Co-tutelle Individual Agreement between the University of Trento (Italy) and the Pontificia Universidad Catolica de Chile (Chile) Doctoral programme in Engineering Sciences

CLOSING THE GAP BETWEEN BUSINESS PROCESS ANALYSIS AND SERVICE WORKFLOW DESIGN WITH THE BPM-SIC METHODOLOGY

CARLA MARINA VAIRETTI

Thesis submitted to the Office of Graduate Studies in partial fulfillment of the requirements for the degree of DOTTORE DI RICERCA IN INFORMATICA E TELECOMUNICAZIONI

Advisors:
ROSA ALARCÓN (PUC, CHILE)
FABIO CASATI (UNITN, ITALY)

Santiago de Chile, June 2016

Co-tutelle Individual Agreement between the University of Trento (Italy) and the Pontificia Universidad Catolica de Chile (Chile)
Doctoral programme in Engineering Sciences

CLOSING THE GAP BETWEEN BUSINESS PROCESS ANALYSIS AND SERVICE WORKFLOW DESIGN WITH THE BPM-SIC METHODOLOGY

CARLA MARINA VAIRETTI

Committee members:
ROSA ALARCÓN (PUC, CHILE)
FABIO CASATI (UNITN, ITALY)
SERGIO OCHOA
MARCOS SEPULVEDA
YADRAN ETEROVIC
LUCA CERNUZZI
GUSTAVO ROSSI
CRISTIAN VIAL

Thesis submitted to the Office of Graduate Studies in partial fulfillment of the requirements for the degree of
DOTTORE DI RICERCA IN INFORMATICA E TELECOMUNICAZIONI

I dedicate my dissertation work to my family and friends. A special feeling of gratitude to my loving parents, Aida and Massimo whose words of encouragement and push for tenacity ring in my ears. My sister Paola has never left my side and is very special.

I also dedicate this dissertation to my many friends who have supported me throughout the process.

I devote this work and give special thanks to my husband Rodrigo, my wonderful kids: Bruno and Emilia for being there for me throughout the entire doctorate program. You three have been my best cheerleaders.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my supervisor Rosa Alarcón for her expert guidance, constant encouragement and enduring patience during my doctoral research.

During of year 2013 and 2014 I was given the opportunity to stay at the Department of Computer Science in Trento University. I would like to thank Professor Fabio Casati who invited me for this visiting. I am lucky to work in his group and have been surrounded by knowledgeable and helpful co-workers. I am grateful to Florian Daniel for his wonderful cooperation and all the interesting discussions, information and inspiration. I would also like to thank the colleagues there, in particular Carlos Rodrigez and Jorge Saldivar for discussions and assistance.

Finally, I would like to thank my family for their love and support. My special thank goes to Rodrigo for his encouragement and support.

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

RESUMEN

Hoy en día las empresas y organizaciones se enfrentan al reto de integrar y automatizar sus procesos de negocio. Un proceso de negocio es un conjunto de tareas relacionadas lógicamente, llevadas a cabo para generar un producto o servicio. Los procesos de negocio se implementan típicamente mediante servicios Web. Los servicios Web son interfaces programables que se pueden invocar a través de protocolos estandares de comunicación. En general, la necesidad de externalizar partes de los procesos de negocio da lugar a un gran número de servicios Web heterogéneos y distribuidos entre varias organizaciones y plataformas.

La capacidad de seleccionar e integrar estos servicios Web en tiempo de ejecución, es una característica deseable pues permitiría a las plataformas basadas en servicios Web reaccionar rápidamente ante cambios de necesidades de negocio y fallas, reduciendo los costos de implementación y minimizando las pérdidas por una pobre disponibilidad. La composición dinámica y automática de servicios Web tiene como objetivo generar un plan de composición (workflow) en tiempo de ejecución, que cumpla cierta meta de negocios. Las técnicas basadas en semántica explotan la anotación especializada de servicios para facilitar el descubrimiento de servicios simples o compuestos (matchmaking) que forman parte del plan. Por lo general, el proceso de matchmaking pone mayor atención en la selección de servicios y no tanto en el comportamiento del servicio compuesto (workflow) que tiende a ser muy simple. En la industria, por el contrario, los servicios compuestos o workflows son definidos manualmente y típicamente siguen patrones de control de flujo complejos que implementan procesos de negocio elaborados.

A pesar de que una técnica de composición dinámica y automática de servicios produzca un workflow ejecutable que implementa un proceso de negocio, éste debe ser validado en relación al objetivo del negocio. Este análisis de alto nivel, por lo general, es realizado por expertos del dominio (BPA, Business Process Analyst) quienes deben coordinar con los expertos (SA: System Architect) la implementación de los procesos de negocio. La conversación entre el BPA y el SA es un requisito fundamental durante el ciclo de creación de un proceso de negocios ejecutable. La falta de comunicación entre ambos participantes no solo genera retrasos en el tiempo de desarrollo, sino que también genera fallas en el producto y ciclos innecesarios que conllevan muchas veces, aumentos de los costos de producción y grandes pérdidas de dinero en las organizaciones.

En este trabajo de tesis, hemos desarrollado tres enfoques que permiten que esta brecha entre el BPA y el SA disminuya y su colaboración sea más efectiva. Por un lado presentamos una técnica de composición de servicios Web, dinámica y automática, teniendo en cuenta la descripción semántica de los servicios. El servicio compuesto corresponde a un workflow ejecutable con control de flujo complejo, facilitando la tarea de implementación del SA. Por otro lado, proveemos una herramienta que permite al BPA verificar y analizar el rendimiento de sus procesos de negocio. Y finalmente explotamos ambas herramientas con el fin de proponer una metodología que integra ambas perspectivas permitiendo transferir conocimientos en ambas direcciones, logrando así resultados prometedores que permiten descubrir inconsistencias en el desarrollo y el diseño del proceso de negocio así como entregar recomendaciones de mejores prácticas en ambos sentidos.

Palabras Claves: Ontología, Composición de servicios, Servicios Web semánticos, Procesos de negocios, Análisis de procesos, Simulación de procesos de negocio.

ABSTRACT

Nowadays companies and organizations are challenged to integrate and automate their business processes. A business process is a set of logically related tasks, carried out to produce a product or service. Business processes are typically implemented using Web services. Web services are programmable interfaces that can be invoked through standard communication protocols. In general, the need to outsource parts of a business processes results in a large number of Web services, which are, generally, heterogeneous and distributed among various organizations and platforms.

The ability to select and integrate these Web services at runtime is desirable as it would enable Web services platforms a quick reaction to changing business needs and failures, reducing implementation costs and minimizing losses by poor availability. The goal of dynamic and automatic Web services composition is to generate a composition plan (workflow) at runtime that meets certain business goal. Semantics based techniques exploit specialized services annotation to facilitate the discovery of simple or composed services (matchmaking) that form part of composition plan. Usually, the process of matchmaking places more attention in the selection of services and much less on the behavior of the composed service (workflow) that tends to be very simple. In the industry, on the contrary, the service compounds or workflows are manually defined and typically follow complex control flow patterns that implement elaborate business processes.

Although a technique of dynamic and automatic service composition produces an executable workflow that implements a business process, it must be validated in relation to the business goal. This high-level analysis is usually performed by domain experts (BPA Business Process Analyst) who must coordinate with the experts (SA: System Architect) the implementation of the business processes. The conversation between BPA and SA is a fundamental requirement for the cycle of creation of an executable business process. The lack of communication between both participants not only causes delays in development time, but also generates product failures and unnecessary cycles involving often increases in production costs and large losses of money in organizations.

In this thesis, we have developed three approaches that allow decreasing the gap between BPA and SA and making their collaboration more effective. On one hand we present a Web service composition technique that is dynamic and automatic and is based on services' semantic descriptions. The composed service corresponds to an executable workflow with complex control flow, facilitating the SAs implementation task. On the other hand, we provide a tool that allows BPAs to verify and analyze the performance of their business processes. And finally, we exploit both tools in order to propose a methodology that integrates both perspectives allowing knowledge transfer in both directions. We obtained promising results that reveal inconsistencies in the development and design of the business processes as well as provide recommendations for best practices in both directions.

Keywords: Ontologies; Services Composition; Semantic Web Services; Business process analysis; business process simulation.

1. INTRODUCTION

1.1. Motivation

Nowadays, companies and organizations implement their business processes and outsource Web applications on the Internet. That is, they publish, store and invoke their services through the Web, so that others can also use them. The ability of service providers to identify business processes and align each task to a particular Web service is not an easy goal, this happen because, in general, the selection of a suitable service (stored in a repository that contains different services) requires efficient search and selection engines.

A simple or atomic service is a service, which is not dependent on others to run. A composed service, however, is a service that publishes a standard interface and internally invokes the execution of other services (components), which could be atomic or composed (Medjahed, 2004). The composition can be static, if the composition model is built at design and compilation time; or dynamic, if the model components and their arrangement are decided at runtime (Alamri, Eid, & El Saddik, 2006). A composition model is a representation of the set of tasks that should be carried out during the execution of the composed service (e.g. workflow or plan), along with existing data dependence and control-flow among them.

Static composition, on the other hand, takes place during design-time when the architecture and the design of the software system are planned. The set of tasks and data dependency between tasks to be used are chosen, linked together, and finally compiled and deployed. (Dustdar & Schreiner, 2005).

There are several advantages of a dynamic service composition approach (W3C) (Dustdar & Schreiner, 2005), for example, given a set of atomic services, a large number of composed services is performed on demand. It is not necessary to maintain a local catalog of available services to create composite services, which often happens with static composition. The functionality offered by composite services, in the dynamic case, can be extended at runtime. Finally, dynamic service composition allows to customize the software according to individual customer needs, without affecting other users of the system (Mennie, 2000).

Web services composition is not a trivial task. In particular, it presents the following complications:

- First, the amount of available Web services is huge, and it is beyond the capacity of human beings to analyze these services manually (Lemos, Daniel, & Benatallah, 2015).
- Second, Web services can be created and updated on the fly, thus the composition systems need to detect these updates at runtime and have the ability to adapt the system to these changes (Rodriguez Mier, Pedrinaci, Lama, & Mucientes, 2015a).
- Third, Web services are often implemented by different organizations using different conceptual models to present the services characteristics. Differences may appear at syntax and semantic level been a layer the harder to resolve. This requires the use of semantic information in Web services description in order to facilitate the matching and composition of heterogeneous Web services (Klusch, Kapahnke, Schulte, Lecue, & Bernstein, 2015).

Some approaches these three problems above using using a planning technique (Hoffmann, Bertoli, & Pistore, 2007; Sirin, Parsia, Wu, Hendler, & Nau, 2004; Klusch, Gerber, & Schmidt, 2005; Pistore, Barbon, Bertoli, Shaparau, & Traverso, 2004; Xu, Chen, & Reiff-Marganiec, 2011) that derives the sequence of actions needed to find a target state (required outputs) from an origin state (inputs and preconditions). These techniques usually work well for small repositories that also have a high number of repository restrictions; they have some drawbacks such as high complexity, high computational cost and the inability to maximize parallel execution of web services. Other approaches such as (Aversano, Di Penta, & Taneja, 2006; Ghafarian & Kahani, 2009; Rodriguez-Mier, Mucientes, Lama, & Couto, 2010), work with a large number of services without guaranteeing an optimal solution; they are also very lengthy and require much memory. An approach that is similar to this thesis proposal, is capable of finding an optimal composition taking into account the matching between inputs and outputs for a specific requirement, at a semantic level. This approach scales much better than other techniques with a large number of services, showing a good performance (short response time) on large repositories (Rodríguez-Mier, Mucientes, Vidal, & Lama, 2012).The problem is that this approach implements simple flow-control patterns and it does not implement elaborate patterns mostly used in the industry.

But not only the selection of the Web service that better fits a task of a business process is important; software in general (such as individual Web Services as well as the entire set of services that implement tasks in the process) should be thoroughly analyzed before being released to users. Unfortunately, many of the processes that go to market, do not have the proper verification by the users who really know the process (e.g. a BPA, *business process analysts*) (Bhat, Gupta, & Murthy, 2006; Sutcliffe, 2012). The truly important thing is that the piece of software delivered to the market is analyzed by someone with proper knowledge, tools and methodologies (Weske, 2007b).

Interestingly, when it comes to business processes this is not a common practice. In fact, the BPA who designs the processes, don't often have the tools to analyze its own artifacts (Sutcliffe, 2012; Wiegers & Beatty, 2013). Of course, analysts can test their models under different execution scenarios manually, but if they lack automated support and advanced testing capabilities (e.g. metrics, statements, test reports, etc) (Galli et al., 2015), their conclusions may be wrong.

In the context of the business process management (BPM), many process models tasks are typically implemented using Web services (Weske, 2007a). This requires the participation of Web service developers and, it may happen that the role of the process analysis is performed by the same developers (Buchwald, Bauer, & Reichert, 2011). This in turn means that the BPA's perspective may be unavailable before implementing and executing processes. The identification of problems in the last phase of a Web service life cycle can be a slow and costly process.

When a BPA analyzes a given process, he or she needs to communicate the analysis goals, requirements and configurations to the software developer that implements and runs the Web service platform. Understanding and implementing well a process may thus require several iterations between the analyst and the developer. This collaboration may be hampered by the same difficulties already extensively reported in the literature on software engineering in general, and requirements analysis in particular, such as ineffective communication channels, inexpressive notations, and its dependent nature (Bhat et al., 2006), (Sutcliffe, 2012), (Wiegers & Beatty, 2013).

To address these problems, we need technologies and methodologies for aligning both roles supporting the complex dependencies that exist between high-level business process models (as used by domain experts) and technical workflow specifications (i.e., service composition schemas), respectively, (as used in IT departments) (Buchwald et al., 2011).

In this thesis, we propose a methodology that integrates these two perspectives. On one hand, we present a mechanism capable of recommending business processes implementations, assisting both the Web service developer and the BPA in the design and implementation of a business model and, on the other hand, a BP modeling and analysis layer, which facilitates the BPA requirements and expertise elicitation and knowledge dissemination so that the SA can enrich and improve the implementation process layer (Web services).

The recommending mechanism was implemented by an *automatic and dynamic Web services composition platform (CompoWS) (Vairetti, Alarcon, & Bellido, 2016). This approach proposes a semantic web service composition framework and algorithm that is evaluated in a web services dataset. It proposes the use of complex control-flow patterns together with service's signature and semantic annotations to identify composite services dynamically at runtime. With this technique is possible to obtain connections between services that are pre-calculated. It means that when a service is published, the algorithm recursively evaluates the stored services according to the previously described patterns,* *and generates all the necessary control-flow relationships, forming a graph that represents the semantic behavior of a potential composite service. These services can be deployed at runtime. The result is a compose service (a workflow) that enables recommendations to the SA when selecting the services to be used in the business model to implement, as well as detects inconsistencies at the implementation and business level.*

The BP modeling and analysis layer was implemented as a system of spreadsheets that allows the analysis of a business process by the BPA (Galli et al., 2015). This approach allows BPAs to draw models, configure the execution of the entire process, define tests and generate execution log in order to analyze all the processes. The characterization of the ideal execution scenario of a business process is analyzed through the definition of metrics and assertions at business-level, the tool allows the BPA to elicit their knowledge of the business process and also to verify its correctness.

1.2. Research Questions

The research questions that this thesis tries to answer are:

1.2.1. Automatic and dynamic functional service composition

Given a set of services, semantically described, can we design an scalable method to discover complex service compositions (i.e. complex workflows)?

To be more specific:

- Is there any semantic model that can be extended in order to represent service behavior following complex control-flow patterns (such as sequence or alternative) that satisfy the needs of business processes?
- How do we specify service behavior within the composed service and what kind of interaction can occur among a set of services implementing a business process?
- What type of information is required to identify this kind of Web services at run-time?

1.2.2. Eliciting BPA expertise

How can we enable BPAs (without development experience) to analyze and improve their BPs on their own, with less reliance on and intervention of SA?

And more specifically:

- How can we map the problem of business process analysis to the problem of configuring and analyzing data using a easy environment for organizing, analyzing and storing data?
- Since spreadsheets is a tool known for BPAs, How can we enable the generation of testing spreadsheets from an extended BPMN editor? How we can select a familiar BPMN editor? How can we enable the BPA to define their own metrics, assertions and test reports?
- It is feasible for the BPA taking the process execution log in order to define their own metrics, assertions and test reports without the intervention of software developers using a familiar tool?

1.2.3. Closing the gap between BPAs and SAs

How can we assist the BPA to transfer their BP knowledge to the SA and conversely, exploit the IT infrastructure to assist the BPA design choices when constructing a BP model?

In order to address the problem in detail:

- Which are the challenges faced by BPAs and SAs and which of them can be assisted by themselves?
- Are there specific tools that bridge the gap between BPAs and SAs? Are these powerful tools? Which validation criteria can be defined to evaluate these tools?

1.3. Problem statement and General Goals

The general goal of this thesis is to allow the definition of service compositions (workflows) that implement business process considering the complexity of BP behavior as well as business level expertise. To achieve this goal, we divide the challenge into three stages corresponding to corresponding to Chapters 3, 4 and 5 of this thesis, it is important to understand that Chapters 3 and 4 serve as the basis of Chapter 5 that satisfy the general goal.

1.3.1. Automatic and dynamic functional service composition

This part of the thesis addresses the problem of automatic and dynamic Web service composition considering only functional requirements. The problem is seen as a graph search problem from the point of view of the semantic input-output message structure matching.

The first goal is the creation of a method able to determine, given a service description, all the possible service composition fragments that the service is part of. The second goal is to determine a set of control-flow patterns that define the behavior of a composition fragment and from that the possible relationships among services. The third goal is the creation of a method able to calculate, given a request, an extended service dependency graph which represents a solution for the request.

1.3.2. Eliciting BPA expertise

The problem we aim to solve in this part of the thesis is devising an approach that enables the BPA to leverage their expertise, characterizing and testing the performance of their business process without the need for software development skills.

The first goal is to enable the BPA to specify their own business metrics M and assertions A to characterize their business process and elicit part of her expertise. The second goal is to allow the BPA to run a simulation of the BP considering such characterization and to obtain a process execution log L, so as to be able to analyze the behavior of a business process BP. The third goal is to do so in a fashion that allows the BPA to easily discuss her findings with the software developer in charge of implementing processes.

1.3.3. Closing the gap between BPAs and SAs

The purpose of this part of this thesis is to propose a methodology that integrates business process and workflow modeling allowing to transfer expertise in both directions.

The first goal is to allow the BPAs to transfer their expertise to the SAs by enriching and refining the service layer. The second goal is to assist the BPA's design choices when constructing a business model using the recommendations from the service capacity deployed (i.e. workflows).

1.4. Hypothesis

The following are the hypothesis of this thesis:

H1: Services signatures (input and output) along with a goal and a set of rules make possible to automatically and dynamically discover service compositions with complex behavior.

H2: Spreadsheets together with a BP execution log allows BPAs to easily analyze their BPs on their own by describing metrics, assertions and test reports.

H3: Signavio-Core BPM editor along with a service engine supporting H1 make possible to detect new business level semantic relationships and inconsistences in order to annotate them on a service engine supporting H1 that match semantically related services.

H4: A service engine supporting H1 that match semantically related services can recommend to BPA a set of services that serve as implementation of BP tasks; and it can also identify possible implementation level inconsistencies between services.

1.5. Thesis Work and Main Contributions

This Section summarizes the results presented in a collection of selected papers published by or submitted to international journals. Each paper is presented as a separate chapter and briefly summarized, together with its relation to the contributions and research questions of this thesis.

1.5.1. Automatic and dynamic functional service composition

The first part of this thesis describes a technique to derive complex composite service behavior semantics extending the MSM ontology (Pedrinaci et al., 2010) in order to allow the specification of simple and complex control-flow patterns based on the service's signature. It also enables the automatic discovery of such patterns through a set of rules and presents the algorithms and queries required to dynamically pre-compute all the possible combinations between services. This part presents also the algorithm and queries required to discover composite services.

The major contribution of this part is the specification of a minimal extension to MSM ontology in order to specify complex control-flow in service composition as well as the specification of a set of signature-based rules that allow us to infer complex control-flow relationships among services. Specific contributions of our work are as follows; first we improve the performance, in terms of response time, of generating composite services without requiring in memory calculus, which may facilitate scalability of our approach through horizontal scalability. Second we allow the generation of more elaborate compositions that correspond to complex business patterns adopted in most real scenarios, without losing performance when compared to approaches that only consider simple business patterns.

These contributions were published in the following journal: C. Vairetti, R. Alarcon and J. Bellido. A semantic approach for dynamically determining complex composed service behaviour. Journal of Web Engineering - Rinton Press. April 2016.

1.5.2. Eliciting BPA expertise

The second part of this thesis describes a spreadsheet-based approach for business process model analysis that maps the problem of business process performance analysis and verification to the problem of configuring and analyzing data in common *spreadsheets*. It enables the *generation* of analysis spreadsheets from an extended business process model editor for BPMN process models (Object Management Group (OMG), 2011); it enables also the BPA to define their own *metrics, assertions* and *analysis reports*; and it automates the *simulation* of BP and the generation of *process execution logs*.

A major contribution of this part is the development and specification of a formal approach to support the analysis and improvement of Business Process Models using Spreadsheets. Some specific contributions of our work are: first we testify that the interaction between the BPAs and the developers were well disposed and facilitated by our approach, confirming the suitability of the approach for collaborative BP analysis. Second, we allow BPAs to obtain a concrete feeling of how their processes behave if deployed in a real BP system by emulating web services and visualizing process progress in a process-monitoring dashboard.

These contributions were published in the following journal: J. Saldivar, C. Vairetti C. Rodriguez , F. Daniel, F. Casati and R. Alarcon. Analysis and Improvement of Business Process Models Using Spreadsheets. Information Systems - Elsevier. January 2016.

1.5.3. Closing the gap between BPAs and SAs

The third part of the thesis focuses on the importance of understanding the requirements inherent in bridging the software perspective (i.e. Web service implementation) and the business perspective (i.e. which tasks can be executed by a particular service and which conditions shall be considered during the process execution).

A major contribution of this part is the development and specification of six bridging strategies to exploit the two perspectives and propose a methodology that integrates both perspectives allowing to transfer expertise in both directions.

These contributions were submitted to the following journal: C. Vairetti, R. Alarcon, C. Rodriguez , F. Daniel and F. Casati . Closing the gap between IT and business stakeholders: The case of web service reuse, composition and analysis for service-based business processes. Information Systems Journal - Wiley Online Library - June 2016.

1.6. Document Structure

The remainder of this thesis is organized as follows:

In Chapter 2, fundamental definitions and terminology are given to the reader regarding the problem of how can we shorten the gap between BPAs and SAs. When we chose this subject, different research directions inspired us. They are the researches in Web services, Service Oriented Architecture, Functional Testing and Business/Information Technology areas. Those directions are presented here as the background of the research subject in this thesis.

Chapter 3 presents the paper that summarizes the results of the first part of the thesis about *automatic and dynamic web service composition*. It consists of the paper *A semantic approach for dynamically determining complex composed service behaviour*. The paper presents a technique to derive complex composed service behaviour semantics, such semantics make possible to dynamically and automatically discover complex services compositions. We have implemented and tested this technique with a known dataset with better performance when compared to simple service composition strategies.

Chapter 4 explores the ability to test the business process model execution by a BPA. It consists of the paper *Analysis and Improvement of Business Process Models Using Spreadsheets*. This article presents a tool that faces the lack of suitable instruments for business process analysts, who design the processes, and aims to provide them with the necessary instruments to allow them to analyze their processes. A spreadsheet paradigm that allows performing business process analysis tasks is presented. BPAs write metrics and assertions, run performance analysis and verification tasks, and generates reports on the outcomes. The results of two independent user studies demonstrate the viability of the approach.

Chapter 5 presents a methodology to enrich and close the gap between BPA and SA in both directions. It consists of the paper *Enriching Workflows with BPM expertise and vice versa*. In this paper two perspectives, the one of the BPA and the one of the SA, are integrated under a methodology that allows expertise transference in both directions. We consider also that the underlying software infrastructure is able to automatically infer workflows (service composition) so that it is capable of recommending certain implementations. We propose six bridging strategies, which were tested with promising results.

Chapter 6 presents the main conclusion of this thesis and future research. The chapters of this thesis are self-contained, that is, they can be read independently.

2. BACKGROUND

2.1. Introduction to Web Services

The World Wide Web is more and more used for application-to-application communication. Web services are programming language independent pieces of software that offer a programmable interface that can be invoked via standardized Web communication protocols, such as HTTP, in order to deliver the functionality they encapsulate (Erl, 2008).

A more precise definition is provided by the UDDI consortium, which characterizes Web services as self-contained, modular business applications that have open, Internet-oriented, standards-based interfaces (ORG, 2000). According to the World Wide Web consortium (W3C), and specifically, to the group involved in the Web Service Activity: a Web service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A Web service supports direct interaction with other software agents using XML-based messages exchanged via Internet-based protocols (Austin, Barbir, Ferris, & Garg, 2004).

2.2. Service Oriented Architecture

Web Services technologies encourage the reuse of software and software interoperability not only across organizational boundaries but also within a large organization, by exposing some functionality via the infrastructure of the Web, following open Web standards. It allows also the reuse of legacy systems functionality, which is particularly attractive in business scenarios. Web services facilitate the reuse of business components and native Web applications. They allow consumers to find, combine and invoke functionality with more accuracy, supporting automation, integration and the creation of B2B (Business to Business) processes in a more dynamic, scalable and low cost manner.

Service-oriented computing (SOC) (Papazoglou, Traverso, Dustdar, & Leymann, 2008, 2007) main promise focuses on the notion of services as an essential element for the development of software applications. Services are pieces of software that are published by service providers who are responsible for its maintenance and implementation. Services must be described and these descriptions are used to publish service capabilities. These capabilities must be discovered so that services could be used as components of new services.

Services may be atomic or composed if they invoke other services in order to complete their actions. Composition is the action of combining the services functionality (components) on a new service (compound). This new service provides added value and also can be part of other compounds. For example, a business transaction may require three steps: credit verification, the availability of a product and the issuing of a purchase order. If each of these tasks is provided as an atomic service, the service that implements the whole, complex transaction (invoking other services) is called a composite service. Web services allow the use and combination of functional components distributed inside and across enterprise boundaries. These services are loosely coupled between them, which allow to create scalable and heterogeneous applications that join organizations and diverse computing platforms (Papazoglou et al., 2008).

Web service architecture includes three components: clients, providers and service repositories. Providers publish their services in these repositories and clients discover these registered services through queries. Traditionally, Web services infrastructure is based on three basic standards, WSDL (Web Services Description Lenguages) (Chinnici, Moreau, Ryman, & Weerawarana, 2007), SOAP (Simple Object Access Protocol) (Box et al., 2000) and UDDI (Universal Description, Discovery and Integration) (ORG, 2000). WSDL is an XML-based language used to describe the service interface (input and output parameters, data types and the service URI endpoint). SOAP is a standard protocol for the exchange of messages transmitted over HTTP between the service and the client. UDDI allows the definition of global repositories where service information is published. Currently, UDDI is not used. The idea behind UDDI was to discover web services and their capabilities in some sort of web services marketplace. In the real world, description a web service, it can be placed on the same server that is hosting the web service. Web services promote the reuse of software, which reduces development costs, increases maintainability, and enables organizations to create composed services (Erl, 2008).

As mentioned before, service composition can be static and dynamic, but in addition, there are three strategies that are applied to both modes (Dustdar & Schreiner, 2005): a) Model-driven composition where business processes can be built dynamically by composing web services in a model driven fashion where the design process is controlled and governed by a series of business rules, b) Declarative composition where composability rules are used to determine whether two services are composable; c) Automated vs. manual web services composition which differ from a) and b) in the way the components are selected during the process of composition. Among the techniques for implementing dynamic composition, we can find wrappers, adapters, specific languages, workflows, declarative and semantics composition as well as hybrid systems which have several advantages and disadvantages (Dustdar & Schreiner, 2005).

2.3. Semantic Web Services

In order to compose services automatically, additional information provided by WSDL is needed because the WSDL annotation elements are essentially syntactic and meaningless so that it is not possible to infer their purpose and way of use automatically. A popular way of representing semantic aspects of a system is ontologies. Formally, an ontology is formal and explicit specification of a shared conceptualization (Gruber, 1993). *Conceptualization* refers to an explicit definition of an abstract model that characterizes a phenomenon in the world through the identification of relevant concepts and restrictions of the phenomenon. *Formal* represents the fact that the ontology should be understandable by machines. *Shared* reflects the notion that an ontology captures consensual knowledge of a community (Gruber, 1993), (Studer, Benjamins, & Fensel, 1998), (Borst, 1997).

There has been proposed various languages based on different logical formalisms for representing ontologies. For instance KIF (Knowledge Interchange Format), IMCO (Operational Conceptual Modeling Language) (Tanasescu, Domingue, & Cabral, 2004) and F-Logic (Frame Logic) (Th & Schlepphorst, n.d.).

OWL is one of the most popular formal languages for representing ontologies; it is presented as an XML schema that allows to describe concepts (e.g. Classes) and instances (i.e. Objects) as well as relationships among them (Booth & Liu, 2007). There are three variants of the language: OWL-Lite, OWL-DL and OWL-Full, in order of expressivity. Each of these incorporates additional constraints (cardinality properties, transitivity, and others). RDF (Lassila, Swick, et al., 1998) (Resource Description Framework) is another language for representing ontologies but since it is less expressive than OWL, it requires simpler parsers and query engines becoming very popular. Other popular languages are N-Triples (Grant & Becket, 2004) or N3 and Turtle which have a compact format for expressing ontology triples. A triple is a statement in a $< subject, predicate, object >$ form, where the *predicate* semantically relates the *subject* and the *object*. The object can be a concept (similar to the *subject*) or a *literal* (a simple datatype). In OWL and RDF, the three elements are represented through IRIs (Dürst & Suignard, 2004).

Regarding the infrastructure required by ontology-based systems, we can distinguish some key components like parsers and serializers that transform ontologies between various formats (eg. XML to N-Triples); Repositories (Sesame (Schenk & Petrák, 2008), JENA (Reynolds, 2004)) that store both ontologies and their instances in a database that is specialized in the storage and retrieval of triples; languages and search engines (SPARQL, SeRQL (Broekstra & Kampman, 2004)) for retrieving information according to structured queries and the applications on top of all these components. JENA is a Java framework used for building Semantic Web applications, it consists of an API that provides mechanisms for reading and writing triples in different formats (RDF N-Triples, Turtle, etc.), as well as persistency and memory storage; it also provides an implementation of an SPARQL query engine.

2.3.1. OWL, WSMO, WSDL-S, OWL-S, SAWSDL

Service semantics refer to the purpose and meaning of the service and the elements that characterizes it (e.g. WSDL elements), as well as the meaning or interpretation of the conditions that must be considered when invoking it and its results. Many research have arisen in the area of Semantic Web Services, including proposals to W3C standards such as WSMO (Domingue, Cabral, Hakimpour, Sell, & Motta, 2004), OWL-S (Martin et al., 2004) and SAWSDL (Lausen & Farrell, 2007).

WSMO is a framework that comprises a meta-ontology, an architecture, and a composition model; it presents some implementations in different contexts such as education (IRS III (Domingue et al., 2004)). In WSMO, the domain ontology remains separate from the services implementation and relies on expressive knowledge representation formalisms (WSML (De Bruijn, Lausen, Polleres, & Fensel, 2006a)) as well as rule languages to support tasks such as service discovery. These ontologies are rich and complex, and they demand a strong knowledge of ontological design, as well as of the ontologies that describe specific services (which are based on F-Logic) and of the platforms and tools for implementing them (such as WSMX (Haller, Cimpian, Mocan, Oren, & Bussler, 2005)). Furthermore, WSMO is based on a sophisticated logical reasoning for automatic discovery, mapping, composition and execution of composite services, requiring to centralize all service descriptions which represents a significant limitation on the scalability of these approaches. Nevertheless, the core limitation of WSMO is that it fails to comply with W3C standards (e.g. it requires WSML services description instead of WSDL), making it impossible to extend existing Web services with WSMO technology although efforts have been made to provide support for interoperability (Vitvar, Kopeckỳ, Viskova, & Fensel, 2008a).

OWL-S, on the other hand, proposes a meta-ontology of services, but does not prescribe an architecture, a mechanism of composition or a way to connect existing Web services with the ontology. OWL-S requires a domain ontology and a semantic description of services. The domain ontology remains separate from the services implementation and relies on expressive knowledge representation formalisms (Martin et al., 2004). Like WSMO, OWL domain ontologies are rich and complex, and demand a strong knowledge of ontological design as well as the ontologies themselves. The meta-ontology proposed by OWL-S involves three perspectives: support for Web service discovery (*Service profile*), service usage (*Service Model*) and service access (*Service Grounding*) (Burstein et al., 2004).

The *Service Profile* includes a description of what does the service, its limitations, applicability, quality and requirements that the service requester must satisfy in order to use the service successfully. *Service Grounding* specifies the details of how an agent could access a service, it specifies a communication protocol, message formats and other specific details of service communication, such as port numbers used to contact the services. The *Service Model* describes the service capabilities that enable service orchestration and defines two components: Process that describes the service properties such as inputs, outputs, conditions, parameters and effects; and Process Control that describes the process state for atomic (simple request-response service invocation) or composed services (Burstein et al., 2004). Process Control for composed services requires to specify the composite structural semantics through control-flow instructions such as sequence, split, split-join, any-order, choice, if-then-else, while and repeat-repeat-until (Coalition, 2004).

In addition, there are two recommendations of the W3C for semantic description of Web Services, WSDL-S (Verma & Sheth, 2007) and SAWSDL (Lausen & Farrell, 2007) that extend the WSDL language through lightweight semantic annotations. Semantic Annotations for WSDL (SAWSDL) is a W3C recommendation that allows to add semantic annotations to the terms of a WSDL description. This semantics, when expressed through formal languages, can help to disambiguate the description of Web services. These terms refer to an ontology to allow classification, discovery, matching, composition and invocation of Web services. The advantage of this approach is the possibility to extend existent WSDL descriptions with semantic representations (e.g. RDF, OWL, WSML, or any other proprietary language). SAWSDL does not prescribe what a semantic description of a Web service could be (e.g. OWL-S, WSMO or any other model). SAWSDL introduces three new annotations or elements of WSDL extensibility: the *modelReference* to specify the association between a WSDL component and a concept in some semantic model, the *liftingSchemaMapping* and *loweringSchemaMapping* elements that are added to WSDL elements (parameters and data types) to associate them with conceptual definitions of an ontology and XML data schemas respectively.

2.4. Service composition

The most popular technique for dynamic service composition is based on service signature; it considers the dependencies between inputs and outputs in order to generate a composition plan at runtime. Plan generation uses techniques such as dynamic forward or backward search, data dependency and control-flow. The plan can be represented as a directed acyclic graph where nodes are services and arcs correspond to the dependencies between them. Such dependency is established from the service signature (Input, Output), services semantic distance, or services similarity (perfect-match, plug-in, subsume, y zero-match, etc.) (D'Mello, Ananthanarayana, & Salian, 2011). The composition plan includes the services invocation control-flow. Some approaches, such as Linked-OWL (Ahmad & Dowaji, 2013), require that the user defines the composition plan as an abstract workflow, services are stored in a repository and each service component is searched for using a SPARQL query. Others propose a Petri net-based algebra for modelling control-flow (Hamadi & Benatallah, 2003). Service behaviour is modelled through six control-flow patterns: sequence, alternative, iteration, arbitrary sequence, parallel with communication, discriminator, selection, and refinement. OWL-S is a well-know semantic service model that includes basic control flow patterns (invocation, sequence, alternative, join and split) (Zhang, Zhang, & Liu, 2010). A UML- based universal language considering simple and complex interaction has been also proposed (Kylau, Stollberg, Weber, & Barros, 2012). Workflows are considered complex interaction patterns (sequence), but the authors do not specify how such workflows could be modelled. In (van Der Aalst, Ter Hofstede, Kiepuszewski, & Barros, 2003a; N. Russell, Ter Hofstede, & Mulyar, 2006) 43 control-flow patterns are presented in great detail.

Service discovery is achieved by defining a matching degree between services (matchmaking) and the client's request. Matchmaking is considered logics-based, if an ontological structure is used to determine a similarity between the services and the request; non-logical, if syntactic, structural or numeral strategies are used, or hybrid. In YASA (Chabeb, Tata,

& Ozanne, 2010) a hybrid matchmaking mechanism pre-selects a set of services, and determines a services matching degree using a IOPE (Input, Output, Preconditions, Effects) logical approach (exact match, subsumption, etc.); finally, the matching degree is weighted using non logical strategies (min-average, cupid, and combinatorial algorithms). Logical and statistical matching degree is broader in IO OWLS- MX (Klusch, Kapahnke, & Zinnikus, 2009) the user determines the matching degree, and the similarity threshold, and the service provider, requester and matchmaker share a minimal vocabulary with mapping rules (synonyms based on a WordNet thesaurus) to classify service request's input and output concepts. In (Bener, Ozadali, & Ilhan, 2009), a hybrid PE (Preconditions, Effects) algorithm ranks published services according to the semantic distance of concepts (counting edges). WordNet is used to determine synonymy and concept subsumption. SAM is an IO matchmaking algorithm for OWL-S that considers semantic descriptions for requested and provided services (Brogi, Corfini, & Popescu, 2008). The algorithm simplifies services into trees and creates a BF-hypergraph representing the dependencies among matched services; the dependency graph in analysed to determine whether it satisfies the user request or additional input requests are required from the client to produce a full match.

Semantic-based service composition techniques rely on semantically annotated services to facilitate service discovery. Annotations make explicit the semantics of the input and output parameters, as well as the service goals among others. For instance in Kill and Nam (Kil & Nam, 2013), conceptual relationships between services parameters are used to find the semantically close services that satisfy the user's requirement. This technique is based on model checking and a matchmaking process. Other solutions are based on various description languages capturing different service's semantics. For instance, OWL- S proposes three ontologies specifying a service (a service profile indicating the service goals, limitations, quality and requirements for the service consumer), usage (a service model), and access (service grounding). A service model component, the Process, describes properties such as inputs, outputs, preconditions, parameters, and effects; the Process Control component, describes the processing state and together they allow designers to create workflows. Unlike OWL-S, the Web Service Modeling Ontology (WSMO) includes an execution framework and a set of four ontologies that describe the information used by the other ontologies; the objectives fulfilled when executing the service; the services' capabilities and signature; and the mapping between components. Service discovery is a three-step process involving the service signature and goal, whereas service composition is generally addressed through forward-chaining techniques that determine the set of valid state transitions (i.e. service invocation) in order to achieve a goal (Domingue, Galizia, & Cabral, 2005). WSMO-Lite service ontology (Vitvar, Kopeckỳ, Zaremba, & Fensel, 2007) is an extended Web service specification stack, adding semantic layers that offer richer descriptions for Web services with the goal of the maximal compliance with Web standards. SAWSDL (Semantic Annotations for WSDL) is a W3C recommendation for semantic service descriptions, which extends XML-based WSDL with semantic annotations without imposing a representation language (e.g. RDF, OWL, WSML, etc.), and without prescribing a service semantic model (i. e. could be compatible with OWL-S, WSMO or other models). Some WSDL elements can be annotated with a modelReference attribute that refers to the equivalent concept in some semantic model through a URI.

