, Marlon Dumas, Reina Uba, and Remco M. Dijkman. Business process model merging : An approach to business process consolidation. *ACM Transactions on Software Engineering and Methodology (TOSEM), in press*, 2012.

[38] Ivanna M Lazarte, LucinéIa Heloisa Thom, Cirano Iochpe, Omar Chiotti, and Pablo D Villarreal. A distributed repository for managing business process models in cross-organizational collaborations. *Computers in Industry*, 64:252–267, 2013.

[39] Z. Ma, B. Wetzstein, D. Anicic, S. Heymans, and F. Leymann. Semantic business process repository. In *Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM 2007)*, volume 251, pages 92–100, 2007.

[40] T. W. Malone, K. Crowston, and G. A. Herman. *Organizing Business Knowledge: The MIT Process Handbook*. The MIT Press, 2003.

[41] S. Minton, M. D. Johnston, A. B. Philips, and P. Laird. Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.

[42] Ugo Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132, 1974.

[43] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[44] Ikujiro Nonaka. A dynamic theory of organizational knowledge creation. *Organization Science*, 5(1):14–37, 1994.

[45] A. Polyvyanyy, S. Smirnov, and M. Weske. Process model abstraction: A slider approach. In *IEEE 12th International Enterprise Distributed Object Computing Conference, EDOC'08*, pages 325–331. IEEE, 2008.

[46] Jan Recker. Opportunities and constraints: the current struggle with bpmn. *Business Process Management Journal*, 16(1):181–201, 2010.

[47] ReportLinker. Business process management (BPM) market shares, strategies, and forecasts, worldwide, 2012 to 2018. Website, 2012. `http://reportlinker.com/p0963519-summary/Business-Process-Management-BPM-Market-Shares-Strategies-and-Forecasts-Worlc html`.

[48] M. Rosemann and J. Brocke. The six core elements of business process management. *Handbook on Business Process Management 1*, pages 107–122, 2010.

[49] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, New Jersey, 3rd edition, 2010.

[50] S. Sakr and A. Awad. A framework for querying graph-based business process models. In *Proceedings of the 19th International Conference on World Wide Web*, pages 1297–1300. ACM, 2010.

[51] Daniel Schleicher, Tobias Anstett, Frank Leymann, and David Schumm. Compliant business process design using refinement layers. *On the Move to Meaningful Internet Systems: OTM 2010*, pages 114–131, 2010.

[52] K. Shahzad, B. Andersson, M. Bergholtz, A. Edirisuriya, T. Ilayperuma, P. Jayaweera, and P. Johannesson. Elicitation of requirements for a business process model repository. In *Business Process Management Workshops*, pages 44–55. Springer, 2009.

[53] K. Shahzad, M. Elias, and P. Johannesson. Requirements for a business process model repository: A stakeholders perspective. In *Business Information Systems*, pages 158–170. Springer, 2010.

[54] S. Smirnov, H.A. Reijers, and M. Weske. From fine-grained to abstract process models: A semantic approach. *Information Systems*, 2012.

[55] Minrong Song, John A Miller, and Ismailcem B Arpinar. *Repox: An xml repository for workflow designs and specifications*. PhD thesis, Citeseer, 2001.

[56] M. Strohmaier and E. Yu. Towards autonomic workflow management systems. In *Proceedings of the 2006 Conference of the Center for Advanced Studies on Collaborative Research*, page 34. IBM Corp., 2006.

[57] W. M. P. van der Aalst and K. M. van Hee. Framework for business process redesign. In *Proceedings of the Fourth Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 36–45. IEEE Computer Society Press, 1995.

[58] Wil MP van der Aalst. *Structural characterizations of sound workflow nets*. Eindhoven University of Technology, Department of Mathematics and Computing Science, 1996.

[59] Wil MP van der Aalst. Interorganizational workflows: An approach based on message sequence charts and petri nets. *Systems Analysis Modelling Simulation*, 34(3):335–368, 1999.

[60] Wil MP van der Aalst and Twan Basten. Inheritance of workflows: an approach to tackling problems related to change. *Theoretical Computer Science*, 270(1):125–203, 2002.

[61] Wil MP Van der Aalst, Mathias Weske, and Dolf Grünbauer. Case handling: a new paradigm for business process support. *Data & Knowledge Engineering*, 53(2):129–162, 2005.

[62] Kees van Hee, Olivia Oanea, Natalia Sidorova, and Marc Voorhoeve. Verifying generalized soundness of workflow nets. *Perspectives of Systems Informatics*, pages 235–247, 2007.

[63] Jussi Vanhatalo, Jana Koehler, and Frank Leymann. Repository for business processes and arbitrary associated metadata. *BPM Demo Session 2006*, pages 25–31, 2006.

[64] R. Vidgen and X. Wang. From business process management to business process ecosystem. *Journal of Information Technology*, 21(4):262–271, 2006.

[65] H.J. Wang and H. Wu. Supporting process design for e-business via an integrated process repository. *Information Technology and Management*, 12(2):97–109, 2011.

[66] Y. Wang, J. Yang, and W. Zhao. Change impact analysis for service based business processes. In *2010 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 1–8. IEEE, 2010.

[67] B. Weber, M. Reichert, and S. Rinderle-Ma. Change patterns and change support features–enhancing flexibility in process-aware information systems. *Data & Knowledge Engineering*, 66(3):438–466, 2008.

[68] Steve Weber. *The success of open source*, volume 368. Cambridge Univ Press, 2004.

[69] M. Weidlich, M. Weske, and J. Mendling. Change propagation in process models using behavioural profiles. In *IEEE International Conference on Services Computing, SCC'09*, pages 33–40. IEEE, 2009.

[70] Mathias Weske. *Business Process Management - Concepts, Languages, Architectures*. Springer-Verlag Berlin Heidelberg, second edition, 2012.

[71] Mathias Weske, WMP Van der Aalst, and HMW Verbeek. Guest editorial : advances in business process management. *Data & Knowledge Engineering*, 50(1):1–8, 2004.

[72] S. A. White and D. Miers. *BPMN: Modeling and Reference Guide*. Future Strategies Inc., 2008.

[73] Stephen A White. Introduction to bpmn. *IBM Cooperation*, pages 2008–029, 2004.

[74] Celia Wolf and Paul Harmon. The state of business process management. Website, 2012. `http://bptrends.com/surveys_landing.cfm`.

[75] M. Zur Muehlen. Organizational management in workflow applications–issues and perspectives. *Information Technology and Management*, 5(3):271–291, 2004.