PhD Dissertation

International Doctorate School in Information and Communication Technologies

DISI - University of Trento

Exploiting Business Process Knowledge for Process Improvement

Carlos Dario Rodríguez Hermosa

Advisors:

Prof. Fabio Casati - Dr. Florian Daniel

Università degli Studi di Trento

Co-Advisor:

Prof. Luca Cernuzzi

Universidad Católica "Nuestra Señora de la Asunción"

March 2013

To my grandmother Lucila,
my parents,
my brothers and sister,
and Delsi.

Acknowledgement

I would like to express my sincere gratitude to Prof. Fabio Casati and Dr. Florian Daniel for their support and guidance throughout the whole course of my doctoral studies. They truly helped me to develop the critical and methodological thinking needed to undertake a good research work. Their scientific, professional and academic excellence and example inspired me to grow as a researcher and professional.

I am also grateful to Claudio Bartolini for giving me the opportunity to join and work with his research group at Hewlett-Packard Research Labs, Palo Alto (CA). The internship that I undertook in this lab was very useful and insightful as a first research experience in an industrial environment.

Most importantly, I am infinitely grateful to my parents, grandmother Lucila, brothers and sister for their unconditional love and support throughout my entire life. My immense thanks goes also to my fiancée Delsi, for her emotional support and caring throughout all the years that we have been together. I would not have been able to reach this far without the love, encouragement and support of all them, and, therefore, this accomplishment is also theirs.

It is a pleasure to thank also my colleagues and friends with whom I have shared all these years. In particular, my gratitude goes for the Paraguayan community here Trento, Italy with whom I have shared so many good times, which helped me to overcome the homesickness produced by the distance from my country and family.

Finally, I would like thank the Department of Information Engineering and Computer Science, and the International Doctorate School in Information and Communication Technologies of the University of Trento for the financial support of my research work.

Carlos Rodríguez

Abstract

Processes are omnipresent in humans' everyday activities: withdrawals from an ATM, loan requests from a bank, renewals of driver's licenses, purchases of goods from online retail systems. In particular, the business domain has strongly embraced processes as an instrument to help in the organization of business operations, leading to so-called business processes. A business process is a set of logically-related tasks performed to achieve a defined business outcome. Business processes have a big impact on the achievement of business goals and they are widely acknowledged as one of the more important assets of any organization next to the organization's customer basis and, more recently, data. Thus, there is a high interest in keeping business processes performing at their best and improving those that do not perform well.

Nowadays, business processes are supported by a wide range of enabling technologies, including Web services and business process engines, which enable the (partial) automation of processes. Information systems supporting the execution of processes typically store a wealth of process knowledge that includes process models, process progression information and business data. The availability of such process knowledge gives unprecedented opportunities to get insight into business processes, which leads to the question of how to exploit this knowledge for facilitating the improvement of processes.

In order to answer this question, we propose to exploit process knowledge from two different but complementary perspectives. In the first one, we take the process execution perspective and leverage on process execution data generated by information systems to analyze and understand the actual behavior of executed processes. In the second one, we take the process design perspective and propose to extract process model patterns from existing models for reuse in the design of processes. The final goal of this thesis is to facilitate process improvement by exploiting existing process knowledge not only for gaining insight into and understanding of processes but also for reusing the resulting knowledge in the improvement thereof. We have successfully applied our approaches in the context of service-based business processes and assisted dataflow-based mashup development. In the former, we validated our approach through a end-user study of the usability and understandability of our approach and tools, while in the latter the evaluations were performed through experiments run on a dataset of models from the mashup tool Yahoo! Pipes.

Keywords
Business process, process model pattern, pattern mining, service-based business process, compliance, mashup, uncertainty.

Contents

Chapter 1

Executive Summary

Processes are part of every human's daily activities. Consciously or unconsciously, we are constantly participating in one or more processes. Take the example of a very simple process for shopping goods on an online, Web-based retail system. Typically, a costumer logs into his account, searches for the goods he is interested in, adds the goods to a shopping cart, provides shipping information, and pays for the goods. A customer goes through these steps guided by the application, in most cases, just by clicking the "next step" button and following this "shopping process". In the case of a small online shop with few clients, a small delay in one of the steps of the process will probably have no big impact in the overall performance of the business. Instead, if we consider big online shops such as Amazon.com where up to millions of items are ordered in busy days [12], such delays may severely damage not only the revenue of the company but also its reputation [7].

The example process above corresponds to what is known as a *business process*. As defined by Davenport and Short [4], a business process is "a set of logically-related tasks performed to achieve a defined business outcome." The set of business processes of an organization is considered as one of its more important assets [7], and, with their core business operations running as business processes, organizations have a high interest in keeping processes performing at their best. For this reason, *process improvement*, i.e., the endeavor of making processes effective (producing the desired outcomes) and efficient (minimizing the resources used), has always been a key concern for organizations.

Process improvement initiatives typically aim to eliminate defects that may prevent processes from achieving these goals [8]. Doing so requires both a careful *design* of processes so as to minimize the presence of defects and the ability to *identify problems* and analyze them to find and understand their *root causes* and then fix them [2]. In the following, we discuss these concerns in more details.

- **Process design.** Process design is the task of formalizing the (informal) business process descriptions using a business process modeling notation [27]. The design of effective and efficient business processes has always been a top interest for organizations since its introduction as an instrument for the organization of business operations [6]. Process design is typically not a one-time activity. It is rather a part of a broader business process management methodology where process design is repeatedly visited for making adjustments. Typically, a process is designed, then implemented and tested, deployed, and executed. Each phase that follows process design may encounter issues that, in turn, require going back to the process design phase to address such issues from a design perspective. Thus, designing a process that is both effective and efficient is not always straightforward and it can be a difficult task even for expert process designers, taking several iterations before these goals are satisfied. The delay introduced by these iterations may be then translated into additional costs or even failure in fulfilling the organization's business goals. Therefore, a careful design of processes is needed to avoid defects that may lead processes to fail in achieving their desired outcomes.

- **Identification of problems in processes.** Even when a process has been carefully designed to make it effective and efficient, in practice, the actual behavior of the process may be different from what was designed or expected, due to problems that emerge and become evident only at process execution time. The evidences are typically found in the process progression information and business data, and the types of problems that can be found include excessive task execution times, violations of compliance rules, process instance abortions, among other problems.

  The correct identification of such problems is a key requirement to analyze the underlying reasons of the underperformance of processes. Doing so can be a difficult endeavor for three main reasons. First, process execution data may be spread throughout the information systems of the company, and generated by fully and partially automated as well as manual processes, which makes the identification of process-relevant events hard. Second, the execution data generated by the running processes may be extensive and uncertain. It may be extensive because process instances may generate a large number of evidences during process progression. It may be uncertain because we may have cases in which we are not sure whether the generated evidences correctly represent the process progression. These issues complicate the identification of problems because we need to understand which data to look at and how much we can trust such data in the presence of uncertainty. Finally, spotting problems is in many cases subjective: a small underperformance of a process during one range of time may not be as critical as in another range of time. These issues hint at the need for an appropriate management of potentially uncertain process execution data and a mechanism to effectively identify and report problems to assist the process analyst.

- **Root cause analysis.** Once a problem has been identified, the next step is to understand what are the root causes of the problem. Finding root causes typically involves an analysis that start from a high level symptom down to fine-grained details of process executions, which typically asks again for the inspection of process progression information and business data. For example, we may start the analysis from an indicator that shows excessive execution times for process instances and go down to the execution times of each of the individual tasks of the process to identify the task where the root cause is found.

  There are cases in which the root cause is not associated to the problem in an obvious way. The challenge here resides in being able to associate these potentially large, fine-grained and non-obvious root causes to the identified problems. Just like for the identification of problems, finding root causes is also a difficult task because of the typically extensive execution data generated by process executions. A manual inspection of process execution evidences and their dynamics is in most cases not practical, and, therefore, we need to find a way to (at least partially) automate this work. Moreover, the imperfections present in process execution data may further complicate this work, not only in finding the root causes, but also in their interpretation by the human expert because the obtained models may be not only complex but also inaccurate. We need, therefore, to create awareness on the human expert about possible imperfections, such that the analysis performed on top of the root causes found and the decided improvement actions take such imperfections into account.

Many attempts has been undertaken to address the issue of process improvement. In the early nineties, these topics attracted a lot of interest with seminal works such as the ones from Harrington [7] and Hammer et al. [6], in which the focus was put on business process improvement, total quality and business process reengineering. In that same decade, the total or partial automation of business processes started to gain interest as a way of fostering process improvement, leading to technologies such as workflow management systems [11]. In more recent times, leveraged by Service-Oriented Computing (SOC) technologies such as Web services, service-based business processes turned into a key instrument for the organization of business operations in modern business [1] [14]. Medium and large sized organizations, both private and public, rely on such processes, which may expand within and across many of their business units and partners.

The technologies that emerged from the works above turned into the cornerstone for the support of business operations in organizations. Such technologies typically store a wealth of *process knowledge* including (i) process models, (ii) process progression information, and (iii) business data. A process model represents the activities that are part of the process and the constraints between them and it is typically expressed through a notation such as the Business Process Modeling Notation (BPMN) [28]. Process progression information refers to the actions and events that took place during process execution, and they are usually represented as events that are stored in an event log [26]. Finally, business data refer to data related to the domain of the business as produced during process progression and they are typically stored in the operational database or in the payload of the events recorded in the event log of the supporting information system.

The availability of such rich process knowledge creates unprecedented opportunities and challenges to gain visibility and insight into business processes. The research question we address in this dissertation is *how to exploit such knowledge to facilitate process improvement.*

1.1 Research Questions

The process knowledge discussed in the previous section can be studied from two different perspectives: On the one hand, we have the *process design (or static) perspective* in which the knowledge comes in the form of process models. On the other hand, we have the *process execution (or dynamic) perspective* where the knowledge comes in the form of process progression information and business data. The main objective of this dissertation is thus to exploit such process knowledge from these two perspectives to facilitate process improvement. From this, we formulate the following key research questions (RQs), which we address in this dissertation:

- **RQ1:** *How can we identify and extract process-relevant events within an organization information system?* Most approaches used for the identification of problems and root-cause analysis from process execution data assume the existence of logs that store evidences of process executions. For example, most algorithms used for process discovery require event logs where each event carries information such as the process instance identifier, the task name an event is related to, timestamp of its occurrence, among other process execution data [26]. In practice, this is not always the case since in many situations the information system supporting the business processes is not instrumented to generate such event logs.

  Typically, however, an information system stores both process progression information and business data produced by process executions in different formats and for different purposes. For example, *operational databases* [10] (also known as production databases) store data that comprise process progression data, process state data, business data produced throughout the process, data related to the regular operations of an organization, as well as their related business facts and objects. In a situation where there is no event log, we ask the question of how to reconstruct the necessary process execution data from such alternative datasources to enable process execution analysis techniques such as process discovery [26]. Many challenges emerge when trying to reconstruct an event log from such datasources including the identification of events related to process execution, the determination of the ordering of events, the grouping of events into process instances (event correlation), and the mapping of data to the payload of events.