On other hand, the DSD (Klein, Konig-Ries, & Mussig, 2005) language describes services from a pure state-based approach; it requires services to declare its effects and pre-conditions. Service's domain is modelled through a hierarchical ontology specialized in various layers down-to instances. Instance sets (instances subgraphs with constrained attributes) serve as a medium to specify unambiguously consumer's request and provider's capabilities. A service is composed dynamically from a request. In DSD, and such request is an instance set that specifies the expected state of the world after a successful service execution. For YASA, a query-formatted document (i.e. an extended SAWSDL description with annotations on the interface, operation, input and output elements) representing an abstract description of the expected service is used as the request specification. For WSMO, the desired goals as well as the input values are specified (Domingue et al., 2005) through abstract goal templates describing functional capabilities and constraints that are instanced and customized by users.

Services orchestration aims to generate a composition plan determining the service components, the data to be interchanged and the control-flow regulating services interaction. Service orchestration is the most popular paradigm in REST service composition research. For instance, the JOpera framework (Pautasso, 2009a) proposes a visual language and an execution platform for building large applications including multiple REST services. In JOpera, the orchestrator is implemented as a central (composite) resource that drives the control and data flow. Both flows are visually modeled as two separate design documents producing a BPEL compatible executable program for orchestration engines. Other works such as a BPEL extension for REST (Pautasso, 2009b) and a BPEL-inspired workflow composition language called Bite (Rosenberg, Curbera, Duftler, & Khalaf, 2008) are used to describe control/data flow and data transformations for web service composition and control flow dependencies are modeled and implemented using a Petri Net in (Alarcon & Wilde, 2010). A set of control-flow patterns that implement stateless REST service composition are described in (Bellido, Alarcón, & Pautasso, 2013). Authors describe a technique for decentralized REST services composition that takes into account the constraints of REST architectural style in the composition process. The implemented control flows follow a choreography paradigm implemented through callbacks and redirections.

Semantic REST service composition approaches model the composition as graph patterns (Krummenacher, Norton, & Marte, 2010). For instance, Verborgh et al. propose an RDF-based approach for describing RESTful services where a service composition is implemented through SPARQL queries (N3). The control-flow is modeled as query patterns following the RESTdesc language; data flow is dynamically resolved when the query is performed and the served representations can be later processed (Verborgh et al., 2012). Mismatches between data formats, fully supported in JOpera, are not considered (Verborgh, Steiner, Deursen, Van de Walle, & Vallés, 2011). Semantic Web technologies are used to model contextual information from users, sensors and things so that machine-clients can make sense from the responses (He, Zhang, Huang, & Cao, 2012).

Other approaches for REST service composition focus on service description. Proposals include WADL (Hadley, 2006), WSDL 2.0 (Chinnici et al., 2007), and SA-REST (Lathem, Gomadam, & Sheth, 2007). These descriptions facilitate the automation of machine-client and RESTful services interaction. These languages are strongly influenced by existing imperative service description languages (input/output) and do not capture well the resource-centric nature of RESTful WSs (transitions of resources). ReLL (Alarcon & Wilde, 2010) differs from the other three in that it is hypermedia-centric, supports REST architectural constraints requiring less coupling between clients and services. Semantic REST service descriptions have been also proposed. For instance, hRESTS (Maleshkova, Pedrinaci, & Domingue, 2009) proposes a microformat to annotate HTML service descriptions that can be used also by crawlers and search engines to find services. The microformat extends the HTML description with semantic annotations so that RESTful services can be discovered, composed and invoked automatically. Four aspects of service semantics: information model, functional semantics, behavioral semantics and nonfunctional descriptions, instances are modeled by MicroWSMO (Maleshkova et al., 2009). SWEET (Maleshkova et al., 2009) supports users in searching for suitable domain ontologies and in making semantic annotations in MicroWSMO in order to provide a higher level of automation on tasks with RESTful services, such as discovery and composition. WSMO-Lite (Vitvar, Kopeckỳ, Viskova, & Fensel, 2008b) ontology is used for describing the content of semantic annotations in WSDL.

2.5. Functional Testing

In software development, especially in quality control, one principal aspect is software Testing and particularly, functional tests (Beizer, 1995). Functional testing is responsible for verifying whether all the system works correctly or whether a particular application behaves as expected; for this purpose, this kind of test needs a given set of controlled execution scenarios or test cases (Gutiérrez, Escalona, Mejías, & Torres, 2005).

Functional testing typically involves three main steps (Howden, 1980), (Beizer, 1995):

- Planning what should be tested and the corresponding approach. In this step, it is necessary to identify the functions that the software is expected to perform.
- Execution of a test. It also involves preparing the testing environment, to complete or execute the test and to determine test results. For this purpose the creation of input data, based on the function's specifications, and the determination of expected output, based on the function's specifications, are required.
- Evaluation. It means to compare the actual test outcome with what the correct outcome should have been. The expected (correct) outcome should be easy to record for this purpose.

There are three main testing methods: black box testing that bases its test cases on the specifications of the software component under test. This type of test is characterized by loading the input and examining its output without considering the program structure, the outside world comes into contact with the test item only through a specified interface. Task descriptions are necessary for creating all the test cases needed. On the other hand, in white-box testing, the test cases are created based on the knowledge of the input-outputs and the internal program structure. This method attempts to verify the correct behavior knowing the whole structure of the system. Finally, gray-box testing is a combination of both methods mentioned before. According to (Kaner, Falk, & Nguyen, 2000) functional testing is a type of black box testing.

There are different testing levels based on the test target or in the objectives of testing (Kaner et al., 2000). We have unit testing, integration testing, system testing and system integration testing, which testing individual units of source code, testing software modules that are combined and tested as a group, testing a completely integrated system and testing software system's coexistence with others, respectively. We also have regression testing, acceptance testing, automated testing, alpha testing and beta testing.

In order to automate the testing process, it is possible to write a program that performs the testing process, obtaining therefore a long maintenance that records the software life cycle, because even minor patches over the lifetime are subject to testing. Software testing utilizes a variety of tools to automate the testing process (Burnstein, 2006).

Alpha testing consists of simulated or actual operational testing performed by different developers; it is a kind of internal acceptance testing that is executed before the software goes to beta testing. Beta testing comes after alpha testing and can be considered a form of external user acceptance testing, usually performed by the open public, with the purpose of increasing the feedback field to a maximal number of future users (Beizer, 1995) .

An entirely different way of distinguishing among testing types is to look at the range of possible input data. One way is to use input data generated at random to confront the application under test, for instance, since typically testers pay no attention to expected data types, a testing scenario is to feed a random sequence of numbers, letters and characters into numeric data fields. Another way is to use spot check testing that resembles random data testing, but in this approach, input data is selected from a mass of real data that the software will encounter in its future use. To make this approach work, we need the largest quantity of real world data possible. Then, we can use a random algorithm to select a manageable subset from this pool of data and feed the data to the system. Finally, boundary value tests are specific tests, which check the most extreme values of input data. Boundary value tests are always individual tests with expected outcomes that can be precisely specified and checked (Burnstein, 2006).

2.5.1. Testing in Business process

For the purpose of defining a business process as a collection of activities/tasks that produces a specific service/product for a particular customer, it is common to think that all this activities need a functional testing in order to check if the application behaves as expected and all the components are working properly (Young, 2008).

The analysis of business process has triggered many research efforts, yielding a variety of different approaches. In the following, we discuss those related works that fall into the context of this chapter, which includes service-based BP testing, BP simulation, compliance checking, process mining and modeling and testing spreadsheet.

The problem of **service-based BP testing** took significant relevance with the SOA and its use to support the operation of BPs. In particular, many approaches have been proposed to address this problem in the context of BPs represented with BPEL (Andrews et al., 2003). For example, some of the works in this context are dedicated to perform *unit tests* of web service compositions. Unit tests in this context means testing each web service and their corresponding interfaces, i.e., each operation offered and invoked by the service (Mayer & Lübke, 2006; Li, Sun, & Du, 2008). A side problem associated to the testing of BPEL processes is the generation of test cases. Works like (García-Fanjul, Tuya, & De La Riva, 2006; Yuan, Li, & Sun, 2006; Yan, Li, Yuan, Sun, & Zhang, 2006) propose approaches for the generation of test cases using techniques from model checking, graph search and concurrent path analysis. Other types of tests performed on service-based BPs include regression testing (Li, Tan, Liu, Zhu, & Mitsumori, 2008) and integration testing (Bucchiarone, Melgratti, & Severoni, 2007). All these approaches require special software testing and development skills.

BP simulation has been employed for the purpose of testing BPs. For example, (Aguilar, Rautert, & Pater, 1999) propose a BP simulation methodology to analyze the performance of financial BPs in unforeseen, potential situations. In the context of service-based BPs, (Narayanan & McIlraith, 2003) propose the use of simulation to test the preservation of properties (e.g., safety conditions) associated to the services responsible for the BP execution by combining Petri-Nets and DAML-S. (Chandrasekaran, Miller, Silver, Arpinar, & Sheth, 2003) use simulation to monitor and analyze the performance of individual web services involved in a BP. (Tan & Takakuwa, 2007) and (Wynn, Dumas, Fidge, Ter Hofstede, & van der Aalst, 2008) use simulation to evaluate the impact of BP re-engineering tasks on process performance.

In the context of *process mining* (van der Aalst, 2011), techniques such as conformance checking and process discovery have been employed to check whether or not a BP behaves as expected. **Conformance checking** (Rozinat & van der Aalst, 2008; van der Aalst, Dumas, Ouyang, Rozinat, & Verbeek, 2008) verifies whether the traces of execution of a BP conform with a given BP model. In order to do so, the approach proposes to *replay* the real process execution data on the BP model to detect if there are mismatches between the two. Conformance checking therefore checks if the event log structurally matches the process model, or, in other words, it checks if the control flow that underlies the event log matches that of the process model. It therefore only considers the structure of the BP model as the specification of the expected behavior and does not focus on process-specific metrics. Moreover, while the main use case of conformance checking consists in using a real event log to check against a predefined process model *a posteriori* (i.e., after a real execution of the process), in our case we use a simulated log to check *a priori* (i.e., before the real execution of the process) if a BP behaves as expected.

Process discovery is the task of inferring a BP model from process execution data (van der Aalst, Weijters, & Maruster, 2004; Motahari-Nezhad, Saint-Paul, Casati, & Benatallah, 2011). Testing a BP with process discovery can be done by first inferring the BP model from the process execution data and then comparing if the inferred model corresponds to the expected model. This comparison can be done either manually or using automatic techniques such as those based on BP similarity (R. Dijkman, Dumas, & García-Bañuelos, 2009). There are a set of commercial (e.g., ARIS PPM, HP BPI, and ILOG JViews) and academic (e.g., EMiT, Little Thumb, InWoLvE, Process Miner, and MinSoN) process mining tools. The main goal of the academic tools is to extract knowledge from event logs for discovering processes. The commercial tools are more oriented to the design, analysis and optimization of BPs using for example, charts and dashboards. In addition, HP BPI can discover a BP from event logs. Our approach differs from these process mining techniques in that they are meant to be used *after* the real process has been executed. This contradicts our purpose of using BP testing as an instrument to prevent unwanted behaviors. Moreover, these two approaches focus only on the structure of the BP, while our approach tests the dynamics and data produced by the execution of the BP through the use of user-defined metrics and assertions.

Compliance checking is the problem of verifying whether a BP model or its execution adheres to a set of compliance rules (i.e., the *expected behavior*) that typically emerge from laws, regulations and standards. This problem has been addressed both statically and dynamically. In *static compliance checking*, only the model of the BP is checked against the compliance rules. For example, (Liu, Muller, & Xu, 2007) address the problem by expressing the BP model in Pi-Calculus and the corresponding compliance rule in Linear Temporal Logic. Using this representation, model checking techniques are used to check whether the BP model complies with the compliance rules. (Governatori, Milosevic, & Sadiq, 2006) proposes a logic-based formalism to represent both the semantics of contracts and compliance checking procedures. The formalism used is Format Contract Language, which is based on Deontic Logic and helps in representing and checking contrary-to-duty obligations in contracts. In *dynamic compliance checking*, BP execution evidences are used to check for compliance. Works by (Rodríguez et al., 2013) and (Silveira et al., 2010) propose the use of so-called Key Compliance Indicators (KCIs) to measure the compliance level of service-based BPs from process execution data, e.g., to measure the fulfilment of Service-Level Agreements (SLAs). In a similar approach, (Casati, Castellanos, Dayal, & Salazar, 2007) and (Sayal, Casati, Dayal, & Shan, 2002) propose to warehouse process execution data to enable the monitoring, analysis and reporting on the performance of BPs, e.g., to check the duration of process execution instances when they are constrained in time. The approach we present in this chapter is similar to dynamic compliance checking, with the difference that we enable the use of simulated data, next to real data, to check different execution scenarios. Our approach can thus be used for simulation-based compliance checking if process properties provide access to the data necessary to express compliance concerns (assertions).

A topic that is also related to our work is that of **spreadsheet modeling and testing**. Here, the focus is put on modeling and testing the spreadsheet content itself. In particular, spreadsheet testing and debugging is important because it positively influences spreadsheet accuracy (Kruck, 2006). Burnet et al., who coined the term “end-user software engineering”, proposed an approach to support assertions in spreadsheets (M. Burnett et al., 2003). The assertions are built on top of cells to check the execution of formulas contained in cells. The approach provides the possibility to create assertions by the end-user, following an abstract syntax that is implemented through both graphical and textual concrete syntaxes. Hermans proposes Expector, an Excel-based tool for helping users in improving their testing practices, e.g., by helping them in achieving better testing coverage, more meaningful names for outcomes of the testing, among other features (Hermans, 2013). In the same paper, the author presents an interesting study on the use of testing within spreadsheets. They found out that 8.8% of the spreadsheets from the EUSES corpus (Fisher & Rothermel, 2005) contain testing formulas that use only the spreadsheet's built-in functions. Rothermel et al. present a methodology for the test adequacy criterion in form-based visual programs (the authors place spreadsheets under this category) (Rothermel, Li, DuPuis, & Burnett, 1998). In their methodology, the authors propose to check for the definition-use adequacy of a test suite based on the *all-uses* data flow adequacy criterion. The prototype, implemented in the research language Forms/3 (M. M. Burnett & Ambler, 1994), provides visual feedback to the users about the "testedness" of their spreadsheet. The research works presented above focus more on modeling and testing the spreadsheets itself. Our approach, instead, focuses on testing an external artefact with the help of spreadsheets. Yet, the contributions made in these works can complement our solution because they can help to improve the accuracy of the spreadsheets we used for BP analysis.

2.6. Reducing the gap between business and Information Technology areas

Enterprises and organizations focus their activities on business processes in order to optimize or adapt their business processes to the new organizational needs, which are widely recognized and supported by Business Process Management (BPM). Especially, on business and technology areas are adopting this paradigm of Service Oriented Computing (SOC), which is based specifically on the development of services that are implementing business processes and help in reducing the gap between these two areas, easing the communication and understanding of business needs (Delgado, Ruiz, de Guzmán, & Piattini, 2010).

A business process model (BPM) is used to document the representation of a business requirements that appears during a business meeting been between users and process owners and BPAs (Weske, 2007a). They are typically representing following a graphical notation describing activities interconnected through control-flow elements. Business processes are implemented through Web services by SAs creating a workflow or service composition schema or executable model, which is run by a workflow engine. An algorithm that exploits some service's description characteristics can create the service composition schema automatically.

Reducing the gap between business and Information Technology (IT) areas is helped through the implementation of business processes as services into a specific technology. Achieving in this way promote the independence between the definition and modeling of business processes and allowing changes in the implementation with minimal impact over the processes.

Some of the most relevant works taken into account (1) the integration of Model Driven Development (MDD) and SOC paradigms to business processes and (2) the introduction of a middleware that sits between the models and executable workflows.

In the area comprising the integration of MDD and SOC paradigms to business processes are as follows: (Chen & Buchs, 2006) defines a methodological framework based on Petri Nets for modeling, verification and prototyping of business processes; (De Castro, Marcos, & Lopez Sanz, 2006) focus on the development of service oriented Web Systems defining models, metamodels and transformations to obtain a service composition model which expresses the interaction of services. In (Castro, Mesa, Herrmann, & Marcos, 2008) they integrate a business value model adding the business view and models, and transformation to use case model. In (Delgado, Ruiz, et al., 2010) the idea is very similar but it based the generation of services on business process models using BPMN models as first input. (Touzi, Benaben, Pingaud, & Lorré, 2009) proposes a model driven approach but for collaborative SOA, to transform BPMN models into UML models and BPEL models.

Secondly, some in the area comprising the introduction of a middleware are as follows: (Buchwald et al., 2011), propose to reduce the gap between business process models and service compositions using EPCs as a graphical notation and Petri nets as executable workflow. The authors reconcile both worlds using transformations based on patterns that are identified in the models; however, this approach does not focus on improving the business process models and does not attempt to improve the selection and discovery of services either.

AbuJarour et al. are also focused on both worlds, trying to find fine-grained linkage patterns among the web services used in a BPM. The main objective of this approach is to improve service discovery during the configuration of an executable process (AbuJarour & Awad, 2014) (AbuJarour & Awad, 2011). The approach generates also additional information about web services based on the configuration of the business processes that consume the web services. This information enriches the technical descriptions released by the service providers. Various approaches to discover relationships between services are also discussed by the authors, for instance input/output matching (Dong, Halevy, Madhavan, Nemes, & Zhang, 2004), semantics (Lecue & Leger, 2006) (Lin, Kwong, & Perni, 2006), service-based compositions (Basu, Casati, & Daniel, 2008) and consumer-consumer similarity (Rong, Liu, & Liang, 2009), been the third one the approach chosen by the authors.

Smirnov et al. use behavior profiles in order to detect patterns or association rules over tasks models, and thus, it is possible to suggest to the BPA some improvements and help him to detect errors in the models (Smirnov, Weidlich, Mendling, & Weske, 2009).

Taking into account the lack of suitable instruments for business process analysts (BPAs) and their abilities to design only business processes (with a graphical notation), Saldivar (Galli et al., 2015) uses the spreadsheet paradigm to allow the BPA to verify and analyze the performance of business processes without the need for software development skills. In this context, the BPA can write his own metrics and assertions in order to obtain a process execution log, and he can design analysis reports to be able to autonomously analyze the behavior of his business process. In addition, the BPA can easily discuss his findings with the SA in charge of implementing processes. This work emphasizes the role of the BPA, not only in providing input for the design of processes but also in analyzing them.

Closing the gap between the two worlds of the BPA and the SA is an active research field. Some approaches, such as Buchwald et al. (Buchwald et al., 2011), propose to reduce the gap between business process models and service compositions through the introduction of a middleware that sits between the models (using EPCs as a graphical notation) and the executable workflows (using Petri nets). The authors reconcile both worlds using transformations based on patterns that are identified in the models, however, this approach does not focus on improving the business process models (e.g. through suggestions to the BPA), and does not attempt to improve the selection and discovery of services either.

3. A SEMANTIC APPROACH FOR DYNAMICALLY DETERMINING COMPLEX COMPOSED SERVICE BEHAVIOUR

Web service composition is the process of combining the functionality of diverse services (components) into a new service that provides aggregated value and can be part of another composed service (Dustdar & Schreiner, 2005). Service composition requires defining the order and conditions to selected, bind and invoked services. These tasks can be performed automatically or manually, at design-time (static) or at run-time (dynamic). Dynamic and automatic composition is desirable because it contributes to reduce the development costs of creating new services. It can also assists developers to discover services among a myriad of existing services and to deal with the failure of a component or a whole composed service on real time, facilitating composed services to adapt to contextual changes.

A popular strategy for supporting dynamic and automatic service composition exploits service signature, that is, service's input and output to determine services dependencies, deriving a composition plan that can be seen as a graph (D'Mello et al., 2011). Most research focus on enriching services' signature with additional information (pre and post conditions, quality, conceptual semantic models, business rules, etc.) in order to improve services' dependencies. Standards such as SAWSDL (Kopecky, Vitvar, Bournez, & Farrell, 2007) allow service providers to annotate web service descriptions (WSDL) with references to semantic elements without prescribing a semantic model, which is kept separated from the description. Popular semantic approaches such as OWL-S and WSMO describe service semantics reling on expressive knowledge representation formalisms such as OWL (McGuinness, Van Harmelen, et al., 2004) and WSML (De Bruijn, Lausen, Polleres, & Fensel, 2006b) respectively, along with rule languages. Domain ontologies for both OWL-S and WSMO are rich and complex and the development on either platform demands significant expertise and knowledge from designers and developers on subjects such as the corresponding domain ontology, the platforms, and the tools that enable the execution of semantic Web Services. These characteristics imply an important limitation to the scalability of these approaches (Pedrinaci et al., 2011; Alowisheq, Millard, & Tiropanis, 2009), for this reason, lightweight approaches such as WSMO-Lite (Dietze et al., 2010), and the Minimal Service Model (MSM) (Pedrinaci et al., 2010). Research on semantic-based dynamic composition place a strong emphasis on the discovery of suitable candidates for a composition (Brogi et al., 2008), while the behaviour of the composed service is either highly complex, over-simplified (Kylau et al., 2012) or ignored, for instance, WSMO-Lite does not support a control-flow infrastructure but instead this one is provided by WSMO. Manual techniques on the other hand, allow full control on the specification of the service behaviour, resulting into a variety of complex control flows patterns that satisfy the various needs and constraints of the business processes (N. Russell et al., 2006; N. C. Russell, van der Aalst, & Ter Hofstede, 2009).

Automatic composition is a challenge that tends to become more difficult when the number of services increases, which is worsened if connections between services are complex (i.e. when complex control-flow patterns are included). Some approaches that follow artificial intelligence planning (Hoffmann et al., 2007; Sirin et al., 2004; Klusch et al., 2005; Pistore et al., 2004; Xu et al., 2011) derive the sequence of actions required to reach a goal state (required outputs) from a initial state (inputs and preconditions). These techniques typically work well for small repositories with a high number of constraints. Most of these proposals have some drawbacks: high complexity, high computational cost and inability to maximize the parallel execution of web services. Others (Aversano et al., 2006; Ghafarian & Kahani, 2009; Rodriguez-Mier et al., 2010) deal with a huge number of services but they do not guarantee to obtain an optimal solution, are extremely slow and memory intensive. An approach that is similar to us (Rodríguez-Mier et al., 2012) finds a valid composition considering the matching of the input-output message at a semantic level. The approach scales better than other techniques with huge number of services, and also shows a great performance over large repositories. However, they can discover only two of the most important control-flow patterns: sequence and parallel.

In this chapter, we present our approach for dynamic service composition (CompoSWS) that exploits service signature and semantic annotations along with rules to identify simple and complex control-flow patterns between services at publishing-time (i.e. when a service provider makes its service available in our platform). Services are connected through such patterns forming a graph that is pre-calculated and represent the behavioural semantics of a potential composite service. A composite service can be dynamically and automatically discovered and assembled into an executable service at consuming-time (i.e. when an consumer requests a non existent service but whose functionality can be provided through a services subgraph). We propose also to extend the Minimal Service Model (MSM), which is a lightweight ontology that captures (part of) the semantics of both Web services and Web APIs in a common model focuses on service's signature and facilitates our approach's scalability. We validate our approach theoretically through a complexity analysis and experimentally on a known dataset of 980 services, both at publishing-time and consuming-time, in terms of performance (response time), and scalability (compositions of various sizes). Our results are promising and suggest that our approach could be used in an online fashion. Our experience indicated some limitations of SPARQL 1.1. Specification when querying subgraphs (Arenas, Conca, & Pérez, 2012) that was resolved by defining incremental queries (i.e. progressively reducing the search space).

This chapter describes a technique to derive complex composed service behaviour semantics that:

- Extends the MSM ontology in order to allow the specification of simple and complex control-flow patterns based on the service's signature;
- Enables the automatic discovery of such patterns through a set of rules;
- We also present the algorithms and queries required to dynamically pre-compute all the possible combinations between services taking into account service behaviour (derived from the control-flow patterns); and the algorithm and queries required to discover composite services.

The contributions of this chapter are two; first we improve the performance, in terms of response time, of generating composite services without requiring in memory calculus, which may facilitate scalability of our approach through horizontal scalability. Second we allow the generation of more elaborate compositions that correspond to complex business patterns adopted in most real scenarios, without losing performance when compared to approaches that only consider simple business patterns.

3.1. Composing Web services considering complex control-flow patterns

Web service composition requires determining the service components as well as the order in which services are invoked. Such choices can be made dynamically and automatically at consuming-time (i.e. when a consumer requests a non existing service) by examining the characteristics of a set of known services. As described before, services' signature can be used to determine both service components and the dependencies among them. Typically such dependencies are simple sequence and alternative control-flow patterns (e.g. consume service A first in order to produce and output that serves as an input for the subsequent service B).

Composite services in the real world, however, follow complex control-flow patterns in order to fulfil the requirements and constraints of real world business processes (Ter Hofstede, van der Aalst, Adams, & Russell, 2009). Furthermore, business process modelling comprehends up to 43 well-known control-flow patterns (van Der Aalst et al., 2003a; N. Russell et al., 2006). Semantic Web service composition, on the other hand, considers various properties to determine a composite service, however, the few service model ontologies (OWL-S) that contain elements that make possible to produce complex control-flow patterns are extremely complex and verbose and control-flow related concepts and relationships cannot be derived automatically but have to be included in the model manually, at design-time.

In order to face such problem, in this chapter we extend a well-known and simple semantic Web service ontology (MSM, the minimal service model), with minimal concepts and relationships that make possible to represent relationships among services corresponding to complex control-flow patterns. In this way it is possible to discover composite services as subgraphs where services are interlinked following complex control-flow patterns.

In this section, we present a real-world business process model that includes various control-flow patterns as a motivating example (subsection 3.1.1). Then, we extend the MSM ontology to support complex control-flow patterns (subsection 3.1.2) and then we present our approach to derive 6 control-flow patterns and the corresponding semantic relationships (subsection 3.1.3).

3.1.1. Motivating and example: Finding a service to apply for a travel reimbursement

We introduce a business case scenario that is used along the chapter to illustrate our approach. It is inspired on the University of Minnesota travel reimbursement process ($https://www1.umn.edu/ohr/pay/reimbursements/index.html$). For demonstrating the effect of all the composition patterns we have added complexity to the final step (12) of the process. Figure 3.1 presents a business process model for the business case; it comprehends several steps that we summarize as follows:

(i) An employee must retain detailed itemized receipts for expenses of $25 or more, excluding meals, and s/he must prepare an Employee Expense Worksheet.

(ii) The employee must sign the worksheet.

(iii) The employee must attach the receipts to the worksheet and for each receipt, the system must validate if the costs are within the margins accepted by the university.

(iv) After the worksheet is completed, it is sent to a Preparer and s/he verifies that the expenses meet the University (and/or applicable sponsored fund) policy and procedures.

(v) The Preparer ensures that receipts are included as required and asks the employee for any missing receipts.

(vi) If any rates claimed for applicable charges (hotel, mileage, per diem, etc.) exceed the University limits, the Preparer contacts the employee, and informs him/her of any adjustments made to the total reimbursement.

(vii) The Preparer prints a barcoded Expense Report from the financial system after submitting it for approval.

(viii) The Preparer also attaches the worksheet, receipts, and other support documentation to the printed Expense Report and forwards it to the Approver(s).

(ix) The Approver reviews the Employee Expense Worksheet to verify if inadequate substantiation exists for any expense item.

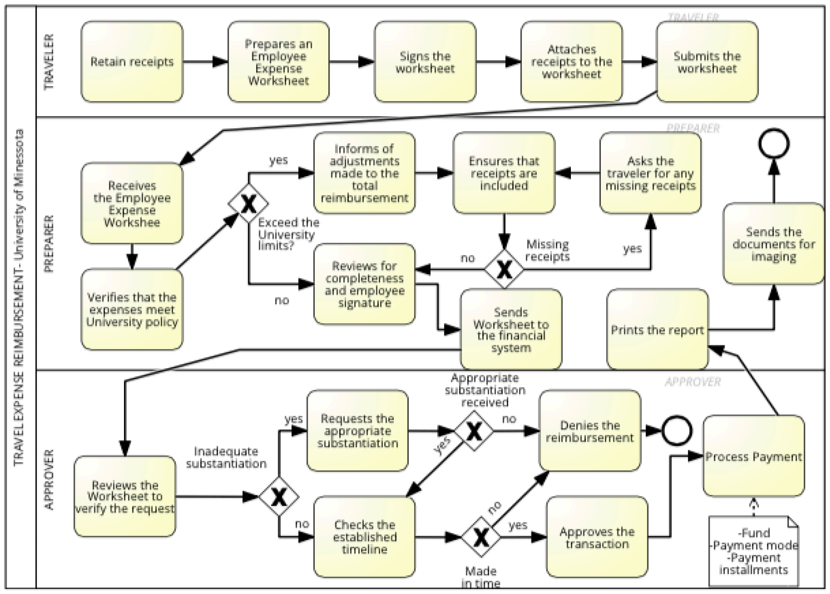

TRAVEL EXPENSE REIMBURSEMENT- University of Minnesota
TRAVELER
Retain receipts
Prepares an Employee Expense Worksheet
Signs the worksheet
Attaches receipts to the worksheet
Submits the worksheet
PREPARER
Receives the Employee Expense Workshee
Verifies that the expenses meet University policy
Exceed the University limits?
yes
no
Informs of adjustments made to the total reimbursement
Reviews for completeness and employee signature
Ensures that receipts are included
Missing receipts
Asks the traveler for any missing receipts
Sends Worksheet to the financial system
Prints the report
Sends the documents for imaging
APPROVER
Reviews the Worksheet to verify the request
Inadequate substantiation
Requests the appropriate substantiation
Appropriate substantiation received
Denies the reimbursement
Checks the established timeline
Made in time
Approves the transaction
Process Payment
-Fund
-Payment mode
-Payment installments

FIGURA 3.1. BPMN model of a travel expense reimbursement process based on the Minnesota University reimbursement process.

(x) If there is an inadequate substantiation, the Approver must request the appropriate substantiation for the items in question. In addition, the Approver will deny any unsubstantiated expense reimbursement if it is not accompanied by an appropriate substantiation.

(xi) The Approver may choose to deny any reimbursement request not submitted within the established timeline.

(xii) Once the Employee Expense Worksheet and attached information is appropriate, s/he approves the transaction and determines the characteristics of the reimbursement such as the sponsored funds from where the money must be transferred, the payment mode (bank, cash or Paypal) and the number of payment installments.

Let's suppose that even though various services provide a partial solution for the problem, a composite service providing the whole functionality is not yet available. In this scenario, users (e.g. a business specialist, a software engineer, an IT analyst, etc.) issue a composition request, which in our approach is an XML file indicating the desired characteristics of a service, as seen in Figure 3.2. The request can be determined through dialogs

```
1  <?xml xmlns:sb=“http://soc.ing.puc.cl/CompoWS/ServiceBehavior”
    xmlns:reimbursement=“http://www.university.org/finance/reimburse.owl”
    targetNamespace="http://example.com/requestReimbursement.xls"
    xmlns:tns="http://example.com/requestReimbursement.xls"
    xmlns:rq=“http://www.soc.ing.puc.cl/CompoWS/request”
    version="1.0" encoding="UTF-8"?>
    <xsd:complexType name="Persona">
                <xsd:sequence>
                    <xsd:element name=“names” type="xsd:string" />
                    <xsd:element name=“surnames” type="xsd:string"/>
                    <xsd:element name="dateBirth" type="xsd:date"/>
                    <xsd:element name=“personalAccount” type="xsd:integer”/>
                </xsd:sequence>
    </xsd:complexType>
2  <rq:request>
3   <sb:Goal> <!-- Describes the goal to be achieved -->
4          <rq:modelref>reimbursement:ProcessedReimbursement </rq:modelref>
5   </sb:Goal>
6   <rq:parameters><!-- Describe inputs and the output to be obtained -->
7          <rq:paramIn> <!-- Inputs can be more than one -->
8              <rq:name>receipts</name>
9              <rq:modelref>reimbursement:Receipts</rq:modelref>
10             <rq:value>[id2014,id2023,id2314,id2456]</rq:value>
11         </rq:paramIn>
12         <rq:paramIn> <!-- Inputs can be more than one -->
13             <rq:name>person</name>
14             <rq:modelref> reimbursement:PersonalData</rq:modelref>
15             <rq:value type="tns:Persona">P</rq:value>
16         </rq:paramIn>
17         <rq:paramOut>
18             <rq:name>result</name>
19             <rq:modelref>reimbursement:ReimbursementResult </rq:modelref>
20         </rq:paramOut>
21  </rq:parameters>
22  <rq:guard> <!-- Used to contain the service behaviour of composite. XPath expressions -->
23   <rq:expression> math:max(reimbursement:FundApprovalLimit) </expression>
24   <rq:expression> contains:(reimbursement:ReimburseByBankAccount") </expression>
25   <rq:expression> for-each(reimbursement:NumPaymentInstallments,3) </expression>
26  </rq:guard>
27 </request>
```

FIGURA 3.2. The user request as specified in an XML document.

as the composition is built, but since our focus is the composition itself, we will let out this feature and will assume that the service request contains all the required information.

Let’s assume that a user issues the service request as described in Figure 3.2. In this request, a goal element (Figure 3.2, line 4) is used to describe the desired activity that an atomic or composite service will provide (e.g. to obtain a ProcessedReimbursement). The parameters element (line 6) describes the input and output information that the user requesting the service is providing. Note that we refer to the concept (semantics) associated to such parameters, instead of considering it a data type or a value since the latter will be provided at runtime. In addition, our algorithm requires at most one output but zero or more input concepts (Figure 3.2, lines 6 to 21). Additional constraints may be provided, for instance the user chooses the sponsored fund that allows maximum approval limits (Figure 3.2, line 23), to reimburse through a bank account (payment mode: bank, cash or paypal) (line 24) and the number of payment installments, in this case the request specifies only three installments (line 25). These expressions follow an XPath notation, they are resolved dynamically and bound to the appropriate control-flow pattern depending on the concepts they refer to (i.e. input, output or goal).

As we can see in Figure 3.1 and the request issued by the client (Figure 3.2), a model (e.g. an ontology) that supports real world service composition must be able to represent complex control-flow patterns such as those arising in the example (alternative service selection, parallel invocation, and various synchronization patterns), as well as certain constraints (conditional selection of responses, and iteration) that affect or result in additional control-flow patterns (e.g. iteration). Existing Web services ontologies that consider control-flow do not consider complex control-flow (such as iteration or conditional selection of responses) and do not provide extensibility elements to model such new patterns easily.

3.1.2. Extending MSM to support complex control-flow patterns

In order to support complex control-flow patterns, we propose a simple extension to the MSM service ontology (Figure 3.1). In MSM, a Service (MSM:SERVICE) has an endpoint represented by a URL (RDF:RESOURCE) that exposes one or more operations (MSM:OPERATION) with Input/Output parameters (MSM:MESSAGECONTENTS and MSMMESSAGEPART); these parameters refer to concepts in an application domain (RDF:RESOURCE). In Figure 3.1, rounded rectangles represent concepts (e.g. MSM:SERVICE), arcs represent relationships between concepts (e.g. MSM:HASINPUT), and the squared rectangles represent literals (e.g. XSB:STRING).

In Figure 3.3, we can see our extension to the MSM service ontology in dotted line and blue colour. We try to be minimalistic in our extension so that it can be applied to other Web service ontologies as well. We use the sb (service behavior) namespace prefix to refer to the elements of our proposed extension. Service goals (SB:GOAL) represent the

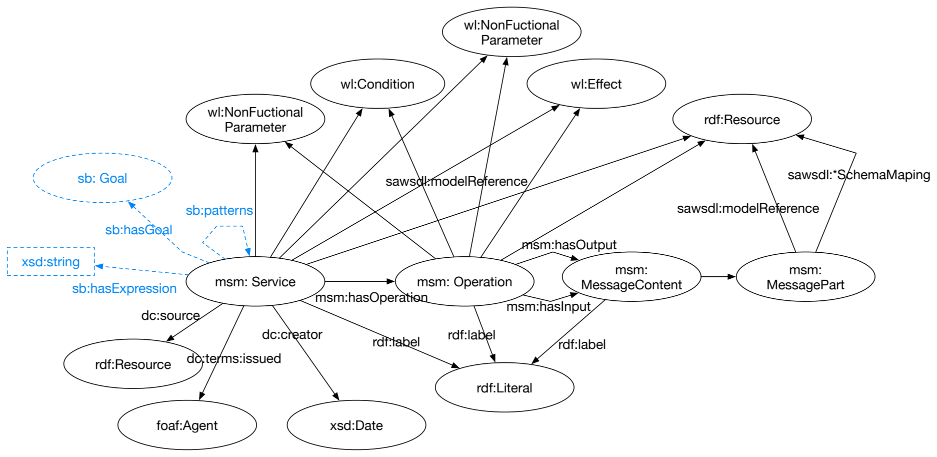


FIGURA 3.3. An MSM ontology extension considering control-flow patterns and guard expressions in order to model service behavior.

activity that is performed when executing a service, at a high level of abstraction (i.e. is not a service effect) described according to a domain specific ontology. The goal is related to the service through a SB:HASGOAL relationship. Service composition may be restricted according to certain constraints or guard expressions (SB:HASEXPRESION), and services are related to other services through relationships that represent the semantics of control-flow patterns (SB:PATTERNS). In this chapter we model six control-flow patterns that are sub-properties of SB:PATTERNS (i.e. they specialize the SB:PATTERNS relationship), each of them represent a relationship between two services: SB:SEQUENCE, SB:ALTERNATIVE, SB:SYNCHRONIZE, SB:DISCRIMINATOR, SB:SELECT and SB:ITERATOR, which are detailed in section 3.1.3. Some constraints or guard expressions that use the SB:HASEXPRESION relationship can be seen in Figure 3.2, lines 23 to 25. These are XPath expressions that are traduced to specific control-flow patterns, that is, they contribute to generate an SB:PATTERNS relationship, and the guard expression itself is stored as XSD:STRING related to the service.

With these three specialized relationships and one concept, we are capable of introducing complex control-flow patterns support in the MSM semantic service model. That is, services can relate to each other specifying the type of dependency between them as well as refer to constraints and the goal they pursue. Furthermore, if we consider these elements in addition to the service signature it is possible to determine such relationships automatically. In the following subsection we extend the example presented in subsection 3.1 by including the ontology extension presented in this section. We use the resulting service implementation to illustrate the application of a set of rules, which are also detailed. The rules exploit our ontology extensions to derive control-flow patterns automatically.

3.1.3. Control flow patterns

In the case of the SAM algorithm (Brogi et al., 2008), the dependencies among atomic services are modelled as an in-memory tree. The SAM algorithm is executed at run-time for each client request. Since the graph of services can grow significantly as companies merge, evolve and change their needs, we pre-compute the possible graph of services dependencies and store the new graph in a specialized database (a NoSQL, graph oriented database). A fragment of the resulting graph will serve as the basis of a new composite service if it is eventually required. Our approach generates new relationships (triples) between services that are stored for later consumption. These relationships are sub-properties of SB:PATTERN represented previous in the ontology (Figure 3.1).

Figure 3.4 illustrates a composition graph both at publishing and consuming time for the previously introduced scenario. The control-flow patterns to derive are based on a set of rules, detailed in the remainder of this section.

The widely known *Workflow Patterns Initiative* (van Der Aalst et al., 2003a) identifies 43 control-flow patterns divided into 8 categories. In our approach, we consider 3 of 5 basic patterns, 2 out of 14 advanced patterns and 1 pattern of the remaining 6 categories in order to illustrate our approach. They are described below.

3.1.3.1. Basic Control Flow Patterns

In this section patterns capturing elementary aspects of process control are discussed. The patterns we consider are sequence, synchronization, and exclusive choice.

*Pattern 1: Sequence (*SB:SEQUENCE*)*

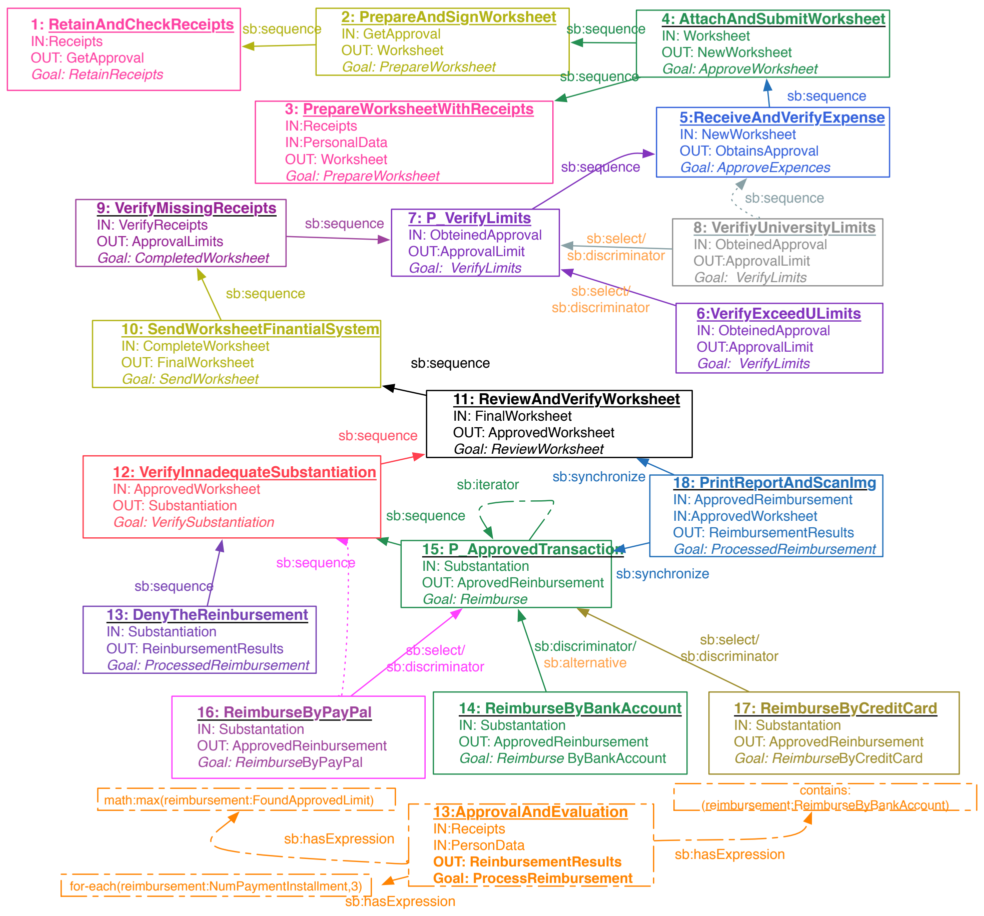


FIGURA 3.4. A composition example for the travel reimbursement scenario: Services are progressively published into our triplestore as indicated by the numbers. The composite service (19) is built from bottom to top (backwards) when a user request is made.

A sequence pattern models dependencies between services so that a service s2 cannot start before service s1 finishes. An SB:SEQUENCE operator is inferred when the goals of both services (s1 and s2) are different, and service (s1) generates an output, which can be used as an input service (s2).

Figure 3.4 shows an example including nineteen services, which are progressively published by the provider. The publication process requires service descriptions to be annotated with SAWSDL expressions that are taken into account to produce SPARQL 1.1 Update sentences. These sentences generate triples that are stored into a triplestore implemented in

Jena. In the example, the process begins right after service 1 (*RetainAndCheckReceipts*) is published. At this point, no relationships are generated since there are no other services. When service 2 (*PrepareandSignWorksheet*) is published a SB:SEQUENCE relationship is generated between both services because they have different goals and *RetainAndCheck-Receipts*' output matches *PrepareandSignWorksheet*'s input.

*Pattern 2: Exclusive Choice (*SB:ALTERNATIVE*)*

Pattern 2 is applied to services with the same goal and output, however; in this case the condition is applied to the goal and optionally to the input parameters. The condition is evaluated to determine which services will be actually invoked and it can be known only when the user issues a request. In our example, an SB:ALTERNATIVE relationship is created between services 14 (*ReimburseByBankAccount*) and 15 (*P_ApprovedTransaction*), since the client requests to pay using a bank account (*Reimburse* goal), which impacts only service 14. The other candidate services, 16 and 17, will be discarded because their goals are different to the request (Figure 3.3 line 24). In summary, depending on the guard expression, some services may be selected, others may be ignored and new relationships may be created. Expressions applied to services with the same goal and output that evaluate the output results, will cause the inclusion of services related through SB:SELECT relationships. However, expressions that evaluate only the goal (and optionally the input) will cause to ignore those services whose evaluation is negative, such services will be related through an sb:alternative relationships. If no guard expressions are applied, then the services will be related through the sb:discriminator relationship.

*Pattern 3: Synchronization (*SB:SYNCHRONIZE*)*

This pattern is applied when the inputs of a service can be obtained from the outputs of other services, and not a single service can provide all the inputs. In the example, when service 18 (*PrintReportAndScanimg*) is published, this pattern is applied; it requires the execution of service 11 (*ReviewAndVerifyWorksheet*) and service 15 (*P_ApprovedTransaction*) in order to start its execution. Hence, an sb:synchronize relationship is created between service 18 and 11, and service 18 and 15. In the latter case, predecessors (e.g. service 15) are preferred to final services (e.g. services 14, 16 and 17).

3.1.3.2. Advanced Branching and Synchronization Patterns

Advanced patterns refer mainly to parallel invocation. These patterns refer to the various ways that the split and join part of a parallel invocation can arise in business processes. We considered the *Structured Synchronizing Merge*, and the *Structured Discriminator patterns*.

*Pattern 4: Structured Discriminator (*SB:DISCRIMINATOR*)*

In this pattern the thread of control is passed to next service when the first incoming service finishes its execution. That is, only the output of the first service providing a response is considered. When a service is published, the algorithm searches for services with the same goal and output. If there exists more than one service that share the same goal and output, a predecessor node is created (or reused if already exists) and the services are related to the predecessor with an sb:discriminator relationships.

In Figure 3.4 when the provider publishes service 8 (*VerifyUniversityLimits*) and service 6 (*VerifyExceedULimits*), pattern 4 is applied, since the goal and outputs of both services are the same. Note that service 7 (*P_VerifyLimits*) is the predecessor service created by the system. The same case applies to services 14 (*ReimburseByBankAccount*), 16 (*ReimburseByPayPal*) y 17 (*ReimburseByCreditCard*), which cause the generation of service predecessor 15 (*P_ApprovedTransaction*). Notice that in this case, the goals of services 14 (*ReimburseByBankAccount*), 16 (*ReimburseByPayPal*), and 17 (*ReimburseByCreditCard*) are specializations (inheritance) of the goal of service 15 (*Reimburse*) as defined in the reimbursement ontology. Predecessors are not executable services (empty services); they are generated automatically using the grouped services' goal (or the super goal in the case of inheritance) as the predecessor name.

*Pattern 5: Structured Synchronizing Merge (*SB:SELECT*)*

Similarly to pattern 4, services with the same goal and output are grouped together under a predecessor using an sb:select relationship. However, unlike pattern 4, a condition applied to the services' output must be evaluated at runtime in order to choose the proper response and such condition can be known only when the user provides a service request. That is, this pattern is not pre-computed at publishing time, but calculated when the consumer issues its request (see Section 5, *Connect2* algorithm).

In Figure 3.3, when the consumer issues a request and specifies a guard expression on the *FundApprovalLimit* output parameter (line 23), pattern 5 is applied to services 6 (*VerifyExceedULimits*) and 8 (*VerifyUniversityLimits*) since the goal of both services and their output parameters are the same and the output parameter is *FundApprovalLimit*. Since a predecessor was generated in the previous example, the system connects services 6 and 8 with the predecessor with a SB:SELECT relationship.

3.1.3.3. Iteration Patterns

The following pattern deals with capturing repetitive behaviour in a workflow. We only considered the *Structured loop pattern*.

*Pattern 6: Structured Loop (Pre-Test) (*SB:ITERATOR*)*

The iteration pattern occurs when the user requests that a simple or composite service is executed more than once. This need can be only determined from the user request, at composing time, based on the for-each guard expression applied on some goal. In Figure 3.4, the client specifies that the payment shall be performed in 3 installments only (Figure 3.3 line 25). Then, an sb:iterator relationship is applied to service 15 (*P_ApprovedTransaction*).

3.2. COMPO-SWS

In order to test our approach we designed and built *Compo-SWS*, a Web service composer that follows a two sides approach. First, it acknowledges the different roles of the service *publisher* and the service *consumer*, and for the former case it takes advantage of

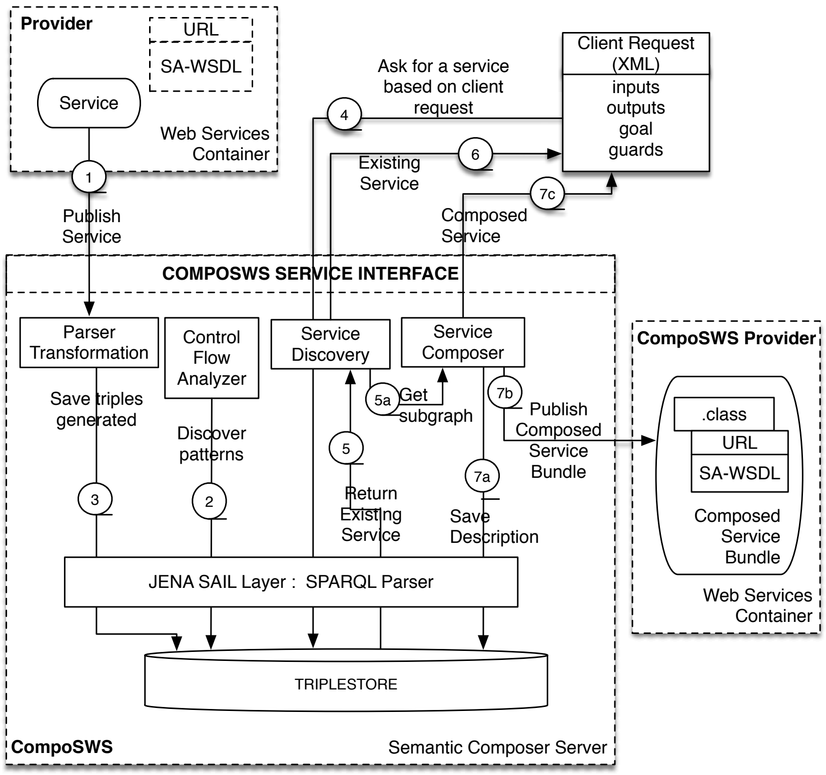


FIGURA 3.5. *Compo-SWS* Architecture.

the service availability by pre-calculating all possible relationships, so that, at consuming time, the chances of finding and identifying a composite service are higher.

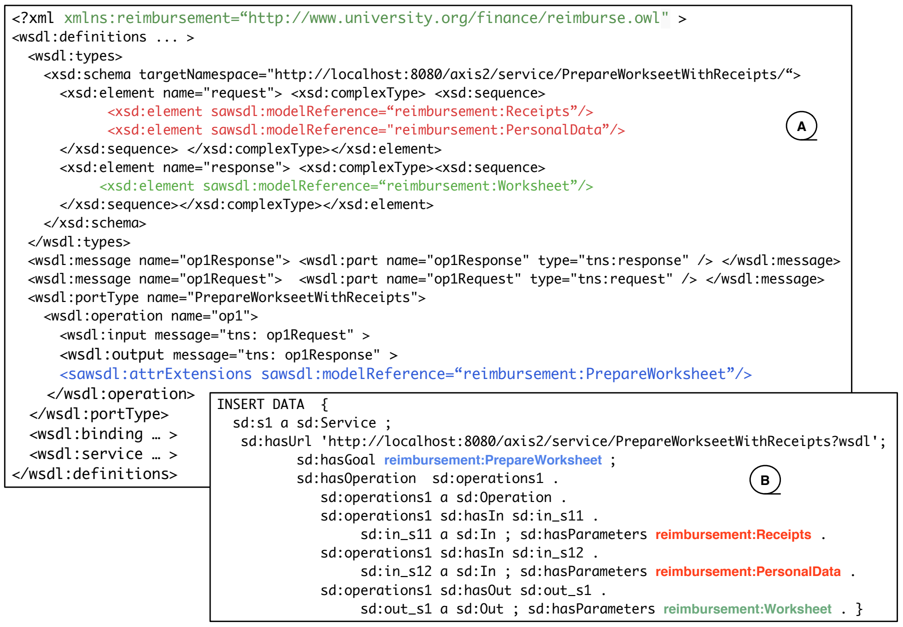
In Figure 3.5, we summarize the major architectural components of *Compo-SWS*. The dotted rectangle (A) represents the Web services container on the side of the service *publisher*. The publishing process (step 1) requires that the provider interact with *Compo-SWS* interface in order to submit the service description. Such description must be annotated with SA-WSDL expressions in order to be transformed, according to our ontology, into triples (step 2).

The SAWSDL descriptions contain annotations related to the service goal and data types. The *service goal* annotation is an attribute of the WSDL's *portType* element; data types (used as input and output) annotations refer to concepts defined in an external application domain through a *modelReference* element. Figure 3.6(A) shows a SAWSDL description for the *PrepareWorksheetWithReceipts* service. The service's goal is to allow an employee to request a process REIMBURSEMENT (*#PrepareWorksheet*), the service's input includes the receipts (*#Receipts*) and the employee personal data (*#PersonalData*); and the service's output is the worksheet (*#Worksheet*) registered by the service. The SAWSDL description is transformed (step 2) into a SPARQL 1.1 Update expression that populates the triplestore. Figure 6(B) presents the SPARQL query generated to populate the triplestore for the SAWSDL description shown in Figure 3.6(A).

The generated triples are stored into a Jena's TDB triplestore. We use Apache Jena, which is a Java framework providing functionality such as RDF and N3 parsers, and a SPARQL engine among other features. It also provides a programming environment for RDF, RDF(S), OWL, and SPARQL and includes an inference engine based on rules and triplestores. Once translated, the service description is analysed by the *Control Flow Analyser* component, which is responsible of executing the *Connect* algorithm, which connects the services together (step 3 in Figure 3.12) using the different relationships corresponding to the control-flow patterns. These relationships become new triples that are stored in the database.

The client request can be provided as an XML document (see Figure 3.2) describing the expected goal and output, and providing some inputs and guards (Figure 3.5, step 4). The *FindService* algorithm is performed by the *Service Discovery* component (step 5), which executes the SPARQL queries on the triplestore. If a service is found, the user is informed (step 6). If no service is found, the *FindService* algorithm is executed (step 5a) by the *Service Composer* component, obtaining a subgraph of services.

The identified subgraph is returned to the user (step 7a) who is asked for approval. If the composite service is approved, a SAWSDL description is created and stored in the triplestore (step 7b). An executable file (Java .class) implementing the component services invocation under the control-flow patterns is created. The bundle, including the executable file and the service description, is deployed on the *CompoSWS Provider Web services* container (step 7c) in order to expose the created service's endpoint. The composite service's



```
<?xml xmlns:reimbursement="http://www.university.org/finance/reimburse.owl" >
<wsdl:definitions ... >
  <wsdl:types>
    <xsd:schema targetNamespace="http://localhost:8080/axis2/service/PrepareWorkseetWithReceipts/">
      <xsd:element name="request"> <xsd:complexType> <xsd:sequence>
            <xsd:element sawsdl:modelReference="reimbursement:Receipts"/>
            <xsd:element sawsdl:modelReference="reimbursement:PersonalData"/>
      </xsd:sequence> </xsd:complexType></xsd:element>
      <xsd:element name="response"> <xsd:complexType><xsd:sequence>
           <xsd:element sawsdl:modelReference="reimbursement:Worksheet"/>
      </xsd:sequence></xsd:complexType></xsd:element>
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="op1Response"> <wsdl:part name="op1Response" type="tns:response" /> </wsdl:message>
  <wsdl:message name="op1Request">  <wsdl:part name="op1Request" type="tns:request" /> </wsdl:message>
  <wsdl:portType name="PrepareWorkseetWithReceipts">
    <wsdl:operation name="op1">
      <wsdl:input message="tns: op1Request" >
      <wsdl:output message="tns: op1Response" >
      <sawsdl:attrExtensions sawsdl:modelReference="reimbursement:PrepareWorksheet"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding … >
  <wsdl:service … >
</wsdl:definitions>
```

```
INSERT DATA  {
  sd:s1 a sd:Service ;
   sd:hasUrl 'http://localhost:8080/axis2/service/PrepareWorkseetWithReceipts?wsdl';
          sd:hasGoal reimbursement:PrepareWorksheet ;
          sd:hasOperation  sd:operations1 .
             sd:operations1 a sd:Operation .
             sd:operations1 sd:hasIn sd:in_s11 .
                  sd:in_s11 a sd:In ; sd:hasParameters reimbursement:Receipts  .
             sd:operations1 sd:hasIn sd:in_s12 .
                  sd:in_s12 a sd:In ; sd:hasParameters reimbursement:PersonalData  .
             sd:operations1 sd:hasOut sd:out_s1 .
                  sd:out_s1 a sd:Out ; sd:hasParameters reimbursement:Worksheet . }
```

FIGURA 3.6. A: SAWSDL description for the *PrepareWorksheetWithReceipts* service; goal, input and output are annotated. B: The N3 query using SPARQL 1.1 Update generated form the SAWSDL.

URL is also supplied to the user (step 7d). Our algorithms have been fully implemented in Java and SPARQL using the Jena's SAWSDL4J API and the OWL API as well as the Pellet reasoner as inference engine for logic-based matchmaking. In the following section, the composition algorithms for both sides, the publisher and the consumer, are presented in detail.

3.3. Composition Algorithms

In this section we present the algorithms that implement the described control-flow patterns, from the publisher and consumer perspective. When a service is published in our platform (Figure 3.5, steps 1 and 2), the system pre-calculates all the possible relationships between the services (Figure 3.5, step 3) through the *Connect* algorithm (See subsection 5.1). The resulting graph includes the service's goals, input and output characteristics, at the semantic level, including the presented control-flow patterns and rules.

When a consumer requests a service (Figure 3.5, step 4), the system looks for an existing service (Figure 3.5, step 5) executing the *Connect2* algorithm. If no service can be found, the system finds a graph fragment that satisfies all or most of the user's requests (Figure 3.5, step 5a). The latter task is accomplished by executing the *FindService* algorithm (See subsection 5.2). If the consumer approves the proposed service, the graph is used as the behaviour (control-flow) of a composite service, which is created, deployed, and publisher later in our system.

3.3.1. Pre-computing the graph: the Connect algorithm

SPARQL is an RDF query language that operates over the data graph model underlying a triplestore. It has some limitations for expressing queries where the length of the path of the consulted graph model is variable, that is, every arc of a graph must be statically and explicitly defined in a SPARQL query. Since we are modelling service dependencies as graphs, our workflows have unpredictable lengths. The *Connect* algorithm addresses this issue by breaking down the graph query in two steps. Figure 3.7 presents the algorithm following Gooneratne (Gooneratne, Tari, & Harland, 2007).

In lines 2 to 12 (Figure 3.7), the algorithm uses a SPARQL query to look for the occurrence of the select, discriminator and alternative patterns. That is, it looks for services with a goal and output that is equal to the published service's goal and output. For instance, when service 3 is published, the algorithm searches for services with a *#PrepareWorksheet* goal and *#Worksheet* output (see Figure 3.8 (Q1)).

The resulting graph is evaluated to determine if all the nodes that share the same goal and output are associated to a predecessor through a SB:SELECT, SB:DISCRIMINATOR, or SB:ALTERNATIVE relationships. Figure 3.8 (Q2) presents a SPARQL query looking for a predecessor for a specific service (*PrepareWorksheetWithReceipts*). If the predecessor exists but the relationships are missing, nodes and predecessor are connected. If there is not an available predecessor, it is created and the relationships are established (Figure 3.8 (Q3)).

```
1   Connect (S) {
2      // FIND DISCRIMINATOR PATTERNS
3      // Search services witch the same goal and output (Q1)
4    C ← FindNode(Goal(S), Output(S))
5    for each S' ∈ C do
6        // find the predecessor of S'
7        P ← Predecessor(S') (Q2)
8        if P is Null then
9              P ← CreatePredecesor(S') (Q3)
10         // connect the service S with P
11       ConnectDiscriminator(S, P);
12   end
13   // FIND SEQUENCE AND SYNCHRONIZE PATTERNS
14   // Search services with different goal and same output of S
15   C ← FindNode(Goal(S), Output(S)) (Q4)
16   for each  S'∈ C do
17     if relationship(S', S) = sequence then
18            connectSequence(S, S');
19     else
20       if relationship(S', S) = synchronization then
21            connectSynchronize(S, S');
22    end
23   I ← FindInput(S)
24   for each I'∈ I do
25     C ← FindNode(Goal(S), I) (Q5)
26     for each S' ∈ C do
27        if relationship(S', S) =  sequence then
28            connectSequence(S, S');
29        else
30          if relationship(S', S) = synchronization then
31             connectSynchronize(S, S');
32    end
33   end
```

FIGURA 3.7. Connect algorithm, step 1 (lines 1 to 12) and Connect algorithm, step2 (lines 13 to 30).

In lines 13 to 32 (Figure 3.7), the sequence and synchronize patterns are discovered using two queries (see Figure 3.8 (Q4 and Q5)). Q4 looks for services with a goal different than the published service's goal and with an input parameter that matches the published service's output parameter (Figure 3.8 (Q4)). Q5 looks for services with a goal different than the published service's goal and with an output parameter that matches at least one of the published service's input parameters (Figure 3.8 (Q5)).

3.3.2. Consuming services

When a consumer requests a service, the system attempts to find an atomic service providing the requested functionality. Otherwise, the system looks for a subgraph of services of variable length that satisfies client's needs. The subgraph is a set of interrelated services containing all or most of the information provided by the user (input), called *origin*

```
SELECT  ?x ?url ?op ?in ?inT ?out
        (reimbursement:PrepareWorksheet AS ?goal)
        (reimbursement:Worksheet AS ?outT)
WHERE{ ?x  rdf:type sd:Service. ?x sd:hasUrl ?url.
      ?op rdf:type sd:Operation .
      ?x sd:hasOperation ?op .                       (Q1)
      OPTIONAL { ?in sd:hasParameters ?inT .
                 ?op sd:hasIn ?in.}
      ?out sd:hasParameters reimbursement:Worksheet.
      ?op sd:hasOut ?out  .
      ?x sd:hasGoal reimbursement:PrepareWorksheet .
      FILTER NOT EXISTS { ?y  rdf:type sd:Service.
                          ?y sd:discriminator ?x.
                          FILTER (?x = ?y )} }
```

```
SELECT  ?x ?url ?goal ?op ?in ?inT ?out ?outT
WHERE { ?x  rdf:type sd:Service. ?x sd:hasUrl ?url.
        ?op rdf:type sd:Operation.
        ?x sd:hasOperation ?op.
        OPTIONAL { ?op sd:hasIn ?in .
              ?in sd:hasParameters ?inT.}    (Q2)
        ?op sd:hasOut ?out .
        ?out sd:hasParameters ?outT .
        ?x sd:hasGoal ?goal .
        ?x2  rdf:type sd:Service .
        ?x sd:discriminator ?x2.
        ?x2 sd:hasUrl
        "http://localhost:8080/ws/
                PrepareWorksheetWithReceipts?wsdl".}
```

```
INSERT DATA  {
 sd:s3P a sd:Service ;
 sd:hasUrl
   'http://localhost:8080/ws/sP5?wsdl';
 sd:hasGoal reimbursement:VerifyLimits;
 sd:hasOperation  sd:operation3 .
 sd:operation3 a sd:Operation .
 sd:operation3 sd:hasOut sd:out .
 sd:out a sd:Out ; sd:hasParameters
      reimbursement:Worksheet.} (Q3)
```

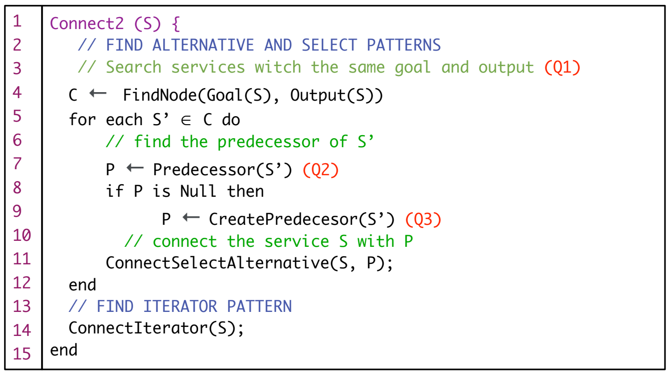
```
SELECT   ?x ?url ?goal ?op ?in
  (reimbursement:ApprovalLimit AS ?inT) ?out ?outT
WHERE{ ?x  rdf:type sd:Service . ?x sd:hasUrl ?url .
   ?op rdf:type sd:Operation .                    (Q4)
   ?x sd:hasOperation ?op .
   ?in sd:hasParameters reimbursement:ApprovalLimit.
   ?op sd:hasIn ?in.
   ?out sd:hasParameters ?outT. ?op sd:hasOut ?out.
   ?x sd:hasGoal ?goal.
       FILTER(?goal != reimbursement:VerifyLimits).
       FILTER NOT EXISTS { ?y rdf:type sd:Service.
                      ?y sd:discriminator ?x.
                      FILTER (?x = ?y )} }
```

```
SELECT ?x ?url ?goal ?op ?in  ?inT
       ?out (reimbursement:ObtainedApproval AS ?outT)
WHERE{ ?x  rdf:type sd:Service . ?x sd:hasUrl ?url .
   ?op rdf:type sd:Operation.                       (Q5)
   ?x sd:hasOperation ?op.
   ?op sd:hasOut ?out.
   ?out sd:hasParametersreimbursement:ObtainedApproval.
   ?op sd:hasIn ?in.?in sd:hasParameters ?inT.
   ?x sd:hasGoal ?goal .
        FILTER(?goal != reimbursement:VerifyLimits) .
        FILTER NOT EXISTS { ?y  rdf:type sd:Service.
                            ?y sd:discriminator ?x.
                            FILTER (?x = ?y )} }
```

FIGURA 3.8. Q1 query finds all the services with the same goal and output. Q2 query looks for a specific service predecessor. Q3 creates a predecessor in no one is available. Q4 and Q5 look for services with a goal other than the published service's goal, in particular (Q4): Finds services with an input parameter that matches the published service output and (Q5): Looks for services with an output parameter that matches one of the published service' input parameter.

nodes; and containing the expected goal and result (output) required by the user, called *target* nodes. Notice that it may be necessary various services in order to cover all the user request's input parameters, and there may be some parameters that no service in the system support. Our approach minimizes the number of services required to cover the user request, and additional parameters shall be required to the user in an interactive fashion if needed, but such feature is out of the scope of this chapter.

For instance, let's consider the example shown in the business scenario previously proposed (See Figure 3.1). The requested goal is to determine the *Reimbursement Conditions* (*ProcessedReimbursement* goal, *ReimbursementResults* output) given certain *receipts an employee personal data* (*Receipts*, and *PersonalData* input parameters respectively). The user also prefers that the fund maximum approval limit is granted (*FundAppovalLimit*,



```
Connect2 (S) {
   // FIND ALTERNATIVE AND SELECT PATTERNS
   // Search services witch the same goal and output (Q1)
  C ← FindNode(Goal(S), Output(S))
  for each S' ∈ C do
      // find the predecessor of S'
      P ← Predecessor(S') (Q2)
      if P is Null then
            P ← CreatePredecesor(S') (Q3)
        // connect the service S with P
      ConnectSelectAlternative(S, P);
  end
  // FIND ITERATOR PATTERN
  ConnectIterator(S);
end
```

FIGURA 3.9. The *Connect2* algorithm generates sb:select, sb:alternative and sb:iterator relationships as defined by the corresponding control-flow patterns.

see line 23 in Figure 3.2), and the payment option is through a bank account (*Reimburse-ByBankAccount*, see line 24 in Figure 3.2), and indicates that 3 (*NumPaymentInstalment*, see line 25 in Figure 3.2) will be the maximum number of payment instalments. Let's consider as well that only services 1 to 19, as described in the example (see Figure 3.4), have been published in our system. That is, services 1 to 19 have been related through the SB:SEQUENCE, SB:ITERATOR, SB:SYNCHRONIZE, SB:ALTERNATIVE, SB:SELECT and SB:DISCRIMINATOR control-flow patterns.

Inputs, outputs and goals are described through concepts in an ontology. Some researches (Chabeb et al., 2010) exploit the ontology structure and the concept syntax in order to determine a more relaxed similarity degree among concepts, which increases the candidate's set. In this chapter we consider only exact similarity among concepts since our focus is the composition that takes place once candidates have been found. We plan to include such hybrid approaches as future work. As discussed in Section 3.1.3, patterns 3 (SB:SELECT), 4 (SB:ALTERNATIVE), and 5 (SB:ITERATOR) can be only applied when the user issues his or her request. That is, the *Connect2* algorithm looks for services with the same goal and output and creates the predecessors if necessary connecting the services with SB:SELECT, SB:ALTERNATIVE or SB:ITERATOR patterns. The *Connect2* algorithm is shown in Figure 9.

```
1   FindService (S)
2     G ← Goal(S)
3     O ← Output(S)
4     I ← Inputs(S)
5     // Search atomic service with similar goal, inputs and output (QF1)
6     S* ← FindNode(G, I, O)
7     if S* not is Null then
8       C <- FindNode(G, O) (QF2)
9      for each S' ∈ C do
10        Q ← CreateAndEnqueue(S)
11       P = Ø
12        while not is empty Q do
13            S ← FirstInQueue(Q)
14            if S not visited then
15               Visited(S)
16              C ← FindRequireServices(S)
17             for each S' ∈ C do
18                 R ← Next(C)
19                 if R not visited then
20                      Enqueue(R)
21                      If R has equal input  I  then
22                         P ← P + R
23          S* ← S*  +  P
24      return S*
25    end
```

FIGURA 3.10. The *FindService* algorithm is responsible for finding a simple service or discovers the subgraph between a target and origin nodes, generating a subgraph that represents the composed service behaviour.

Once the *Connect2* algorithm completes the graph, the *FindService* algorithm seeks for an atomic service that matches the request's input, output and goal (line 5, query QF1 in Figure 3.10). If such service cannot be found, the algorithm searches for the set of nodes that contains the goal and the output defined in the user request that is the set of target nodes, using a SPARQL 1.0 Query (line 8, query QF2 in Figure 3.10).

Figure 3.10, lines 11 to 22 is a backtracking algorithm that, starting from a target node (first element of a queue Q), builds a graph until the set of origin nodes are reached. The algorithm incrementally finds services leaving out those that cannot allow it to arrive to a valid solution (i.e. includes only services containing at least one input that matches the user request input I). The resulting graph includes the services related through the defined control-flow relationships.

Considering the patterns of our study, there are only two ways that services can create compositions that include more than one service, that is, either they form a sequence (at least 2 services) or they are invoked in parallel (at least 2 services). These cases correspond to the SB:SEQUENCE and SB:SYNCHRONIZE patterns. The other patterns represent services that are connected either to themselves (SB:ITERATOR) or to an abstract service (predecessor) but have connections among them (SB:DISCRIMINATOR, SB:SELECT, SB:ALTERNATIVE). Since the nodes in the resulting subgraph are interlinked with control-flow relationships, it is possible to create a composed service that implements the corresponding logic. In our case, we generate a Java Web Service class that implements the new composed service. That is, the composite is a bundle containing a SAWSDL description and the functional modules (i.e. Java classes) implementing the invocation of services according to the workflow represented by the subgraph. The description contains the set of inputs and the output defined by the user request; it is also stored in our triplestore. It will be possible to generate a BPEL description supporting the proposed control-flow patterns, however such alternative will be considered for future work.

```
SELECT  ?url                                                (QF1)
 WHERE{?x sd:hasOperation ?op.  ?x sd:hasUrl ?url.
     ?in0 sd:hasParameters reimbursement:Receipts.
     ?op sd:hasIn ?in0.
     ?in1 sd:hasParameters reimbursement:PersonalData.
     ?op sd:hasIn ?in1. ?out sd:hasParameters
        reimbursement:ReimbursementResult.
     ?op sd:hasOut ?out.
   ?x sd:hasGoal reimbursement:ProcessedReimbursement. }
```

```
SELECT  ?x ?url ?operation ?in ?inT ?out                     (QF2)
WHERE { ?x sd:hasOperation ?op .
  OPTIONAL { ?in sd:hasParameters ?inT .
             ?op sd:hasIn ?in  .}
  ?out sd:hasParameters. ?x sd:hasUrl ?url.
  ?operation sd:hasOut ?out reimbursement:ReimbursementResult.
  ?x sd:hasGoal reimbursement:ProcessedReimbursement. }
```

FIGURA 3.11. Query (QF1) seeks for an atomic service that matches the request's input, output and goal. Query (QF2) searches for the set of nodes that contains the goal and the output defined in the user request that is the set of target nodes.

In our example, the origin service is service 3 (*PrepareWorksheetWithReceipts*) because it contains two input parameters (*Receipts* and *PersonalData*) that match the user request input. The subgraph contains related services that include the output parameter (*ReimbursementResult*) and the goal (*ProcessedReimbursement*) as requested by the user.

In our example, the algorithm finds one possible solution starting from service 3 (*Prepare-WorksheetWithReceipts*) to service 18 (*PrintReportAndScaimg*) passing through services 3, 4, 5, 6, 9, 10, 11, 12, 14 and 18. Hence, a new composite service *ApprovalAndEvaluation* will be created, and the guard expressions (line 23 for services 6 and 8; 24 for service 14; and 25 for service 15 in Figure 3.2) will be triggered and evaluated at run-time, depending on the user preferences (at run-time), additional control-flow relationships could be created for the composite graph.

3.4. Evaluation

In this section, we evaluate our approach theoretically, through an analysis of complexity, and experimentally by measuring performance and scalability. Our analysis considers one operation per service, although it can be extended to include more operations. We also consider a single output parameter and zero or more input parameters.

3.4.1. Provider complexity: publishing a new service

Complexity is calculated considering V, the number of nodes in a graph (services); E, the edges between the nodes (relationships); and k, the number of input parameters for each node. As described before, when a new service is registered in the platform, the possible relationships between services are calculated. The worst-case time complexity analysis of the *Connect()* algorithm, $connect(V, E, k)$, considers three phases, a) finding the nodes matching the new service goals and outputs (line 4 to 13, Figure 3.5); b) finding the services with a goal that differs from the new service goal, but has at least one input that matches the output of the new service (line 17 to 25, Figure 3.7), and c) finding services with a goal different than the new service but with an output that matches the new service's inputs (line 27 to 36, Figure 3.7).

Lets consider M, the number of nodes representing services with the same goal as the new service, and M' the number of services with different goal, let be V the total set of nodes, such that $V = M + M'$.

For the case of a) the worst-case time complexity analysis occurs when $M = V$, that is all the nodes matches the new service goal, hence, the order of this step is calculated as $TConnect(V, E, k) = V$, that is the process of creating a relationship between the new service and the previously existing services. For the case of b), the worst-case scenario occurs when the new service output matches all the previously stored services input, in this case, the order is $TConnect(V, E, k) = V$. For the case of c), the worst-case scenario occurs when given the new service's k inputs, every V node output matches the new service's input, hence the order is $TConnect(V, E, k) = V * k$. That is, for each input of the new service, a relationship is established with all the existing nodes. Therefore in the worst case, the algorithm has order $TConnect(V, E, k) = V * k$ time complexity. Hence, the algorithm is lineal.

3.4.2. Consumer complexity: atomic or composed (on the fly) service

When consuming a service, the algorithm *FindService()* recursively finds a graph of services providing the desired functionality. The worst-case time complexity for *FindService()* is defined as $T\textit{FindService}(V, E, k) = 1$, that is, it performs a query searching for an atomic service that matches users criteria (Figure 3.9, QF1, line 3). If there is no atomic service, the algorithm will perform also a single query searching for the services matching the user's request goal and output. In this case the time complexity is calculated as $T\textit{FindService}(V, E, k)$ the query result will include a list of nodes $N < V$, the algorithm performs a depth-first recursive search. The end of the recursion occurs when a node's or a set of nodes' inputs (k) matches the user requirements inputs. The worst case time complexity of the depth-first search is E (all the edges) and, since this search must be performed for all the results obtained in the previous query the time complexity is $TCreatePath(V, E, k) = E * N$. Hence, the order of complexity for the consumer phase is $E * N$ time complexity, that is, $O(N^2)$ complexity.

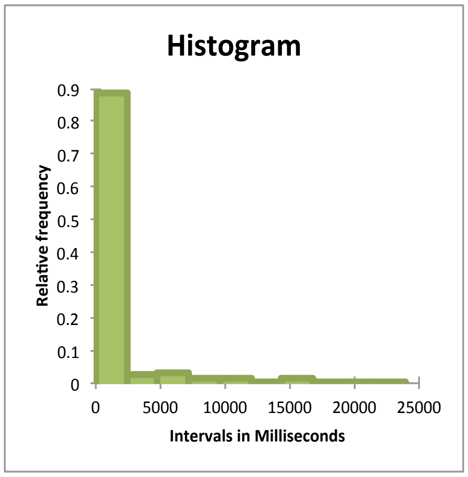
3.4.3. Experimental evaluation

In order to measure the performance and scalability of our approach, we used a SAWSDL test-bed collection semi-automatically derived from a SAWSDL public dataset (SAWSDL-TC3 WSDL11). Descriptions were annotated with a goal concept since the collection considered only input/output concepts. The original collection consisted of 1080 Web services covering different application domains: education, medical care, food, travel, communication, economy and weaponry. We only used 980 services for this test and discarded all services with no outputs. We ran our experiments on an Intel Xeon E5620 with 2,4 Ghz 4Core and 3 GB RAM, running on Linux Ubuntu 11.04. We performed the tests 10 times and we averaged all the results in order to obtain a reliable measure. We evaluated the pre-computing response time when publishing a new service (the connect algorithm) and the response time when requesting a service (service discovery and composition).

3.4.3.1. Performance analysis: Publishing time

We measured the time it takes to add a new service to the graph, varying the number of web services from 1 to 980. In order to avoid additions with no effects (no relationships) we added first the biggest set of unique nodes arranged in a deep relationship (i.e. sequence or synchronize) conforming a composite. In our dataset, the largest possible composed service corresponds to a set of four nodes connected with a sequence relationship (three edges). We added these services first and the remainder nodes were added in random order, one by one. The experiment was run 10 times and the results averaged.