- **RQ2:** *How can we exploit process execution data for the identification of problems and root-cause analysis?* The behavior of a process can significantly differ from what was originally designed. This is of no surprise if we consider that most processes are partly automated and partly executed by humans. In the former case, many factors can affect process execution and they range from hardware issues to runtime error in business applications. In the latter case, humans can make mistakes either intentionally or unintentionally. Problems that emerge from these two situations typically impact on the process performance. In the case of processes that need to adhere to *compliance* rules (e.g., compliance with laws, regulations or standards), these problems may lead to compliance violations that may in turn be translated into hefty penalties to the company.

  Typically, these problems manifest themselves in the process execution data generated as processes are executed. It is relevant, therefore, to make use of this data to analyze and understand the actual behavior of process executions. However, doing so is not trivial: we need to understand what insights we want to gain from process execution data (e.g., compliance level of process executions), what techniques to use for the analysis (e.g., root cause analisys), which data to use from the available process execution data, how to prepare the data to bring them to a format that is suitable for the analysis, how to deal with the presence of *uncertainty* in data, and how to report on the results in order to properly inform the process analysts.

- **RQ3:** *How can we reuse the process modeling wisdom of the crowd to assist process designers in defining processes?* In the previous research questions we focused on the process knowledge in the form of process execution data. Here, we look at the problem from the process design perspective and we ask how we can exploit existing process models to acquire knowledge that can help us in the identification of common practices in a given domain. The intuition behind this research question is that, by identifying such practices, we can analyze and understand how processes are modeled and incorporate such knowledge in future process designs. The identification of common practices involves analyzing the set of existing process models and finding model patterns that are recurrently used in such models. When the number of models is large, manually finding such patterns may turn into a daunting task and therefore automated techniques are needed in order to facilitate the work. The key research challenges that emerge in this problem are, first of all, the identification and definition of the types of patterns that can be useful from a process design perspective and the development of algorithms that are able to find interesting patterns from a model repository.

  Given that we are able to discover useful knowledge from a set of process models, the next question is how to reuse such knowledge for the improvement of processes. Since these patterns are obtained from process models that are the results of the design phase of a business application, we are interested in investigating on how to reuse this knowledge for assisting process modelers in designing their processes. To do so, we need to understand when we can assist users in designing their processes. For example, we need to decide if the assistance will be provided before, during or after process design. Then, we need to investigate on the form (e.g., model patterns) and granularity (e.g., model fragments) in which the assistance will be delivered, and, finally, we need to decide how to operationalize the delivery of such assistance.

The research questions discussed previously untangle the problem of how to exploit existing process knowledge for process improvement from two different perspectives, namely, the process execution and process design perspective. Although different, these perspectives complement each other and represent the foundation of a holistic approach to process improvement.

1.2 Contributions

Addressing the research questions discussed in the previous section asks for models, techniques and tools that facilitate process improvement by leveraging on the process knowledge generated by information systems. In this dissertation, we propose to exploit such process knowledge in order to produce analysis models that serve not only for the analysis and understanding of processes but also for reuse in the improvement of processes. The contributions (Cs) of this dissertation are:

- **C1: Process execution log reconstruction.** The contributions we make in addressing research question **RQ1** can be summarize as follows: (i) we characterize the problem of process execution log reconstruction from operational databases and identify the key challenges in addressing it, (ii) we propose an approach for addressing this problem, which includes core steps such event identification, event ordering, data mapping and event correlation, (iii) we propose a set of *eventification patterns* that can be applied in each of these steps, and (iv) we provide a tool, the *Eventifier*[1], that assists human experts in reconstructing process execution logs from operational databases.

- **C2: Root cause analysis and process discovery from uncertain process execution data.** In order to address research question **RQ2**, we first identify and analyze the information needed to carry out our analysis. As result, we propose a *data warehouse model* for storing process execution data. Here, we also introduce the ideas of *uncertain events* and present a model to express and store uncertainty metadata inside our data warehouse. Using these models, we propose the idea of *uncertain key indicators* and a tool to effectively compute an report on such indicators and show how to analyze process execution data using root cause analysis and process discovery techniques for the case where uncertainty is present in the data. Overall, our main contribution here is in providing the basis for uncertainty in process execution data analysis and business intelligence applications.

- **C3: Compliance reporting and analysis suite.** Leveraging on the contributions discussed above, and still in the frame of the research question **RQ2**, we show how we successfully applied our approach for *SOA-enabled compliance management*, in particular, for the case of service-based business processes. Here, we propose an assisted compliance management methodology based on the Deming cycle [9] and an event-based compliance management architecture that instruments our compliance management methodology. We also propose a reporting and analysis suite to report on compliance and support root cause

[1]https://sites.google.com/site/dbeventification/

analysis to provide better informed decision making.

- **C4: Process model pattern discovery.** In order to address the research question **RQ3**, we propose to leverage on community process modeling knowledge by discovering process model patterns from existing models. We investigate this research question in the context of dataflow-based mashups [29]. While this technology serves a different purpose than business processes, they both share a process-oriented approach and therefore our contributions can be also adopted for the case of business processes. In concrete, our contributions can be summarized as follows: (i) we propose a *canonical mashup model* that is able to represent in a single modeling formalism a variety of dataflow-based mashup languages with the goal of mining dataflow patterns from multiple source languages by implementing the necessary algorithms only once, (ii) based on our canonical mashup model, we define a set of *mashup pattern types* that resemble the modeling actions of typical dataflow-based mashup environments, (iii) we develop a set of data mining algorithms that discover composition knowledge in the form of reusable dataflow mashup patterns from a repository of mashup models, and (iv) we describe an architecture that can be used for mining patterns and building a pattern knowledge base.

- **C5: Process model pattern reuse for assisted mashup development.** The results of our research on process model pattern discovery has been used as the basis for building the knowledge base of our assisted mashup development approach. In this approach, we propose to assist users in designing mashups by means of interactive, contextual recommendations of composition knowledge that comes in the form of reusable model patterns. Thus, we show how to reuse the discovered process model patterns in practice for assisting process designers in defining their processes (research question **RQ3**).

In summary, the contributions described above leverage on existing process knowledge taking the perspectives of both process design and execution. The concepts, models, algorithms and tools described above provide a powerful mechanism for aiding in the analysis, understanding and reuse of process knowledge for facilitating process improvement.

1.3 Adoption of Results and Impact

The novel contributions discussed in the previous section regarding the computation of indicators, root cause analysis and process discovery from uncertain data, and compliance reporting and analysis have been adopted in the European projects MASTER[2] and COMPAS[3]. Inside these projects, we developed prototype tools that served us as proof of concepts and allowed us to perform user studies. The results obtained from our work on process log reconstruction is currently being adopted in the research project Ianus [13] from the Province of Trento (Italy) to enable process discovery.

In the project OMELETTE[4], the novel contributions regarding process model pattern discovery and reuse in assisted mashup development serve as basis for and are being adopted in the assisted development for the tools MyCocktail and OMELETTE's Live Environment (the details about these tools can be found at the project's website). These contributions has been also adopted for building the knowledge base of Baya[5], our assisted mashup development tool for Yahoo! Pipes.

[2]Managing Assurance Security and Trust for Services - http://www.master-fp7.eu

[3]Compliance-driven Models, Languages, and Architectures for Services - http://www.compas-ict.eu/

[4]Open Mashup Enterprise Service Platform for Linked Data in the Telco Domain - http://www.ict-omelette.eu/home

[5]http://www.lifeparticipation.org/baya.html

1.4 How to Read this Dissertation

This dissertation takes the form of an executive summary with a collection of research papers that have been peer-reviewed and published in international journals, conferences or books. The collection of papers can be found in the Appendix and is divided into two parts. In Part I, we discuss how we exploit process knowledge from the process execution perspective, while Part II does the same but from the process design perspective. Each part presents a list of publications that address the research questions and discuss in more details the approach and contributions presented previously. Figure 1.1 presents a graphical representation of our publications where we include a brief abstract for each of them and their interdependencies, and shows which publication falls into each of the process execution or process design perspective. In the following, we discuss the development of our research in each of these parts. Our aim is to use the learning process we went through as a guide on how to read this dissertation.

Part I - Process execution perspective. Our first interest was to analyze the behavior of processes from a high-level view by the use of key indicators (i.e., numbers that summarize different aspects of the process) computed over process execution data. We acknowledged that process execution data is not always perfect and it may be subject to uncertainty. Thus, to address this problem, we introduced an uncertain event model and key indicators together with a model for storing and a tool for computing and visualizing uncertain key indicators [16] [17]. The lessons learned from these works led us to the conclusion that it is not enough to just compute the right indicators in the right way, but we also need to visualize the results appropriately in order to deliver the right message to the users of such information. In the light of this observation, and starting from a compliance management problem, we therefore modeled the problem of