In Figure 3.12 the response time obtained when publishing services is shown as a histogram of 10 intervals; tables a) and b) presents some descriptive analysis. The response time is 0 for a total of 79% of the added services; this result varies from 69% to 94% according to the applied pattern. The average response time is 7 milliseconds, again varying according to the pattern, the maximum response time (average) obtained is 105 milliseconds corresponding to the addition of a service that causes the generation of various select (or discriminator) relationships. The standard deviation is about 17 milliseconds, which is, three times the average.



(a)

	Lower bound	Upper bound	Frequency	Relative frequency
1	0.0	2.4	866	0.885
2	2.4	4.8	24	0.025
3	4.8	7.2	32	0.033
4	7.2	9.6	12	0.012
5	9.6	12.0	15	0.015
6	12.0	14.4	4	0.004
7	14.4	16.8	17	0.017
8	16.8	19.2	5	0.005
9	19.2	21.6	2	0.002
10	21.6	24.0	2	0.002

(b)

	Response time				
Pattern	0's	>0	Average (ms)	St. Dev. (ms)	Max (ms)
Sequence	86%	14%	12	32	128
Select	69%	31%	8	14	110
Discriminator	69%	31%	8	14	110
Synchronize	94%	6%	2	8	73
Total	79.5%	20.5%	7	17	105

FIGURA 3.12. Descriptive analysis of the performance results when publishing services.

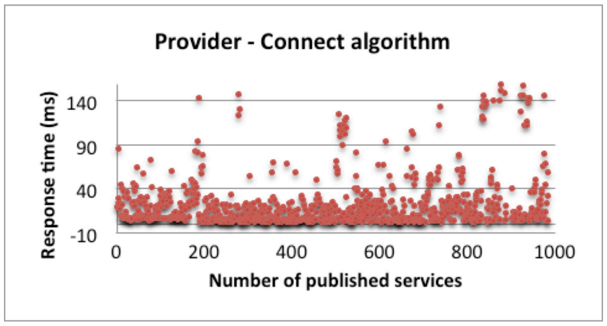


FIGURA 3.13. Accumulated maximum response time obtained from the connect algorithm.

Figure 3.14 shows the time response results (y-axis) when publishing the Web services (x- axis), in seconds. The figure shows the distribution of such values. As can be seen in the figure, the time for pre-computing services composition increases with the number of web services, this is explained since the more available services, the more comparisons must be performed and probably the more relationships must be created. Notice also that the select and discriminator patterns have the same behaviour; this is because both relations are created at the same time. In addition, the alternative relationship is created only when a consumer requests this relation, hence it was not included in our analysis.

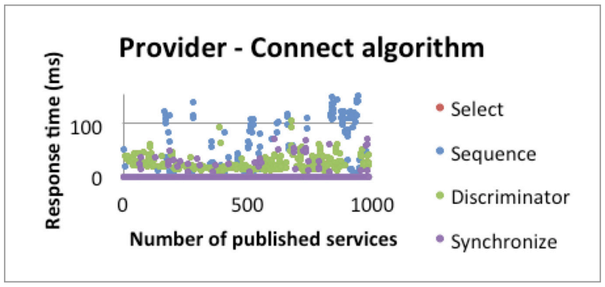


FIGURA 3.14. Response time considering the sequence, select, discriminator, and synchronize patterns.

3.4.3.2. Performance analysis: Consumer time

We measured the response time needed to process Web services requests. Following a similar strategy, we published first a set of four connected services and then added the remainder services randomly. We performed queries asking for services that we knew included 1 (SS), 2 (2CS), 3 (3CS) and 4 (4CS) services, however, we did not stored the composite services into the triplestore (so that, they need to be discovered every time a query is performed). The experiment was ran 10 times and the results averaged. Figure 3.14 shows the mean execution time required for processing the queries; as we can observe the response time increases as the number of Web services in the triplestore increases. This is explained because we perform deep and breadth searches, so that, the more services are published, the more likely they conform complex composites and hence the time spent by the createPathComposedService algorithm increases.

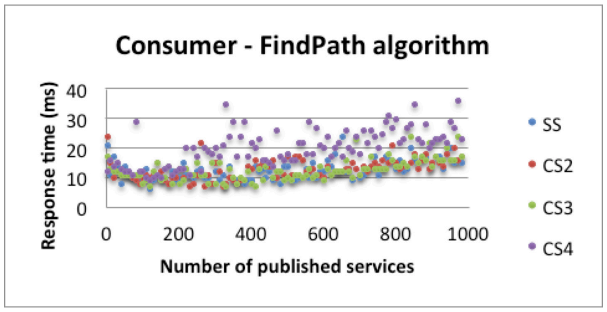


FIGURA 3.15. Response time when searching for atomic and composite services including 1 (SS), 2 (2CS), 3 (3CS) and 4 (4CS) services.

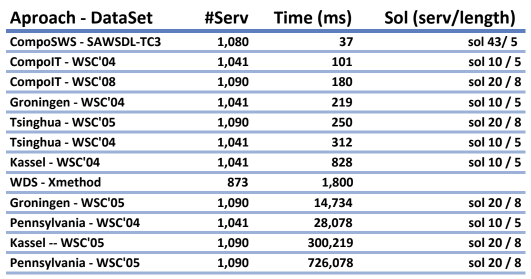
3.4.3.3. Performance evaluation and metrics

We compared our approach using eight public repositories from Web Service Challenge 2008 (Bansal et al., 2008), CompoIT (Rodríguez-Mier et al., 2012; Rodriguez Mier, Pedrinaci, Lama, & Mucientes, 2015b) and WSD (Nayak & Bose, 2015), since the datasets present the same size and perform a similar task. However, notice that service composition implemented in such approaches correspond to simple control-flow patterns, namely sequence and alternative, requiring deep search instead of both deep and breadth search, which is our case. In addition, the WSC challenges as well as WSD perform only the search task, leaving out the composition step (and time); CompoIT and WSD considers search by similarity whereas we are limited to exact matches which causes that they can obtain a high number of composite services while we are limited to exact matches. Figure 3.16(a) summarizes the results of these approaches in terms of the number of relevant services (Serv), that represents, the number of discovered services used in the generated service compositions, dynamic service composition or discovery time (Time(ms)) in milliseconds; that is, the time required to process a user service request and perform the discovery of services and composition if possible. Figure 3.16(b) presents a comparison of the top-8 approaches. The number of I/O parameters however is around 5700 (taking into account semantic concepts) for WSC while we keep 7 I/O parameters.

The quality of each composition includes also the complexity of the composite services. The depth of a composite service in the WSC dataset falls between 5 to 8, comprehending also 10 to 20 services, whereas the deepest composite service in our dataset includes 5 composition layers and 43 services. However, our composite services include the sequences/synchronize pattern (depth) as well as the select/discriminator pattern (breadth). For the latter case, the broadest composite service includes 21 services.

3.5. Conclusions

In this chapter we propose a technique for automatically deriving simple and complex composite service behaviour from the component service's characteristics, dynamically. In



Aproach - DataSet	#Serv	Time (ms)	Sol (serv/length)
CompoSWS - SAWSDL-TC3	1,080	37	sol 43/ 5
CompoIT - WSC'04	1,041	101	sol 10 / 5
CompoIT - WSC'08	1,090	180	sol 20 / 8
Groningen - WSC'04	1,041	219	sol 10 / 5
Tsinghua - WSC'05	1,090	250	sol 20 / 8
Tsinghua - WSC'04	1,041	312	sol 10 / 5
Kassel - WSC'04	1,041	828	sol 10 / 5
WDS - Xmethod	873	1,800	
Groningen - WSC'05	1,090	14,734	sol 20 / 8
Pennsylvania - WSC'04	1,041	28,078	sol 10 / 5
Kassel -- WSC'05	1,090	300,219	sol 20 / 8
Pennsylvania - WSC'05	1,090	726,078	sol 20 / 8

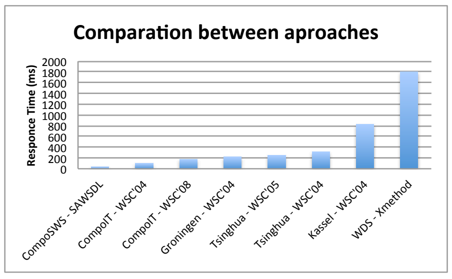


FIGURA 3.16. A comparison, based on number of services composed or discovered versus response time and the quality of the solution, among our approach (CompoSWS) and the WSC challenge, CompoIT and WDS.

order to deal with the resulting complexity, we also propose a strategy for pre-computing possible relationships resulting in a control-flow graph. Later, queries can identify graph fragments as potential candidates for a complete or partial composite service, automatically and dynamically. Our results provide good evidence of the potential of our approach. Despite the increasing response time at publishing-time, 75% of such responses took almost 0 seconds. Regarding the consumer-time, our observations testify that as the composites are stored, the service response time also decreases. An important limitation of our approach is the need for providers, and consumers to know beforehand the ontologies describing the concepts associated with inputs, outputs and goals as well as properly writing the request and annotating the services. Possible solutions for such challenges include the emergence of popular ontologies in various niches such as FOAF describing social relationships, *Good Relations* describing *e-Commerce*, among others.

In this chapter we used semantic Web technologies but we placed an emphasis on the graph nature of the data model rather than the semantic aspects. SPARQL 1.1 presents some limitations to perform queries such as those needed in our work but other NoSQL databases and languages such as Neo4J, Cipher, and Gremlin may serve to provide an alternative implementation with better performance. In addition, we exploited only concept specialization in order to implement goal queries as described in the ontologies, however, other techniques that range from logical (plugin, subsumed-match, subsumed-by-match) to statistical (similarity by nearest neighbour, pearson, jacquard, etc.) or a hybrid, will be applied as future work. In such cases, we expect an explosive growth in the number of relationships between services and possible a degrading performance and scalability. Finally, we just explored 6 control-flow patterns out of the numerous existing and ones in order to prove the feasibility of our approach, this work should be extended to determine the feasibility of automatically deriving the remaining patterns.

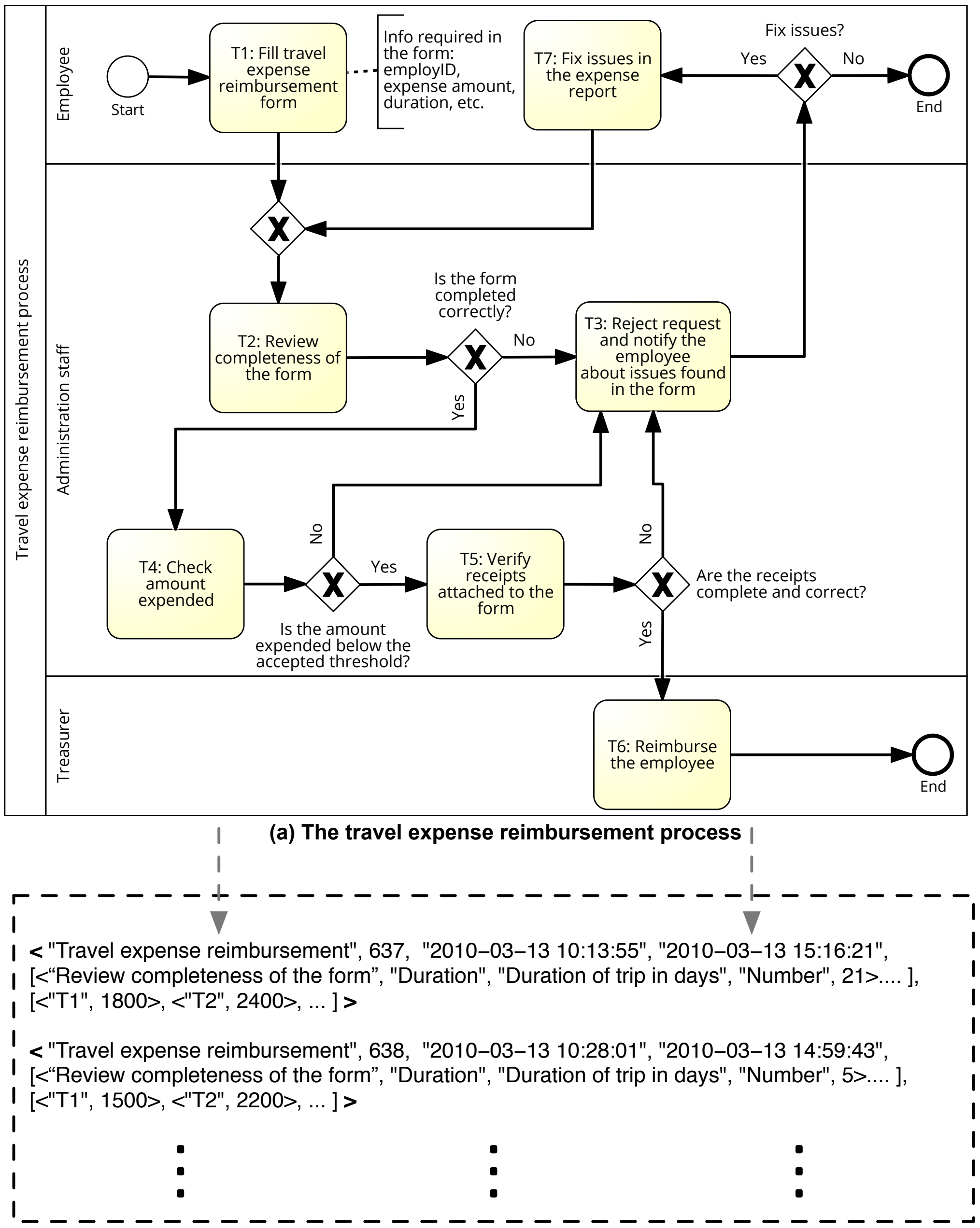
4. ANALYSIS AND IMPROVEMENT OF BUSINESS PROCESS MODELS USING SPREADSHEETS

The analysis of a piece of software, e.g., an algorithm or a mobile app, is a highly technical and daunting task typically performed by *developers* or *testers* who have the necessary technical background to know what to analyze and how. What is important is that the piece of software is analyzed by someone with the right skills, tools and methodologies.

Interestingly, when it comes to **business processes** (BPs) this is not common practice. In fact, the *BP analysts*, who design the processes to be executed, often do not have the necessary instruments to analyze their artifacts, i.e., the *business process models*.

In the context of Business Process Management Systems (BPMSs), the tasks in the process models are typically implemented using web services (Weske, 2007a). The web services can be either fully automated or it can provide a web application that allows human operators to perform the tasks through suitable user interfaces. For this type of business processes, which implementation requires involving developers, the analysis is, therefore, done again by the developers, if at all. This in turn means that the concerns of the actual owners of the artifacts, the BP analysts, may not be properly taken into account before implementing and running the production processes. Identifying issues at this late stage of the process lifecycle can be time-consuming and costly.

Let us consider, for example, the *travel expense reimbursement process* in Figure 4.1(a). Furthermore, let us assume that the process is currently in use in a service-based BPM system and that some problems has been identified by the BP analyst of the company. More concretely, he has noticed that with the current resources assigned to operate this process, only 70% of all the reimbursement requests are processed on time. The BP analyst would like to change the process in order to improve its performance without the need of having to increase the amount of resources assigned to the process. In addition to this, he has also noticed that the amount of many reimbursement requests are far below the operational costs of having to run the BP to process the request and that, in such cases, it


Travel expense reimbursement process
Employee
Administration staff
Treasurer
Start
T1: Fill travel expense reimbursement form
Info required in the form: employID, expense amount, duration, etc.
T7: Fix issues in the expense report
Fix issues?
Yes
No
End
T2: Review completeness of the form
Is the form completed correctly?
No
Yes
T3: Reject request and notify the employee about issues found in the form
T4: Check amount expended
No
Yes
Is the amount expended below the accepted threshold?
T5: Verify receipts attached to the form
No
Are the receipts complete and correct?
Yes
T6: Reimburse the employee
End

(a) The travel expense reimbursement process

```
< "Travel expense reimbursement", 637,  "2010–03–13 10:13:55", "2010–03–13 15:16:21",
[<“Review completeness of the form”, "Duration", "Duration of trip in days", "Number", 21>.... ],
[<"T1", 1800>, <"T2", 2400>, ... ] >

< "Travel expense reimbursement", 638,  "2010–03–13 10:28:01", "2010–03–13 14:59:43",
[<“Review completeness of the form”, "Duration", "Duration of trip in days", "Number", 5>.... ],
[<"T1", 1500>, <"T2", 2200>, ... ] >

        ⋮                    ⋮                    ⋮
```

(b) Process execution evidences stored in a log

FIGURA 4.1. BPMN model of a travel expense reimbursement process and a possible execution log.

may just be better to immediately reimburse the employee without having to run the whole process and incur in costs that are not justified by the requested amount.

Before investing the necessary effort for implementing and deploying changes in the process, the BP analyst needs to find answers to key questions, such as *how many reimbursement requests, per quarter of the year, fall within the 30% of requests that are not processed on time, what should the value be for the amount requested under which the request is immediately reimbursed, and whether all these requests can be reimbursed without exceeding the maximum amount of 15K euros imposed by the accounting department*. These are business questions that require the possibility to try different process execution scenarios that reproduce different execution outcomes. The BP analyst needs to be able to specify the typical behavior per quarter of the year, check whether the new *fast track* reimbursement will comply with the constraints above, analyze and visualize the results to propose a fine tuning of the process, and communicate with the software developer working on the implementation of the process.

These tasks can be done manually if the BP is simple and the number of issues to be analyzed are small. Otherwise, analyzing a BP can turn into a daunting task that requires automation, programming and IT skills. BP analysts usually do not have these skills, and, in practice, BPs are therefore mostly analyzed by software developers that, by nature, focus more on implementation than on business aspects.

If the BP analyst nonetheless wants to analyze a given process, he needs to communicate his analysis goals, requirements and configurations to a software developer, who is able to implement and run the analysis on behalf of the analyst. Once analysis results are ready, the developer needs to communicate them back to the BP analyst, who in turn may ask for a re-run of the analysis under new settings, and so on. Understanding well a process may thus require several iterations between the analyst and the developer. This is not optimal and suffers from the same difficulties already extensively reported in the literature on software engineering in general and requirements analysis in particular, such as ineffective communication channels, inexpressive notations, and its reliant nature (Bhat et al., 2006; Sutcliffe, 2012; Wiegers & Beatty, 2013). We propose therefore an approach that enables the BP analyst to analyze BP models on his own, with less reliance on and intervention of software developers.

We rely on *spreadsheets* to accomplish this. Created in 1979, spreadsheets are nowadays a common business tool and the most widely used end-user programming environment (M. Burnett, Cook, & Rothermel, 2004). Scaffidi and colleagues had estimated that only in the United States, by 2012, more than 55 million people would have been using spreadsheets at the workplace, mainly for business purposes (Scaffidi, Shaw, & Myers, 2005). Considering the ubiquity of electronic spreadsheets in today's business landscape, these tools represent an ideal environment to build powerful user-centric solutions, such as BP analysis instruments that target BP analysts.

This chapter describes a spreadsheet-based approach for business process model analysis that:

- maps the problem of business process performance analysis and verification to the problem of configuring and analyzing data in common *spreadsheets*;
- enables the *generation* of analysis spreadsheets from an extended business process model editor for BPMN process models (Object Management Group (OMG), 2011);
- enables the BP analyst to define own *metrics, assertions* and *analysis reports*;
- automates the *simulation* of BP executions and generates *process execution logs*.

Two independent *user studies* demonstrate the viability of the approach, which was implemented in a prototype *tool for spreadsheet-based BP model analysis*, and a detailed *qualitative analysis* of the state of the art in BP model analysis highlights the benefits of the tool.

Before outlining the details of the approach (Section 4.2), next we formalize the context and problem statement of the work. In Sections 4.3–4.5, we then explain the design, execution and analysis of BP models. In Section 4.6, we report on how we implemented our prototype tool, which we assess in Section 4.7. We conclude the article with a discussion of related works and our final considerations on the results achieved.

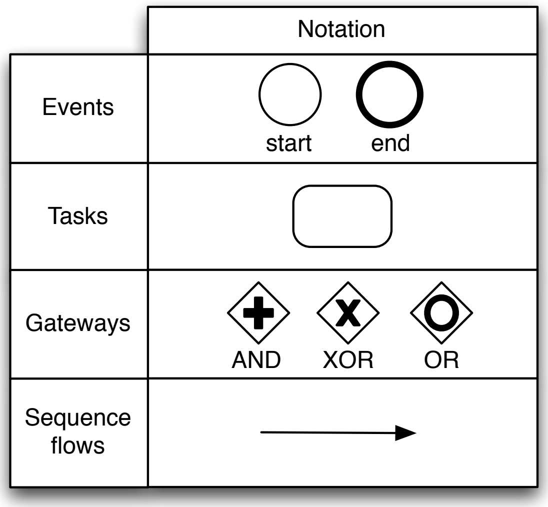


FIGURA 4.2. BPMN elements related to control-flow specification.

4.1. Preliminaries and background

4.1.1. Business processes

As illustrated in Figure 4.1(a), in practice, BPs are typically expressed through process models. In this chapter, we represent processes using BPMN (Object Management Group (OMG), 2011), a BP modeling notation widely used both in industry and academy. The core elements of BPMN related to control-flow specification include events, tasks, gateways and sequence flows (R. M. Dijkman, Dumas, & Ouyang, 2008) (see Figure 4.2). *Events* can be used to signal the start (*start event*) and end (*end event*) of a process. *Tasks* represent atomic activities to be performed as part of the process. *Gateways* are routing constructs that determine the execution flow of the process. They can be one of *AND gateway* (for creating concurrent execution flows), *XOR gateway* (to select one of a number of mutually exclusive flows), or *OR gateway* (to select any number of flows from the set of all outgoing flows). *Sequence flows* are used to represent the ordering relationship between any two elements (events, tasks and gateways) presented before. BPMN includes a richer set of elements, but in this chapter we focus only on the ones presented here.

For notational convenience, we define a **business process model** as a tuple $BP = \langle pid, start, end, N, E, P \rangle$, where pid is a unique identifier, $start$ and end are the events that represent the start and end of a process, $N = T \cup G$ is the set of nodes of the process, with T being the set of tasks and G being the set of gateways of the process, $E \subseteq N \times N$ is the set of edges that connect pairs of nodes, and P is the set of properties that store business data produced and consumed during the execution of BP. A a task $t \in T$, $t = \langle tid, tname \rangle$ is an activity of the process, where tid is a unique identifier and $tname$ is the name of the task. A gateway $g \in G$, $g = \langle gid, C, gtype \rangle$ is a control flow node, where gid is a unique identifier, C is the set of conditions that controls the gateway, and $gtype \in \{AND, XOR, OR\}$ is the type of the gateway that derives from C. Each condition $c \in C$ is a tuple $c = \langle cid, e, expr \rangle$, with cid being a unique identifier, $e \in E$ being an outgoing edge of the gateway, and $expr$ being a Boolean expression over process properties in P specifying the condition to follow the outgoing edge e. Process properties are of type $p = \langle nid, pname, pdesc, datatype, pvalue \rangle$, with nid being the identifier of the node producing a value for p, $pname$ being the name, $pdesc$ being the description, $datatype$ being the data type, and $pvalue$ being the value of the property ($pvalue$ may be empty at the design time).

The BP model we introduce here is mapped to BPMN as follows. *start* and *end* in our model corresponds to the *start event* and *end event* in BPMN. Tasks T and gateways G map to the *gateways* and *tasks* in BPMN, respectively. Furthermore, the element $gtype \in \{AND, XOR, OR\}$ in a gateway $g \in G$ determines whether g refers to an AND, XOR or OR *gateway* in BPMN. Edges E in our model corresponds to the sequence flows in BPMN. Finally, the conditions C and properties P in our model are represented in BPMN through attributes associated to gateways and the process itself, respectively. It is worth noticing that the formalization and mapping introduced here are simple and straightforward, and that they are tailored to the notation needs of this chapter. The interested reader can find a more detailed treatment of the formal semantics and analysis of BPMN process models (e.g., using Petri Nets) in (R. M. Dijkman et al., 2008; R. M. Dijkman, Dumas, & Ouyang, 2007).

The execution of a BP is a business process instance (or process instance for short). A **business process instance** bpi is a concrete case of operation of BP and can be represented as a tuple $\langle pid, piid, startTs, endTs, PI, TD\rangle$, where pid identifies the process model BP, $piid$ identifies the process instance, $startTs$ and $endTs$ are the start and end timestamps of the execution, PI is the set of process properties produced by the execution of BP, and TD is a set of task durations $td = \langle tid, d\rangle$, with tid being the task identifier and d being the execution time of the task.

Process instances are typically tracked in the form of a process execution log (van der Aalst, 2011) for later inspection and analysis. We define a **process execution log** as a set of process instances $L = \{bpi_j\}$. For example, Figure 4.1(b) shows a possible log of the travel expense reimbursement process with two process instances. This representation differs from the more common, event-based representation of process logs, in that it proposes an already aggregated view on execution events. As we will see later, this choice helps us to simplify the presentation of process execution data to the BP analyst.

4.1.2. Business process analysis

The term business process analysis has a broad meaning and includes many different types of analyses such as simulation, diagnosis, verification and performance analysis (Van Der Aalst, Ter Hofstede, & Weske, 2003). In this chapter, we focus our analysis on the combination of the last two. More specifically, we take the *dynamic* perspective of verification and performance analysis, i.e., we run our analysis based on the execution of the business process models.

From this dynamic perspective, verifying a business process model means analyzing whether or not the behavior of its instances matches a given expected behavior. For example, in the scenario described in the introduction, the BP analyst may want to verify whether, under different execution conditions, the sum of the amounts for the reimbursement processed using the *fast track* option is kept under 15K euros in each quarter of the year. In order to perform this verification, the BP analyst needs to be able to (i) specify the *expected behavior* of the business process, (ii) provide the inputs for the process, run it, and track its *observed behavior*, and (iii) *analyze* the expected and observed behavior in order to verify if they match. We call the joint realization of these tasks **business process verification and performance analysis**. For conciseness, in the rest of the chapter we refer to this simply as **BP analysis** and explicitly refer to verification and performance analysis when needed.

The *expected behavior* of the process is partly specified by the process model BP. However, the process model provides only static, structural information about the process; if instead the object of the verification are the dynamics and data produced by the execution of the process as we discuss here, additional constructs are necessary. This is where the **performance analysis** part comes into play. More concretely, we use **metrics**, i.e., measures that capture the performance of a process starting from process execution evidences. Formally, we can define a metric as a function $m(L) = val$, where L is the process execution log and $val \in \mathbb{R}$ is the metric's value. Although this definition allows for the computation of cross-process metrics, i.e., of metrics computed over execution evidence from different process models, for simplicity in this article we limit our attention to single-process metrics only.

The availability of metrics further allows one to express expected behavior in terms of conditions over metric values. Such conditions can be expressed as **predicates**, which are Boolean statements over a metric m. Formally, we can represent a predicate as a function $pred(L, m) = bool$, where L is a process execution log, m is a metric and $bool \in \{true, false\}$ holds the evaluation of the predicate. Predicates can be combined using standard logical operators, such as AND, OR and NOT, to build assertions. An **assertion** can be defined as a function $a(L, Pred) = bool$, where L is a process execution log, $Pred$ is a set of predicates and $bool$ is as defined before. Assertions allow one to write arbitrarily complex combination of predicates to specify and check the expected behavior of a process.

In order to assess the behavior of a process, we need to run the process and record its *observed behavior*. For already implemented processes or services, this behavior can be extracted from the log L of the process. For processes or services that have not yet been implemented, we need to find the way of generating L by exercising the process model BP. We will get back to this issue in Section 4.1.3

The *analysis* of BP now requires evaluating the assertions and metrics over the collected execution evidence L and visualizing the respective outcomes. Doing so requires setting up a suitable analysis report. We define a **analysis report** as a function $r(L, M, A) = V$, where L is a process execution log, M is a set of metrics, A is a set of assertions and V is a set of tables and charts that summarize the analysis outcomes. For example, $v \in V$ can be a pie chart that shows the percentages of $true$ and $false$ evaluations of an assertion $a \in A$ over the set of process instances in L. Such reports serve not only as a means to convey the outcomes to the analyst but also as a communication tool between the analyst and the software developer implementing the process.

4.1.3. Business process simulation

One way to obtain the event log L when it is not possible or convenient to run the real process to generate it is to use *business process simulation* (van der Aalst, Nakatumba, Rozinat, & Russell, 2008), which mimics the execution of process instances, given a business process model BP and a suitable configuration. We propose to use this approach to obtain the process execution log L.

We define a **BP simulator** as a function $BPsim(BP, SS) = L$, where BP is a process model, SS represents the settings used to simulate the BP, and L is the process execution log generated by the simulation. BP simulation thus enables the BP analyst to mimic different process execution scenarios and obtain corresponding execution evidences.

4.1.4. Problem statement

The problem we aim to solve in this article is devising an approach that enables the BP analyst to verify and analyze the performance of business processes without the need for software development skills. The first goal is to enable the BP analyst to write own metrics M and assertions A, to obtain a process execution log L, and to design own analysis reports r, so as to be able to autonomously analyze the behavior of a business process BP. The second goal is to do so in a fashion that allows the BP analyst to easily discuss his findings with the software developer in charge of implementing processes. Our hypothesis is that mapping the BP analysis problem as defined in this article to the design of a spreadsheet calculation allows us to achieve both goals at the same time, in particular, given that spreadsheets are omnipresent in business and well-known by average BP analysts (Deloitte, 2009).

4.2. Spreadsheet-based business process analysis

We specifically consider the case of service-based BPs, where activities are executed by web services; human actors are hidden behind web service interfaces. Obtaining a log file for this type of BPs requires either invoking real web services (if such are available and do not have any persistent side effects) or simulating web service invocations (if the web services do have side effects or are not available at all). We assume that the BP analyst is capable of designing coarse BP models using the Business Process Modeling Notation (BPMN) and that he is familiar with spreadsheet tools like Microsoft Excel or Google Spreadsheets. We also assume that there is a software developer implementing the process and its web services, starting from the coarse BP models.

4.2.1. Requirements

Given these assumptions and the above problem statement, we identify a set of *functional requirements*. We group them into categories that correspond to the BP analysys phases (Section 4.1) they are related to:

Specification of expected behavior:

- **R1:** The solution shall enable the design of the BP model BP, along with its process properties P.
- **R2:** The solution shall enable the writing of metrics M and assertions A over the process execution log L.

- **R3:** The solution shall enable the storage of metrics and assertions for later reuse.

Obtainment of observed behavior:

- **R4:** The solution shall enable the configuration and simulation of BP to obtain a corresponding log L.
- **R5:** The solution shall enable the use of existing implementations of web services used by BP that do not produce unwanted side effects.
- **R6:** The solution shall enable the configuration and simulation of web services used by BP that do have unwanted side effects or that do not exist yet.
- **R7:** The solution shall enable the storage of the generated log L.

Analysis and reporting:

- **R8:** The solution shall enable the creation of analysis reports r based on BP, M, A and L.
- **R9:** The solution shall enable the storage of reports for future reuse, e.g., for re-running the analysis under different conditions.
- **R10:** The solution shall enable the sharing of BP analysis configurations and results with other stakeholders (e.g., with software developers).

The implicit, *non-functional* requirement is that the solution's tools that target the BP analyst shall not need any software development skills.

4.2.2. Approach

The overall approach proposed in this chapter takes into account the fact that there are tasks that require specific technical skills that BP analysts may not have and that may prevent them from being able to analyze BPs. For example, the detailed design of *executable* process models in BPMN and the configuration of the more technical aspects is usually out of the reach of typical BP analysts. The design of executable BP models may in fact require the developer to use a larger set of modeling constructs than introduced in Section

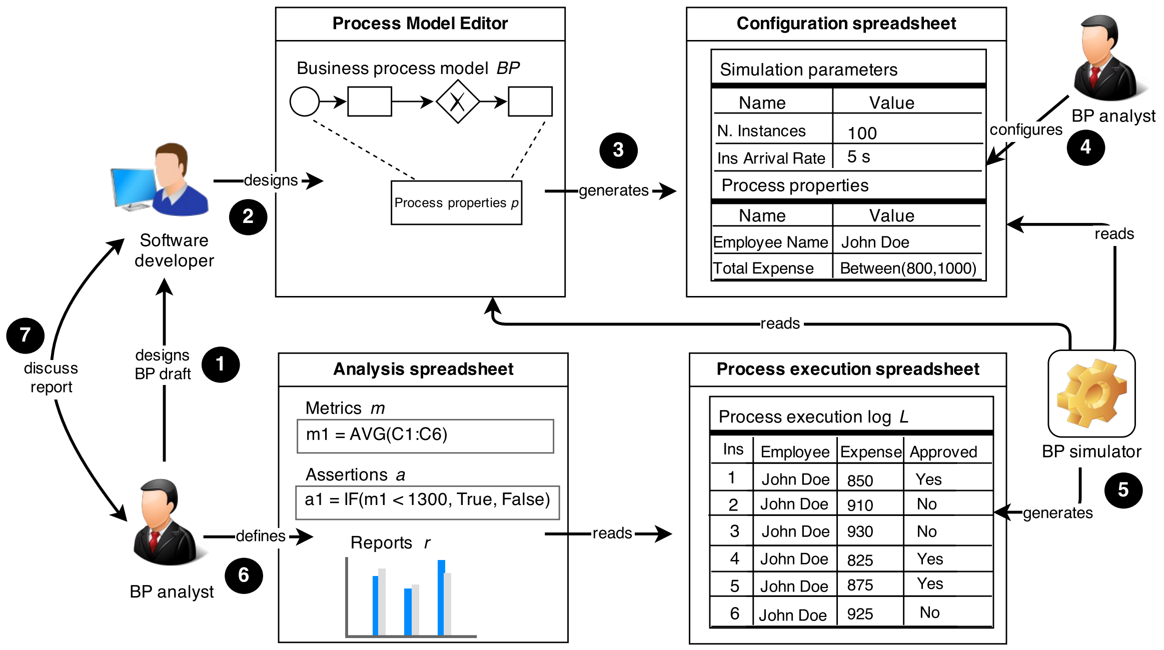


FIGURA 4.3. Spreadsheet-based approach to BP analysis.

4.1.1 (e.g., events or messages); the analyst only needs to master the subset of constructs introduced in Section 4.1.1 to be able to run his analysis. We therefore propose to separate the tasks related to (i) the design and configuration of executable process models, and (ii) the analysis of BPs. Task (i) is assigned to software developers, while task (ii) to BP analysts (see Figure 4.3). This separation of duties is not only practical and realistic, but it also leverage on the skills and interests of each role.

In order to approach the requirements discussed in Section 4.2.1, we leverage on BPMN and the *spreadsheet paradigm* to provide an approach for the analysis of BPs. BPMN is used to model executable BPs. The spreadsheet is used as interface toward the BP analyst and as communication instrument between the analyst and the developer. Simulation is used to safely generate behavioral information for those web services of the service-based BP that may have persistent side effects in the system or do not yet have a readily usable implementation. Figure 4.3 illustrates our approach.

Starting from a *draft of* BP (step 1), the *software developer* refines and implements the process (2) using an extended *BPMN editor* (**R1**). This produces BP, including the set of process properties P that can be used for analysis. Given these ingredients, the BPMN editor generates a so-called *configuration spreadsheet* (3), which contains the process properties and a set of simulation parameters. Simulation parameters are used to configure the dynamics of the simulation (**R4**); they include parameters such as the number of process instances to be simulated, the rate at which instances are to be generated, and the execution time of simulated web services (**R6**). Process properties P are used to configure business data for different process execution scenarios; they are associated to the nodes of BP and may refer to both real or simulated web services (**R5**, **R6**). Activities of BP that refer to real services are marked as such and pre-configured by the developer in the extended BPMN editor; the BP analyst can configure the behavior only of simulated services. He does so simply by editing the spreadsheet and defining values for the simulation parameters and process properties (4). Once the simulation is configured, the *BP simulator* reads the configuration and BP and runs the simulation, mimicking the web service behaviors defined by the BP analyst and invoking existing web services. As a result, it generates (5) a process execution log L that contains the observed behavior, which is again stored as a *process execution spreadsheet* (**R7**). The actual verification and performance analysis, is again done by the BP analyst using an *analysis spreadsheet* (6). In this spreadsheet, the analyst defines the metrics M and assertions A over the generated log L as standard spreadsheet functions (**R2**). The spreadsheet automatically performs the necessary calculations, and allows the BP analyst to define charts or tables for the visualization of results (V), turning the analysis spreadsheet into a report r that can easily be saved (**R3, R8, R9**) and shared with the developer (**R10**) (7).

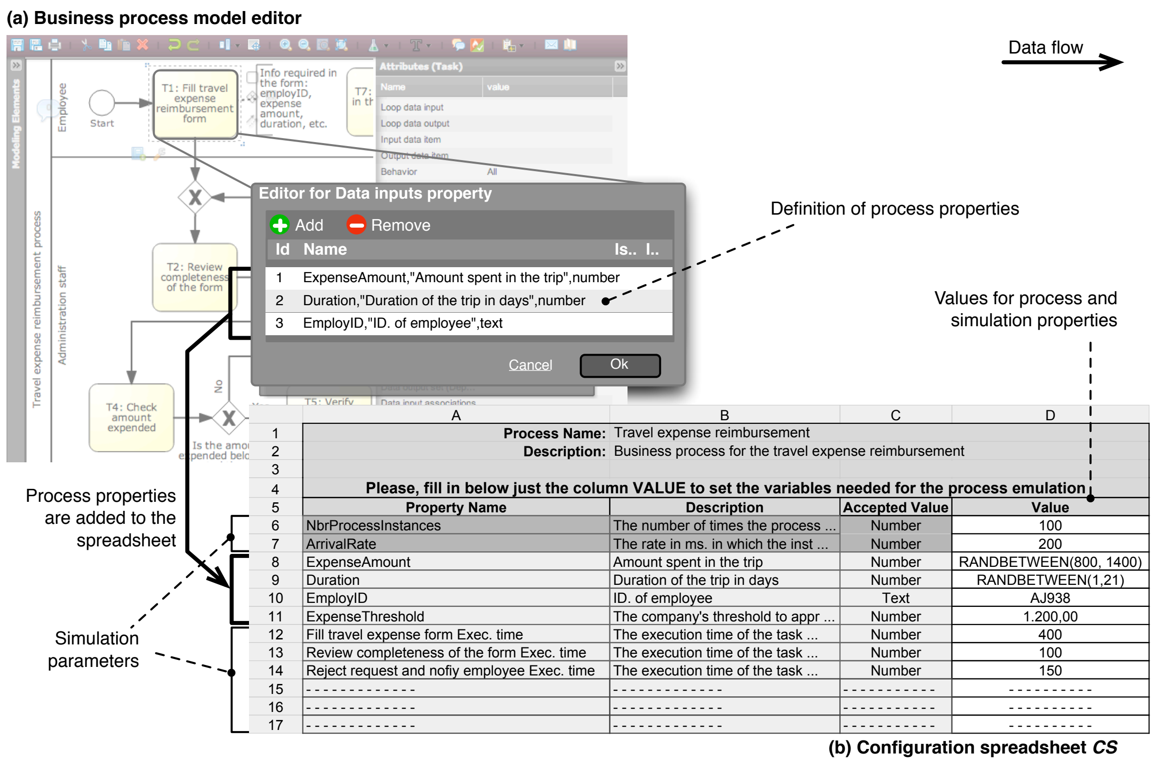
In the following, we detail how processes are modeled, how the simulation is performed, and how predicates, assertions and analysis reports are calculated.

4.3. Business process modeling and simulation configuration

Setting up a BP analysis requires a suitable design of BP and the configuration of the simulation to be performed. The definition of the process properties P by the developer and their configuration by the BP analyst play an important role in setting up the BP analysis. The relevance of process properties its twofold: First, gateway conditions that determine the control flow of a process are defined over process properties. That is, they determine the runtime scenarios of process instances. Second, they are the starting point for the design of metrics and, hence, for the actual design of the verification and performance analysis. The simulation parameters allow the BP analyst to define the time behavior of simulated tasks and the number of task instances to be generated.

The developer models BPs using BPMN (Object Management Group (OMG), 2011), which is a standard process modeling notation targeting both BP analysts and software developers. The BPMN constructs presented in Section 4.1.1 allow him to associate process properties to tasks. For example, Figure 4.4(a) shows the definition of three process properties for the task *Fill travel expense reimbursement form*, namely, $ExpenseAmount$, $Duration$ and $EmployID$. Each property requires a $name$, $description$ and $datatype$, separated by commas, matching the model $p = \langle nid, pname, pdesc, datatype, pvalue\rangle$ introduced before. The value for nid is automatically derived by selecting a task in the process editor; $pvalues$ are defined by the BP analyst using a *configuration spreadsheet* CS. In the same vein, by using standard BPMN constructs the developer can define conditional rules to control the execution of the process. Conditions are defined over process properties. Figure 4.5(a) shows the conditions that regulate the execution flow over the outgoing arcs of the highlighted decision point. Each condition is set by the developer, who defines a Boolean expression, e.g., $ExpenseAmount \leq ExpenseThreshold$, over each gateway's outgoing arc.

A **spreadsheet** is a bi-dimensional array s where each element $s(i, j)$, with i representing the column index and j the row index, represents a cell. A cell $s(i, j)$ can contain one of (i) a value that consists of an alfa-numeric datum, such as in $s(i, j) \leftarrow$ "$AJ487$", (ii)



(a) Business process model editor
Data flow
Definition of process properties
Values for process and simulation properties
Process properties are added to the spreadsheet
Simulation parameters
(b) Configuration spreadsheet *CS*
Travel expense reimbursement process
Employee
Administration staff
Start
T1: Fill travel expense reimbursement form
Info required in the form: employID, expense amount, duration, etc.
T2: Review completeness of the form
T4: Check amount expended
T5: Verify
Is the amount expended below
No
Attributes (Task)
Loop data input
Loop data output
Input data item
Output data item
Behavior
All

Editor for Data inputs property

Add Remove

Id	Name	Is..	I..
1	ExpenseAmount,"Amount spent in the trip",number		
2	Duration,"Duration of the trip in days",number		
3	EmployID,"ID. of employee",text		

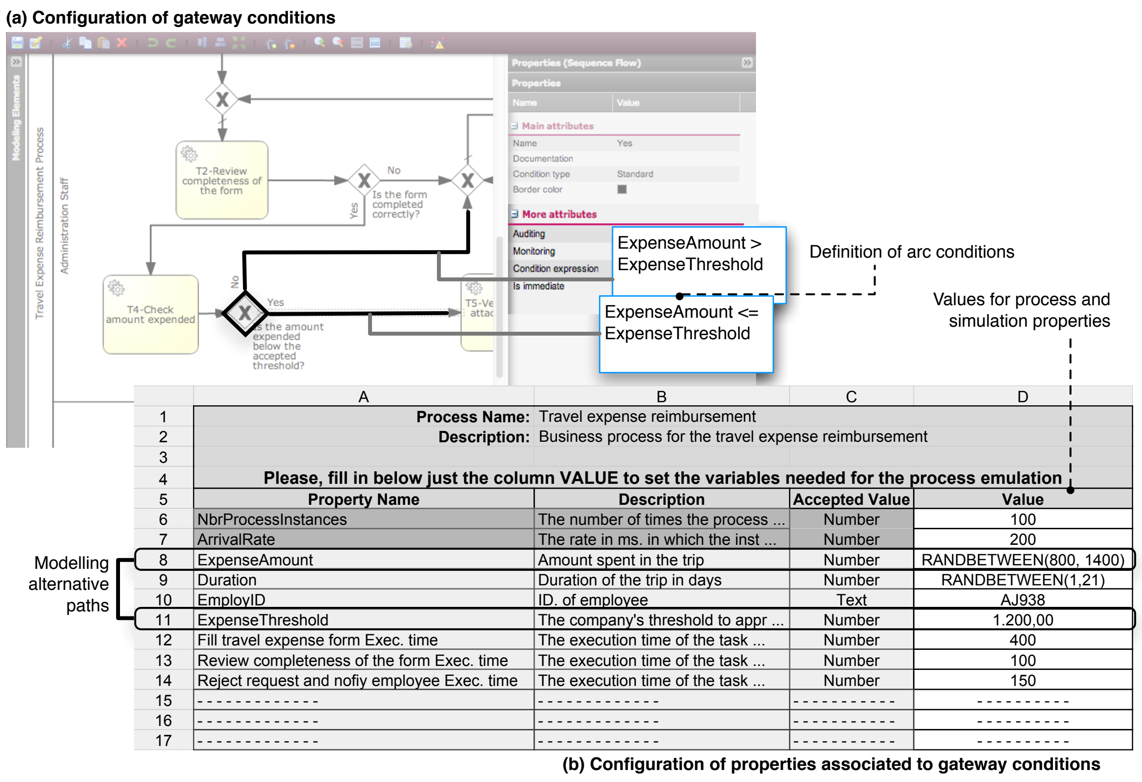
Cancel Ok

	A	B	C	D
1	Process Name:	Travel expense reimbursement		
2	Description:	Business process for the travel expense reimbursement		
3				
4	Please, fill in below just the column VALUE to set the variables needed for the process emulation			
5	Property Name	Description	Accepted Value	Value
6	NbrProcessInstances	The number of times the process ...	Number	100
7	ArrivalRate	The rate in ms. in which the inst ...	Number	200
8	ExpenseAmount	Amount spent in the trip	Number	RANDBETWEEN(800, 1400)
9	Duration	Duration of the trip in days	Number	RANDBETWEEN(1,21)
10	EmployID	ID. of employee	Text	AJ938
11	ExpenseThreshold	The company's threshold to appr ...	Number	1.200,00
12	Fill travel expense form Exec. time	The execution time of the task ...	Number	400
13	Review completeness of the form Exec. time	The execution time of the task ...	Number	100
14	Reject request and nofiy employee Exec. time	The execution time of the task ...	Number	150
15	- - - - - - - - - - - - -	- - - - - - - - - - - - -	- - - - - - - - - - -	- - - - - - - - - -
16	- - - - - - - - - - - - -	- - - - - - - - - - - - -	- - - - - - - - - - -	- - - - - - - - - -
17	- - - - - - - - - - - - -	- - - - - - - - - - - - -	- - - - - - - - - - -	- - - - - - - - - -

FIGURA 4.4. The instruments used for BP modeling and simulation configuration.

a reference to a different cell, such as in $s(i,j) \leftarrow s(p,k)$, or (iii) a formula that combines functions, values and references to other cells such as in $s(i,j) \leftarrow sum(13, s(p,k))$.

The **configuration spreadsheet** CS imports the process properties P of BP for their configuration. To do so, we can follow the simple rule of mapping each property $p_j \in P$ to one spreadsheet row, like in $CS(1,j) \leftarrow p_j.pname$, $CS(2,j) \leftarrow p_j.pdesc$ and $CS(3,j) \leftarrow p_j.datatype$. Figure 4.4(b) shows how the properties are represented in the spreadsheet. The spreadsheet's cells are indexed using letters for columns and numbers for rows. For example, the process property $ExpenseAmount$ associated to the BP model is mapped to row 8 in the spreadsheet using the mapping `CS(A,8)` $\leftarrow$ `"ExpenseAmount"`, `CS(B,8)`$\leftarrow$ `"Amount spent in the trip"`, and `CS(C,8)`$\leftarrow$ `"Number"`. The last cell (`CS(D,8)`) is used to set the values of the property. The rest of the properties are mapped following the same mapping logic.


(a) Configuration of gateway conditions
Modeling Elements
Travel Expense Reimbursement Process
Administration Staff
T2-Review completeness of the form
Is the form completed correctly?
No
Yes
T4-Check amount expended
Is the amount expended below the accepted threshold?
No
Yes
T5-Ve attac
Properties (Sequence Flow)
Properties
Name
Value
Main attributes
Name
Yes
Documentation
Condition type
Standard
Border color
More attributes
Auditing
Monitoring
Condition expression
Is immediate
ExpenseAmount > ExpenseThreshold
ExpenseAmount <= ExpenseThreshold
Definition of arc conditions
Values for process and simulation properties
Modelling alternative paths
(b) Configuration of properties associated to gateway conditions

	A	B	C	D
1	**Process Name:**	Travel expense reimbursement		
2	**Description:**	Business process for the travel expense reimbursement		
3				
4	**Please, fill in below just the column VALUE to set the variables needed for the process emulation**			
5	**Property Name**	**Description**	**Accepted Value**	**Value**
6	NbrProcessInstances	The number of times the process ...	Number	100
7	ArrivalRate	The rate in ms. in which the inst ...	Number	200
8	ExpenseAmount	Amount spent in the trip	Number	RANDBETWEEN(800, 1400)
9	Duration	Duration of the trip in days	Number	RANDBETWEEN(1,21)
10	EmployID	ID. of employee	Text	AJ938
11	ExpenseThreshold	The company's threshold to appr ...	Number	1.200,00
12	Fill travel expense form Exec. time	The execution time of the task ...	Number	400
13	Review completeness of the form Exec. time	The execution time of the task ...	Number	100
14	Reject request and nofiy employee Exec. time	The execution time of the task ...	Number	150
15	- - - - - - - - - - - - -	- - - - - - - - - - - - -	- - - - - - - - - - -	- - - - - - - - - -
16	- - - - - - - - - - - - -	- - - - - - - - - - - - -	- - - - - - - - - - -	- - - - - - - - - -
17	- - - - - - - - - - - - -	- - - - - - - - - - - - -	- - - - - - - - - - -	- - - - - - - - - -

FIGURA 4.5. Modelling and configuring gateway nodes.

To define values for properties, the BP analyst can use a constant value, as in the cell `CS(D,10)`, or he can choose to write a formula that generates values for the cell. For example, the cell `CS(D,8)` uses the function `RANDBETWEEN(800, 1400)` to compute random values between 800 to 1400. By assigning non-constant values to the properties involved in the conditional expressions of gateways, the BP analyst can model the execution of alternative paths. As Figure 4.5(b) illustrates, the random function `RANDBETWEEN`, used to define the values of the property *ExpenseAmount*, is what models the conditional execution of either the *Yes* or *No*-labeled outgoing arcs of the gateway highlighted in Figure 4.5(a). The values obtained from these formulas are computed for each process instance at BP simulation time.

Simulation parameters are configured similarly to process properties. The parameters we consider are three: (i) the number of process instances to be simulated, (ii) the arrival rate for process instances, and (iii) the task duration for each task $t \in T$ in the process model BP. These properties do not need to be explicitly defined by the developer or BP analyst. They are added automatically to the spreadsheet at its generation time. Figure 4.5(b) shows that rows 6 and 7 are filled with the parameters $NbrProcessInstances$ and $ArrivalRate$, respectively. From row 12 on, rows are filled with parameters that define the duration of the tasks. Values for these parameters are defined like values for process properties.

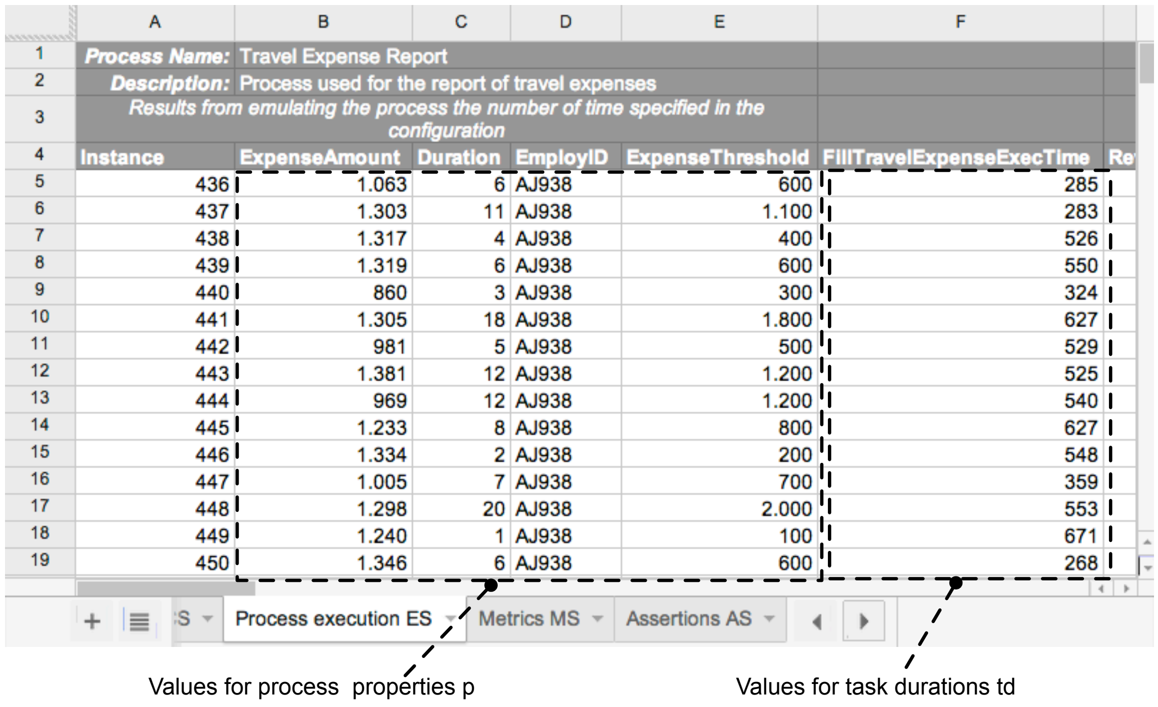
Basic nondeterministic human-behaviors, such as task completion time, can be modeled directly by the BP analyst on the spreadsheet. He can set, for example, a fixed or a normally distributed task completion time. Modern spreadsheet software offers a vast collection of built-in mathematical and statistical functions that enable BP analysts to model the dynamics of human-based tasks by approximating it via mathematical or statistical formulas. Predefined function can be employed also to define process properties, conditional statements and simulation parameters of almost any complexity. If more complex human interventions are required the software developer needs to implement additional pieces of software, e.g., a text recognition algorithm.

The results of the business process modeling and analysis design are a business process model BP and a configuration spreadsheet CS that are consumed by the BP simulator.

Our approach supports only the BPMN constructs introduced in Section 4.1.1. Tasks of type service, exclusive, parallel and inclusive gateways and, sequence flows are supported, while events of types different from *start* and *end* are not yet considered in this work.

4.4. Business process simulation

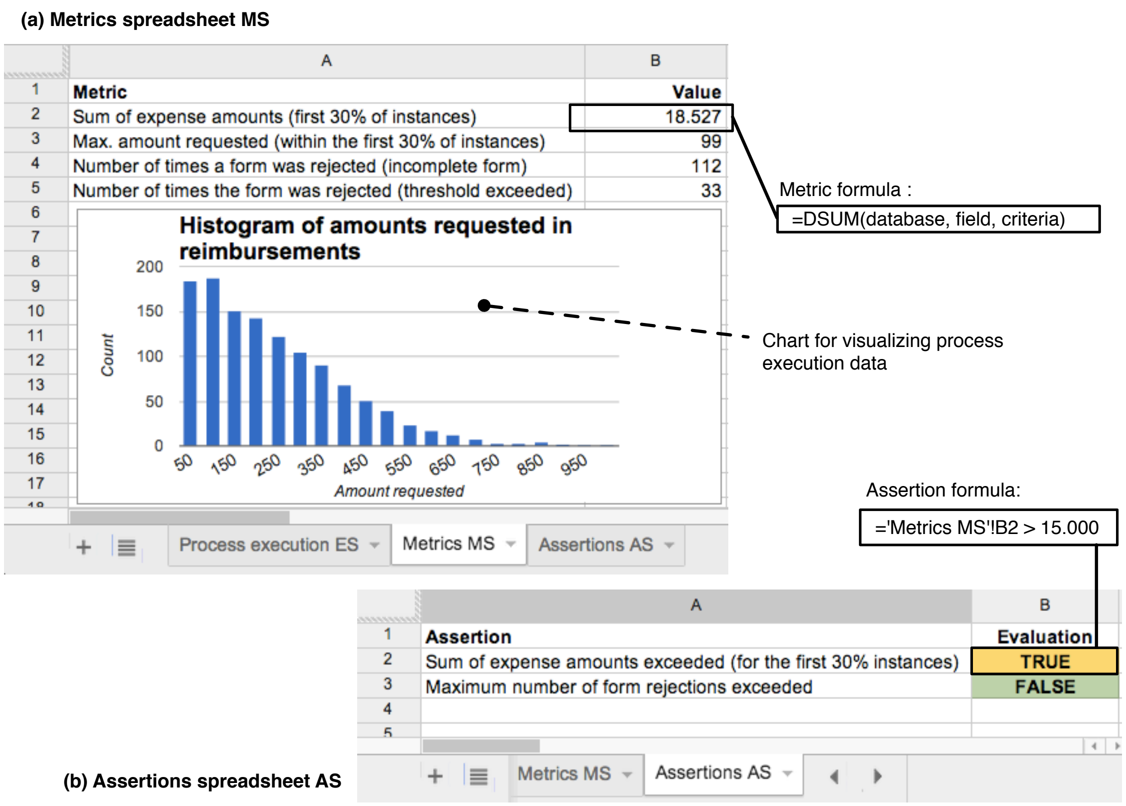
The BP simulator is in charge of simulating the execution of BP based on the configurations provided in CS, thereby producing a process execution log. Refining our definition of Section 4.1, the BP simulator can be seen as $BPsim(BP, CS) = ES$, where BP is the process model, CS is the configuration spreadsheet and ES is the resulting process execution spreadsheet (the log in spreadsheet format).



	A	B	C	D	E	F	
1	***Process Name:***	Travel Expense Report					
2	***Description:***	Process used for the report of travel expenses					
3	*Results from emulating the process the number of time specified in the configuration*						
4	**Instance**	**ExpenseAmount**	**Duration**	**EmployID**	**ExpenseThreshold**	**FillTravelExpenseExecTime**	**Re**
5	436	1.063	6	AJ938	600	285	
6	437	1.303	11	AJ938	1.100	283	
7	438	1.317	4	AJ938	400	526	
8	439	1.319	6	AJ938	600	550	
9	440	860	3	AJ938	300	324	
10	441	1.305	18	AJ938	1.800	627	
11	442	981	5	AJ938	500	529	
12	443	1.381	12	AJ938	1.200	525	
13	444	969	12	AJ938	1.200	540	
14	445	1.233	8	AJ938	800	627	
15	446	1.334	2	AJ938	200	548	
16	447	1.005	7	AJ938	700	359	
17	448	1.298	20	AJ938	2.000	553	
18	449	1.240	1	AJ938	100	671	
19	450	1.346	6	AJ938	600	268	

FIGURA 4.6. Process execution spreadsheet ES containing logged process progression information.

The **process execution spreadsheet** ES is a spreadsheet that holds process execution data that results from the simulation of BP. Each tuple in ES represents a business process instance bpi, and each cell within the tuple stores the runtime values of the elements of bpi as defined in Section 4.1.1. The idea of using this representation, as opposed to an event-based representation, is to keep the querying of business process instances simple and intuitive for the BP analyst and to avoid the need for writing complex event aggregation logics to reconstruct process instances. Thus, to store a business process instance bpi_j in ES, where the associated BP has a number of k properties and l tasks, we map the elements of bpi_j to ES as $ES(1, j) \leftarrow bpi_j.piid$, $ES(2, j) \leftarrow bpi_j.P[1].value$, ..., $ES(k+1, j) \leftarrow bpi_j.P[k].value$, $ES(k+2, j) \leftarrow bpi_j.TD[1].dur$, ..., $ES(k+l+2, j) \leftarrow bpi_j.TD[l].dur$. In other terms, we store the $piid$ in the first column, followed by all the process properties of BP and by the durations of the tasks participating in the BP.


(a) Metrics spreadsheet MS

	A	B
1	**Metric**	**Value**
2	Sum of expense amounts (first 30% of instances)	18.527
3	Max. amount requested (within the first 30% of instances)	99
4	Number of times a form was rejected (incomplete form)	112
5	Number of times the form was rejected (threshold exceeded)	33

Metric formula :

=DSUM(database, field, criteria)

Chart for visualizing process execution data

Assertion formula:

='Metrics MS'!B2 > 15.000

(b) Assertions spreadsheet AS

	A	B
1	**Assertion**	**Evaluation**
2	Sum of expense amounts exceeded (for the first 30% instances)	**TRUE**
3	Maximum number of form rejections exceeded	**FALSE**
4		
5		

FIGURA 4.7. Designing analysis reports: spreadsheets for defining (a) metrics m and (b) assertions a.

Figure 4.6 shows an example of how ES looks like for our travel expense reimbursement process. Using letters to index columns, we have that row 8 holds the process instance bpi_{439}. The mapping is done as follows: $piid$ is mapped to the first column `ES(A, 8)` $\leftarrow$ 439, then, we have the mapping for the process properties $ExpenseAmount$, $Duration$, $EmployID$ and $ExpenseThreshold$ as `ES(B, 8)` $\leftarrow$ `1.319`, `ES(C, 8)` $\leftarrow$ `6`, `ES(D, 8)` $\leftarrow$ `"AJ938"` and `ES(E, 8)` $\leftarrow$ `600`, respectively. Finally, we have the task durations, of which we show in Figure 4.6 only the one corresponding to the task *Fill travel expense reimbursement form* as `ES(F, 8)` $\leftarrow$ `550`.

4.5. Analysis and Visualization of Results

Recall the definition of metrics as a function $m(L) = val$, where L is the process execution log and $val \in \mathbb{R}$ is the metric's value. Within a spreadsheet, m corresponds to a formula that can be specified using the standard spreadsheet functions provided by the adopted spreadsheet tool, L corresponds to the process execution log ES, and val corresponds to the output produced by the spreadsheet formula. Figure 4.7(a) shows the **metrics spreadsheet** MS we use for computing metrics.

In this example, if we consider Google Spreadsheets[1] as our spreadsheet tool, the BP analyst computes the metric *Sum of expense amounts (first 30% of instances)* using a combination of the spreadsheet functions `SORT(range, sortColumn, isAscending, sortColumn2,isAscending2)` and `DSUM(database, field, criteria)`. The function `SORT(...)` sorts the process instances in ascending order based on the expense amount. The function `DSUM(...)` sums the amount requested for first 30% number of instances that appear in the sorted list. Due to the lack of space to fully explain the use of the aforementioned spreadsheet functions, we put in Figure 4.7(a) only the reference to the formula used to compute the sum.

An assertion was defined as a function $a(L, Pred) = bool$, where L is the process execution log, $Pred$ is a set of predicates and $bool \in \{true, false\}$. In turn, a predicate is a function $pred(L, m) = bool$, where L, m and $bool$ are as defined before. In order to define assertions in a spreadsheet, we use the standard logical operators provided by spreadsheet tools. Figure 4.7(b) shows an example of an **assertions spreadsheet** AS that can be used to compute assertions. For instance, the spreadsheet formula `'Metrics MS'!B2 > 15.000` checks whether or not the sum of expenses for the first 30% of the instances exceeds the maximum amount allocated for the *fast track* reimbursement. This assertion is composed of a single predicate that compares the metric *Sum of expense amounts (first*

[1] http://docs.google.com/spreadsheet/

30% of instances) (cell `MS(B, 2)`) with the maximum amount allowed (i.e., 15K euros). While this is a simple assertion, the BP analyst can construct fairly complex ones by combining logical operators (such as `AND`, `OR` and `NOT`) and predicates.

Also the definition of **analysis reports** relies on the built-in data visualization tools (charts and tables) provided by the spreadsheet. Recall that reports are of the form $r(L, M, A) = V$, with L being the process execution log and M and A being the sets of metrics and assertions, respectively. V are the visualization widgets (e.g., charts or tables) used to construct the report. Charts and tables conveniently summarize the results obtained from the computation of metrics and assertions. For example, Figure 4.7(a) uses a simple bar chart that plots an histogram of the expense amounts requested. This enables the visual analysis of its distribution. The dataset for this chart was prepared using metrics computed with standard spreadsheet formulas (the details are skipped in this article). The exact design of the report is up to the BP analyst, who knows best how to design the report so as to most effectively communicate his findings to the developer.

Thus, using metrics, assertions and charts, operationalized with the help of the built-in functions of the spreadsheet tool, the BP can analyze the outcomes of the simulations generated by different configuration settings that reproduce the various execution scenarios in study. Back to our reimbursement process, we can see in Figure 4.7 that under the configuration settings of the simulation parameters, the BP analyst can learn that reimbursing all 30% of the of lowest request amounts is not possible (the total sum of amounts exceeds the budget), and that, for example, either the target percentage should be lowered, the budget for the *fast track* reimbursement option should be incremented, or the company should still tolerate a delay in a (fewer) number of reimbursement requests.

4.6. Implementation

Figure 4.8 illustrates the functional architecture of the prototype of our solution. On the client side, we have the *process model editor* used to design BPs and the *spreadsheet tool* used to work with the spreadsheets CS, ES, MS and AS. On the server side, we have

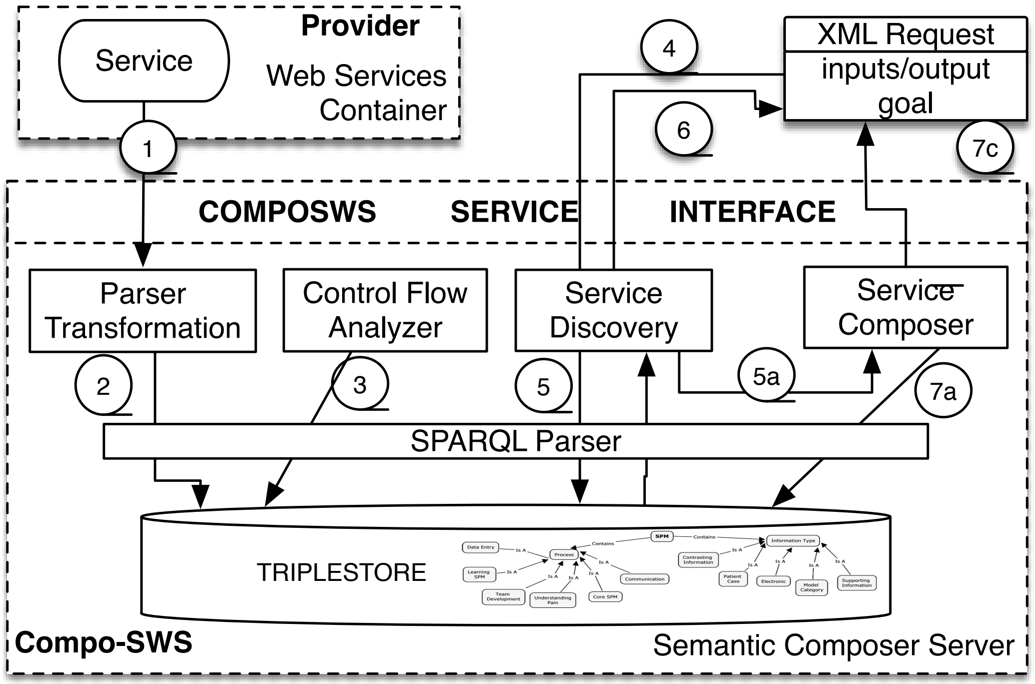


FIGURA 4.8. Architecture of the proposed solution.

all the components that are in charge of configuring the environment for the simulation and analysis of BPs. When a process model BP is ready for simulation, it is sent to the *simulation server*, which is in charge of managing the requests for BP simulations. This request is forwarded to the *simulation configurator*, which takes the model BP and performs three tasks: First, it creates the mock services that mimic the tasks in BP at simulation time and saves them in the *service repository*. Then, it stores BP into the ***BP* *model repository*** for future use. Finally, it requests the *spreadsheet manager* to create the templates for the spreadsheets CS, ES, MS and AS, which are stored into the *spreadsheet repository* for reuse in the following phases.

Back to the client side, the BP analyst can use the *spreadsheet tool* to configure CS and send it to the *simulation server* for the simulation of BP. The *simulation server*, in turn, forwards BP and CS to the *simulation manager*, which is in charge of managing the deployment of BP (using the configurations in CS) into the ***BP simulator***. The latter queries ES, the mock services and BP from the *spreadsheet*, *service* and *BP model repositories*, respectively, simulates the BP, and stores the obtained process execution data into ES. The resulting ES is stored back into the *spreadsheet repository* and made available, together with MS and AS, to the *spreadsheet tool* for analysis.

The current implementation of our analysis suite uses Signavio[2] as *process model editor*, Google Spreadsheets[3] as *spreadsheet tool*, and Activiti[4] as internal engine of the *BP simulator*. Signavio has been extended to enable the generation of the configuration spreadsheet. Google Spreadsheets has been extended to interface with the simulation back-end, acting as user interface of the BP analyst for simulation and BP analysis. Both extensions are implemented via JavaScript; the back-end components are implemented as standard web applications using Java. The screenshots in Figures 4.4–4.7 show the look and feel of the prototype at work. At http://goo.gl/v4k2Yj we show a video of the tool in action. The source code of the tool can be downloaded from https://sites.google.com/site/ssbptester/.

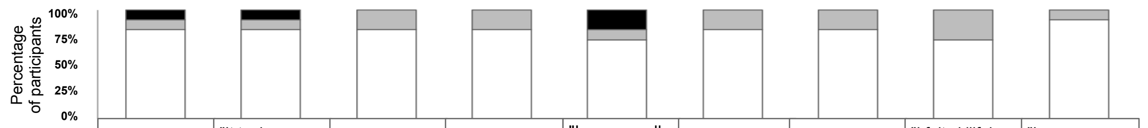
4.7. User studies

We ran two user studies to validate the viability of the proposed approach. First, we assessed the suitability of spreadsheets with real BP analysts, then the whole approach with master students. We summarize both studies next; details of the study (scenarios, questionnaires, raw data) can be found at
http://sites.google.com/site/ssbptester.

4.7.1. Business process analysis with spreadsheets

The objective of this study was to understand whether our approach achieves the first goal of our problem statement, i.e., enabling BP analysts to analyze business processes. This study specifically focused on the configuration of the BP simulation and the analysis of process execution data. The participants to the study were three employees of the

[2]http://code.google.com/p/signavio-core-components
[3]https://docs.google.com/spreadsheet
[4]http://www.activiti.org



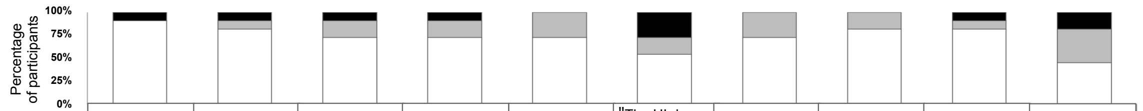
		"I felt comfortable"	"It took a reasonable amount of time"	"It was according to my skills"	"I was able to easily follow all the tasks"	"I am overall satisfied with the experience"	"The tasks were easy to understand"	"I felt confident in doing the given tasks"	"I felt skillful enough to perform the tasks"	"I was successful in setup the properties"
■	Negative	1	1	0	0	2	0	0	0	0
▨	Neutral	1	1	2	2	1	2	2	3	1
□	Positive	9	9	9	9	8	9	9	8	10
	Mean	1,909	2,090	1,545	1,636	2,272	1,818	1,727	1,727	1,636
	Median	2	2	1	1	2	2	2	1	2
	W-value	3,5	8,5	0	0	9	0	0	0	0
	p-value	0,014	0,052	0,006	0,007	0,052	0,007	0,007	0,010	0,005

FIGURA 4.9. Survey results for questions regarding the overall experience of software developers.

Paraguayan subsidiary of DHL, who operate as BP analysts at an everyday basis. Each BP analyst participated separately in a one hour session, conducted within the premises of the company. All participants were familiar with spreadsheets; only one of them knew Google Spreadsheets. None of them had a background in computer science. The user study was structured as a *usability test* followed by a *retrospective probing* in the form of a semi-structured interview (Lazar, Feng, & Hochheiser, 2010).

For the usability test, participants were introduced to our tool and watched a video exemplifying the features of the tool. Then, they were presented with a analysis scenario based on a simplified version of the BP model shown in Figure 4.1(a), provided with a suitable configuration spreadsheet CS, and asked to analyze the BP according to the scenario. For the retrospective probing, we asked participants questions about their experience, thoughts and actions after the usability test.

All participants agreed that the BP analysis tasks were easy to understand, but in some cases difficult to perform. The main reason for this was that, while participants did not have problems in writing simple spreadsheet formulas like sums, averages and standard deviations, they faced difficulties in defining complex formulas that involved conditional and statistical distribution functions. This problem was exacerbated by the fact that the nomenclature of the functions in Google Spreadsheets differs from the one found in Microsoft Excel, which they were more acquainted with. The language of the tool also contributed



		"It was simple to use the software for designing processes"	"I was able to effectively complete my tasks using the software"	"I felt comfortable using the software"	"It was easy to setup process properties for the BP simulation"	"It was easy to generate the spreadsheet from the BP model"	"The UI does NOT differ much with respect to the original version of Signavio"	"It was easy to understand the concepts concepts introduced in Signavio"	"Overall, I am satisfied with the software"	"I would recommend this tool to others"	"I would prefer to use this tool for BP analysis instead of others"
■	Negative	1	1	1	1	0	3	0	0	1	2
▨	Neutral	0	1	2	2	3	2	3	2	1	4
□	Positive	10	9	8	8	8	6	8	9	9	5
	Mean	1,818	1,545	1,909	2,000	1,818	2,636	1,909	2,000	2,000	2,636
	Median	2	1	1	2	2	2	2	2	2	3
	W-value	8,5	1,5	6	6,5	0	15	0	0	8	8,5
	p-value	0,028	0,006	0,046	0,057	0,012	0,390	0,012	0,006	0,047	0,386

FIGURA 4.10. Survey results for questions regarding the BP model editor.

to the issue: the spreadsheets for the analysis of the BP were all in English, and so were the names of the functions. Although all participants declared good knowledge of English, they were used to work with spreadsheets in Spanish, which made it sometimes difficult to find even functions they knew. These problems were however easily overcome with a small help from the person running the study.

As for the general feeling and mood after the study, two of the participants said they felt comfortable, while the third one said the experiment was long and stressful. All three agreed that they had to learn new concepts and terminology they were not familiar with regarding both BP simulation (e.g., arrival rates) and spreadsheet functions (e.g., conditional and statistical distribution functions). All participants agreed that using spreadsheets for analysing BPs is useful and close to their working experience for two main reasons: (i) they are familiar with spreadsheets, and (ii) spreadsheets are suitable to analyze data in a tabular format, helped by the pre-built spreadsheet formulas, filters and charts. When participants were asked if they would use the tool, all of them responded positively and stated that the approach would indeed be effective in helping them to autonomously analyze BPs, provided they have a good working knowledge of the spreadsheet's predefined functions.

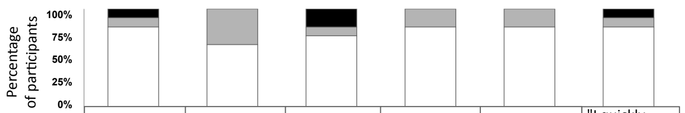
Although this study was conducted with only three BP analysts and, hence, does not have statistical relevance, we nevertheless consider the study a good indicator for the suitability of using our spreadsheet-based approach for analyzing BPs. Participants intuitively understood their tasks and unanimously agreed on the viability of the approach.

4.7.2. Modeling, analyzing and reporting

The second study aimed to provide end-to-end coverage of our approach and statistical tests. The study therefore also aimed to understand whether the approach facilitates the discussion of findings between the BP analyst and the software developer. To do so, we involved a total of 22 MSc students taking part of a BPM course at the University of Trento, Italy, all of them with a background in computer science, BP modeling (BPMN) and spreadsheets. The study lasted around 1 hour and 15 minutes and it was carried out in the laboratory of the university.

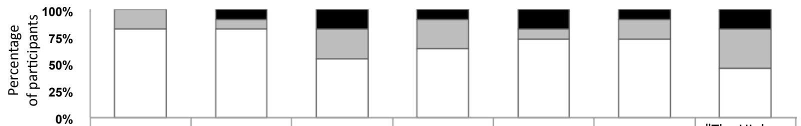
This study was also structured as a *usability test* with *retrospective probing*. We again offered a training session in the form of a live demo to introduce the tool to the students. The retrospective probing took the form of an online survey. Students were paired up, one playing the role of the BP analyst and one the role of the software developer. Each role was assigned a number of tasks related to the role, specified in two scenarios based on a simplified version of the BP model in Figure 4.1(a). The first scenario consisted in performing one analysis cycle as presented in Figure 4.3. The second scenario asked the BP analyst to communicate a change in the BP model to the software developer and to analyze the modified BP again.

To collect feedback, we prepared two surveys with 26 and 19 questions for the BP analyst and software developer, respectively. The questions were answered using a Likert scale of 1 (strongly agree) to 5 (strongly disagree) (Lazar et al., 2010). We consider answers 1 and 2 as *positive answers*, 3 as *neutral answer*, and 4 and 5 as *negative answers*. The survey included also open questions that allowed participants to provide free feedback. The results of the survey are reported in the form of descriptive statistics using the mean and median of the sample. We also use a two-sided hypothesis test to test the significance of the

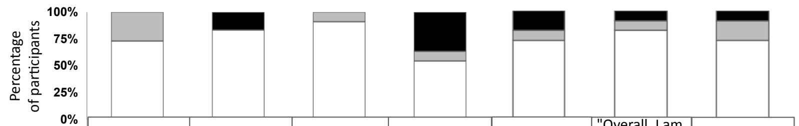


	"I felt comfortable"	"The analysis took a reaso-nable amount of time"	"The analysis was accor-ding to my skills"	"I was able to easily follow all the tasks"	"I am overall satisfied with the experience"	"I quickly acquired the necessary knowledge to perform the tasks"
■ Negative	1	0	2	0	0	1
▨ Neutral	1	4	1	2	2	1
□ Positive	9	7	8	9	9	9
Mean	1,818	2,091	1,909	1,636	1,727	1,727
Median	2	2	1	1	2	1
W-value	3	0	5	0	0	2,5
p-value	0,012	0,019	0,021	0,007	0,007	0,010

FIGURA 4.11. Survey results regarding the BP analysts' overall experience.