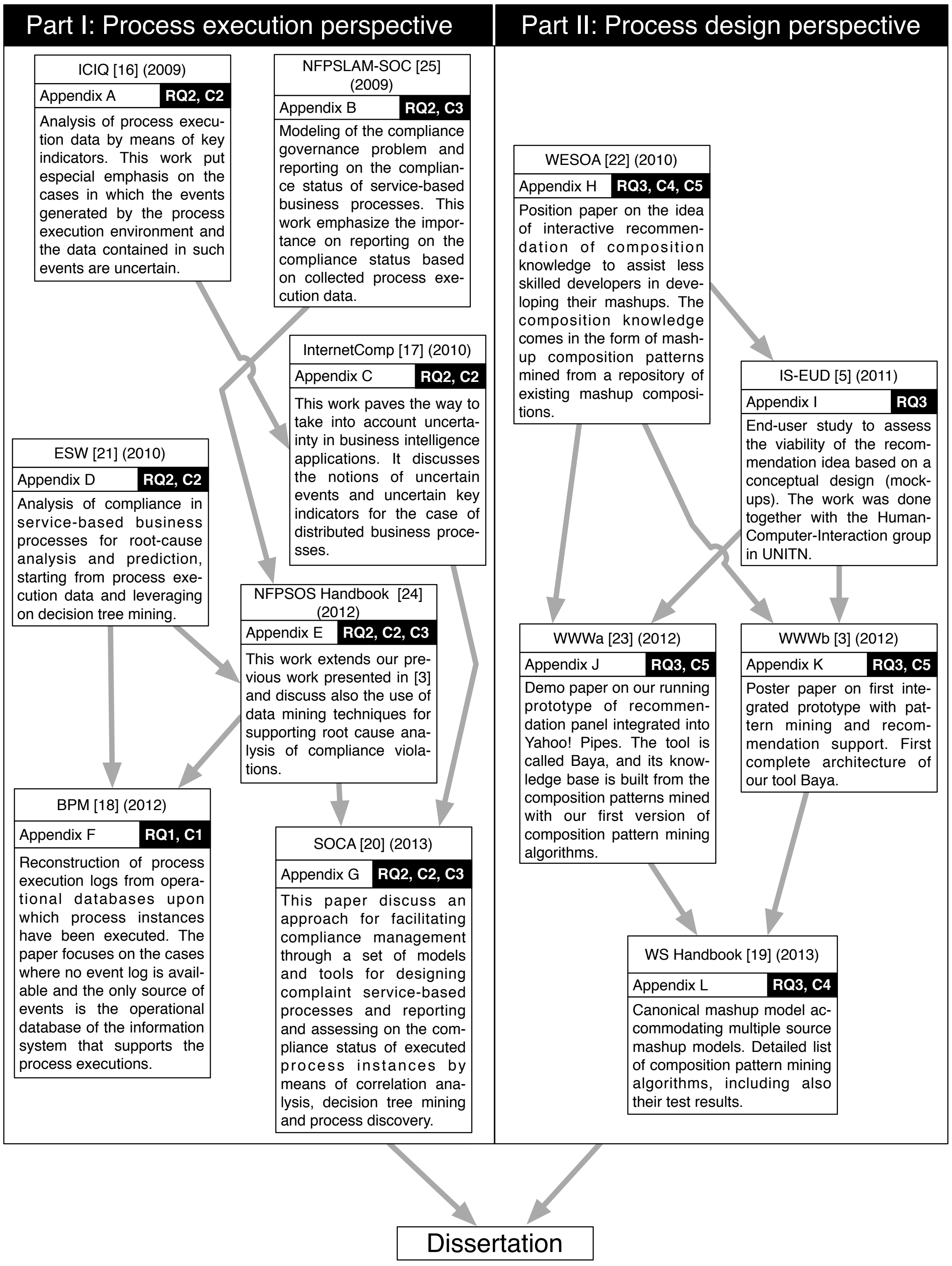

Part I: Process execution perspective
Part II: Process design perspective
ICIQ [16] (2009)
Appendix A RQ2, C2
Analysis of process execution data by means of key indicators. This work put especial emphasis on the cases in which the events generated by the process execution environment and the data contained in such events are uncertain.
NFPSLAM-SOC [25] (2009)
Appendix B RQ2, C3
Modeling of the compliance governance problem and reporting on the compliance status of service-based business processes. This work emphasize the importance on reporting on the compliance status based on collected process execution data.
InternetComp [17] (2010)
Appendix C RQ2, C2
This work paves the way to take into account uncertainty in business intelligence applications. It discusses the notions of uncertain events and uncertain key indicators for the case of distributed business processes.
ESW [21] (2010)
Appendix D RQ2, C2
Analysis of compliance in service-based business processes for root-cause analysis and prediction, starting from process execution data and leveraging on decision tree mining.
NFPSOS Handbook [24] (2012)
Appendix E RQ2, C2, C3
This work extends our previous work presented in [3] and discuss also the use of data mining techniques for supporting root cause analysis of compliance violations.
BPM [18] (2012)
Appendix F RQ1, C1
Reconstruction of process execution logs from operational databases upon which process instances have been executed. The paper focuses on the cases where no event log is available and the only source of events is the operational database of the information system that supports the process executions.
SOCA [20] (2013)
Appendix G RQ2, C2, C3
This paper discuss an approach for facilitating compliance management through a set of models and tools for designing complaint service-based processes and reporting and assessing on the compliance status of executed process instances by means of correlation analysis, decision tree mining and process discovery.
WESOA [22] (2010)
Appendix H RQ3, C4, C5
Position paper on the idea of interactive recommendation of composition knowledge to assist less skilled developers in developing their mashups. The composition knowledge comes in the form of mashup composition patterns mined from a repository of existing mashup compositions.
IS-EUD [5] (2011)
Appendix I RQ3
End-user study to assess the viability of the recommendation idea based on a conceptual design (mockups). The work was done together with the Human-Computer-Interaction group in UNITN.
WWWa [23] (2012)
Appendix J RQ3, C5
Demo paper on our running prototype of recommendation panel integrated into Yahoo! Pipes. The tool is called Baya, and its knowledge base is built from the composition patterns mined with our first version of composition pattern mining algorithms.
WWWb [3] (2012)
Appendix K RQ3, C5
Poster paper on first integrated prototype with pattern mining and recommendation support. First complete architecture of our tool Baya.
WS Handbook [19] (2013)
Appendix L RQ3, C4
Canonical mashup model accommodating multiple source mashup models. Detailed list of composition pattern mining algorithms, including also their test results.
Dissertation

Figure 1.1: Timeline of scientific, peer-reviewed publications with the respective contributions and interdependencies. For each publication we include the research questions (RQs) and contributions (Cs) the publication is related to.

reporting on compliance and developed an approach for reporting on it together with a prototype reporting dashboard [25]. We also introduced the concept of Key Compliance Indicators (KCIs) and applied techniques from decision tree mining for the analysis of compliance violations [24] [21].

From the lessons learned in these research works, we proposed a methodology for compliance management in a service-based business process environment [20]. The methodology is based on the *Deming cycle* [9] and it aims at the continuous improvement of the compliance of processes. Here, we employ in a unified and harmonized way the approaches we proposed in our previous works and added two additional techniques for the root cause analysis of compliance violations, namely, key indicator correlation and process discovery, both adapted to work on uncertain data. By applying these approaches to the Ianus project, we noticed that, in many real settings, the information systems supporting the execution of processes either do not generate event logs that are suitable for process execution data analysis, or they do not generate event logs whatsoever (this is especially true in the case of legacy systems). This led us to an additional line of work in which we address the problem of obtaining process execution data from an alternative source, in this case, the operational database of the information system supporting the business [18]. While this research work has been developed having in mind the reconstruction of an event log for process discovery, the work can be adapted to support other types of process execution data analysis that requires process execution data in the format of an event log.

Part II - Process design perspective. As shown in Figure 1.1, shortly after we started to work on the approaches based on process execution data, we initiated in parallel our work from the process design perspective. As opposed to the previous perspective, we carried out our studies in a different context, namely, dataflow-based mashups [29]. The purpose of dataflow-based mashups is different from that of business processes, but they both share a process-oriented approach. Moreover, dataflow-based mashups (like in Yahoo! Pipes) are analogous to service-based business processes in that both use a process-based logic to orchestrate services.

More precisely, the scenario we worked with inside the context mentioned above is that of *assisted mashup development.* We started with the proposal of assisting users without programming skills in building their mashups through development recommendation of composition knowledge, where this knowledge comes in the form of composition patterns that are mined from a repository of mashup models [22]. In order to better understand the requirements from our target users, we run an end-user study where the viability of the proposed approach was analyzed based on conceptual designs (mockups) [5].

Based on the results obtained from the study above, we continued with the progress on the research work and many artifacts where created: we designed a canonical mashup model for dataflow-based mashups, identified a set of mashup composition patterns types, developed algorithms for mining these patterns and provided a tool for the interactive recommendation of composition patterns for the mashup tool Yahoo! Pipes [23] [3]. The tool is called *Baya*, and its knowledge base, made of mashup composition patterns, is built with the mining algorithms we describe in details in [19]. While our algorithms developed for automatically mining composition patterns were effective in finding patterns that structurally make sense, a main limitation of the resulting patterns was the lack of rich semantics. This led us to a new research direction that we are currently investigating and that we will further discuss as part of the future work.

We conclude this dissertation with Chapter 2 where we outline the lessons learned, limitations of our approach and directions for future work.

Chapter 2

Lessons Learned

In this chapter, we discuss the lessons learned during the development of our research work. To do so, we group together our discussion based on the topics we investigated for addressing the research questions discussed in this dissertation. We include not only the lessons learned during the development of our research work, but also the limitations we identified in each case and the ongoing and future work.

2.1 Uncertain Events and Uncertain Business Intelligence

Uncertainty is a real issue in modern data management. Indeed, the database community (both academia and industry) has already started investing considerable amount of effort into research on uncertain and probabilistic databases, yet there is a lack of business process management and business intelligence applications that are able to profit from the results of such research. The research we did in this area follows in the footsteps of other scientific areas where uncertainty has become a key ingredient when modeling reality. As opposed to what is typically done when facing uncertain data, such as, data cleaning and other types of approaches with the purpose of providing clean and complete data to the end-users, we model this imprecision in terms of uncertain events and uncertain indicators and propose an approach to store uncertainty metadata and compute uncertain indicators. The bottom line of all this is that we do not want to hide this uncertainty from end users. Instead, we want to expose this information to them in order to create awareness about the existence and extent of this uncertainty.

Among the lessons learned from this research work we recognize that dealing with uncertain data is a non-trivial task. We acknowledge that we need to find the right level of complexity for the analysis models and indicators computed on top of the uncertainty models we proposed, in order to make them both informative and very clear to the process analysts. Failing to do so may not only put in risk the objective of creating awareness about uncertainty in the data, but also complicate the work of the process analyst. The application of our approach in practice does not come for free. On the one hand, making computations that take uncertainty into account requires more sophisticated algorithms, which, in turn, are computationally more intensive than traditional algorithms. On the other hand, there is a need for instrumenting the necessary mechanisms to obtain uncertainty metadata. This is, however, not very different from what existing approaches from data quality management have to face, where the data quality systems and experts need to address similar issues to make sure they provide clean and complete data [15]. Moreover, to address this challenge, we too can benefit from the approaches coming from data quality management.

2.2 Reporting on Compliance

Compliance with regulations, laws and standards has become a very relevant concern in modern business. We are witnessing how compliance concerns are permeating more and more business operations and, thus, how they are invading business processes. With the increasing number of compliance concerns a modern business has to deal with, the effective visualization and reporting on the compliance status of processes is becoming a very important aspect in any organization. In this regard, our contribution consists in the conceptualization of the issues involved in the design of compliance dashboards for service-based business processes and a model for both storing compliance-related data and supporting the dashboard navigation in a drill-down, roll-up fashion for exploring the compliance concerns at different levels of detail. Our solution has been devised having in mind the needs of auditors (both internal and external ones) and with the help of experts from auditing companies, namely, PriceWaterhouseCoopers and Deloitte.

The main findings in this line of work includes that reporting on compliance is not a straightforward task especially when we need to combine compliance concerns with a process-oriented reporting approach. In order to effectively report on the compliance status of business processes we need to find the appropriate aggregation of compliance-related information and correct association of such information to process-related concepts. A key concern expressed by the auditor experts is that no matter how good a reporting solution is, if auditors identify a compliance violation or a flaw in the process or controls, they will go right to the logs for a fine-grained analysis. A reporting solution that provides support for this requirement benefits the company in two ways: on the one hand, it allows internal auditors and business process analysts to drill down to the events that originated the indicator values for finding the root causes of compliance violations, and, on the other hand, it positively impacts external auditors by showing that there are mechanisms in place for monitoring the compliance status of business processes.

In the solution we proposed for reporting on compliance we focus on the compliance of business processes, and the highest level of abstraction we reach is the business unit. Thus, with our solution we do not address higher level concerns such as organizational compliance and the problem of how it relates to compliance at the business process level.

2.3 Compliance Management in Service-based Business Processes

We now switch to discuss our work in a relevant and critical issue in todays business reality: compliance management. In this dissertation, we also consider the problem of process improvement from a compliance management viewpoint taking into account the peculiarities of service-oriented architecture and distributed business contexts. In addressing this problem, we kept the perspective of auditors (both internal and external ones) and focus on the design of compliant processes and the assessment and improvement of their compliance. The models and instruments we propose in our work complement existing monitoring and enforcement approaches and provides a comprehensive approach to service-based compliance management.

From the realization of our approach we learned that compliance management cannot be *automated* completely and that we always need a human expert in the loop and, therefore, the most we can do is to *facilitate* compliance management through instruments that help human experts in its various phases. We also learned that a key factor to successful compliance management is the appropriate internalization of compliance concerns into business processes, i.e., a proper translation of the compliance sources into compliance rules that business processes must comply with.