	"It was simple to use the spread-sheet"	"I was able to effectively complete my tasks"	"I felt comfortable using the spreadsheet "	"I find spread-sheets convenient to simulate BPs"	"I find spread-sheets intuitive to simulate BPs"	"It was easy to setup the properties"	"The UI does NOT differ much with respect to the original version of the spread-sheet"
■ Negative	0	1	2	1	2	1	1
▨ Neutral	2	1	3	3	1	2	4
□ Positive	9	9	6	7	8	8	6
Mean	1,545	1,818	2,364	2,000	1,909	1,818	2,273
Median	1	2	2	2	1	1	2
W-value	0	0	6	2	5	2	5
p-value	0,006	0,012	0,097	0,025	0,021	0,014	0,135



	"I would recommend this tool to others"	"The process property names and descriptions were clear"	"The process property names and descriptions were useful"	"I would prefer to use this tool for BP analysis instead of others"	"It was easy to under-stand the concepts and elements used"	"Overall, I am satisfied with the spreadsheet for the simulation of BPs"	"The spread-sheet was convenient to analyze the output data"
■ Negative	0	1	0	4	2	1	1
▨ Neutral	3	0	1	1	1	1	2
□ Positive	8	10	10	6	8	9	8
Mean	1,545	1,727	1,727	2,455	1,909	1,727	1,727
Median	1	2	2	2	1	1	1
W-value	0	3,5	0	14	5	2,5	1,5
p-value	0,006	0,008	0,005	0,172	0,021	0,010	0,010

FIGURA 4.12. Survey results regarding the use of spreadsheets for BP analysis.

answers for each single question. The test makes use of a *Wilconox, signed rank test* with a significance level of $p = 0,05$ and a null hypothesis $H_0 : \eta = 3$ (where η represents the median) and an alternative hypothesis $H_A : \eta \neq 3$. In other words, the null hypothesis is that participants of the study have a neutral answer for each question, against the alternative hypothesis that they are lean towards positive or negative answers.

Software developers: Figure 4.9 shows the feedback from software developers regarding their overall experience. Most questions were answered positively (which is confirmed by our tests: $p - values$ are lower than 0,05), with the exceptions of the questions regarding the time taken by the experiment and the overall satisfaction with the experience, in which we obtained rather neutral answers (for these two questions, the $p - values$ where slightly higher than 0,05). Figure 4.10 shows the feedback regarding the BP model editor. Most questions were again answered positively, confirmed by our hypothesis tests. The exceptions are the questions related to the easiness for setting up process properties, the similarity of the BP editor's UI w.r.t. the original version of Signavio and the preference of our tool over the others, for which we obtained rather neutral answers (we cannot reject H_0). One participant pointed out positively that the joint use of the BP model editor with Google Spreadsheets "permits teams to work together in real-time," while another participant recommended to improve the debugging functionalities of the editor "in order to be able to find errors in the BP model before running the BP simulation."

BP analysts: Figure 4.11 summarizes the overall experience by BP analysts. All questions were answered positively and confirmed by our statistical tests. Regarding the use of spreadsheets, Figure 4.12 reports that most questions were also answered positively. The three exceptions we have are the questions regarding the comfortability with the use of spreadsheets, the similarity of the UI of the spreadsheet w.r.t. to the original version of the tool, and the preference of the proposed tool over the others. For these questions, we have a split preference on the answers and the average results yields a neutral answer (with $p - values$ equal to 0,097; 0,135 and 0,172, respectively, for each question). The reason for this neutrality, despite the positive answers in the previous questions, may be motivated by both the paradigm shift in the approach used to analyze BPs and the fact that our tool is a prototype implementation, and thus, not yet meant to be ready for use in a production environment.

4.7.3. Discussion of results

The results of both user studies provide good evidence of the potential of our approach. Our observations testify that the interaction between the BP analysts and the developers were well-disposed and facilitated by our approach, confirming the suitability of the approach for collaborative BP analysis. The company's BP analysts were more inclined towards the use of mainstream spreadsheet tools, such as Microsoft Excel, given their familiarity with such tools. BPM students, instead, appreciated more the use of Google Spreadsheets, due to its suitability for real-time collaboration. Given their higher familiarity with conditional expressions and complex statistical functions compared to the BP analysts, the BPM students felt much more comfortable in using spreadsheets to analyze the process execution log. This suggests that, in order to bring our tool to a real setting, it is necessary to make sure BP analysts have the necessary training in using spreadsheets. However, it is important to note that all participants in both user studies easily understand the mapping of the BP analysis problem to the problem of analyzing data in spreadsheets. Once participants figured out the right spreadsheet functions to use, they were able to easily define metrics and assertions over the process execution log organized into process instances. In conclusion, we accept our hypothesis that mapping the BP analysis problem to the design of a spreadsheet calculation enables the BP analyst to analyze BPs autonomously.

4.8. Qualitative Analysis

We complement the above user studies with a qualitative analysis of our tool (Spreadsheet-based BP Analyzer). The analysis consists in a comparison of our tool against state of the art solutions used for BP analysis and simulation.

The analysis includes today's most representative commercial tools in the realm of BP analysis as well as academic and open source solutions. In particular, we considered Websphere Business Modeler (v. 6.2) (IBM, 2009), the licensed solution of IBM for simulation and analysis of BPs, TIBCO Business Studio (v. 3.9) (TIBCO, 2014), the proposal of TIBCO Software Inc. for modeling, analysis, and simulation BPs. Free alternative solutions are also included in our analysis. In concrete, we consider BizAgi Modeler (BizAgi, 2015), the free solution offered by BizAgi to document and analyze BPs, Bonita Open Solution (v. 5.6) (Bonitasoft, 2011), the open-source proposal of BonitaSoft for modeling and simulating BPs represented in BPMN, and Adonis Community Edition (BOCGroup, 2015), the free modeling and simulation BPM tool offered by BOC Group. Two academic tools are also present in the comparative analysis. The first one, Signavio Process Editor (v. 9.0) (Signavio, 2015), started as an academic project and recently turned into a commercial solution. The proposal targets business practitioners in the context of modeling and simulation of BPs. The second one, SimQuick (Hartvigsen, 2004), is an entirely academic BP simulation tool that uses spreadsheets and Microsoft Excel macros to enable the design and simulation of BPs.

4.8.1. Comparison Framework

The comparison framework used to conduct the analysis is based on five categories of functionalities, namely, *BP modeling*, *simulation configuration*, *simulation*, *analysis*, and *reporting*. These five categories represent the typical phases of the BP design and analysis process proposed by the analyzed tools. The framework in particular aims to highlight those concerns that are related to the BP analyst inside this BP design and analysis process.

Although BP modeling is not the focus of our work, it was included in the analysis given its crucial role in the proposed process lifecycle. The **BP modeling** capabilities of the tools are analyzed under three dimensions: *notation*, i.e., which BP modeling notation is supported by the tool, *model verification*, i.e., what types of modeling errors can be detected, and *debugging*, i.e., what types of features are available to find and fix the modeling errors.

The analysis of functionalities offered by the tools for the **BP simulation configuration** is mainly focused on the statistical facilities offered by the tools to configure *process tasks*, *resources*, and *domain-specific parameters* (e.g., expenses, interest rate). We also analyze the variety of domain-independent variables (e.g., number of simulation instances, instances arrival rate, loading period) that are possible to setup with the tools.

We capture the **simulation** capabilities of the tools through two dimensions, namely, *runtime monitoring* and *task behavior*. The fist one is related to the features offered by the tools to monitor the execution of the simulation, e.g., animation and runtime statistics, while the second one is associated to the simulation capacities of the tools w.r.t. the simulation of task behavior.

Regarding the **analysis** capabilities of the tools, we focus on understanding the flexibility of the tools in giving BP analysts the freedom to write their own *analysis instruments* (metrics and assertions) and the power of the tools in *assisting analysts* in obtaining information about the behavior of the process.

The **reporting** dimension includes the flexibility of the tools to create *custom analysis reports*, the features offered to foster *collaborative analysis and reporting*, and whether the *simulation output* is accessible and in which format.

Because not all the tools could be installed for their study, we exclusively base our analysis on the official documentation provided for each solution. We therefore highlight the situations in which the documentation provides insufficient information to draw a conclusion about a particular dimension.

Figure 4.13 presents the analysis in a table where columns represent the description of each tool. The gray-shaded column contains the descriptions of our Spreadsheet-based BP Analyzer.

4.8.2. Analysis

BP modeling. All the tools offer similar modeling functionalities. Almost all of them have BPMN editors equipped with features that perform automatic model checking (syntax validation and deadlocks verification). They also provide model debugging functions through warning and errors messages and by coloring the conflicting elements. In addition, TIBCO provides features to semi-automatically fix simple syntax errors. SimQuick

		IBM WebSphere Business Modeler	TIBCO Business Studio	BizAgi Modeler	Bonita Open Solution	Adonis Community Edition	Signavio Process Editor	Spreadsheet-based BP Analyzer	SimQuick
BP Modelling	Notation	BPMN	BPMN	BPMN	BPMN	BPMN	BPMN	BPMN (Signavio)	BP SimQuick specific notation
	Model Verification	Syntax, dead-locks	Syntax, dead-locks	Syntax, dead-locks	Syntax, dead-locks	Syntax, dead-locks	Syntax, dead-locks	Same as signavio	Unavailable information
	Debugging	Error messages, highlighting of conflicting elements	Warning and error messages, highlighting of conflicting elements, auto correction	Error messages, highlighting of conflicting elements	Error messages, highlighting of conflicting elements	Error messages, highlighting of conflicting elements	Warning and error messages, highlighting of conflicting elements	Same as signavio	Unavailable information
BP Simulation Configuration	Task Configuration	Execution time through constants and statistical distributions. Constant cost	Execution time through historical data, constants and statistical distributions. Constant cost	Execution time through constants and statistical distributions. Constant cost	Execution time through constants and statistical distributions. Constant cost	Execution time and cost through constants	Execution time through constants and statistical distributions. Constant cost	Execution time and cost through constants and statistical distributions	Execution time and cost through constants and statistical distributions
	Domain-specific parameters	Unavailable information	Constants, statistical distributions, expressions	Not supported	Constants, statistical distributions, expressions	Constants, expressions	Not supported	Constants, statistical distributions, expressions	Not supported
	Domain-independent variables	Timespan, workload calendar	Number of instances, workload calendar	Number of instances, arrival rate, workload calendar	Number of instances, workload calendar, arrival rate	Number of instances, workload calendar	Number of instances, arrival rate, timespan, currency	Number of instances, arrival rate	Number of instances, instances duration
	Resource settings	Quantity, work schedule and cost per units through constants	Quantity and cost per units through constants	Quantity, work schedule and cost per units through constants	Quantity, work schedule through constants	Unavailable information	Quantity and cost per units through constants	Not supported	Quantity through constants and statistical distributions
BP Simulation	Runtime monitoring	Interactive animation, runtime statistics (max, min, avg, total)	Interactive animation, runtime statistics (max, min, avg, total)	Runtime statistics (max, min, avg, total)	Not supported	Not supported	Not supported	Not supported	Not supported
	Tasks behavior	Simulation of execution time, cost and resource utilization	Simulation of execution time, cost, domain-specific parameters and resource utilization	Simulation of execution time, cost, and resource utilization	Simulation of execution time, cost, domain-specific parameters and resource utilization	Simulation of execution time, cost, domain-specific parameters and resource utilization	Simulation of execution time, cost and resource utilization	Simulation of defined tasks behavior or invocation of real services, simulation of domain-specific parameters	Simulation of execution time and resource utilization
BP Analysis	Metrics	Pre-defined standard metrics and user defined metrics based on arithmetic operations, statistical distributions	Pre-defined metrics, such as duration, cost, throughput, resources utilization and idle time	Pre-defined standard metrics (e.g., duration, cost, throughput, resources utilization, bottleneck)	Pre-defined standard metrics	Pre-defined standard metrics	Pre-defined standard metrics	User-defined through arithmetic operations, statistical distributions, expressions	User-defined through arithmetic operations, statistical distributions, expressions
	Assertions	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported	Boolean expression over metrics	Boolean expression over metrics
	Analysis Assistance	What-if-analysis, As-Is and To-Be comparison	Resource utilization comparison between instances	What-if-analysis	Not provided	Not provided	Not provided	Not provided	Not provided
BP Analysis Report	Raw simulation output	Not provided	Not provided	Event-based, tabular, exportable	Accessible; the format is not specified in the documentation	Trace-based, tabular, exportable	Event-based, downloadable in MXML format	Accessible and trace-based, tabular, exportable	Accessible and trace-based, tabular, exportable
	Type of reports	User-defined	Pre-defined. Textual and graphical	Pre-defined. Textual and graphical	Pre-defined. Textual and graphical	Pre-defined. Textual and graphical	Pre-defined. Textual and graphical	User-defined	User-defined
	Features for collaborative analysis and reporting	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported	Interactions through comments, notes, and real-time chat	Built-in support for comments and notes

is an exception within this category because it has its own modeling notation implemented through Microsoft Excel macros. In our case, the full modeling functionalities is based on Signavio Core Components, the free and open-source version of Signavio Process Editor and thus our modeling features are equivalent to that of Signavio.

BP simulation configuration. No big differences are appreciated when comparing the features for configuring the simulation. All the tools enable the use of statistical distributions to configure the duration of the tasks. Spreadsheet-based BP Analyzer and SimQuick allow for the use of distributions to set up the costs associated to the execution of the process tasks while the rest of the tools allow only for the use of constant values to define task execution costs. TIBCO provides features to set up tasks duration using historical data.

Half of the tools, namely TIBCO, Bonita, Adonis, and Spreadsheet-based BP Analyzer, offer the possibility to model, through constants, expressions, and even by statistical distributions, the value of domain-specific parameters, such as interest rate, expenses, return of investment. All the tools enable the definition of domain-independent variables. In SimQuick and Spreadsheet-based BP Analyzer a couple of variables can be configured, i.e., number of simulation instances and arrival rate, while in the rest of the tools a large variety of parameters can be defined, such as timespan in which the instances are created and workload calendar.

Almost all the tools, excluding ours, enable the definition of quantity, work schedule and cost per units of the resources associated to the process execution. Being a tool that is constructed on top of a spreadsheet editor (Microsoft Excel), SimQuick provides the chance to employ statistical distributions to model resources while in the other tools only constant values can be used.

BP Simulation. Only TIBCO, WebSphere and BizAgi support runtime-monitoring functions. The first two provide interactive animation features that allow users to follow the execution of the simulation step-by-step. In addition, all of them display descriptive statistics at runtime, e.g., average tasks duration, most expensive simulation instance, number of current idle resources and average waiting time. The main advantage of Spreadsheet-based BP Analyzer in relation to the others is the possibility to go from 0% to 100% on the simulation of the behavior of the process tasks. In other words, Spreadsheet-based BP Analyzer provides the flexibility to choose whether a task in the BP should be simulated or carried out by a real web service. This feature, which is not available in any other state-of-art tools that offer the chance of simulating the execution time, cost, domain-specific process parameters and resource utilization of the tasks, provides the possibility to verify whether the real services operate as expected.

BP Analysis. Spreadsheet-based BP Analyzer and SimQuick are the only ones that fully empower BP analysts to define their own metrics and assertions. In order to provide such flexibility, they leverage on the power of the spreadsheet's built-in functions. Similarly, WebSphere provides user-defined metrics but from the documentation is not clear whether this functionality is available to measuring simulation data or real data.

The rest of the tools offer standard, predefined metrics, e.g., duration, cost, throughput and resource utilization, through which the analyst can get information about process behavior. Also, only SimQuick and Spreadsheet-based BP Analyzer are the only ones that give BP analysts the chance of writing assertions on top of the metrics computed over the process execution data.

A useful and important feature is the possibility to assist analysts in the evaluation of their business processes. Only TIBCO, IBM WebSphere and BizAgi offer additional analysis functions. WebSphere and BizAgi provide features to conduct what-if-analysis while TIBCO enables the comparison of resource utilization between simulation instances. In addition, IBM WebSphere provides features that ease the comparison between the ideal BP model (to-be) and the current version of the model (as-is).

BP Analysis Report. Most of the tools provide predefined reports. However, SimQuick and Spreadsheet-based BP Analyzer offer the users the possibility to create their own reports. By exploiting the graphical and analytical features of spreadsheets editor programs

(Excel, Google Spreadsheet) SimQuick and Spreadsheet-based BP Analyzer allow BP analysts to draw custom reports of any complexity. Following a more restricted approach, WebSphere also offers user-defined report features. By employing a drag-and-drop designer, BP analysts are able to build custom reports which are based on a limited set of predefined elements, such as text-fields, tables, and summary statistics (counts, sums, and averages).

The spreadsheet editors used by SimQuick and Spreadsheet-based BP Analyzer offer, in addition, built-in functionalities that enable the possibility to add comments and notes on the reporting documents, which can facilitate the communication between BP analysts and developers. In the case of Spreadsheet-based BP Analyzer this communication may even happen in real-time thanks to the chat functionalities of Google Spreadsheet.

Neither IBM Websphere nor TIBCO provide the raw output of the simulations, which limits the possibility of running further BP analyses. The rest of the tools do allow users to access the simulation output promoting the use of alternative tools to further analyze BPs. In this sense, BizAgi provides the output in a tabular, exportable and event-based format; Signavio in an event-based, downloadable MXML format (van der Aalst, 2011), while Adonis, Spreadsheet-based BP Analyzer, and SimQuick offer the raw simulation output in a tabular, trace-based and exportable scheme. Bonita's documentation states that simulation outputs can be downloaded but their content and format are not specified.

4.8.3. Discussion

The above analysis helps to understand better the distinctive and innovative aspects of the Spreadsheet-based BP Analyzer. The combination of BP simulation and a spreadsheet-based UI (Google Spreadsheet) accompanied with a standard BPMN process model representation equips also BP analysts with limited technical skills with a single tool that enables them to simulate, verify and analyze the performance of BPs through personalized instruments (metrics, assertions and custom reports). The use of a full-fledged BP engine to run the simulation enables the flexible invocation of both simulation web services that mimic the behavior of other web services as well as real, production web services. This turns the

Spreadsheet-based BP Analyzer into an instrument that can be used both by the BP analyst for his BP verification and simulation and by the developer for the testing of how real web services perform inside a simulated process. As an add-on, the live collaboration support by Google Spreadsheet offers a new and dynamic communication alternative that can ease the interaction between BP analysts and developers.

Two limitations of the Spreadsheet-based BP Analyzer w.r.t. to some of the tools presented in this comparative analysis are the lack of support for the simulation of human resource utilizations (given our focus on service-based processes) and the lack of additional analysis aids such as what-if-analyses. We plan to extend the Spreadsheet-based BP Analyzer with these functionalities in future work.

4.9. Conclusion

In this article we approached a relevant and timely issue in today's business process management practice, i.e., that of analysing processes. We specifically emphasized the role of the BP analyst, not only in providing input for the design of processes but also in analyzing them. In order to enable BP analysts to perform own analyses without the need for programming skills, we conceived a technique that is specifically tailored to the average skills of BP analysts. The intuition we followed for the implementation of the technique is to adopt common spreadsheets as abstraction and user interface for both setting up analyses and computing analysis reports.

As confirmed by our user studies, the spreadsheet abstraction has indeed the potential to enable BP analysts to perform fairly complex analyses autonomously and to effectively discuss findings with software developers, so as to iteratively improve their models. The qualitative analysis of the approach complements the user studies with a discussion of its strengths and weaknesses compared to the state of the art in BP model analysis.

The positive results we obtained encourage further extensions of the approach. Specifically, we would like to allow BP analysts to also obtain a concrete feeling of how their processes behave if deployed in a real BP system by emulating web services and visualizing process progress in a process monitoring dashboard. Comparing log data produced during the analysis of a process with real log data obtained after the deployment of the process would further enable the fine-tuning of the simulation/emulation. This, in turn, would improve analysis precision and turn the simulator into a viable tool, for example, to produce synthetic data for the testing of process mining algorithms.

5. CLOSING THE GAP BETWEEN IT AND BUSINESS STAKEHOLDERS: THE CASE OF WEB SERVICE REUSE, COMPOSITION AND ANALYSIS FOR SERVICE-BASED BUSINESS PROCESSES

It is very hard to imagine today a medium or large company that runs its business operations without IT support. More specifically, it is hard to imagine one that does not provide its own set of IT-supported services or interact with external ones in some way or another. Even a simple e-mail sent by an Acquisition Manager of a company to a service or product provider may trigger the execution of complex IT-supported, business process that helps in fulfilling his needs.

Such processes are typically implemented through a combination of so-called web services, i.e., software that expose functionalities through standardized Application Program Interfaces (APIs) via the web. The design and instrumentation of such business processes require a close involvement of both business and IT stakeholders, who have to join efforts in order to come up with a solution that fulfills the business goals in a efficient way. Two relevant stakeholders in the context of service-based business processes include the Business Process Analyst (BPA) and System Architect (SA). A BPA is a business domain expert that is typically concerned with the design and analysis of business processes that are employed to achieve a set of business goals. He usually does not have IT-specific expertise such as software engineering and development skills. A SA is the responsible for instrumenting a business process through IT. He typically has software engineering and development skills and is able to translate business process models into a software implementation thereof. A close collaboration of these two roles is considered a healthy practice that can contribute to building solutions that matter for a company.

Even though the literature (Buchwald et al., 2011) (AbuJarour & Awad, 2014) (AbuJarour & Awad, 2011) has acknowledged the existence of a gap between these stakeholders and it has proposed a number of approaches to close that gap, as of today, the problem still remains open. We ascribe this to the lack of an appropriate methodology and a set of instruments that facilitate the mutual communication and coordination of the work between these two key roles. We argue that in order to build successful business process solutions, we necessarily need an approach that allows them to convey their needs and expose their expertise to each other in an iterative manner. In this article, we specifically focus on the case of web service reuse, composition and analysis for service-based business processes. This case is an interesting one because it poses two key challenges. The first one relates to the requirement of reusing and composing existing web services to exploit existing IT resources. This is a challenging task because we need to align the implemented IT resources to the business needs. The second one relates to the analysis of the reused and composed web services. The challenge here lies in the analysis and validation of the proposed service compositions from a business perspective and the provision of feedbacks to the IT experts for the improvement thereof.

We therefore need to bring together both the BPA and SA perspective in order to address any inconsistency or missing functionality that may emerge in the process. In order to address these problems, in this article, we propose a methodology and set of tools aimed at supporting the decision-making process and enabling the information to flow between BPAs and SAs. We rely on two of our previous work, namely, Emulator (Galli et al., 2015) and CompoSWS (Vairetti et al., 2016), to provide the support for the phases we propose in our methodology. The concrete contributions of this article can be summarized as follows:

- We propose a *methodology* that facilitates the work performed by BPAs and SAs in the reuse, composition and analysis of service-based business processes.
- We extend the *Emulator* and *CompoSWS* to support the information flow between BPAs and SAs. CompoSWS focuses on the pre-calculation and recommendation of plausible relationships between services, while the Emulator provides support in the analysis and simulation of business processes built with such services.
- We propose an *integrated platform* that (1) allows the BPAs to transfer their expertise to the SA by enriching and refining the service layer and (2) assists the BPA design choices when constructing a business model using the recommendations created from the workflow layer.

- We specify six bridging requirements and strategies to exploit the two perspectives and propose a methodology that integrates both perspectives allowing for the transferring of expertise in both directions.
- We validate our approach through a *user study* that demonstrates the viability and usefulness of the integrated approach.

This chapter is organized as follows. Section **??** presents a summary of related work in the areas of bridging the BPA/SA gap and dynamic and automatic Web services workflow generation. Section 5.1 presents a motivating scenario that will be used along the chapter to explain and evaluate our approach. Section 5.2 identifies key bridging requirements and the changes in the scenario that can be done. Section 5.3 presents our approach for aligning business process models with workflow specifications and the methodology adopted. Section 5.4 presents our approach and implementations for supporting such requirements. Section 5.5 describes the experiment followed to test our approach. Section 5.6 presents an analysis and discussion of results from the bridging requirements perspective, and, finally, Section 5.7 presents our conclusions.

5.1. Motivating scenario

As our motivating scenario, let us consider the travel reimbursement processes (TR) followed by the departments of a university. In this scenario, the employees working for the university, such as the faculty members and research staff, typically need to travel to different cities and countries to attend project meetings, conferences and other business trips. As part of the rules of the university, such travels are fully organized and paid in advance by the employee who then needs to ask for the reimbursement of the associated costs. There are 3 main steps required by the university in order to successfully get the reimbursement:

(i) Before the trip, the employee must **prepare a travel authorization request**, including the budget for the trip, which needs to be approved by the head of the department.

(ii) After the trip, the employee must **prepare an expenses report** including all the expenses and the corresponding receipts and submit it to the accounting office.

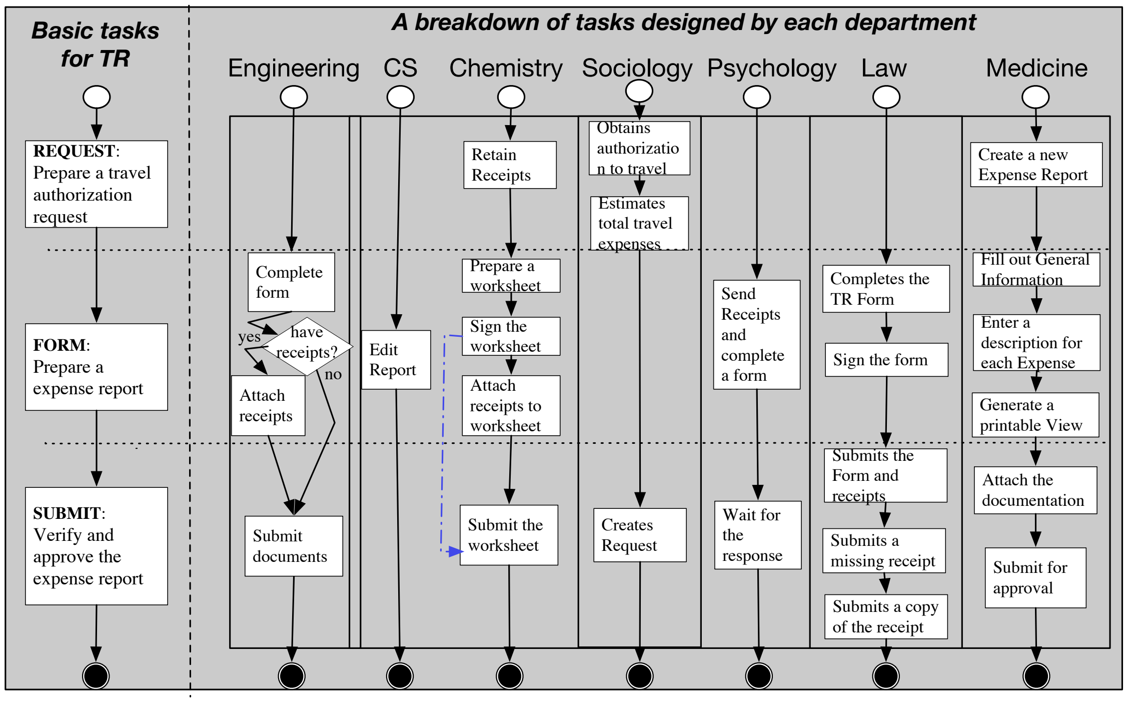
(iii) The accounting office needs to **verify and approve the expenses report** based on the university's rules. If the expenses are approved, the reimbursement is applied.

Traditionally, each department of this university has been granted the freedom to implement the corresponding business processes in order to comply with these general steps. As a result of this, over the years, each department has produced its own implementation of the process using web service-oriented technologies that are composed in a way that the above requirements are met. Figure 5.1 presents an outline of the business processes followed by each department. In this figure, we can see that there are activities coming from different departments that seem to pursue the exact same goal, such as the *Request authorization* from the Engineering and CS departments. In other cases, a single activity in a department subsumes more than one activity from another department. For example, the activity Start TR procedures from the Law department includes both the *Obtain authorization to travel* and *Estimate total travel expenses* from the Sociology department. Other similar examples can be found, but the key message we want to convey here is that the various departments follow a somewhat similar process that consists of activities that match across different departments to different extent.

Let us now consider a new mandate from the university that establishes that from next year, all the processes from the different departments must be homogenized in order to facilitate the control and auditing of the TR processes. In order to do so, the CS department has been appointed as the main responsible for the analysis of both the business processes and the web services in place and to propose a homogenized TR process. Given that each department has invested time and money in designing and implementing their own TR processes, the university has established that a key requirement for the proposed solution is that of leveraging as much as possible on the existing TR processes (and best practices) and technological implementations thereof so as to reduce the transition costs for each department. This initiative requires the involvement and joint effort of both business and IT stakeholders from the various departments in order to come up with a homogeneous business process model that complies with the university mandate and the corresponding IT instrumentation of web services that will support such model. In line with what we discussed before, we assign the former tasks to BPAs while the latter falls into the domain of SAs.

The scenario we present in this section provides a good foundation for the key challenges faced in similar, real-world situations such as the merger of companies or the reengineering of processes within large companies. Other examples include multinational companies, which have similar processes deployed in different companies with services that can be reused and adapted across different deployments, and government institutions that may require the versioning of their processes across different regions or states based, e.g., on local legislation. In the next sections, we will elaborate more on the requirements of this scenario that will serve as the starting point for the analysis of the problem and the proposed solution.

In order to frame this scenario from a technological perspective, let us consider that the tasks in these processes are implemented through single or composed (more than one) web services in a SaaS (Sofware as a Service) solution. SaaS means that an application is running on a cloud infrastructure and is accessible from various client devices through a thin client interface such as a web browser (e.g., web-based application). The consumer does not manage or control the underlying cloud infrastructure (Mell & Grance, 2011). Composed web services are workflows that result from the interconnection of more than one service following simple (sequence, alternative, etc.) or complex (parallelism, iteration, etc.) control-flow patterns such as those described in (van Der Aalst, Ter Hofstede, Kiepuszewski, & Barros, 2003b).



Basic tasks for TR
A breakdown of tasks designed by each department
Engineering
CS
Chemistry
Sociology
Psychology
Law
Medicine
REQUEST: Prepare a travel authorization request
FORM: Prepare a expense report
SUBMIT: Verify and approve the expense report
Complete form
have receipts?
yes
no
Attach receipts
Submit documents
Edit Report
Retain Receipts
Prepare a worksheet
Sign the worksheet
Attach receipts to worksheet
Submit the worksheet
Obtains authorization to travel
Estimates total travel expenses
Creates Request
Send Receipts and complete a form
Wait for the response
Completes the TR Form
Sign the form
Submits the Form and receipts
Submits a missing receipt
Submits a copy of the receipt
Create a new Expense Report
Fill out General Information
Enter a description for each Expense
Generate a printable View
Attach the documentation
Submit for approval

FIGURA 5.1. TR business processes from different departments of the university.

5.2. Requirements for closing the gap between BPAs and SAs

Sharing expertise between the person who knows about the business process (BPA) and the one who implements it (SA) is a difficult task (Galli et al., 2015). In general, (1) the BPA draws a business process model and sends it to the SA who first (2) creates or reuses web service implementations for each task of a process model or a whole workflow implementation and then (3) executes the implemented process (4) obtaining some results that need to be validated by a BPA. Steps (1) and (4) are performed by a BPA and steps (2) and (3) by a SA.

Discovering Web services and automatically building workflows based on a set of requirements is a well established research topic that reduces the SA effort to choose and interconnect Web services that satisfy a BPA request. This typically results in the automatic discovery of workflows that satisfy current and potential needs. It is therefore possible under this scenario to suggest, to the BPA, business process model design alternatives that are enabled by the technological infrastructure but were not considered when modeling the business processes. Whether such recommendations make sense or not depend on the business constraints defined by the current business context. Let's examine the effect of this feature in our motivating scenario.

5.2.1. Changes in the scenario

According to what we discussed before, step (2) is performed after the BPA draws a business process model and sends it to the SA. The SA, in turn, either creates or reuses Web service implementations for each task of a process model. To do so, the SA needs to identify services from a services workflow repository, validate the services' signature (inputs, output and goal) and choose the proper service for each task in the BP model. In the example shown in Figure 5.2, we can see that each task has an implementation of a web service that is stored in a repository. For instance, the task *EditReport* from the CS department is implemented by service WS1. The service has only one input (*receipts*), one output (*approveReimb*) and one goal (*obtainReimb*).

If we, playing the role of a BPA from the Law department, want to model a new TR business process, we can leverage on the knowledge that stem from the various departments of the university. Assuming that our goal is to obtain a reimbursement (*obtainReimb*), the plausible recommendations are the CS atomic service WS1, the Chemistry workflow comprehending services WS2 to WS6, or the Psychology workflow comprehending services WS7 to WS8. According to Figure 5.1, CS basic process lacks the request and submits process, whereas Chemistry process implements the three main steps. Psychology, instead, implements the form and submits steps, which is very similar to the Law process. However, Law differs from Psychology in the detail of its steps. Thus, Law may require the implementation of additional services since existing services (WS7 and WS8) lack the internal tasks required by Law. Law contributes to the community with a new alternative of implementing the TR steps.

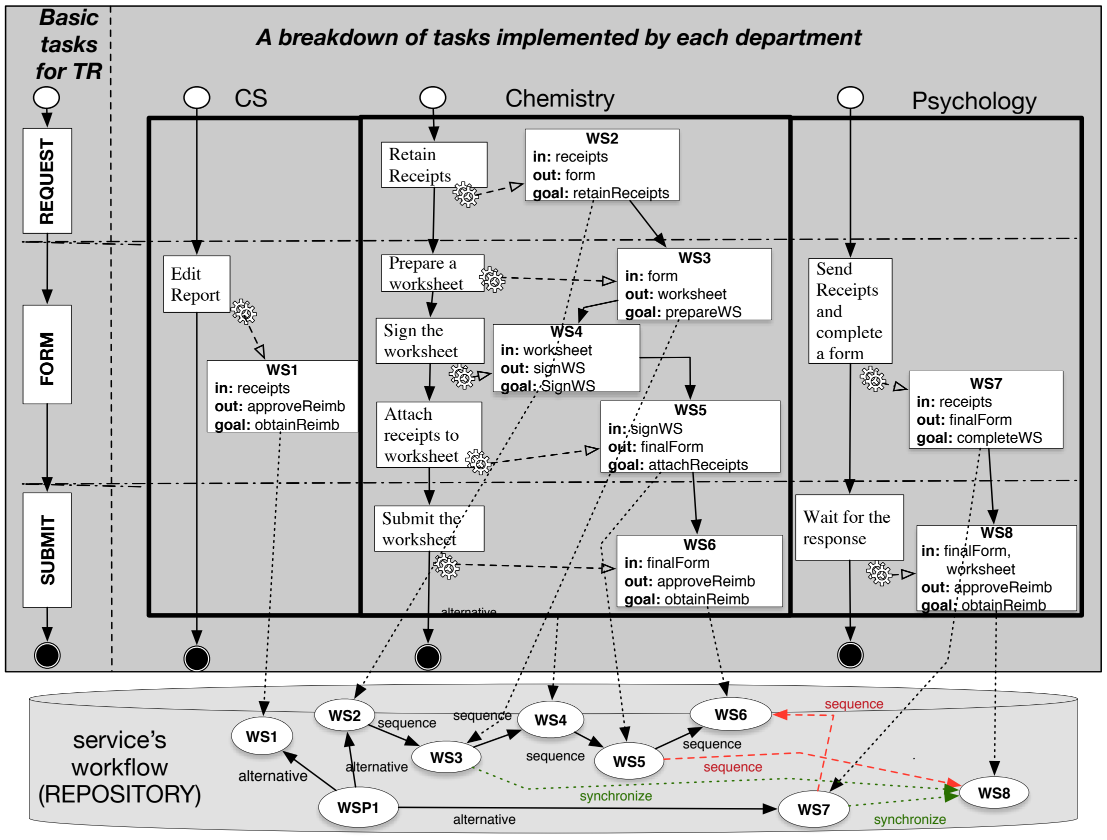

Basic tasks for TR
A breakdown of tasks implemented by each department
CS
Chemistry
Psychology
REQUEST
FORM
SUBMIT
Retain Receipts
WS2 in: receipts out: form goal: retainReceipts
Edit Report
Prepare a worksheet
WS3 in: form out: worksheet goal: prepareWS
Send Receipts and complete a form
Sign the worksheet
WS4 in: worksheet out: signWS goal: SignWS
WS1 in: receipts out: approveReimb goal: obtainReimb
WS7 in: receipts out: finalForm goal: completeWS
Attach receipts to worksheet
WS5 in: signWS out: finalForm goal: attachReceipts
Submit the worksheet
WS6 in: finalForm out: approveReimb goal: obtainReimb
Wait for the response
WS8 in: finalForm, worksheet out: approveReimb goal: obtainReimb
alternative
service's workflow (REPOSITORY)
WS1
WS2
WS3
WS4
WS5
WS6
WS7
WS8
WSP1
sequence
alternative
synchronize

FIGURA 5.2. Implementation of the TR business model as a service workflow for various universities.

5.2.2. Bridging BPA and SA key requirements

There are essentially two bridging directions: the first one refers to transferring expertise from the BPA to the SA, and thus, to the workflow implementation. The second direction is that of transferring workflow knowledge from the SA to the BPA, by leveraging on the knowledge that stems from the workflow implementation. In the former direction, the transfer occurs either in the typical case in which the BPA passes the BP model to the SA for its instrumentation, or, when the BPA provides feedbacks to the SA about the implementation of a process model. In the latter direction, the SA can learn from the workflow implementation, e.g., through available service discovery or composition features of the platform, and then transfer this knowledge to the BPA in the form of new BP model designs. Below, we present a list of requirements that need to be considered so as to build the bridge between these two roles:

5.2.2.1. Transferring business knowledge from BPA to SA and workflow platform.

BR1: Extending BP model implementation based on the feedback provided by the BPA.
Connections that are not evident in the workflow can be derived from the BP models created by the BPA; some services should be connected at the business level even though at the software level they cannot, e.g., because data transformations are required. In addition, let's suppose that the BP is analyzed generating an execution log that indicates the most frequent path followed by the end user. This can be done because the simulation is based on actual business constraints determined by the BPA.

BR2: Identifying inconsistencies between the workflow implementation and the BP model.
Connections that are conceptually inconsistent at the business level can also be detected. It is well known that users perform workarounds when the software does not match the actual business process. The identification of mismatches like these can be very important for a company, e.g., because it may allow for the prevention of situations of non-compliance with the law.

5.2.2.2. Transferring implementation knowledge from the workflow platform to the SA and BPA.

BR3: Assisting SAs in identifying Web service candidates for reuse. As discussed in Section **??**, in CompoSWS is possible to determine the most suitable candidates to be associated with a service from the service's signature (i.e. input, output and goals). Sometimes a workflow fragment (a composed service) instead of a single service is required to implement a task.

BR4: Recommending plausible alternatives in the design of BP models based on the workflow platform and existing web services.

Similarly, if the BPA adds a task to the BPM model, the service infrastructure can be used to recommend single or composed services that may add business functionality (or alternative implementations with various QoS) not envisioned by the BPA but already available in the platform. Thus, the deployed capabilities can be recommended to the BPA in order to improve the BPM models.

BR5: Recommending plausible alternatives in the design of BP models based on their execution costs.

As discussed before, BP analysis may be implemented through simulations. In this case, a business process log obtained during the analysis contains information about its completion time, which goes beyond computational execution time (e.g., the verification of a document from a clerk takes two business days to complete). This information can be used to enrich the BPM and its implementation. The BPA can analyze the costs associated to a process and determine whether or not to make changes towards greater efficiency and lower cost.

BR6: Identifying inconsistencies between BP models and its implementation as a workflow.

The workflow platform, enriched with the BPA expertise, captures business knowledge that can prevent other BPAs to introduce business inconsistencies due to new requirements, at different time frames. In addition, it is possible to prevent changes that could have negative impact on other areas of the business.