The generation of the right process progression information, typically, in the form of event logs, is just as important as the internalization of compliance concerns since they are the key sources of data for compliance assessment. Yet, there are cases in which the generated logs are not suitable for the analysis techniques we proposed in our work or they are simply not generated at all. This is especially true when parts of the business processes of a company are supported by legacy systems, and thus, sources different from an event log need to be queried to obtain the process progression information needed. In this context, we learned that a clear eventification process and a set of sensibly designed heuristics and tools can enable a domain expert to reconstruct an event log even from an operational database, an IT component that most companies running an information system have. We also recognized that a fully automated approach is not possible and it requires involving the domain expert into the decisions to be taken. Finally, while we focus on event logs that enable process discovery, the approach we proposed can reconstruct process execution logs that meet the requirements of other types of process execution data analysis. The main task here is in identifying the right event types that we need to reconstruct as required by the analysis to be performed.

2.4 Process Model Pattern Mining for Reuse

In our approach of mining process model patterns for reuse, we leverage on community composition knowledge. As explained before, we took the scenario of assisted mashup development in which we aimed to help users in building dataflow-based mashups through interactive recommendation of composition knowledge that comes in the form of mashup composition patterns. Thus, one of the key challenges we faced here was the identification of composition patterns that are useful from the point of view of the typical mashup composition actions made by users and that can be interactively recommended to them during composition. While we were able to identify such patterns using Yahoo! Pipes as an exemplary tool, we soon realized that other similar tools for dataflow-based mashups use very similar models for representing their mashups and this led us to the development of a mashup canonical model on top of which we developed a set of pattern mining algorithms for the pattern types we had identified in the previous phases. The purpose of doing this was the development of a single set of mining algorithms that is abstracted from the specific mashup representations used by different tools. Time proved us right as we were indeed able to reuse the same algorithms to mine patterns from two other mashup tools developed in the context of the European project OMELETTE.

As for the discovery of model patterns, we learned that even patterns with very low support carry valuable information. Even though they do not represent generally valid solutions or complex best practices in a given domain, they still show how its constructs have been used in the past. This property is a positive side-effect of the sensible, a-priori design of the pattern types we are looking for. Without that, discovered patterns would require much higher support values, so as to provide evidence that also their pattern structure is meaningful. Our analysis of the patterns discovered by our algorithms shows that, in order to get the best out them, semantically rich information inside the composition patterns is crucial.

2.5 Final Remarks and Future Work

The managerial view on business process improvement is wide and includes improvement in business processes, manufacturability, market share, customer satisfaction, profits, employee performance, among other issues. In this dissertation, we take the view of process improvement from an IT viewpoint and propose to exploit business process knowledge managed by the supporting information systems from the perspectives of process design and process execution. We consider these two perspectives as proactive and reactive, respectively. The process design perspective is proactive because having a good design in the first place is important to prevent processes from being ineffective or inefficient. The process execution perspective is reactive because we analyze the actual behavior of the processes after they were executed and only then we identify problems and their corresponding solutions and improvements. The models, algorithms and tools we propose in this dissertation for each of these perspectives allow for gaining great visibility into and understanding of the processes of an organization and also profiting from this knowledge, thus, facilitating business process improvement.

The lessons learned and outcomes of this dissertation ask for a continuation of our research work. In the following, we discuss the future work that aim to address some of the limitations discussed above along with other new concerns:

- In the context of *uncertain business intelligence*, investigate the adoption and extension of further data mining algorithms that can take advantage of our uncertainty data model. Additionally, since the algorithms operating on uncertain data are computationally more intensive, we need to study how our approach scales for large volumes of data.

- Extend our approach for *process log reconstruction* to consider cases where no timing information is associated to the records in the database. Study also how to automatically extract and associate task names to events, and how to recommend tables where potentially relevant, process progression events can be found.

- Investigate how to adapt our *compliance reporting* approach to consider the case of reporting on and navigating through compliance concerns and indicators at multiple levels.

- Adapt our mashup model pattern mining algorithms for their use on business processes models that include control flow constructs.

- In order to address the lack of rich semantics in the patterns resulting from our model pattern mining algorithms, study how to use human computation in a crowdsourcing environment for actively involving human experts in finding patterns with semantically rich information from process models. We have already started to work on this interesting research issue.

Bibliography

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web services: concepts, architectures and applications.* Springer, 2003.

[2] K.S. Cameron, S.J. Freeman, and A.K. Mishra. Downsizing and redesigning organizations. *Organizational change and redesign*, pages 19–63, 1993.

[3] Florian Daniel, Carlos Rodríguez, Soudip Roy Chowdhury, Hamid R. Motahari Nezhad, and Fabio Casati. Discovery and reuse of composition knowledge for assisted mashup development. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 493–494. ACM, 2012.

[4] T.H. Davenport and J.E. Short. Information technology and business process redesign. *Operations management: critical perspectives on business and management*, 1:1–27, 2003.

[5] Antonella De Angeli, Alberto Battocchi, Soudip Roy Chowdhury, Carlos Rodríguez, Florian Daniel, and Fabio Casati. End-user requirements for wisdom-aware eud. *End-User Development*, pages 245–250, 2011.

[6] M. Hammer and J. Champy. *Reengineering the corporation: A manifesto for business revolution.* HarperBusiness, 2003.

[7] D.H.J. Harrington. *Business process improvement: The breakthrough strategy for total quality, productivity, and competitiveness.* McGraw-Hill, 1991.

[8] J.M. Juran. *Juran on leadership for quality.* Free Press, 2003.

[9] G.K. Kanji. Implementation and pitfalls of total quality management. *Total Quality Management*, 7(3):331–343, 1996.

[10] R. Kimball and M. Ross. *The data warehouse toolkit: the complete guide to dimensional modelling.* Wiley, 2002.

[11] F. Leymann and D. Roller. *Production workflow: concepts and techniques.* Prentice Hall PTR, 2000.

[12] Mail Online. Santa's very busy at Amazon UK: Website had 3million transactions on 'Cyber Monday' with a frenzy for the site's Kindle. URL: http://goo.gl/OeEed. Last access: Jan 16, 2013.

[13] Opera21. Ianus Project. URL: http://goo.gl/GN9Gr. Last access: Jan 16, 2013.

[14] M.P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: State of the art and research challenges. *Computer*, 40(11):38–45, 2007.

[15] E. Rahm and H.H. Do. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 23(4):3–13, 2000.

[16] Carlos Rodríguez, Florian Daniel, Fabio Casati, and Cinzia Cappiello. Computing uncertain key indicators from uncertain data. In *Proc 14th Intl Conf. Information Quality*, pages 106–120, 2009.

[17] Carlos Rodríguez, Florian Daniel, Fabio Casati, and Cinzia Cappiello. Toward uncertain business intelligence: The case of key indicators. *Internet Computing, IEEE*, 14(4):32–40, 2010.

[18] Carlos Rodríguez, Robert Engel, Galena Kostoska, Florian Daniel, Fabio Casati, and Marco Aimar. Eventifier: Extracting process execution logs from operational databases. In *Proc. Demo Track 10th Int'l Conf. on Business Process Management*, pages 17–22, 2012.

[19] Carlos Rodríguez, Soudip Roy Chowdhury, Florian Daniel, Hamid R. Motahari Nezhad, and Fabio Casati. Assisted mashup development: On the discovery and recommendation of mashup composition knowledge. *Web Services Handbook, Springer (in press).*

[20] Carlos Rodríguez, Daniel Schleicher, Florian Daniel, Fabio Casati, Frank Leymann, and Sebastian Wagner. SOA-Enabled Compliance Management: Instrumenting, Assessing and Analyzing Service-Based Business Processes. *Journal of Service Oriented Computing and Applications (SOCA), Springer.* DOI: 10.1007/s11761-013-0129-3, URL: http://link.springer.com/article/10.1007%2Fs11761-013-0129-3.

[21] Carlos Rodríguez, Patrícia Silveira, Florian Daniel, and Fabio Casati. Analyzing compliance of service-based business processes for root-cause analysis and prediction. *Current Trends in Web Engineering*, pages 277–288, 2010.

[22] Soudip Roy Chowdhury, Carlos Rodríguez, Florian Daniel, and Fabio Casati. Wisdom-aware computing: on the interactive recommendation of composition knowledge. *Service-Oriented Computing*, pages 144–155, 2011.

[23] Soudip Roy Chowdhury, Carlos Rodríguez, Florian Daniel, and Fabio Casati. Baya: assisted mashup development as a service. In *Proceed-*

ings of the 21st international conference companion on World Wide Web, pages 409–412. ACM, 2012.

[24] Patrícia Silveira, Carlos Rodríguez, Alexander Birukou, Fabio Casati, Florian Daniel, Vincenzo DAndrea, Claire Worledge, and Zouhair Taheri. Aiding compliance governance in service-based business processes. *Non-Functional Properties for Service-Oriented Systems: Future Directions (NFPSLA-BOOK-2011) Edition, IGI Global*, 2011.

[25] Patrícia Silveira, Carlos Rodríguez, Fabio Casati, Florian Daniel, Vincenzo DAndrea, Claire Worledge, and Zouhair Taheri. On the design of compliance governance dashboards for effective compliance and audit management. In *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*, pages 208–217. Springer, 2010.

[26] W.M.P. Van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes.* Springer, 2011.

[27] M. Weske. *Business Process Management.* Springer, 2012.

[28] S.A. White et al. Business process modeling notation. *Specification, BPMI. org*, 2004.

[29] J. Yu, B. Benatallah, F. Casati, and F. Daniel. Understanding mashup development. *Internet Computing, IEEE*, 12(5):44–52, 2008.

Part I

Process Execution Perspective

Appendix A

Computing Uncertain Key Indicators from Uncertain Data

Computing Uncertain Key Indicators from Uncertain Data *

Carlos Rodríguez Florian Daniel Fabio Casati Cinzia Capiello

Abstract

Key indicators, such as key performance indicators or key compliance indicators are at the heart of modern business intelligence applications. Key indicators are metrics, i.e., numbers, that help an organization to measure and assess how successful it is in reaching predefined goals (e.g., lowering process execution times or increasing compliance with regulations), and typically the people looking at them simply trust the values they see when taking decisions. However, it is important to recognize that in real business environments we cannot always rely on fully trusted or certain data, yet indicators are to be computed. In this paper, we tackle the problem of computing uncertain indicators from uncertain data, we characterize the problem in a modern business scenario (combining techniques from uncertain and probabilistic data management), and we describe how we addressed and implemented the problem in a European research project.

1 Introduction

Facilitated by the extensive use of Information Technology (IT) in today's companies, business environments have become highly dynamic and responsive. Especially the growing availability of business data that are accessible for analysis and interpretation has changed the way business people read the performance of their company: increasingly, they base their decisions on summaries, reports, and analyses coming from Business Intelligence (BI) applications. In order to gain competitive advantage over their competitors, BI applications allow them to get insight into the changes in the business environment, to rapidly react to changes, and to keep performance under control. With the advent of so-called

*The final publication is available at http://mitiq.mit.edu (http://mitiq.mit.edu/iciq/iqpapers.aspx?iciqyear=2009) process-aware information systems (such as Business Process Management systems), we are now also in the presence of large amounts of data regarding the execution of business processes, and, hence, business people also have the possibility to access not only business data (e.g., the amount of sales in a particular month) but also execution data (e.g., who executed a given activity in a business process and how long did it take him to complete the task). The analysis of such kind of business process execution data is the focus of so-called Business Process Intelligence (BPI) applications and of this paper.