5.3. BPM-SIC: Business Process Model - Service Implementation Collaboration methodology

This chapter presents a flexible approach for aligning business process models with workflow specifications, so as to maintain the complex dependencies that exist between high-level business process models (as used by BPAs) and technical workflow specifications (as used in IT departments by SAs). We propose a methodology that integrates these two perspectives: A modeling layer allows BPAs to transfer their expertise to the SAs, and, vice versa, through recommendations built from the workflow implementation and integrated by the SA, assist BPAs by providing design alternatives when constructing a business model.

In order to do that we combine two tools: CompoSWS (Vairetti et al., 2016) and the Emulator (Galli et al., 2015). CompoSWS pre-computes all the possible relationships between the services so as to provide recommendations on the possible process design alternatives, while the Emulator focuses on the analysis and simulation of business processes.

5.3.1. CompoSWS server

CompoSWS (Vairetti et al., 2016) is an approach for dynamic service composition that exploits service signature and semantic annotations along with rules to identify simple and complex control-flow patterns between services at publishing-time. Services are connected through such patterns forming a graph that is pre-calculated and represent the behavioral semantics of a potential composed service. A composed service can be dynamically and automatically discovered and assembled into an executable service at consuming-time.

Services' signature can be used to determine both service components and the dependencies among them. Typically such dependencies are simple sequence and alternative control-flow patterns (e.g. consume service A first in order to produce and output that serves as an input for the subsequent service B). However, in the real world, follow complex control-flow patterns in order to fulfill the requirements and constraints of real world business processes (Ter Hofstede et al., 2009). .

In this approach we extended a well-known and simple semantic Web service ontology (MSM, the minimal service model), with minimal concepts and relationships that make possible to represent relationships among services corresponding to complex control-flow patterns. Such relationships are inferred from the dependencies between services signature such as Input, Output and Goal. In this way it is possible to discover composed services as subgraphs where services are interlinked following complex control- flow patterns. The control flow relationships between services considered are sequence, alternative, synchronize, discriminator, select, and iterator and they are briefly described below.

(i) A `sequence pattern` models the dependency between two services that pursue different goals; one service produces the output to be used as an input for the subsequent service. Both services are linked through an SB:SEQUENCE relationship operator.

(ii) A `alternative pattern` is identified when two services share the same goal and output, and they will be invoked depending on certain condition. Such services are linked through an SB:ALTERNATIVE relationship.

(iii) A `synchronize pattern` is inferred when the inputs of a service can be obtained from the outputs of other services, and not a single service can provide all the inputs. The sb:synchronize relationship links all the involved services.

(iv) In a `dicriminator pattern` is identified when two services share the same goal and provide the same output, but only the output of the first service providing a response is considered. Services are linked with an SB:DISCRIMINATOR relationship.

(v) A `select pattern` is similar to a discriminator, but a condition is applied to choose one of the services output. The relationship in this case is SB:SELECT

(vi) The `iteration pattern` occurs when a simple or composed service is executed more than once. The relationship in this case is SB:ITERATOR.

CompoSWS is a Web service composer that follows a two sides approach: service *publisher* and the service *consumer*. When a new service is published in the platform (or in the cloud if we use SaaS technology), the system pre-calculates all the possible relationships between the services through a *Connect* algorithm described in (Vairetti et al., 2016). The resulting graph includes the service's goals, input and output characteristics, at the semantic level, including the presented control-flow patterns and rules.

When a SA requests a service, the system looks for an existing service. If no service can be found, the system finds a graph fragment that satisfies all or most of the SA requests. The graph fragment is a set of interrelated services containing all or most of the information provided by the user (input), called *origin* nodes; and containing the expected goal and result (output) required by the user, called *target* nodes. The latter task is accomplished by executing a *FindService* algorithm described on (Vairetti et al., 2016). If the SA approves the proposed service, the graph is used as the behavior (control-flow) of a composed service, which is created, deployed, and published later in our system.

In the bottom part of Figure 5.2 we can see possible Web services implementations of the TR business process of three departments: CS, Chemistry and Psychology. In this case, CompoSWS, a SaaS cloud-based Web services workflow repository stores the services from various institutions and their connection to the BP tasks. CompoSWS is capable of exploiting the process knowledge implicitly derived from the service's workflow in order to infer possible control-flow relationships among them and recommend them.

Let's suppose that at the beginning, WS1, which implements CS's task *EditReport* is uploaded to CompoSWS and then, WS2 that implements Chemistry's *RetainReceipts* task is uploaded. CompoSWS will analyze service's signatures and will detect that both services pursue the same goal and have the same output. They will be considered as *alternative* implementations of the same goal that can be consumed according to the evaluation's result of some condition; this behavior corresponds to the *discriminator* pattern. In this case, an abstract service WSP1, which is a superset of both services, will be created. If service WS3 (*prepareWS* task) is uploaded to CompoSWS, the *sequence* pattern will be detected since WS2's output (*form*) is the input of WS3 and both goals are different. Something similar will occur when uploading the remaining implementations of Chemistry's task, namely WS4, WS5, and WS6. However, when uploading service WS7 corresponding to PUC's task *completeWS*, CompoSWS will detect the existence of another *alternative* to WS2 and WS1 and will link WS7 to the abstract service WSP1.

5.3.2. Emulator analyzer

The Emulator (Galli et al., 2015) is a spreadsheet-based approach for business process model analysis. This approach enables BPAs to simulate BP executions and generates process execution logs in order to define own metrics, assertions and obtain analysis reports.

The overall approach takes into account the fact that there are tasks that require specific technical skills that BP analysts may not have and that may prevent them from being able to analyze BPs.

BPMN is used to model executable BPs. The spreadsheet is used as interface toward the BPA and as communication instrument between the analyst and the developer. Simulation is used to safely generate behavioral information for those web services of the service-based BP that may have persistent side effects in the system or do not yet have a readily usable implementation.

The approach uses BP simulation for the generation of the *observed behavior* of the process (van der Aalst, Nakatumba, et al., 2008). BP simulation allows us to safely and easily generate different process execution scenarios to test different expected behaviors. The combination of the ease with which spreadsheets can be used to analyze data and the ability to quickly generate different process execution scenarios through BP simulation provides the BPA with a powerful instrument for BP testing.

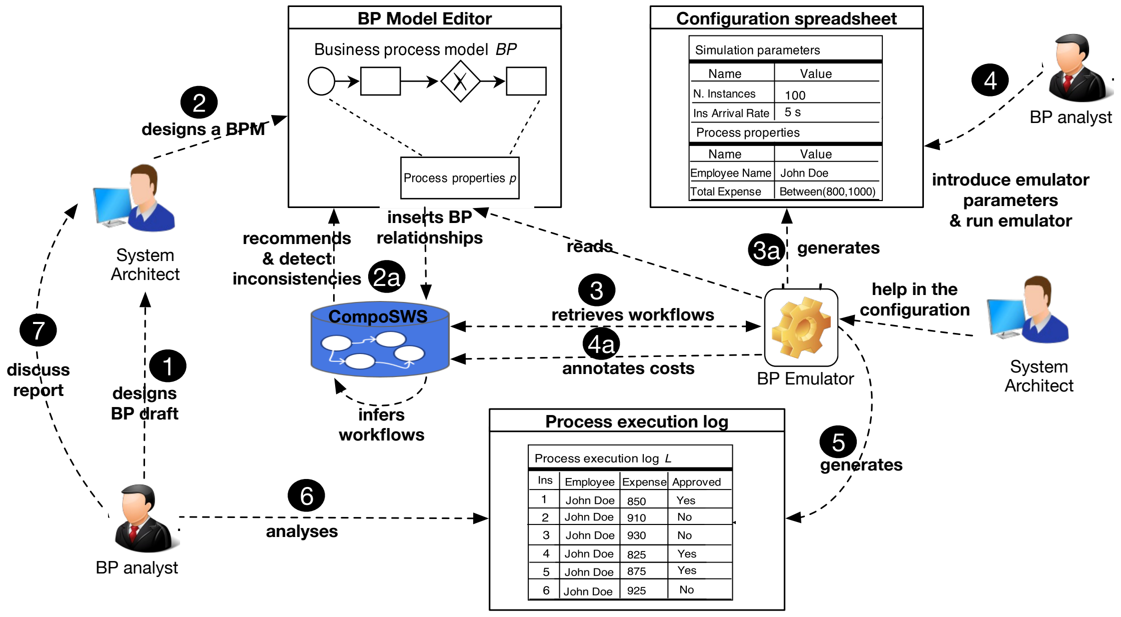
Starting from a *draft of* a BP model, the SA refines and implements the process using an extended *BPMN editor*. Setting up a BP analysis requires also the configuration of the simulation to be performed. Activities of BP that refer to real services are marked as such and pre-configured by the developer in the extended BPMN editor; the BPA can configure the behavior only of simulated services. Once the simulation is configured, the *BP simulator* reads the configuration and BP and runs the simulation, mimicking the web service behaviors defined by the BP analyst and invoking existing web services. As a result, it generates a process execution log that contains the observed behavior, which is again stored as a *process execution spreadsheet*. The BP analyst verifies and analyses processes performance using an analysis *analysis spreadsheet*.

5.3.3. BPM-SIC methodology

Our methodology for bridging the roles of BPA and SA is represented in Figure 5.3. The methodology advocates for a back and forth interaction between these two roles so as to share their expertise and domain knowledge when reusing, composing and analyzing workflows and their associated web services. Starting from a *BP model draft* (step 1) designed by the BPA, the *SA* refines and implements the process using a *BPMN editor* (step 2). When a task is added to the editor, a query is sent to CompoSWS in order to ask for services (or workflows) that could serve as the task implementation (step 2a). CompoSWS will recommend services based on its knowledge (similar workflows, BPs, and costs). If the BPA has identified a relationship that does not exists in the CompoSWS platform, such relationship is enforced and created between the corresponding services (i.e., a + relationship). If no service implementing the task is found, CompoSWS creates a service description with no implementation. Whenever a new service or relationship is uploaded to CompoSWS, the platform infers plausible, new workflows that can be created considering the new situation.

Once the BP model is finished, the SA runs the BP emulator in order to generate a *configuration spreadsheet* (step 3a). During this step, the BP emulator queries the workflows stored in CompoSWS and associated to the BP created by the SA (step 3). For each workflow, it creates an entry in the configuration spreadsheet containing the process properties and a set of simulation parameters.

Once the configuration spreadsheet is created, it is send to the BPA who introduces simulation parameters to configure the simulation dynamics (step 4). They include parameters such as the number of process instances to be emulated, the rate at which instances are to be generated, and the execution time of simulated web services. Process properties P are used to configure business data for different process execution scenarios; they are associated to the BP tasks and may refer to both real or simulated web services. Once the spreadsheet is configured, the BPA runs the simulation, mimicking the web service behaviors defined by



Simulation parameters	
Name	Value
N. Instances	100
Ins Arrival Rate	5 s
Process properties	
Name	Value
Employee Name	John Doe
Total Expense	Between(800,1000)

Process execution log *L*

Ins	Employee	Expense	Approved
1	John Doe	850	Yes
2	John Doe	910	No
3	John Doe	930	No
4	John Doe	825	Yes
5	John Doe	875	Yes
6	John Doe	925	No

FIGURA 5.3. BPA and SA bridging methodology.

the BPA and invoking existing web services. This action consists in calculating the workflows execution costs, which are fed to CompoSWS to update its graphs (step 4a). The BP emulator produces also an execution log (step 5) recording execution paths costs as well as inconsistencies and CompoSWS recommendations to be analyzed by the BPA (step 6). This analysis may create changes in the BP model, which are discussed with the SA (step 7).

When a SA requests that a service or task is added to the editor, the system looks up for an existing service and if no service can be found, the system finds a graph fragment that satisfies all or most of the user's requests. If the SA approves the proposed service, the graph is used as the behavior (control-flow) of a composed service, which is created, deployed, and published later in the system. In order to determine a graph fragment costs we use the Emulator tool.

In summary, we complement the Emulator with CompoSWS that is capable of discovering alternative workflows from services' signature but also use the Emulator to analyze workflow recommendations. Thus, the Emulator and CompoSWS are jointly used to assist in closing the gap between BPA and SA and to foster the interaction between these two roles.

5.3.4. Key requirements examples

In order to better understand each of the presented requirements and, considering the previously defined scenario, in this section we will show some real examples our applying the BPM-SIC methodology.

5.3.4.1. Transferring business knowledge from BPA to SA and workflow platform.

BR1: Extending BP model implementation based on the feedback provided by the BPA.

Let's assume that the department of Chemistry decided to modify the diagram shown in Figure 5.1 in order to allow the user to directly submit the signed worksheet rather than requiring him to attach the receipts. This change is reflected in the process model by an arrow generated from the Sign the worksheet task to the *Submit the worksheet* task.

In addition, let's suppose that after running Missouri's tasks *Complete form* and *Attach receipts*, the execution log demonstrates that users prefer not to attach receipts frequently. In this case, the BPA may dialogue with the SA to request a new verification service in order to determine whether the requested reimbursement amount is acceptable.

BR2: Identifying inconsistencies between the workflow implementation and the BP model.

In Figure 5.2 we can notice a connection between services WS5 (*attachReceipts* goal) and WS8 (*obtainReimb* goal); this occurs since CompoSWS detects that WS5 output (*finalForm* goal) is an input of WS8. Also, WS8 is an alternative implementation of WS6 (*obtainReimb* goal). Even though at an implementation level such relationships are possible, at the business level, the Chemistry's BPA must evaluate whether to reuse Psychology's service WS8 or using service WS6 which falls under its business scope. In the latter case, even though both services coexist, Chemistry's BPA must consider the relationship between WS5 and WS8 inconsistent at the business level.

5.3.4.2. Transferring implementation knowledge from the workflow platform to the SA and BPA.

BR3: Assisting SAs in identifying Web service candidates for reuse. Solutions such as CompoSWS can automatically calculate workflows and recommend it to the SA, facilitating his tasks and fostering service reuse. For example, during the implementation of Psychology's BP model, CompoSWS will recommend either service WS7 or a composite of services WS3, WS4 and WS5 for implementing task 1 (*Send receipts and complete form*).

BR4: Recommending plausible alternatives in the design of BP models based on the workflow platform and existing web services.
In our example, the platform can recommend to the department of Chemistry the possibility of ignoring the tasks associated to services WS3 to WS5 and instead implementing a BP similar to Psychology's since the composed service WS7, WS8 has the same input, output and goal of a composition including WS2, WS3, WS4, WS5 and WS6 (i.e. ignoring the internal operations).

BR5: Recommending plausible alternatives in the design of BP models based on their execution costs.
Let's suppose that in our scenario, the task associated to CS's WS1 has less performance than the task associated to the Psychology's composite WS7 and WS8. Since both satisfy the same goal, CS's BPA may decide to mimic Psychology's BP. Similarly, the SA shall receive the recommendation of reusing WS7 and WS8 and shall implement CS's task *Edit Report* as a service composite.

BR6: Identifying inconsistencies between BP models and its implementation as a workflow.
At an implementation level, an automatic composition tool such as CompoSWS may detect that WS8 requires two input parameters (*finalForm, worksheet*) that

can be provided by services WS2 to WS3 (*worksheet*) and services WS2 to WS5 (*finalForm*) or WS7 (*finalForm*). Let's suppose that the user chooses the implementation of services WS2 to WS3 that will be executed in parallel to WS7, since it requires a shorter execution time. At certain point in time, WS7 may fail and an alternative implementation should be chosen, in this case, WS2 to WS5. Hence WS2 to WS3 will be executed in parallel to WS2 to WS5 (the former is a subset of the latter), such possibility is detected as a inconsistency since WS2 to WS3 will be executed twice. Hence, WS2 to WS3 in parallel to WS7 is not a valid workflow for CompoSWS.

5.4. Implementation

We extended the Emulator and CompoSWS to support the information flow between BPAs and SAs. CompoSWS focuses on the pre-calculation and recommendation of plausible workflows, while the Emulator provides support in the analysis and simulation of business processes built with such services. Figure 7 shows the integration of both approaches. Both architectures complement each other and use the same service repository. The extensions made to both architectures in order to achieve the proposed integration are described below.

5.4.1. Extending CompoSWS

CompoSWS is a Web service composer that acknowledges the service publisher role and the service consumer role. For the former case CompoSWS pre-calculates all possible workflows whenever a new services is available, so that, when the consumer requests a composition (or workflow), the chances of finding an already identified workflow are higher.

Figure 5.4 summarizes the major architectural components of CompoSWS. The publishing process (step 1) requires that the provider uploads a service description containing annotations related to the service goal and data types (inputs/output). The description is transformed (step 2) into a SPARQL expression and stored in a repository. Then, the service description is analyzed (Control Flow Analyzer) to determine plausible workflows (step 3) that follow various control-flow pattern relationships. These relationships become new triples that are stored in the repository. A client request (e.g., an XML document) includes an expected goal and output, and provides some inputs and guard conditions. The client request is processed by the Service Discovery component (step 5), informing the client if an existing service provides a solution (step 6), otherwise it creates a workflow through the Service Composer component (step 5a). If the composed service (workflow) is approved, a new workflow is created and stored in the triplestore (step 7b).

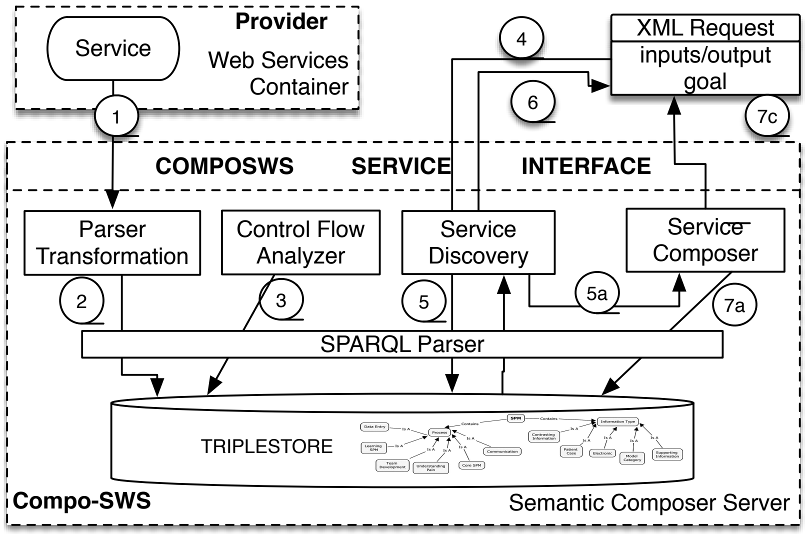


FIGURA 5.4. CompoSWS architecture.

The basis of CompoSWS is an ontology designed as a simple extension to the MSM service ontology as shown in Figure 5.5. Circles represent concepts (e.g., *msm:Service*), arcs represent relationships (e.g., *msm:hasInput*) between concepts, and the squared rectangles represent literals (e.g., *xsd:string*) (Figure 5.5). The *sb* namespace refers to elements modeling service behavior, whereas msm represents the MSM ontology elements. A service goal (*sb:Goal*) represents the activity performed when executing a service, according to a domain specific ontology. The goal is related to the service through an *sb:hasGoal* relationship. Service composition may be restricted according to certain constraints or guard expressions *sb:hasExpresion*, and services are related to other services through relationships that represent the semantics of control-flow patterns *sb:patterns*. Control-flow patterns considered are: *sb:sequence, sb:alternative, sb:synchronize, sb:discriminator, sb:select* and *sb:iterator* as detailed in section 5.3.1.

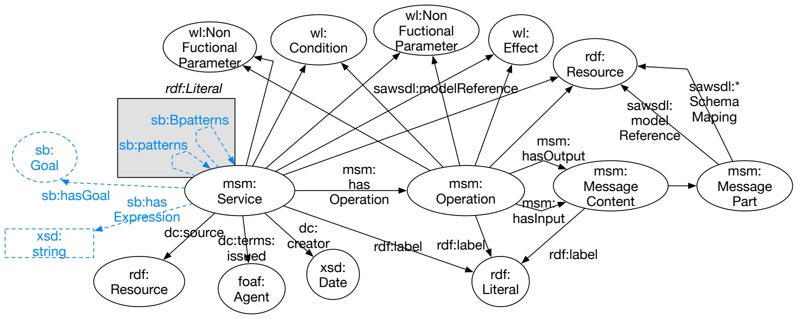


FIGURA 5.5. CompoSWS extended ontology supporting BPM-SIC strategy.

In order to face the BPM-SIC methodology we extended CompoSWS ontology as detailed in Figure 5.5. We extended the proposed ontology with relationships identified at business level (*sb:Bpatterns*), which are variations of the original control-flow patterns (i.e., we support the same patterns but annotated with the + symbol). Triples of the form ⟨*msm:service, sb:pattern* || *sb:bpattern, msm:service*⟩ are annotated with a literal (in this case a number) indicating the workflow fragment cost (e.g., 1200 milliseconds) or inconsistencies at the business (using the the *?* symbol followed by the BP model id) and implementation level (using the * symbol); hence, with this additional annotations, they become quads (Harth & Decker, 2005). Examples of the produced quads are presented in table 5.1. For readability purposes, we present the quad elements including a namespace prefix. This approach allows for higher flexibility in querying data, improving performance and providing the possibility to save a weight between services. The former version of CompoSWS is not capable of annotating relationships or triples so that it cannot assign a weight to fragments of the workflow which is a requirement generated by the business process analysis. Likewise it makes impossible to annotate relationships or triples to indicate an inconsistency or to force a control-flow relationships detected at a business level. This extension

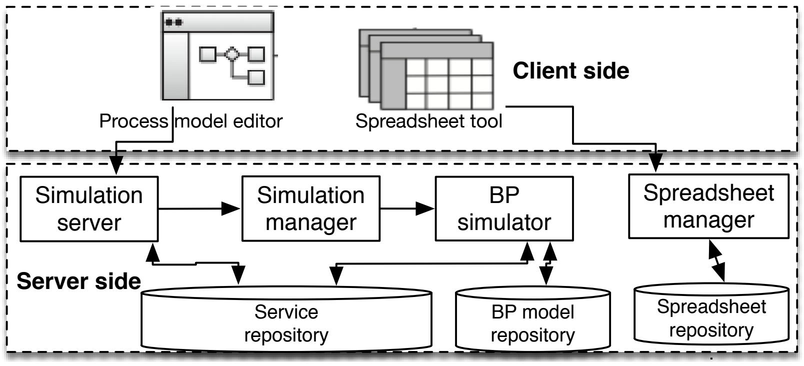


FIGURA 5.6. Emulator architecture.

required modifying all the original queries and algorithms in order to support quads and workflow paths weights (the forth parameter) when pre-calculating workflows.

5.4.2. Emulator server

Figure 5.6 illustrates the principal architectural components of the Emulator tool. On the client side, we have a *process model editor* used to design BPs and a *spreadsheet tool* used to configure the simulation and perform the BP analyses. When a process model is ready for simulation, it is uploaded into the Simulation Server, which configures the simulation environment, and performs three main tasks: First, it creates or selects the required services, saving new services in the repository; then it stores the BP model into the corresponding repository. Finally, it requests the *spreadsheet manager* to create spreadsheet templates and stores them into the *spreadsheet repository* for reuse in the following phases. The Simulation Manager deploys the BP into the BP simulator, which runs the simulation.

The process execution data is stored back into the spreadsheet repository and made available to the spreadsheet tool for analysis. The current implementation uses Signavio[1] as process model editor, Google Spreadsheets[2] as the spreadsheet tool, and Activity[3] as the internal engine of the BP simulator.

[1]http://code.google.com/p/signavio-core-components
[2]www.google.com/App
[3]http://activiti.org/

5.4.3. Extensions made to support the key requirements.

In order to support the extension of the BP model implementation based on the feedback provided by the BPA (BR1), we enforce a business level relationship (*sb:Bpatterns*) between services following the control-flow pattern established in the BP model. Hence, relationships, such as *sequence+, synchronize+, discriminator+, select+, alternative+, or iterator+*, between the services are supported. For example, a BPA may indicate that two services with slightly different goals such as *atachReceipts* and *completeWS* can be invoked in parallel following a synchronizing, conditional merge pattern. We relate such services following a *synchronization+* relationship.

In table 5.1 we can visualize an example of *sb:Bpatterns*; *msm:Service1* is a sequence that is not detected for the platform, *sb:sequence+* relationship with service *msm:Service3* was indicated by the BPA and stored in the platform on the triplestore (step 2). In addition, as mentioned in Section 5.3.3, after performing a BPM analysis, the logs are examined to determine the workflow paths' cost. The services involved in such paths are marked with a number indicating its priority which is stored in the forth parameter of a quad (subject, predicate, object, weight). Such information will serve for the platform to determine deployment properties (e.g., replication, allocated resources, among other properties).

TABLA 5.1. Quad examples for CompoSWS extension.

Subject	Predicate	Object	Context
msm:Service1	sb:sequence	msm:Service2	”1200”^^xsd:string
msm:Service1	sb:sequence+	msm:Service3	”980”^^xsd:string
msm:Service4	sb:synchronize	msm:Service3	”?bpm12”^^xsd:string
msm:Service3	sb:synchronize	msm:Service7	”*”^^xsd:string

Whenever the BPA can identify inconsistencies between the workflow implementation and the BP model (BR2), a copy of the offending relationship is created and annotated in the forth element of the quad to indicate the inconsistency (using a symbol *?* followed by the BPA's BP model id). This way, the relationship is marked as inconsistent only in the BP model of such BPA. When such workflow fragment is recommended to other BPAs, they are notified of the inconsistency so they can evaluate whether it applies to their

BP model or not. For instance, transparency laws may require that governmental funded institutions make all its expenses publicly available, so that, a public university may require that teachers' expenses bills become available, while this may be unnecessary for private institutions. An implementation example is presented in table 5.1: *msm:Service4* is an alternative service in relation to *msm:Service3*, but there is also another BP model in the repository (*bpm12*) where such relationship is considered inconsistent at the business level.

The BR3 requirement: Assisting SAs in identifying Web service candidates for reuse is supported in a straightforward way by CompoSWS, since the SA only requires to specify the composition requirement (goal, input and output) so that CompoSWS finds a single service, an existing workflow or creates a new workflow to satisfy such requirement.

In order to face BR4 request (Recommending plausible alternatives in the design of BP models based on the workflow platform and existing web services), CompoSWS can identify the various workflows that implement a request and suggest such workflows to the BPA, so that he can enrich his BP model with alternative tasks and paths. In this case, a BPA can extend the BP model in Signavio.

Since relationships between services are weighted, it is possible to calculated the cost of a workflow path and determine a workflow priority when CompoSWS recommends services to the BPA and SA. This means that it is possible to support BR5 (Recommend plausible alternatives in the design of BP models based on their execution costs).

Recall that inconsistencies may arise when the platform can detect potential problems at an implementation level that may have an impact at business level (e.g., a workflow subset may be chosen to implement an alternative path of the whole workflow). The workflow fragments that are inconsistent according to CompoSWS are marked with a * symbol. Every time a BPA creates a model including a path marked as inconsistent at the platform level, the platform will notify the BPA and SA about it. Inconsistencies at this level comprehend potential wrong patterns such as cycles between services, impossibility of implementing sequences, since they require additional information, or incomplete fragments

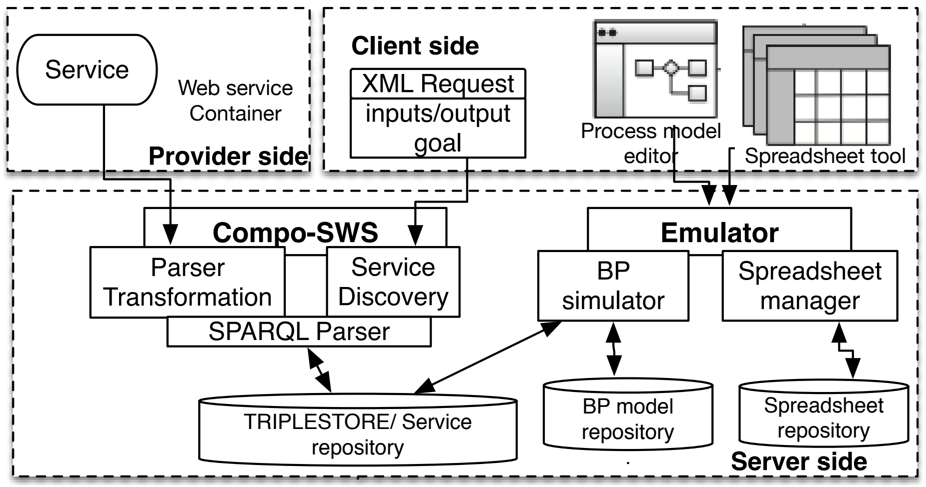


FIGURA 5.7. BPM-SIC architecture.

such as the example of WS7 in section 5.2. Hence, BR6 (Identifying inconsistencies between BP models and its implementation as a workflow) is also supported.

5.5. Experimental setting

In order to test the BPM-SIC methodology we created a testing scenario including BP experts in an educational domain. For the case of the BP models, we created a dataset including 7 BP models for the TR process that are inspired by publicly available descriptions of their practices by the universities of Missouri, Trento, Minnesota, Memphis, PUC (general process), Place and Northwestern. We also created 15 BP models generated from direct interviews with users in charge of processing reimbursement requirements at three Universities in Chile, PUC (specific per department, fund type and labs), Santa Maria and Los Andes as can be seen in Figure 5.1.

For the workflow platform, we used an existing dataset of 250 Web services focused on the educational domain (Klusch, 2012), which included task implementations of TR process.These services are implementations of the tasks in the BP models related to the TR process and are stored in a common repository to be accessed by CompoSWS and the Emulator tools. CompoSWS recommends workflows based on these services and the Emulator analysis is based also on such implementation.

We ran a study to validate the viability of the proposed approach. The study followed the methodology proposed in section 5.3.3, however, we separated the methodology into three steps in order to clearly identify the effects of the recommendation on the BPAs. It included an interview with 15 BPAs to *generate BP models* drawn by hand (step 1 in our methodology), a second interview was performed to gather data in order to run a *business process analysis* using the spreadsheet tool described in (Galli et al., 2015) (step 4 in our methodology). A third interview with BPAs included the validation of the business process analysis logs as well as the recommendations made by CompoSWS, implementing this way the two-way bridge proposed in section 5.3.3 (step 6 in the methodology). In summary, the user study was performed considering a set of 45 working sessions between the BPAs and the SA/platform.

5.5.1. First interview: BPM identification

The objective of this study was to understand how the reimbursement process works in each academic unit of the different universities. The participants of the study were 15 employees of various academy units from 3 different universities, as described in table 5.2, who operate as BPAs. Each BPA participated separately in a 40 minutes session. All participants were familiar with spreadsheets; only 9 of them knew Google Spreadsheets. Seven of them had a background in Computer Science. The user study was organized as a semi-structured interview (Lazar et al., 2010).

The participants were introduced to a small explanation of the purpose of the experiment, they were asked about the TR process and during the following conversation we gathered data characterizing the BPs, and playing a facilitator role, we drew by hand the BP models together with the BPA using BPMN.A list of requirements was recorded in audio and paper notes were taken in this section.

TABLA 5.2. Participants to our study

University	Unit	Total
PUC	Mechanical Engineering	1
	Industrial Engineering	2
	Construction Engineering	1
	Computer Science	1
	Chemistry and Bioprocess Engineering	1
	Engineering Design Laboratory (DILAB)	1
	Education	1
	Agronomy	1
	Semantic Web Research Center (CIWS)	1
Los Andes	Research and Postgraduate Office	2
	Engineering	1
Santa Maria	Computer Science	2
TOTAL		15

5.5.2. Second interview: Business analysis

Playing the SA role, we processed the BP models gathered in the first interview before interview 2 took place. Considering the requirements list, designed the BP models in Signavio BPM editor and fed the BP models to CompoSWS. The resulting implementation is shown in table 3. At this stage, we ignored CompoSWS alternative BPM implementations (workflows). From the generated workflows, we created the configuration spreadsheet for each BP model (steps 2, 2a, 3, 3a in our methodology).

Table 5.3 presents 7 columns showing the participant's identification, the number of workflow generated in the interview, the number of services that was obtained form the modeled workflow, the number of connections between those services and the last 3 columns represent the number of require/synchronize/discriminator patterns generated from the CompoSWS repository.

During the interview, the participants were presented with a small explanation of the purpose of the experiment and the spreadsheet-based tool for performing a BP analysis. At this point, we started a discussion so as to know what problems were present in the models, or the most frequent delays. Thus, the study also aimed at understanding whether the approach facilitates the discussion of findings between the BPA and the SA.

TABLA 5.3. BPA's workflow implementation for each BP model as created by CompoSWS

Participant	#workflow	#services	#connections	#require	#synchronize	#discriminator
1	1	5	4	1	0	0
2	1	5	5	3	2	0
3	1	7	6	5	0	2
4	1	4	5	3	2	0
5	1	9	10	8	0	2
6	1	7	6	6	0	0
7	1	4	5	3	2	0
8	1	5	6	4	0	2
9	1	8	7	7	0	0
10	1	7	8	6	0	2
11	1	6	7	5	2	0
12	1	5	4	4	0	0
13	1	5	6	4	0	2
14	1	5	6	4	0	2
15	1	8	11	7	2	2
TOTAL	15	90	96	70	10	14

Participants were asked questions in order to characterize the BPs (step 4 in our methodology). We started with a discussion of the values of the tasks' input data, expected output, task duration, and delays they knew about when the processes are executed. For instance, if the BPA indicated that simple reimbursements include values between 1.000 and 3.500 US dollar, we configures such behavior in the spreadsheet tool so that the Emulator generates random values within, in this range.

5.5.3. Processing the information

As described before, we uploaded 7 BP models corresponding to the TR process as described in figure 5.2. Then, with the data gathered from the first interview, we created a Web service description per task and participant. This resulted into 15 additional BP models that were fed into CompoSWS, for a total of 22 models. The platform interconnects the services using control-flow patterns, resulting in workflows that implement each participant's BP model as well as alternative implementations that are recommended to the BPA. The recommendations for alternative workflows made by CompoSWS (step 2a in our

TABLA 5.4. CompoSWS recommendations to SA and BPA based on the BPM

Participant	#model	# recommended workflows (BR3, BR4)	#services	#inconsistencies detected by CompoSWS (BR6)	#inconsistencies detected by the BPA (BR2)	Enforced relationships at the BPM level (BR1)
-	7	-	43	2	-	-
1	8	3	17	3	1	1
2	9	5	21	3	1	0
3	10	5	22	2	0	3
4	11	7	31	4	2	2
5	12	5	22	3	1	0
6	13	6	27	1	1	0
7	14	8	35	1	0	1
8	15	8	37	3	1	1
9	16	8	35	1	0	0
10	17	7	32	2	0	0
11	18	9	36	1	0	0
12	19	9	38	2	0	0
13	20	11	45	1	2	0
14	21	13	52	3	1	3
15	22	17	71	5	2	2

methodology) are shown in table 5.4. CompoSWS was capable of identifying a set of workflow inconsistencies at the implementation level (BR6) and inconsistencies detected by the BPA (BR2). In addition, relationships that were present at the business level but could not be inferred by CompoSWS were also detected and enforced into the platform (i.e., a business process pattern +). Table 5.4 has 7 columns that include the participant's identifier, the number of models generated by CompoSWS, the number of workflows recommended by CompoSWS (BR3, BR4), the number of services generated including all the recommended workflows detected by CompoSWS, the implementation level inconsistencies identified by CompoSWS (BR6), the number of inconsistences identified by the BPA participants in the interview (BR2), and the enforced relationships identified by the participant (BR1).

Notice that we generate a configuration spreadsheet but only for the BPAs BP models, not the alternative recommendations. The data gathered from BPAs were used by the BP Emulator to calculate the costs of the paths in the workflow; such data was fed to CompoSWS to annotate the corresponding services' relationships (step 4a in the methodology). Table 5.5 presents the results of the Emulator. Notice that the BPA was able to assign a cost to all the relationships between services in their model; CompoSWS was able to assign such cost whenever a workflow recommendation included a path with a known cost. In addition, CompoSWS was able to detect the less expensive path in a workflow (see Table 5.5) and later calculate the total cost of each workflow in order to recommend the optimized workflow.

TABLA 5.5. CompoSWS recommendations to SA and BPA based on the spreadsheet tool

Participant	#workflows	#recommended workflows	#annotated paths (BR5)	#optimized recommended path
1	7	3	0	0
2	8	5	0	1
3	9	5	0	3
4	10	7	0	2
5	11	5	0	0
6	12	6	0	0
7	13	8	1	2
8	14	8	0	1
9	15	8	0	1
10	16	7	0	2
11	17	9	0	1
12	18	9	1	0
13	19	11	0	0
14	20	13	0	1
15	21	17	0	2

Table 5.5 has 5 columns that represents: the participant's identifier, the number of workflows generated on CompoSWS, the number of recommended workflows that CompoSWS detected, the annotated paths that were reporting for CompoSWS after the execution of different paths and the last column represents the number of recommendations paths

accepted by the participants as an optimal path. Part of this information is taken from the execution log of BP Emulator.

5.5.4. Third interview: Evaluating the recommendation

During this interview, we presented the results to the BPAs, i.e., the enforced relationships at the business level (BR1), the alternative workflows for implementing their BP model (BR3), extensions to the BPM (BR4), paths costs (BR5) and recommendation generated for such paths, as well as inconsistencies detected at the platform (BR6) and business (BR2) level as described in section 5.2. We asked the BPAs to provide feedbacks on this information, which is summarized in table 5.6 and table 5.7. These results are discussed in detail in section 5.6 .

TABLA 5.6. BPAs response to CompoSWS recommendations based on the BPM and the spreadsheet tool

Results	BR1	BR2	BR3
Enforced connection at BPM level	7 require+, 2 synchronize+, 4 discriminator+		
Recommendations from BP Emulator based on the log	7 accepted, 2 rejected, 6 indifferent		
Inconsistencies at BP level		4 BPAs detected 1 or 2 inconsistencies in their models, 7 BPA's indicated their models were similar, and 4 BPAs indicated their models were totally different	
Recommending Web services from CompoSWS			12 participants accepted the recommendations, 3 preferred their own models

5.6. Analyzing the results

5.6.0.1. Transferring business knowledge from BPAs to SAs and the workflow platform.

BR1: Extending BP model implementation based on the feedback provided by the BPA.

We detected 7 BPAs that required the enforcement of a relationship between services that CompoSWS did not manage to identify because the services' signatures differ. 7 participants needed to enforce a *require*+ relationship that occurred when a service's input, $S1$, included a set of receipts that were validated by the subsequent service $S2$; however, if the receipts were invalid a *require*+ relationship needed to be established between services $S2$ and $S1$ creating this way a cycle. CompoSWS considered this an invalid configuration since cycles are considered an inconsistency. This is however a fundamental requirement at the business level.

In 2 cases, CompoSWS detected that the output of a service, $S3$, did not matched the input of another service, $S4$, and therefore no relationship was detected; however, BPAs identified that such parameters (an incomplete form and an unsigned form) were equivalent at the business level and hence a *synchronize*+ relationship should have been established since additional parameters provided from other services were required.

Similarly, due to mismatches between input, output parameters and goals, CompoSWS was unable to detected alternative relationships that BPAs required. In all 4 cases, BPAs indicated that such elements were equivalent at the business level. For instance, three services $S5, S6$, and $S7$ share the same goal, to *verify a worksheet*, however, two of them shared the same output *FinalWorksheet* whereas the third output was labeled as *worksheet*. CompoSWS required that both, goal and output, are the same for identifying such services as related by a *discriminator* pattern. In this case a *discriminator+* pattern is enforced between the services.