One of the most important instruments used to report on the state of a company's business are Key Indicators (KIs), which are metrics that allow a company to measure its performance against specific objectives. Their value mainly lies in their simplicity: with one number they summarize a huge amount of data and, at the same time, intuitively describe a well-specified part of business reality. The use of alarm levels and colours further enhances their readability and (cognitive) accessibility. Typically, indicators like KPIs (key performance indicators) measure the achievement of business objectives (e.g., the average revenue of a given department), but there are also indicators that rather focus on risk (key risk indicators), compliance with laws or regulations (key compliance indicators), and similar. In the last years, great attention has been paid to the automated computation of KIs over business and process execution data.

The advantages provided by BI and BPI applications and the computation of KIs are possible thanks to advanced technologies used to store large amounts of data reflecting the whole lifecycle of a company's business in a continuous form (typically, we talk about data warehouses). However, the speed at which data are generated, combined, and processed by means of various technologies, software tools, and human actors, the quantity of the available data, plus the fact that today's business scenarios are highly interlinked, i.e., companies do not act in an isolated fashion from an IT point of view (e.g., companies share parts of their business processes with strategic partners or they outsource part of their IT infrastructure and business processes to specialized companies), inevitably leads to data with quality problems: logged data may contain errors or noise, incomplete or inconsistent data flows, etc. For example, if the bus or the logging system suffer from bad configuration, overload, performance problems, or downtimes we might not be able to log all the important messages flowing through an enterprise service bus (e.g., to compute indicators).

Computing KIs from data that are characterized by low quality (i.e., uncertain data) demands for novel and sophisticated algorithms, able to take into account the quality of the data. As a matter of fact, KIs themselves will be uncertain. Not taking into account the uncertainty that characterizes an indicator during its computation could give the people looking at the final value of the indicator a wrong perception of the actual performance of the business and might cause them to take wrong decisions, which eventually could negatively affect their business.

In many situations, the huge amount of potentially uncertain data combined with the need for continuously computing and re-computing KIs, makes the effort of running complex correction procedures (if any) prohibitive and impracticable. Yet, business people need to keep computing KIs in order to keep track of business performance while taking into consideration that indicators are computed on uncertain data. That is, decision makers must be aware of the quality of their indicators at the time of taking decisions concerning their business.

Contributions. Computing expressive and meaningful indicators from uncertain data is a challenging and tricky endeavour. In this paper, we approach the problem from both a theoretical and a practical perspective. Specifically, we:

- Characterize the problem of computing key indicators in distributed business environments as a data quality problem that is specifically related to uncertain/probabilistic data;
- Propose an approach to compute values for key indicators from uncertain/probabilistic data based on techniques from uncertain data management;
- Introduce the concept of uncertain/probabilistic key indicator and quantify uncertainties/probabilities starting from the data used in the computation of an indicator;

- Provide a concrete data warehouse model for the data needed to compute key indicators in the context of a European project, along with the corresponding extensions to deal with uncertainty;
- Hint at how to visualize indicators in order to convey to users the likelihood that an indicator takes a particular value considering the uncertainty in the input data.

Structure of this paper. In the next section we introduce a real-life reference scenario that will accompany us throughout the rest of the paper. Then, we conceptualize the described scenario and its business context and formally define the problem addressed in this paper. Based on this formalization, we describe the theoretical foundation for the computation of key indicators from uncertain data and, next, show how we compute uncertain indicators in practice. Finally, we describe our implementation of the proposed solution in the context of a European project, discuss related works, and draw our conclusions.

2 Reference Scenario

Let's consider a Network Information Center (NIC) that provides Internet domain name registration for a Top-Level Domain (TLD) in the Domain Name System (DNS) via the Web. The NIC is in charge of administrating the (fictitious) domain .sci, which is limited to organizations, offices and programs whose main interest resides in any kind of science. For example, the organization abc that does research in nanoscience could register the domain name abc.sci to provide name resolution for Internet resources, such as mailing services or a web site (e.g., http://www.abc.sci).

For our scenario, we consider two business processes used by the company for administrating the TLD. The first business process consists in the delegation of domain names as shown in Figure 1. The model in this figure is a simple flowchart with swim lanes that show the distribution of tasks among stakeholders. The process can be divided into four parts: (i) insertion of the request (client), (ii) verification of the request (NIC), (iii) payment for the domain name (client/bank), and (iv) activation of the domain name (NIC). The

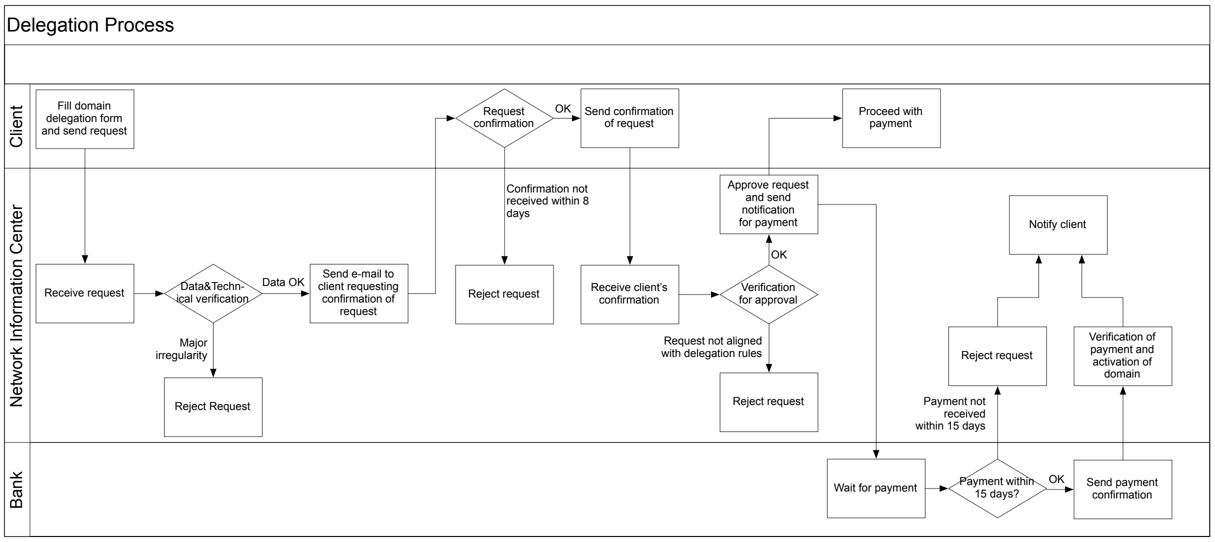

Delegation Process
Client
Network Information Center
Bank
Fill domain delegation form and send request
Receive request
Data&Technical verification
Data OK
Major irregularity
Reject Request
Send e-mail to client requesting confirmation of request
Request confirmation
OK
Confirmation not received within 8 days
Reject request
Send confirmation of request
Receive client's confirmation
Verification for approval
OK
Request not aligned with delegation rules
Reject request
Approve request and send notification for payment
Proceed with payment
Wait for payment
Payment within 15 days?
OK
Payment not received within 15 days
Reject request
Send payment confirmation
Verification of payment and activation of domain
Notify client

Figure 1: Process for the delegation of domain names. The process is operated by the NIC and the bank; clients are involved through the NIC's web application.

management of the payment is performed in conjunction with a bank that interacts with the NIC through web services, i.e., the NIC and the bank share part of their business processes. When the request for delegation is approved, the client proceeds with the payment via one of the channels offered by the bank. Upon reception of the payment, the bank notifies the NIC, which causes the NIC to automatically activate the domain name requested by the client.

The second business process (see Figure 2) is the procedure for modifying information related to the delegation of domains, such as data of the owner and technical details. The process is part of the customer support that the NIC has outsourced to an external company specialized in user support and providing services like a call center and a support web application. Both the call center and the web application are fully detached from the operational system of the NIC and managed by the Customer Support center. Yet, they provide a reduced set of views and operations on users' data.

The NIC is now interested in studying the performance of its business processes, in order to monitor and improve quality of service. For instance, the NIC is interested in computing the following key indicators:

- TBRP (time between request and payment): With this indicator, the NIC wants to capture how long in average it takes to the client to proceed with the payment for the domain name.

- SRSE (number of subsequent requests by the same entity that block a specific domain): This is an important indicator, since, once a request is inserted, no one else can request that same domain name, unless the request is cancelled or expired due to missing payment. This indicator helps the NIC to detect when somebody tries to keep a domain name blocked without willing to pay for it, e.g., to prevent others to acquire it.
- TBAS (time between the activation of a domain name and the first support request): This indicator provides an idea of how long it takes in average a user to contact the first time the support center upon the successful registration of a domain name. This allows the NIC, for instance, to assess the quality of the documentation provided in the phase of registration and to estimate the cost of the support service.

For the computation of the above indicators, the NIC instruments its Delegation process, which is mainly based on web services and the client's web application, so that the process generates the necessary information in form of events (we assume each activity in the process models may generate respective events). For instance, the NIC generates a ReceiveRequest event and has already agreed with the bank on the generation of a corresponding Payment-Confirmation event (along with a respective service level agreement ruling the quality of service of the event delivery), which are at the basis of the TBRP indicator. Similarly, the NIC provides for the events necessary to compute the SRSE indicator, which only involves events under the control of the NIC and, therefore, do not require any negotiation or agreement with either the bank or the support center. The computation of the TBAS indicator, instead, is trickier: the time of the activation of a domain name is easy to track (the NIC has control of that), but for the time of the first support request the NIC could only obtain a best effort commitment by the support center, which is already a good achievement. Indeed, in general the support center could also not have been willing at all to provide that information, able to do so, able to do so

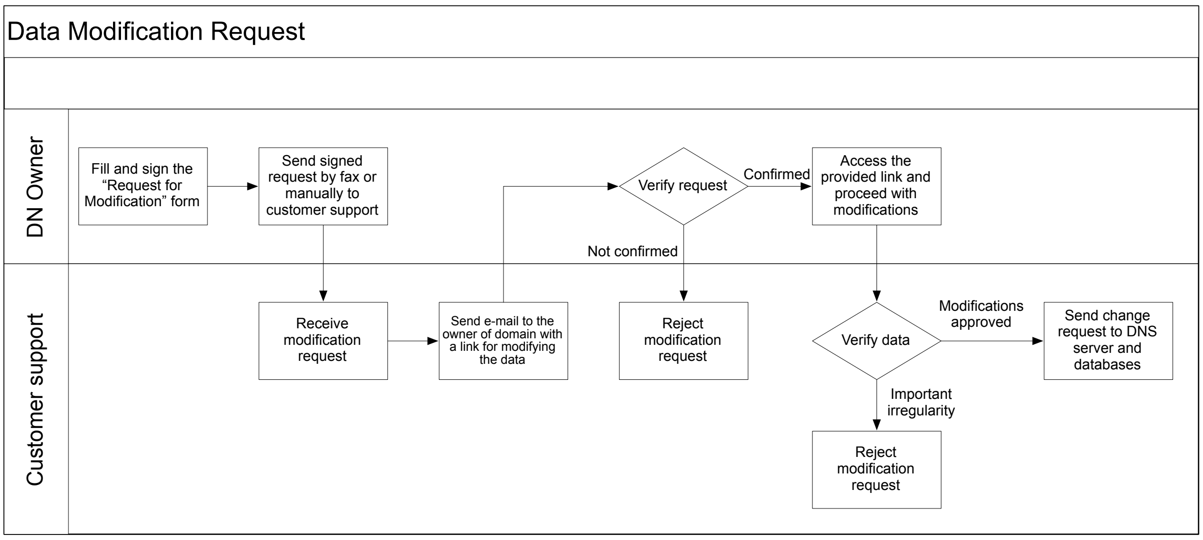

Data Modification Request
DN Owner
Customer support
Fill and sign the "Request for Modification" form
Send signed request by fax or manually to customer support
Verify request
Confirmed
Access the provided link and proceed with modifications
Not confirmed
Receive modification request
Send e-mail to the owner of domain with a link for modifying the data
Reject modification request
Verify data
Modifications approved
Send change request to DNS server and databases
Important irregularity
Reject modification request

Figure 2: The data modification request process.

reliantly, etc. After some days, the NIC looks at the events it could log and sees that, besides the availability or not of events, there are even other problems with the data in the log: events are not always logged correctly; some events seem to be wrong (but the NIC is not fully sure); some data values are not precisely defined, and similar. In short, the event log the NIC would like to use to compute its indicators may present data quality issues, yet the NIC needs to compute its indicators anyway.

3 Uncertainty and Probability in Business Environment

The above scenario demands for the computation of three key indicators related to the two business processes run by the cooperating parties. In this paper, we do not want to pose any restriction on how business processes are executed (e.g., manually vs. semi-automatically vs. automatically). However, in order to be able to automatically compute indicators, we assume that the data necessary for the computation of the indicators are available in the form of events that are generated by the cooperating partner's IT systems and that provide (partial) visibility into the execution of the business processes. We say that the business processes are instrumented in order to generate events. For instance, an ActivityStart event could be automatically generated by a business process engine, or a Reject event could be derived from an email sent by a physical person to an archiving system. In order to be able to compute meaningful indicators, events must carry some piece of business data (e.g., the name of the person approving or rejecting a domain name registration).

These assumptions and minimum requirements are realistic. Especially in presence of companies that cooperate over the Web and typically base their cooperation on web services and the service-oriented architecture, the generation of such kind of events is no big issue. Also, as we want to compute key indicators periodically for reporting purposes (e.g., each night or once a week) we assume that the events we are interested in are logged in a central event log that can be periodically inspected.

We represent a generic event $\bar{e}$ as a tuple $\bar{e} = \langle ID, procID, type, ts, src, dest, \bar{d}_1, ..., \bar{d}_n \rangle$ (note that we use the bar over symbols to indicate that they represent certain data; we will use symbols without the bar when instead they represent uncertain data), where ID is a unique identifier of the event, $procID$ is the unique identifier of both process instance and process model, $type$ specifies which kind of event we have (e.g., ActivityStart or Reject), ts is the timestamp in which the event has been generated, src and $dest$ are the source and the destination (if any) of the event (e.g., the company or business process instance that is the origin of the event), $\bar{d}_1, ..., \bar{d}_n$ are the parameters carrying possible business data values (they are the actual body or payload of the event, their number might vary from event type to event type). More specifically, each $\bar{d}_i$ is characterized as follows: $\bar{d}_i = \langle partype, name, value \rangle$ with $partype$ being the type of the parameter (e.g., integer, enumeration of string values, etc.), $name$ being the name of the parameter, $value$ being the concrete value assigned to the parameter.

Events are generated during the execution of business processes, and each business process in execution (i.e., a process instance) typically generates a multitude of events during its execution. As the only information we have about the executed process instances is the set of events generated by them, we represent a process instance as a trace of chronologically ordered events $\bar{t} = \bar{e}_1, ..., \bar{e}_n$. For instance, the above Delegation process could produce an event trace as follows: $t = \langle 0, DelegationProcess3, ProcessStart, 20090509144209, NIC, NIC, Customer, RegistrationReq\rangle, ..., \langle 27, DelegationProcess3, Notification, 20090604050648, 67, NIC, Client, Msg\rangle$ that tells us that there has been an instance of the Delegation process ($procID = DelegationProcess3$), started on May 9, 2009, which sent a notification with content Msg to the client on June 4, 2009. Finally, we represent the event log as a set $E = \{\bar{t}_i\}$.

Since now a KI summarizes execution data of multiple process instances (e.g., all the executions of the Delegation process of the last week), a key indicator is a function that is computed over a set of process instances, i.e., a set of events. More precisely, a KI is a function $\overline{KI} = \overline{KI}(\{\bar{t}_i\})$ over a set of event traces that assigns to each subset of $\{\bar{t}_i\}$ a real number (the indicator value), i.e., $\overline{KI} : \mathcal{P}(\{\bar{t}_i\}) \rightarrow \mathbb{R}$.

3.1 Data Quality in Event Logs

The problem in practice is that the event log E contains data (or not) that not always are fully aligned with the real world, i.e., with the concretely executed business processes. Inspired by [16], in this paper we distinguish four situations that are characteristic of the described business scenario. In this paper, we address the first three scenarios; we do not explicitly treat the fourth, as it rather represents a design time issue that is out of the scope of this paper:

1. ***Meaningless state*** *= there is an event in the event log, but we are not sure the corresponding real-world event indeed happened:* For instance, in Figure 3a there is an ActivityStart event in the log (row 234) but, as hinted at by the dotted tail of the arrow, we lack the corresponding counterpart in the real world (e.g., an employee sent an email that he started an activity but actually never performed the corresponding task). As a result, there might be events in the log that are uncertain.

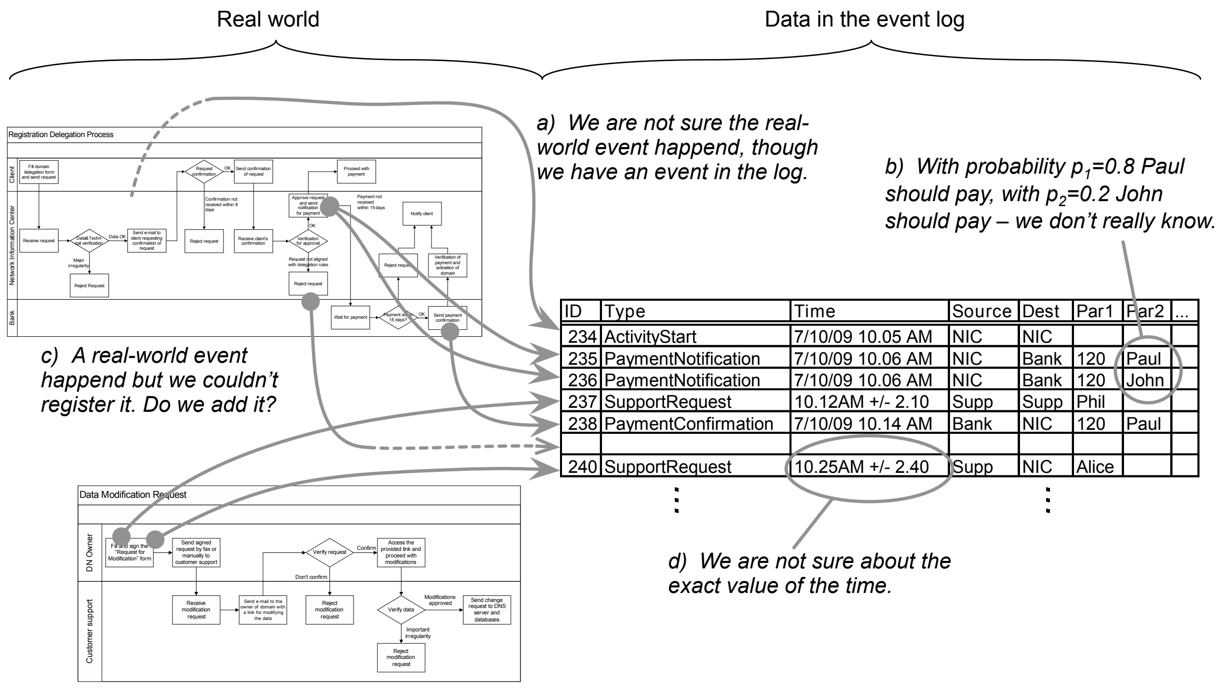
2. ***Uncertain data*** *= there is an event in the event log, but we are not sure about the exact data values carried inside the body of the event or, simply, whether values are correct* (Figure 3b): Rows 235 and 236 derive

from the same real-world event (the notification of the payment details to the client and the bank), yet we are not sure whether the notification has been sent to Paul or to John. Row 240, instead, presents an uncertainty regarding the exact time in which the event was generated.

3. ***Incomplete representation*** = *we think that a real-world event happened, but there is no corresponding event in the log:* As represented by the empty row in Figure 3c, there might be actions in the real world that should have been logged but that lack a corresponding event in the log. Such a lack could for instance be due to system failures or downtimes, network problems, people forgetting to send an email, or the like. In some cases, however, we might be able to derive that a real-world action must have happened from the business context that can be reconstructed from the log. For instance, if the event log contains a ProcessEnd event, very likely there also must have been a corresponding ProcessStart event.

4. ***Lack of representation*** = *we are not able to log all the events that are necessary to compute an indicator:* If a company, for instance, decides to outsource part of its business, it might lose visibility into the details of how an outsourcing partner actually performs its business, practically losing the events associated with the outsourced part of the business process. As a consequence, the company might no longer be able to compute an indicator, and a re-design of the indicator's computation logic is necessary if possible; otherwise, the computation of the indicator can simply not be performed any longer.

This casuistry shows that in realistic settings it is generally not a good idea to think that indicators can be computed straightaway from the data that can be found in an event log. The log might be incomplete (missing events), it might contain noise (wrong events), it might contain uncertainties regarding the correctness of tuples, or it might contain uncertainties regarding the exact value of data cells. Note that the computation of the degrees of uncertainty in the input data is outside of the scope of this paper.


Real world
Data in the event log
Registration Delegation Process
Client
Network Information Center
Bank
a) We are not sure the real-world event happend, though we have an event in the log.
b) With probability p_1=0.8 Paul should pay, with p_2=0.2 John should pay – we don't really know.
c) A real-world event happend but we couldn't register it. Do we add it?
d) We are not sure about the exact value of the time.
Data Modification Request
DN Owner
Customer support

ID	Type	Time	Source	Dest	Par1	Par2	...
234	ActivityStart	7/10/09 10.05 AM	NIC	NIC			
235	PaymentNotification	7/10/09 10.06 AM	NIC	Bank	120	Paul	
236	PaymentNotification	7/10/09 10.06 AM	NIC	Bank	120	John	
237	SupportRequest	10.12AM +/- 2.10	Supp	Supp	Phil		
238	PaymentConfirmation	7/10/09 10.14 AM	Bank	NIC	120	Paul	
240	SupportRequest	10.25AM +/- 2.40	Supp	NIC	Alice		
⋮				⋮			

Figure 3: Discrepacy between the real world and the data we might have in the event log.

3.2 Expressing Uncertainties and Probabilities

In order to be able to compute meaningful indicators from a real event log, we must be able to represent the above problems in the data we use to compute the indicators. The metrics that we use in this paper to keep track of data quality depend on the object of the quality problem; specifically, we associate: (i) reputation to events in order to express the likelihood that an event in the log corresponds to an event in the real world (covering the cases of meaningless states and incomplete representations); (ii) probabilities to data values in order to express alternatives or levels of confidence for discrete values (covering part of the data uncertainty case); and (iii) confidence intervals to data values in order to express doubts about the exact value of continuous, numeric values (covering the other part of the data uncertainty case). Taking into account reputation, probabilities, and confidence intervals demands for an extension of our event formalization.