The feedback from the BP analysis logs indicated the costs related to the various workflows in terms of process execution time (at the business level, not software level); all the processes had very similar costs, so that, only those workflows with less nodes (tasks) presented smaller costs and hence were recommended to the BPAs. 15 recommendations were made, where 7 BPAs accepted the recommendations, 2 BPAs rejected them, while 6 remained indifferent.

BR2: Identifying inconsistencies between the workflow implementation and the BP model.

A total of 11 participants indicated that travel allowances were processed in the same way of general reimbursements whereas four participants indicated their models were totally different: Education(PUC) and Construction Engineering (PUC) required that travel allowances should be granted before in order to initiate a reimbursement process. Computer Science (Santa Maria) and Engineering (Los Andes) required that travel allowances were requested separately from reimbursements. In these cases, the reimbursements were devoted only to expenses other than travel tickets and hotels.

When evaluating their BP models against the others, 4 participants (3 from Los Andes and 1 from Agronomy-PUC) identified inconsistencies in their own BP models. The remaining 7 participants did not consider that their own BP models were inconsistent.

5.6.0.2. Transferring implementation knowledge from the workflow platform to the SA and BPA.

BR3: Assisting SAs in identifying Web service candidates for reuse.

When uploading the services corresponding to BP's tasks into CompoSWS, the platform recommended similar services for implementing the tasks. Thus, 3 to 17 workflows were recommended to the BPAs (see Table 5.4). 12 participants were willing to accept the recommendations and 3 preferred their own models:

TABLA 5.7. BPAs response to CompoSWS recommendations based on the BPM and the spreadsheet tool (second part)

Results	BR4	BR5	BR6
Recommending extensions by CompoSWS	10 BPAs indicated they could extend their BP models and 5 BPAs did not accepted to make changes.		
Workflows suggested by CompoSWS based on their costs (execution time)		13 BPAs accepted de recommendations but 2 participants rejected them	
Inconsistencies detected by CompoSWS			5 BPAs accepted the inconsistencies, 3 BPAs rejected the inconsistencies. 7 remained indifferent.

Agronomy (PUC), Education (PUC) and Research and Postgraduate Office (Los Andes). The more workflows were uploaded the more services could be recommended, making it a burden for the user to choose one service to reuse, or even choose to implement a new one. Visualization strategies to alleviate the recommendation strategy such as filtering (Konstan & Riedl, 2012), emphasis (Waldner & Vassileva, 2014), or sophisticated hierarchical graph maps (Hernando, Moya, Ortega, & Bobadilla, 2014) have been proposed in diverse areas to deal with information overload when recommending items.

BR4: Recommending plausible alternatives in the design of BP models based on the workflow platform and existing web services.

Due to the existence of control-flow patterns such as *discriminator* and *synchronize*, it is possible to identify alternative implementations for the BP's tasks. In the study, 10 of the interviewed BPAs indicated they could extend their BP models so that the steps are followed more strictly. They also accepted that alternative paths should be recommended in case of delays in the execution of tasks. 5 BPAs from Agronomy (PUC), Education (PUC), Computer Science (Santa Maria) and Research and Postgrad Office (Los Andes) did not accept making changes to their own models.

BR5: Recommending plausible alternatives in the design of BP models based on their execution costs.

We presented the workflows' paths costs, in terms of execution time, to the BPAs as well as an optimized workflow suggested by CompoSWS. Thirteen participants showed willingness to change their own BP model partially, in those aspects were their models were more expensive. Among these 13 participants, there were 3 BPAs that formerly rejected our workflow recommendation (BR4) but were willing to accept it when they knew about the workflow's cost (Agronomy (PUC), Education (PUC), Computer Science (Santa Maria)). The remaining two participants (2 from Research and Postgrad Office at Los Andes) preferred to keep their own model unaltered. This demonstrates that our recommendations were effective.

BR6: Identifying inconsistencies between BP models and its implementation as a workflow. CompoSWS detected 4 inconsistencies in the use of the *synchronize* pattern: two of them presented cycles; two required input parameters that were not obtained except by directly asking them to the users. For example, services with 4 input parameters, were 3 of them could be obtained through the invocation of previous services, but 1 missing parameter that could not be obtained.

5 BPAs (Computer Science (Santa Maria), Research and Postgrad Office (Los Andes) and Industrial Engineering (PUC)) indicated that such inconsistencies made sense at the business level. 3 BPAs (Agronomy (PUC), Education (PUC) and CIWS) indicated that even though input data was missing, the workflows

were correct, and obtaining the missing parameter directly from the user could solve it. The remaining 7 BPAs indicated that they did not present such cases in their own BP models so they refrained of making a choice.

5.7. Conclusions

In this chapter we identified 6 mayor requirements for bridging BPA and SA roles during the life cycle of a BP model and its implementation with Web services. Based on such requirements, we propose a methodology that enables both roles to have a closer collaboration. This methodology exploits the benefits of two instruments, CompoWS, a platform capable of composing workflows automatically, and a BP Emulator that makes it possible to analyze the business process models. Together, these instruments enable the capturing of each roles' expertise and generate recommendations that help in enriching both roles.

We tested our approach and found that, in effect, BPAs were able to identify relationships, at the business level, that could not be identified by an automatic tool. This follows the work of (Buchwald et al., 2011), (Delgado, García-rodríguez De Guzmán, Ruiz, & Piattini, 2010) and (Bär, Schmidt, & Möhring, 2014) where they generate relationships directly from the BP model to the workflow. In our case, we identified missing relationships and inconsistencies between the BP model and its implementation and we were able to recommend changes to the implementation that can benefit the SA role. In addition, we enriched the workflow platform with business analysis results so that recommendations can be prioritized from the workflow costs. Knowledge sharing has proved of significant assistance to detect also inconsistencies at the BP model level, by identifying such cases, the BP role was also enriched with information related to the concerned BP model. In summary, the proposed instrumentation (CompoSWS and BP Emulator) made possible to uncover valuable information. Under the framework of our methodology, an iterative, close collaboration scenario between business process and IT stakeholders was set up. This collaboration allowed both stakeholders to exchange their knowledge in order to avoid mistakes, reuse, and refine each other's deliverables.

6. CONCLUSIONS AND FUTURE WORK

6.1. Automatic and dynamic functional service composition

In Chapter 3 we describe an approach to dynamic automatic Semantic Web service composition. Our approach has been directed to meet the main challenges in service composition. First, it is autonomous so that the users do not required to analyze the huge amount of available services manually. Second, it has positive scalability, therefore the composition is better performed in a dynamic environment when compared to simple service composition strategies. Third, since Web services are often implemented by different organizations using different conceptual models, we improve this problem using semantic descriptions and the relationship between services for matching and composing Web services.

We focused on the matchmaking component and the use of W3C recommendations as well as technologies available today for implementing highly scalable semantic solutions. This approach allowed us a simple and scalable implementation, and allows highly sophisticated components (such as SAM algorithm) could be replaced.

The main research question to be addressed in this part of thesis is: *Given a set of services, semantically described, can we design an scalable method to discover complex service compositions (i.e. complex workflows)*

The main conclusion from the experience is that we can discover and compose Semantic Web services considering functional goals by exploiting the services' signature, as has been done in the literature, but to compose services that mimic the complexity of industry level business processes we require to capture such semantics. In particular we have extended the MSM ontology in order to allow the specification of simple and complex control-flow patterns based on the service's signature. Our solution considered some control-flow patterns commonly used in business process modeling as well as the representation of such semantics in a graph and the rules that automatically determine its inference.

In addition, when adding such complexity to determine service composition performance and scalability become relevant issues that must be faced. Our approach was to keep a minimalist semantic model to control the graph growth, to precalculte potential service compounds, and to define a composition algorithm that exploits SPARQL queries. A semantic graph model allows discovering well-formed complex, composed services, automatically and dynamically under a reasonable response time. Furthermore, our performance is twice as fast as the be performance reported, however, our approach is not fully comparable since ether's results consider much more parameters and produce simple service composition (sequences).

Given the specification of available Web services and users requirement, an algorithm can generate a composition of available services, which matches the requirement of the user. The composition process is fully autonomous without the intervention from the user. Additionally, the composite service generation should rely on the specification of Web services that provides significant improvements regarding throughput and availability.

Finally, the presented approach allow the generation of compositions that correspond to complex business patterns adopted in most real scenarios, without losing performance when compared to approaches that only consider simple business patterns.

The fact remains that, we just explored 6 control-flow patterns out of the numerous existing and ones in order to prove the feasibility of our approach, this work should be extended to determine the feasibility of automatically deriving the remaining patterns.

6.2. Regarding eliciting BPA expertise

In Chapter 4, a spreadsheet-based approach for business process model analysis is used to map the problem of business process performance analysis and verification. It means that a user friendly analysis tool of business process was created to allow to the business analyst to easily define metrics, assertions and test reports in order to elicit her expertise, in particular, given that spreadsheets are omnipresent in business and well-known by average BP analysts.

The main research question to be addressed in this part of thesis is: *How can we enable BPAs (without development experience) to analyze and improve their BPs on their own, with less reliance on and intervention of SA?*

The main conclusion is that the combination of a BP simulation and a spreadsheet-based UI (Google Spreadsheet) accompanied with a standard BPMN process model representation empowers BP analysts with limited technical skills to simulate, verify and analyze the execution time and real-time performance of BPs through personalized instruments (metrics, assertions and custom reports).

Second, the presented user studies confirm that the spreadsheet abstraction has indeed the potential to enable BP analysts to perform complex BP analyses and to effectively discuss findings with software developers during the life cycle of a process in an iteratively way. In addition, the qualitative analysis complements the user studies with a discussion of its strengths and weaknesses compared to the state of the art in BP model analysis.

Finally, thanks to the positive results obtained this work should be extended allowing BP analysts to also obtain a concrete feeling of how their processes behave if deployed in a real BP system using as an example a process monitoring dashboard, and also we would like to incorporate the log data comparison produced during the analysis of a process and after the deployment of the process in order to improve analysis precision.

6.3. Regarding closing the gap between BPAs and SAs

In Chapter 5, we join the previous approaches, the first one focused on the automatic and dynamic discovery of workflows from services' signature and the second one focused on the analysis and simulation of a business process, with the purpose to enrich and close the gap between BPA and SA in both directions.

The main research question to be addressed in this part of thesis is: *How can we assist the BPA to transfer their BP knowledge to the SA and conversely, exploit the IT infrastructure to assist the BPA design choices when constructing a BP model?*

The main conclusion is that the combination of a development and analysis supporting platform under a methodological framework increases the flexibility in the development of service- and process-oriented information systems. Our approach enables the realization of business requirements by an IT implementation with a higher quality and more quickly by: ensuring bidirectional traceability between BPA and SA, enabling automatic identification of inconsistencies in the business models and supporting the modeler (BPA and SA) when resolving inconsistencies.

Finally, the development and specification of the six bridging strategies allows to transfer the underlying expertise in both directions demonstrated that in effect the workflow platform with business analysis capability was enriched in their capacity of finding inconsistencies between what was technically possible versus what was required at business level, and also sharing the knowledge gathered by the technical platform have prove of significant assistance to detect also inconsistencies at the implementation and BP model level, for the BPA.

In the future, we plan to extend this work by considering the missing cases and unimplemented details in the Emulator editor such as adding support to new BPMN modules (like event support), adding the ability to simulate resource utilization and logging the loops. In addition a new user testing is needed in order to test the methodology at inter-organizational level with as least 20 real users from a international company.

References

AbuJarour, M., & Awad, A. (2011). Discovering linkage patterns among web services using business process knowledge. In *Services computing (scc), 2011 ieee international conference on* (pp. 314–321).

AbuJarour, M., & Awad, A. (2014). Web services and business processes: A round trip. In *Web services foundations* (pp. 3–29). Springer.

Aguilar, M., Rautert, T., & Pater, A. (1999). Business process simulation: a fundamental step supporting process centered management. In *Winter simulation* (pp. 1383–1392).

Ahmad, H., & Dowaji, S. (2013). Linked-owl: A new approach for dynamic linked data service workflow composition. *Webology*, *10*(1), 21–30.

Alamri, A., Eid, M., & El Saddik, A. (2006). Classification of the state-of-the-art dynamic web services composition techniques. *International Journal of Web and Grid Services*, *2*(2), 148–166.

Alarcon, R., & Wilde, E. (2010). Linking data from restful services. In *Third workshop on linked data on the web, raleigh, north carolina (april 2010).*

Alowisheq, A., Millard, D. E., & Tiropanis, T. (2009). *Express: Expressing restful semantic services using domain ontologies*. Springer.

Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., … Thatte, S. (2003). *Business process execution language for web services.* http://goo.gl/kpBKUS. version.

Arenas, M., Conca, S., & Pérez, J. (2012). Counting beyond a yottabyte, or how sparql 1.1 property paths will prevent adoption of the standard. In *Proceedings of the 21st international conference on world wide web* (pp. 629–638).

Austin, D., Barbir, A., Ferris, C., & Garg, S. (2004). Web services architecture requirements. *W3C Working Group Notes*, 22.

Aversano, L., Di Penta, M., & Taneja, K. (2006). A genetic programming approach to support the design of service compositions. *International Journal of Computer Systems Science & Engineering*, *21*(4), 247–254.

Bansal, A., Blake, M. B., Kona, S., Bleul, S., Weise, T., & Jaeger, M. C. (2008). Wsc-08: continuing the web services challenge. In *E-commerce technology and the fifth ieee conference on enterprise computing, e-commerce and e-services, 2008 10th ieee conference on* (pp. 351–354).

Bär, F., Schmidt, R., & Möhring, M. (2014). Fabric-process patterns. In *Enterprise, business-process and information systems modeling* (pp. 139–153). Springer.

Basu, S., Casati, F., & Daniel, F. (2008). Toward web service dependency discovery for soa management. In *Services computing, 2008. scc'08. ieee international conference on* (Vol. 2, pp. 422–429).

Beizer, B. (1995). *Black-box testing: techniques for functional testing of software and systems*. John Wiley & Sons, Inc.

Bellido, J., Alarcón, R., & Pautasso, C. (2013). Control-flow patterns for decentralized restful service composition. *ACM Transactions on the Web (TWEB)*, *8*(1), 5.

Bener, A. B., Ozadali, V., & Ilhan, E. S. (2009). Semantic matchmaker with precondition and effect matching using swrl. *Expert Systems with Applications*, *36*(5), 9371–9377.

Bhat, J. M., Gupta, M., & Murthy, S. N. (2006). Overcoming requirements engineering challenges: Lessons from offshore outsourcing. *Software, IEEE*, *23*(5), 38–44.

BizAgi. (2015). Bizagi process modeler - user guide [Computer software manual]. (http://download.bizagi.com/docs/modeler/2904/en/Modeler_user_Guide.pdf)

BOCGroup. (2015). Adonis community edition getting started series [Computer software manual]. (http://en.adonis-community.com/welcome/webinars-and-tutorials/adonisce-getting-started-series)

Bonitasoft. (2011, October). Bonita open solution - simulation guide [Computer software manual]. (http://www.bonitasoft.com/system/files/download/bos-5.6-simulation-guide.pdf)

Booth, D., & Liu, C. K. (2007). Web services description language (wsdl) version 2.0 part 0: Primer. *W3C Recommendation*, *26*.

Borst, W. N. (1997). *Construction of engineering ontologies for knowledge sharing and reuse*. Universiteit Twente.

Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., ... Winer, D. (2000). *Simple object access protocol (soap) 1.1*.

Broekstra, J., & Kampman, A. (2004). Serql: An rdf query and transformation language draft.

Brogi, A., Corfini, S., & Popescu, R. (2008). Semantics-based composition-oriented discovery of web services. *ACM Transactions on Internet Technology (TOIT)*, *8*(4), 19.

Bucchiarone, A., Melgratti, H., & Severoni, F. (2007). Testing service composition. In *Proceedings of the 8th argentine symposium on software engineering (assea07).*

Buchwald, S., Bauer, T., & Reichert, M. (2011). Bridging the gap between business process models and service composition specifications.

Burnett, M., Cook, C., Pendse, O., Rothermel, G., Summet, J., & Wallace, C. (2003). End-user software engineering with assertions in the spreadsheet paradigm. In *Proceedings of the 25th international conference on software engineering* (pp. 93–103).

Burnett, M., Cook, C., & Rothermel, G. (2004). End-user software engineering. *Communications of the ACM*, *47*(9), 53–58.

Burnett, M. M., & Ambler, A. L. (1994). Interactive visual data abstraction in a declarative visual programming language. *Journal of Visual Languages & Computing*, *5*(1), 29–60.

Burnstein, I. (2006). *Practical software testing: a process-oriented approach*. Springer Science & Business Media.

Burstein, M., Hobbs, J., Lassila, O., Mcdermott, D., Mcilraith, S., Narayanan, S., ... others (2004). Owl-s: Semantic markup for web services. *W3C Member Submission*.

Casati, F., Castellanos, M., Dayal, U., & Salazar, N. (2007). A generic solution for warehousing business process data. In *Vldb* (pp. 1128–1137).

Castro, V., Mesa, J. M. V., Herrmann, E., & Marcos, E. (2008). A model driven approach for the alignment of business and information systems models. In *Computer science, 2008. enc'08. mexican international conference on* (pp. 33–43).

Chabeb, Y., Tata, S., & Ozanne, A. (2010). Yasa-m: A semantic web service matchmaker. In *Advanced information networking and applications (aina), 2010 24th ieee international conference on* (pp. 966–973).

Chandrasekaran, S., Miller, J., Silver, G., Arpinar, B., & Sheth, A. (2003). Performance analysis and simulation of composite web services. *Electronic Markets*, *13*(2), 120–132.

Chen, A., & Buchs, D. (2006). Generative business process prototyping framework. In *Rapid system prototyping, 2006. seventeenth ieee international workshop on* (pp. 140–148).

Chinnici, R., Moreau, J.-J., Ryman, A., & Weerawarana, S. (2007). Web services description language (wsdl) version 2.0 part 1: Core language. *W3C recommendation*, *26*, 19.

Coalition, O. (2004). *Owl-s 1.1 release.(2004).*

De Bruijn, J., Lausen, H., Polleres, A., & Fensel, D. (2006a). *The web service modeling language wsml: an overview*. Springer.

De Bruijn, J., Lausen, H., Polleres, A., & Fensel, D. (2006b). *The web service modeling language wsml: an overview*. Springer.

De Castro, V., Marcos, E., & Lopez Sanz, M. (2006). A model driven method for service composition modelling: a case study. *International Journal of Web Engineering and Technology*, *2*(4), 335–353.

Delgado, A., García-rodríguez De Guzmán, I., Ruiz, F., & Piattini, M. (2010). Tool support for service oriented development from business processes. In *2nd international workshop on model-driven service engineering (mose?10) in 48th int. conf. on objects, models, components, patterns (tools?10), málaga, spain.*

Delgado, A., Ruiz, F., de Guzmán, I. G.-R., & Piattini, M. (2010). A model-driven and service-oriented framework for the business process improvement. *Journal of Systems Integration*, *1*(3), 45.

Deloitte. (2009). Spreadsheet management: Not what you figured..

Dietze, S., Benn, N., Yu, H. Q., Pedrinaci, C., Makni, B., Liu, D., … Domingue, J. (2010). Comprehensive service semantics and light-weight linked services: towards an integrated approach.

Dijkman, R., Dumas, M., & García-Bañuelos, L. (2009). Graph matching algorithms for business process model similarity search. In *Bpm* (pp. 48–63). Springer.

Dijkman, R. M., Dumas, M., & Ouyang, C. (2007). Formal semantics and analysis of bpmn process models using petri nets. *Queensland University of Technology, Tech. Rep*.

Dijkman, R. M., Dumas, M., & Ouyang, C. (2008). Semantics and analysis of business process models in bpmn. *Information and Software Technology*, *50*(12), 1281–1294.

D'Mello, D. A., Ananthanarayana, V., & Salian, S. (2011). A review of dynamic web service composition techniques. In *Advanced computing* (pp. 85–97). Springer.

Domingue, J., Cabral, L., Hakimpour, F., Sell, D., & Motta, E. (2004). Irs iii: A platform and infrastructure for creating wsmo based semantic web services.

Domingue, J., Galizia, S., & Cabral, L. (2005). Choreography in irs-iii–coping with heterogeneous interaction patterns in web services. In *The semantic web–iswc 2005* (pp. 171–185). Springer.

Dong, X., Halevy, A., Madhavan, J., Nemes, E., & Zhang, J. (2004). Similarity search for web services. In *Proceedings of the thirtieth international conference on very large data bases-volume 30* (pp. 372–383).

Dürst, M., & Suignard, M. (2004). *Internationalized resource identifiers (iris)* (Tech. Rep.).

Dustdar, S., & Schreiner, W. (2005). A survey on web services composition. *International journal of web and grid services*, *1*(1), 1–30.

Erl, T. (2008). *Soa: principles of service design* (Vol. 1). Prentice Hall Upper Saddle River.

Fisher, M., & Rothermel, G. (2005). The euses spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms. In *Acm sigsoft software engineering notes* (Vol. 30, pp. 1–5).

Galli, J. S., Vairetti, C., Rodríguez, C., Daniel, F., Casati, F., & Alarcón, R. (2015). Analysis and improvement of business process models using spreadsheets. *Information Systems*.

García-Fanjul, J., Tuya, J., & De La Riva, C. (2006). Generating test cases specifications for bpel compositions of web services using spin. In *Ws-mate 2006* (p. 83).

Ghafarian, T., & Kahani, M. (2009). Semantic web service composition based on ant colony optimization method. In *Networked digital technologies, 2009. ndt'09. first international conference on* (pp. 171–176).

Gooneratne, N., Tari, Z., & Harland, J. (2007). Verification of web service descriptions using graph-based traversal algorithms. In *Proceedings of the 2007 acm symposium on applied computing* (pp. 1385–1392).

Governatori, G., Milosevic, Z., & Sadiq, S. (2006). Compliance checking between business processes and business contracts. In *Edoc* (pp. 221–232).

Grant, J., & Becket, D. (2004). *Rdf test cases-n-triples* (Tech. Rep.). Tech. rep., W3C Recommendation.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, *5*(2), 199–220.

Gutiérrez, J., Escalona, M., Mejías, M., & Torres, J. (2005). Analysis of proposals to generate system test cases from system requirements. In *Caise'05 forum.*

Hadley, M. J. (2006). Web application description language (wadl).

Haller, A., Cimpian, E., Mocan, A., Oren, E., & Bussler, C. (2005). Wsmx-a semantic service-oriented architecture. In *Web services, 2005. icws 2005. proceedings. 2005 ieee international conference on* (pp. 321–328).

Hamadi, R., & Benatallah, B. (2003). A petri net-based model for web service composition. In *Proceedings of the 14th australasian database conference-volume 17* (pp. 191–200).

Harth, A., & Decker, S. (2005). Optimized index structures for querying rdf from the web. In *Web congress, 2005. la-web 2005. third latin american* (pp. 10–pp).

Hartvigsen, D. (2004). *Simquick, process simulation with excel*. Prentice-Hall, Inc.

He, J., Zhang, Y., Huang, G., & Cao, J. (2012). A smart web service based on the context of things. *ACM Transactions on Internet Technology (TOIT)*, *11*(3), 13.

Hermans, F. (2013). Improving spreadsheet test practices. In *Cascon* (pp. 56–69).

Hernando, A., Moya, R., Ortega, F., & Bobadilla, J. (2014). Hierarchical graph maps for visualization of collaborative recommender systems. *Journal of Information Science*, *40*(1), 97–106.

Hoffmann, J., Bertoli, P., & Pistore, M. (2007). Web service composition as planning, revisited: In between background theories and initial state uncertainty. In *Proceedings of the national conference on artificial intelligence* (Vol. 22, p. 1013).

Howden, W. E. (1980). Functional program testing. *Software Engineering, IEEE Transactions on*(2), 162–169.

IBM. (2009). Tutorials and samples for websphere business modeler version 6.2 [Computer software manual]. (http://www-01.ibm.com/support/docview.wss?uid=swg27013902)

Kaner, C., Falk, J., & Nguyen, H. Q. (2000). *Testing computer software second edition*. Dreamtech Press.

Kil, H., & Nam, W. (2013). Semantic web service composition via model checking techniques. *International Journal of Web and Grid Services*, *9*(4), 339–350.

Klein, M., Konig-Ries, B., & Mussig, M. (2005). What is needed for semantic service descriptions? a proposal for suitable language constructs. *International Journal of Web and Grid Services*, *1*(3-4), 328–364.

Klusch, M. (2012). Overview of the s3 contest: Performance evaluation of semantic service matchmakers. In *Semantic web services* (pp. 17–34). Springer.

Klusch, M., Gerber, A., & Schmidt, M. (2005). Semantic web service composition planning with owls-xplan. In *Proceedings of the aaai fall symposium on semantic web and agents, arlington va, usa, aaai press.*

Klusch, M., Kapahnke, P., Schulte, S., Lecue, F., & Bernstein, A. (2015). Semantic web service search: A brief survey. *KI-Künstliche Intelligenz*, 1–9.

Klusch, M., Kapahnke, P., & Zinnikus, I. (2009). Hybrid adaptive web service selection with sawsdl-mx and wsdl-analyzer. In *The semantic web: Research and applications* (pp. 550–564). Springer.

Konstan, J. A., & Riedl, J. (2012). Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction*, *22*(1-2), 101–123.

Kopecky, J., Vitvar, T., Bournez, C., & Farrell, J. (2007). Sawsdl: Semantic annotations for wsdl and xml schema. *Internet Computing, IEEE*, *11*(6), 60–67.

Kruck, S. (2006). Testing spreadsheet accuracy theory. *Information and Software Technology*, *48*(3), 204–213.

Krummenacher, R., Norton, B., & Marte, A. (2010). Towards linked open services and processes. In *Future internet-fis 2010* (pp. 68–77). Springer.

Kylau, U., Stollberg, M., Weber, I., & Barros, A. (2012). Service functionality and behavior. In *Handbook of service description* (pp. 269–293). Springer.

Lassila, O., Swick, R. R., et al. (1998). Resource description framework (rdf) model and syntax specification.

Lathem, J., Gomadam, K., & Sheth, A. P. (2007). Sa-rest and (s) mashups: Adding semantics to restful services. In *Semantic computing, 2007. icsc 2007. international conference on* (pp. 469–476).

Lausen, H., & Farrell, J. (2007). Semantic annotations for wsdl and xml schema. *W3C recommendation, W3C*.

Lazar, J., Feng, J. H., & Hochheiser, H. (2010). *Research methods in human-computer interaction*. Wiley.

Lecue, F., & Leger, A. (2006). Semantic web service composition based on a closed world assumption. In *Web services, 2006. ecows'06. 4th european conference on* (pp. 233–242).

Lemos, A. L., Daniel, F., & Benatallah, B. (2015). Web service composition: A survey of techniques and tools. *ACM Computing Surveys (CSUR)*, *48*(3), 33.

Li, Z. J., Sun, W., & Du, B. (2008). Bpel4ws unit testing: Framework and implementation. *International Journal of Business Process Integration and Management*, *3*(2), 131–143.

Li, Z. J., Tan, H., Liu, H., Zhu, J., & Mitsumori, N. M. (2008). Business-process-driven gray-box soa testing. *IBM Systems Journal*, *47*(3), 457–472.

Lin, C., Kwong, A., & Perni, R. (2006). Discovery and development of vx-950, a novel, covalent, and reversible inhibitor of hepatitis c virus ns3. 4a serine protease. *Infectious Disorders-Drug Targets (Formerly Current Drug Targets-Infectious Disorders)*, *6*(1), 3–16.

Liu, Y., Muller, S., & Xu, K. (2007). A static compliance-checking framework for business process models. *IBM Systems Journal*, *46*(2), 335–361.

Maleshkova, M., Pedrinaci, C., & Domingue, J. (2009). Supporting the creation of semantic restful service descriptions.

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., … others (2004). Owl-s: Semantic markup for web services. *W3C member submission*, *22*, 2007–04.

Mayer, P., & Lübke, D. (2006). Towards a bpel unit testing framework. In *Workshop on testing, analysis, and verification of web services and applications* (pp. 33–42).

McGuinness, D. L., Van Harmelen, F., et al. (2004). Owl web ontology language overview. *W3C recommendation*, *10*(10), 2004.

Medjahed, B. (2004). *Semantic web enabled composition of web services* (Unpublished doctoral dissertation). Citeseer.

Mell, P., & Grance, T. (2011). The nist definition of cloud computing.

Mennie, D. W. (2000). *An architecture to support dynamic composition of service components and its applicability to internet security* (Unpublished doctoral dissertation). Citeseer.

Motahari-Nezhad, H. R., Saint-Paul, R., Casati, F., & Benatallah, B. (2011). Event correlation for process discovery from web service interaction logs. *VLDBJ*, *20*(3), 417–444.

Narayanan, S., & McIlraith, S. (2003). Analysis and simulation of web services. *Computer Networks*, *42*(5), 675–693.

Nayak, R., & Bose, A. (2015). A data mining based method for discovery of web services and their compositions. In *Real world data mining applications* (pp. 325–342). Springer.

Object Management Group (OMG). (2011). *Business process model and notation (bpmn) version 2.0.* http://www.omg.org/spec/BPMN/2.0/.

ORG, U. (2000). *Uddi executive white paper.* OASIS.

Papazoglou, M. P., Traverso, P., Dustdar, S., & Leymann, F. (2007). Service-oriented computing: State of the art and research challenges. *Computer*(11), 38–45.

Papazoglou, M. P., Traverso, P., Dustdar, S., & Leymann, F. (2008). Service-oriented computing: a research roadmap. *International Journal of Cooperative Information Systems*, *17*(02), 223–255.

Pautasso, C. (2009a). Composing restful services with jopera. In *Software composition* (pp. 142–159).

Pautasso, C. (2009b). Restful web service composition with bpel for rest. *Data & Knowledge Engineering*, *68*(9), 851–866.

Pedrinaci, C., Lambert, D., Maleshkova, M., Liu, D., Domingue, J., & Krummenacher, R. (2011). *Adaptive service binding with lightweight semantic web services*. Springer.

Pedrinaci, C., Liu, D., Maleshkova, M., Lambert, D., Kopecky, J., & Domingue, J. (2010). iserve: a linked services publishing platform. In *Ceur workshop proceedings* (Vol. 596).

Pistore, M., Barbon, F., Bertoli, P., Shaparau, D., & Traverso, P. (2004). Planning and monitoring web service composition. In *Artificial intelligence: Methodology, systems, and applications* (pp. 106–115). Springer.

Reynolds, D. (2004). Jena 2 inference support. *Online manual at http://jena. sourceforge. net/inference/index. html.*

Rodríguez, C., Schleicher, D., Daniel, F., Casati, F., Leymann, F., & Wagner, S. (2013). Soa-enabled compliance management: instrumenting, assessing, and analyzing service-based business processes. *SOCA*, 1–18.

Rodriguez-Mier, P., Mucientes, M., Lama, M., & Couto, M. I. (2010). Composition of web services through genetic programming. *Evolutionary Intelligence*, *3*(3-4), 171–186.

Rodríguez-Mier, P., Mucientes, M., Vidal, J. C., & Lama, M. (2012). An optimal and complete algorithm for automatic web service composition. *International Journal of Web Services Research (IJWSR)*, *9*(2), 1–20.

Rodriguez Mier, P., Pedrinaci, C., Lama, M., & Mucientes, M. (2015a). An integrated semantic web service discovery and composition framework.

Rodriguez Mier, P., Pedrinaci, C., Lama, M., & Mucientes, M. (2015b). An integrated semantic web service discovery and composition framework.

Rong, W., Liu, K., & Liang, L. (2009). Personalized web service ranking via user group combining association rule. In *Web services, 2009. icws 2009. ieee international conference on* (pp. 445–452).

Rosenberg, F., Curbera, F., Duftler, M. J., & Khalaf, R. (2008). Composing restful services and collaborative workflows: A lightweight approach. *Internet Computing, IEEE*, *12*(5), 24–31.

Rothermel, G., Li, L., DuPuis, C., & Burnett, M. (1998). What you see is what you test: A methodology for testing form-based visual programs. In *Proceedings of the 20th international conference on software engineering* (pp. 198–207).

Rozinat, A., & van der Aalst, W. M. P. (2008). Conformance checking of processes based on monitoring real behavior. *Inf. Sys.*, *33*(1), 64–95.

Russell, N., Ter Hofstede, A. H., & Mulyar, N. (2006). Workflow controlflow patterns: A revised view.

Russell, N. C., van der Aalst, W. M., & Ter Hofstede, A. H. (2009). Designing a workflow system using coloured petri nets. In *Transactions on petri nets and other models of concurrency iii* (pp. 1–24). Springer.

Sayal, M., Casati, F., Dayal, U., & Shan, M.-C. (2002). Business process cockpit. In *Vldb* (pp. 880–883).

Scaffidi, C., Shaw, M., & Myers, B. (2005). Estimating the numbers of end users and end user programmers. In *Visual languages and human-centric computing, 2005 ieee symposium on* (pp. 207–214).

Schenk, S., & Petrák, J. (2008). Sesame rdf repository extensions for remote querying. In *Znalosti conf* (Vol. 113, p. 125).

Signavio. (2015). Feature overview a signavio process editor [Computer software manual]. (http://www.signavio.com/docs/en/pe-features.pdf)

Silveira, P., Rodríguez, C., Casati, F., Daniel, F., D'Andrea, V., Worledge, C., & Taheri, Z. (2010). On the design of compliance governance dashboards for effective compliance and audit management. In *Icsoc/servicewave 2009 workshops* (pp. 208–217).

Sirin, E., Parsia, B., Wu, D., Hendler, J., & Nau, D. (2004). Htn planning for web service composition using shop2. *Web Semantics: Science, Services and Agents on the World Wide Web*, *1*(4), 377–396.

Smirnov, S., Weidlich, M., Mendling, J., & Weske, M. (2009). *Action patterns in business process models*. Springer.

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering: principles and methods. *Data & knowledge engineering*, *25*(1), 161–197.

Sutcliffe, A. (2012). *User-centred requirements engineering*. Springer Science & Business Media.

Tan, Y., & Takakuwa, S. (2007). Predicting the impact on business performance of enhanced information system using business process simulation. In *Winter simulation* (p. 2203 - 2211).

Tanasescu, V., Domingue, J., & Cabral, L. (2004). Ocml ontologies to xml schema lowering.

Ter Hofstede, A. H., van der Aalst, W., Adams, M., & Russell, N. (2009). *Modern business process automation: Yawl and its support environment*. Springer Science & Business Media.

Th, J. F. R. H. P., & Schlepphorst, K. G. L. C. (n.d.). Florid: A prototype for f-logic.

TIBCO. (2014, November). Tibco business studio [Computer software manual]. (https://docs.tibco.com/pub/business-studio-bpm-edition/3.9.0/doc/html/index.html)

Touzi, J., Benaben, F., Pingaud, H., & Lorré, J. P. (2009). A model-driven approach for collaborative service-oriented architecture design. *International Journal of Production Economics*, *121*(1), 5–20.

Vairetti, C., Alarcon, R., & Bellido, J. (2016). A semantic approach for dynamically determining complex composed service behaviour. *Journal of Web Engineering*, *15*(3&4), 310–338.

van Der Aalst, W. M., Ter Hofstede, A. H., Kiepuszewski, B., & Barros, A. P. (2003a). Workflow patterns. *Distributed and parallel databases*, *14*(1), 5–51.

van Der Aalst, W. M., Ter Hofstede, A. H., Kiepuszewski, B., & Barros, A. P. (2003b). Workflow patterns. *Distributed and parallel databases*, *14*(1), 5–51.

Van Der Aalst, W. M., Ter Hofstede, A. H., & Weske, M. (2003). Business process management: A survey. In *Business process management* (pp. 1–12). Springer.

van der Aalst, W. M. P. (2011). *Process mining: discovery, conformance and enhancement of business processes*. Springer.

van der Aalst, W. M. P., Dumas, M., Ouyang, C., Rozinat, A., & Verbeek, E. (2008). Conformance checking of service behavior. *ACM Transactions on Internet Technology (TOIT)*, *8*(3), 13.

van der Aalst, W. M. P., Nakatumba, J., Rozinat, A., & Russell, N. (2008). Business process simulation: How to get it right. *BPM Center Report BPM-08-07, BPMcenter. org*.

van der Aalst, W. M. P., Weijters, T., & Maruster, L. (2004). Workflow mining: Discovering process models from event logs. *TKDE*, *16*(9), 1128–1142.

Verborgh, R., Steiner, T., Deursen, D., Van de Walle, R., & Vallés, J. G. (2011). Efficient runtime service discovery and consumption with hyperlinked restdesc. In *Next generation web services practices (nwesp), 2011 7th international conference on* (pp. 373–379).

Verborgh, R., Steiner, T., Van Deursen, D., Coppens, S., Vallés, J. G., & Van de Walle, R. (2012). Functional descriptions as the bridge between hypermedia apis and the semantic web. In *Proceedings of the third international workshop on restful design* (pp. 33–40).

Verma, K., & Sheth, A. (2007). Semantically annotating a web service. *IEEE Internet Computing*, *11*(2), 83.

Vitvar, T., Kopeckỳ, J., Viskova, J., & Fensel, D. (2008a). *Wsmo-lite annotations for web services*. Springer.

Vitvar, T., Kopeckỳ, J., Viskova, J., & Fensel, D. (2008b). *Wsmo-lite annotations for web services*. Springer.

Vitvar, T., Kopeckỳ, J., Zaremba, M., & Fensel, D. (2007). Wsmo-lite: Lightweight semantic descriptions for services on the web. In *Web services, 2007. ecows'07. fifth european conference on* (pp. 77–86).

Waldner, W., & Vassileva, J. (2014). Emphasize, don't filter!: Displaying recommendations in twitter timelines. In *Proceedings of the 8th acm conference on recommender systems* (pp. 313–316). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2645710.2645762 doi: 10.1145/2645710.2645762

Weske, M. (2007a). *Business process management: Concept, languages, architectures*. Springer.

Weske, M. (2007b). *Concepts, languages, architectures* (Vol. 14). Springer.

Wiegers, K., & Beatty, J. (2013). *Software requirements*. Pearson Education.

Wynn, M. T., Dumas, M., Fidge, C., Ter Hofstede, A. H., & van der Aalst, W. M. P. (2008). Business process simulation for operational decision support. In *Bpm* (pp. 66–77).

Xu, J., Chen, K., & Reiff-Marganiec, S. (2011). Using markov decision process model with logic scoring of preference model to optimize htn web services composition.

Yan, J., Li, Z., Yuan, Y., Sun, W., & Zhang, J. (2006). Bpel4ws unit testing: Test case generation using a concurrent path analysis approach. In *Issre 2006* (pp. 75–84).

Young, M. (2008). *Software testing and analysis: process, principles, and techniques*. John Wiley & Sons.

Yuan, Y., Li, Z., & Sun, W. (2006). A graph-search based approach to bpel4ws test generation. In *Software engineering advances, international conference on* (pp. 14–14).

Zhang, Y., Zhang, X., & Liu, F. (2010). Semantic web service matchmaking based on service behavior. In *Anti-counterfeiting security and identification in communication (asid), 2010 international conference on* (pp. 184–188).