So far, we defined an event as a tuple $\bar{e} = \langle ID, procID, type, ts, src, dest,$

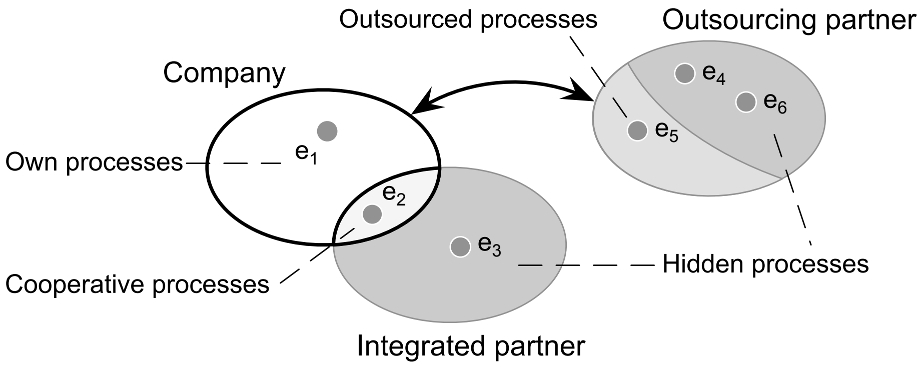


Figure 4: Reputation and visibility into business processes of cooperating partners.

$\bar{d}_1, ..., \bar{d}_n\rangle$, in which both the event and its parameter values were fully trusted. In order to associate a reputation value with each event and probabilities/confidence intervals with data values, we define an uncertain event as a tuple $e = \langle ID, procID, type, ts, src, dest, d_1, ..., d_n, rep\rangle$, where $ID, procID, type, ts, src, dest$ are as defined previously for $\bar{e}$ (note that, rep is the reputation associated to the event), and $d_1, ..., d_n$ are the business data parameter to which we associate probabilities or uncertainties, as described next. Note that in presence of uncertain date we now omit the bar on top of the symbols.

Modelling reputation. The association of a reputation level to an event can, for instance, be done by combing an objective and a subjective measure, i.e., an analysis of the data in the event log and the confidence we have in the correct operation by cooperating partners (i.e., their reputation). The objective measure can be derived by looking at how many meaningless state cases and incomplete representation cases we have in the log. The subjective measure typically stems from the reputation levels we associate to business partners; Figure 4 conceptualizes our cooperative business scenario and highlights reputation and visibility issues. First of all, the company (e.g., the NIC) runs own processes in-house; the probability that events are correctly registered in own processes is typically high (e.g., $p(e_1) = 0.99$). Next, a company might cooperate with an independent partner by means of a shared business process in which both partners participate and of which both have full visibility (e.g., the NIC cooperates with the bank); as the responsibility of the common process is shared among the two parties, the confidence in correct events is typically lower than the one we have in own events (e.g., $p(e_2) = 0.85$). Finally, if part of a business process is outsourced (e.g., to the Customer Support Center), the company has only a very limited visibility into the outsourced part of the business process and, hence, confidence in events might be lower again (e.g., $p(e_5) = 0.70$). For presentation purposes, we manually assign reasonable values to the five types of events; in practice, such values would be derived by a suitable reputation assessment system from historical data about the interacting parties.

We assume that for each event e, we know its provenance, i.e., the company $c \in C$ that generated the event, where C is the set of companies involved in the business processes over which we want to compute our indicators. The reputation level rep of e can then be seen as a function $rep : C \rightarrow [0; 1]$, where $rep(c)$ represents the reputation of company c. Associating reputation levels to events therefore allows us to deal with meaningless states: the level of reputation expresses the likelihood that the events provided by a business partner also have appropriate real-world counterparts. But we can also deal with incomplete representations: if we decide to add an event to the event log because we believe a real-world event is not represented in the log, we can add the presumably missing event to the log, associate it to its respective company, and assign it a low probability (to express that we are anyway not fully sure of our decision).

Modelling uncertainty over data values. As hinted at above, we use two instruments to express uncertainty over data values: confidence intervals and possible worlds. We use confidence intervals to refer to measurements performed over business matters, such as the revenue in a time period, for which we are not sure of their exact value. More precisely, here we focus on event parameters expressed as real numbers that come with an error or confidence represented as a confidence interval or standard deviation σ. This way, an uncertain value is represented as $value \pm \sigma$, which means that the true value lies somewhere in between $[value - \sigma; value + \sigma]$. Instead, we talk about possible worlds in order to denote all the possible values (together with their respective probabilities) that a given data field or indicator might assume. For example, in probabilistic databases, a possible world refers to a particular database instantiation that is aligned to a predefined schema. Here, each instantiation is associated with a probability and the sum of the probabilities of all the possible instantiations must be equal to 1 [8]. In our context, we use the possible world model as a base to represent the various values a measurement can take, together with their corresponding probabilities. More precisely, we opt to use the possible world representation to describe the probabilities of occurrence of discrete, countable values. For example, if we have an event parameter name for which we are not sure about its true value (e.g., we are not sure if its value is *smit*, *shmit*, or smith), the possible worlds for this parameter could be represented by a set of pairs $\langle value, probability\rangle$ as $\{\langle smit, 0.2\rangle, \langle shmit, 0.2\rangle, \langle smith, 0.6\rangle\}$.

The association of confidence interval with a data value can be done by the source of the event if it knows about the probability distribution of the value to be transmitted (e.g., in the case of the timing information logged by a logging system in a distributed environment), or it can be computed from the log by looking at the probability distribution of the value that derives from past values. The association of a probability can be done directly by the source of the event (a company), which might communicate its doubt regarding the data value, or it can be computed from the log, for example, by means of entity resolution algorithms [4] that are typically able to identify similar tuples and to associate probabilities to the identified options.

In order to characterize confidence intervals/probabilities for the parameters $d_1, ..., d_n$, we introduce the concept of uncertain parameter as $d_i = \langle partype,$ $name, (v_{conf}|v_{prob})\rangle$ with *partype* being the type of the parameter, *name* being the name of the parameter, $v_{conf} = \{\langle value_j, \sigma\rangle\}$ being the *value* assigned to the parameter and its standard deviation σ, and $v_{prob} = \{\langle value_j, prob_j\rangle\}$, being the set of possible worlds (in terms of possible values and probabilities) deriving from the probabilistic nature of the data value ($\Sigma prob_j = 1$). In order to express confidence intervals for numeric values, we therefore use the values' standard deviation $\sigma \in \mathbb{R}$ (we do not take into account the whole probability distribution of the value), while for each possible world we use a probability $p \in [0; 1]$. We assume that from the *name* of a parameter we can uniquely tell whether the parameter comes with probabilities or a confidence interval.

It is worth noting that if we have both the value and the probability distribution for an uncertain parameter, we could actually compute its probability.

This would allow us to assign a probability p to the parameter instead of an uncertainty σ. However, in order to keep the number of possible worlds as small as possible, we express uncertainty over data values as $value \pm \sigma$ whenever possible. This decision further allows us to compute uncertainty levels for KIs independently of the probability distributions of the in-volved parameters, as shown in the next section.

The problem now is (i) how to compute an indicator $KI(\{t_i\})$ over a set of traces $t = e_1, ..., e_n$ and that is uncertain itself and (ii) how to visualize and report on indicators that are characterized by uncertainty.

4 Computing Uncertain Key Indicators

In the previous section, we have seen that we characterize events by means of an uncertainty at the event level (its reputation) and an uncertainty at the data level (the confidence and the possible worlds for the data values). The former is strictly related to the reputation of the company involved in the cooperative process and indicates the probability that a logged event indeed corresponds to an event in the real world. In the computation of a KI, we can use this information to weight the data in the events according to their reputation, so as to give more weight to data with high reputation and less weight to data with low reputation. The uncertainty at the data level, instead, carries over from the data in input to the final value of the indicator in form of either a confidence level associated with the indicator value or a set of possible worlds describing all the possible combination of possible worlds we have in the probabilistic data values of the events used in the computation of a KI.

Therefore, given a set of traces $\{t_i\} = \{e_{i1}, ..., e_{ij}\} = \{[e_{ij}]\}$ with $1 \leq i \leq I$, where I is the number of traces in the log, and $1 \leq j \leq J$, where J is the number of events inside a trace i , $e = \langle ID_{ij}, procID_{ij}, type_{ij}, ts_{ij}, src_{ij}, dest_{ij}, [d_{ijk}],$ $rep_{ij}\rangle$ with $1 \leq k \leq K$, where K is the number of parameters inside each event, and $d_{ijk} = \langle partype_{ijk}, name_{ijk}, (\langle value_{ijk}, \sigma_{ijk}\rangle | \langle value_{ijk}, prob_{ijk}\rangle)\rangle$ with $1 \leq l \leq L$, where L is the number of possible worlds characterizing the value of a probabilistic parameter, an uncertain KI can be expressed as follows:

$$KI(\{t_i\}) = \langle\{\langle f_m, \sigma_m, prob_m\rangle\}, conf\rangle \tag{1}$$

where $1 \leq m \leq M$, being M the number of possible worlds that characterize the indicator, f_m is the indicator value, σ_m is the standard deviation associated to the indicator value, $prob_m$ is the probability that characterizes each possible world, and $conf$ is the overall confidence that we can derive for the indicator from the reputations of the events used in f. The definition of uncertain KI, in general, contains both a confidence interval σ_m for the value, and a probability $prob_m$ for each possible world, as the function f might be computed over events with both, parameter values associated with confidence intervals, and parameter values associated with probabilities (possible worlds).

The number of possible worlds M derives from the combination of the possible worlds inside each event. Specifically, $M = \prod_n L_{ijn}$ with L_{ijn} being the number of possible worlds of each probabilistic data parameter d_{ijk} used in the computation of KI; hence, $0 \leq n \leq K$. An uncertain KI is therefore characterized by a set of possible worlds, where each world can be characterized as follows:

$$f_m = f(\{[\langle\{(value_{ijk}|value_{ijkl})\}, rep_{ij}\rangle]\}) \tag{2}$$

That is, the indicator value of an individual possible world can be computed by means of a function f_m over the data values $(value_{ijk}|value_{ijkl})$ of the parameters d_{ijk} over which the KI is defined; the computation might also take into account the reputation rep_{ij} of the events involved in the computation (e.g., to weight data according to reputation). We use $value_{ijk}$ in case d_{ijk} contains an uncertain data value and $value_{ijkl}$ in case we use the l-th possible value of a probabilistic d_{ijk}. In practice, f_m is given by the designer of the KI. In our reference scenario, it is the NIC who defines the KIs it is interested in. Indeed, we have seen that the NIC wants to compute the three indicators TBRP, SRSE, and TBAS.

$$\sigma_m = \sigma_m^f = g(\{[\langle\{\sigma_{ijk}\}, rep_{ij}\rangle]\}) \tag{3}$$

The standard deviation σ_m can be computed by means of a function g from the standard deviations σ_{ijk} associated with the values $value_{ijk}$; g might take into account the reputation of events. We use the notation σ_{ijk} to uniquely identify the standard deviation of each data value; yet, note that data values of a same parameter in different traces or events are characterized by the same standard deviation. We use the notation $\sigma_m = \sigma_m^f$ to highlight that the computation of the standard deviation (and g) depends on how the data values (i.e., the statistical variables) are used in the formula f_m. We will see this property in the example we discuss next.

$$prob_m = h(\{[\langle prob_{ijkl}, rep_{ij}\rangle]\}) \tag{4}$$

We compute $prob_m$ by means of a function h that is specified over all the probabilities $prob_{ijkl}$ associated with the choices of possible data values that characterize the particular set of possible world of the indicator.

$$conf = conf(\{rep_{ij}\}) \tag{5}$$

Finally, we compute the overall confidence $conf$ of the indicator as a function of the reputations associated with the data values considered by f_m. The exact value of $conf$ can be computed by using different aggregation functions (e.g., the minimum of all reputations, the average of them, or similar); in this paper we adopt the minimum, though other functions could be used as well.

In order to better explain the concepts introduced in this section, let us consider the case in which the NIC is interested in monitoring the TBAS indicator that calculates the average time between (i) e_{i1} = registration of a new domain name and (ii) e_{i2} = first time that the customer contacts the customer support center. Let's assume $e_{i1}[TReg] = \langle time, TReg, \langle TReg.value, TReg.\sigma\rangle\rangle$ and $e_{i2}[TSup] = \langle time, TSup, \langle TSup.value, TSup.\sigma\rangle\rangle$, that is, both data values come with a confidence interval. Formally, the $TBAS$ indicator is then characterized by the value obtained by summing the time intervals calculated in each trace and weighed based on the companies' reputation:

$$TBAS(t\{t_i\},)[f] = \frac{\Sigma_{i=1}^{I}(e_{i2}[TSup.value] - e_{i1}[TRep.value])}{I} \tag{6}$$

In this case, the indicator $TBAS(\{t_i\})$ is also characterized by an aggregate σ that can be obtained by applying the rules used for the computation of error propagation from uncertain values to functions computed on these values, such as the one presented below [15]. In this formula, f is the function we want to compute, $TSup.value = TS$ and $TReg.value = TR$ are statistical variables, and, $\sigma(TS)$ and $\sigma(TR)$ are the standard deviations of TS and TR, respectively:

$$f = \frac{\Sigma_{i=1}^{I}(TS - TR)}{I} = \frac{I(TS - TR)}{I} = TS - TR \tag{7}$$

$$\sigma(f) = \sqrt{[\sigma(TS)]^2 + [\sigma(TR)]^2 \pm 2COV(TS, TR)} \tag{8}$$

In our example, we can assume that the time of the support is independent of the time of the registration. Therefore, the events that are completely independent, and the covariance between the two different values $cov(e_{ij}[value_{ijk}],$ $e_k[value_{ijk}])$ is equal to 0. This simplification, together with the assumption that, on one hand, all $TReg$ values are associated with roughly the same standard deviation throughout all traces and, on the other hand, all $Tsup$ are associated with roughly the same standard deviation as well, allows us to define the standard deviation for the $TBAS$ indicator as:

$$TBAS(\{t_i\})[\sigma] = \sqrt{e_{i1}[TReg.\sigma]^2 + e_{i2}[TSup.\sigma]^2} \tag{9}$$

Finally, each parameter discussed above (i.e., f, σ) that characterize the indicator should be weighted by the person looking at the indicator with the confidence that we associated to the overall indicator (in our convention, we use the minimum; other conventions could be used as well):

$$TBAS(\{t_i\})[conf] = Min(e_{i1}[rep]; e_{i2}[rep]) \tag{10}$$

In conclusion, our uncertain indicator $TBAS$ is given by:

$$TBAS(\{t_i\}) = \langle\langle TBAS(\{t_i\})[f], TBAS(\{t_i\})[\sigma]\rangle, TBAS(\{t_i\})[conf]\rangle \tag{11}$$

If instead of having uncertain data values, the two parameters used in the computation of $TBAS$ were of probabilistic nature, e.g., $e_{i1}[TReg]$ having three different alternatives and $e_{i2}[TSup]$ having two differrent alternatives, we would be in presence of six possible worlds for the final indicator. We would therefore need to apply the above procedure to each of the possible worlds of the indicator, and we would need to compute the combined probability for each of the possible worlds as follows (note that the two events e_{i1} and e_{i2} are independent):

$$TBAS(\{t_i\})[prob] = \{e_{i1}[prob_l] * e_{i2}[prob_l]\} \tag{12}$$

with l being the index of the possible worlds of the indicator. The final indicator would therefore look like the following:

$$\begin{aligned} TBAS(\{t_i\}) = \langle\{\langle TBAS(\{t_i\})[f]_l, TBAS(\{t_i\})[\sigma]_l,\\ e_{i1}[prob_l] * e_{i2}[prob_l]\rangle\}, TBAS(\{t_i\})[conf]\rangle \end{aligned} \tag{13}$$

5 Implementation: Key Indicators in Practice

The described approach to the computation of uncertain indicators has been developed in the context of a European FP7 research project (MASTER Managing Assurance, Security and Trust for sERvices[1]), which focuses on methodologies and infrastructures to manage security and compliance of service-based business processes. We in particular focus on the assessment of and reporting on compliance, starting from a log of events generated by the MASTER infrastructure, and the computation of uncertain key compliance indicators is one of our main contributions, along with an analysis of correlations among indicators and process model discovery.

Figure 5 shows a simplified architecture of the MASTER infrastructure; specifically, we focus on the diagnostic infrastructure with its data warehouse and analysis algorithms. The main input to the infrastructure is the (uncertain) Event log, which contains events generated by the operational system (a

[1]For the details, the reader is referred to http://www.master-fp7.eu

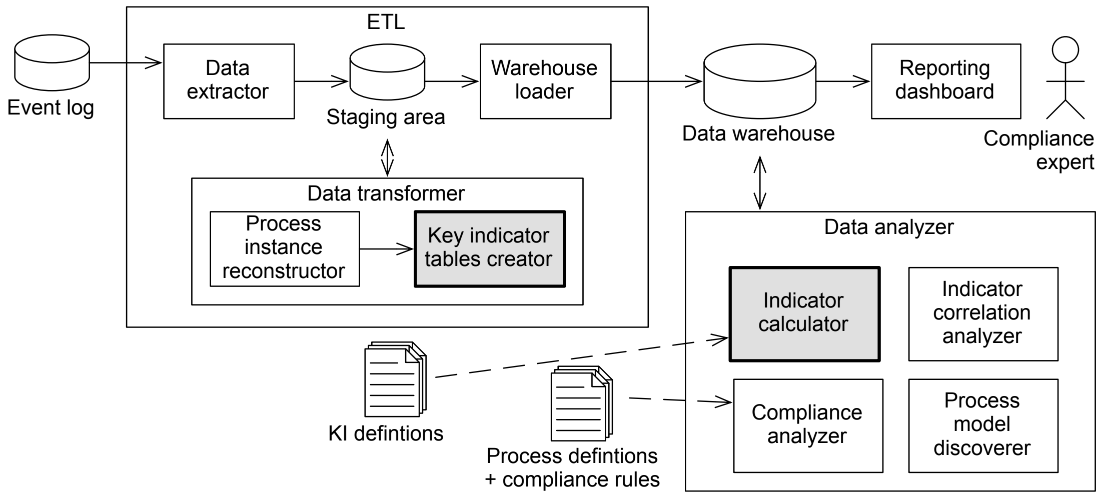


Figure 5: Functional architecture of the diagnostic infrastructure in the MASTER project.

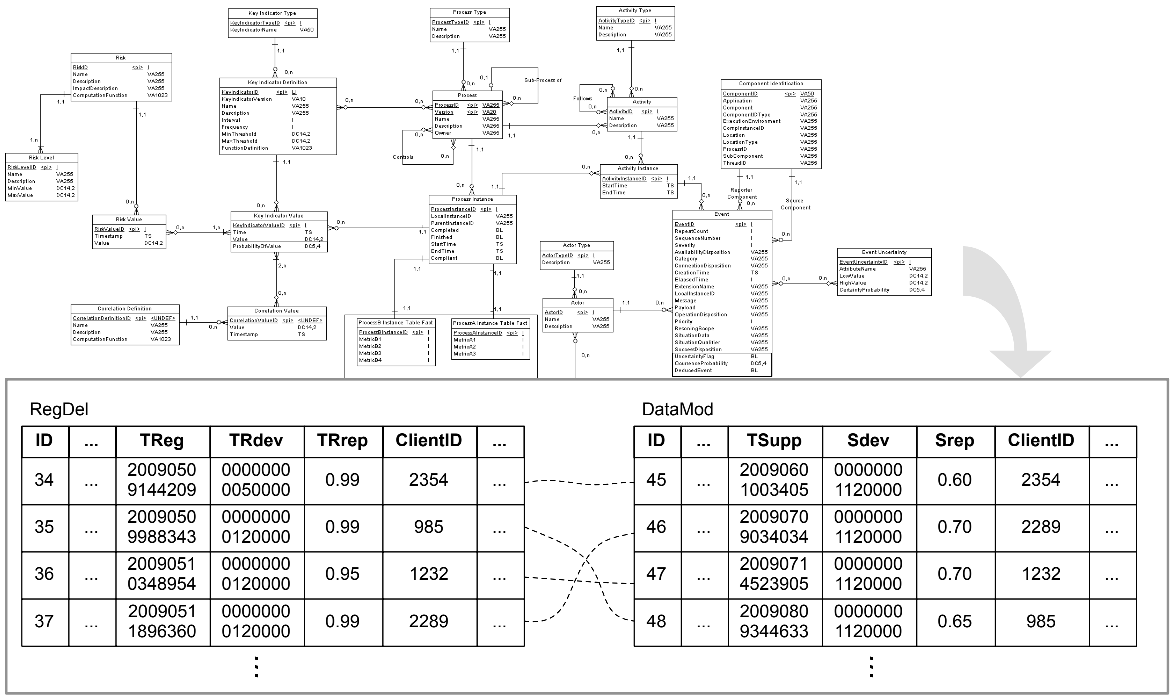
service-oriented architecture). During ETL (Extract-Transform-Load), events are extracted from the log and stored in Staging area for transformation. Of the overall transformation process, we highlight the Process instance reconstructor and the Key indicator tables creator, which reconstruct from the events in the log which process instances have been executed and create auxiliary tables for indicator computation, respectively. Then, we load (Warehouse loader) the data into the Data warehouse, upon which we then run our analysis algorithms to (i) compute uncertain indicators (Indicator calculator), (ii) correlate indicators (Indicator correlation analyzer), (iii) analyze compliance of processes with regulations and laws (Compliance analyzer), and (iv) discover process models from the log (Process model discoverer). All analysis results are stored back into the warehouse and rendered to the user (compliance expert or business analyst) via a Reporting dashboard. In this paper, we focus on the computation of the indicators, which are at the heart of the Reporting dashboard; the respective components are highlighted in the architecture.

In Figure 6 we hint at the conceptual model of the data warehouse underlying the analysis algorithms, yet, for lack of space, we do not describe its details here. For the sake of this paper, it suffices to know that we store all the events in the warehouse, along with the data quality metadata associated to them and to the individual data parameters inside the events (reputation levels, uncertainties, probabilities). We also keep track if an event is a deduced event that we added to the log during ETL to solve incomplete representation cases. Therefore, the warehouse contains a complete historical view over the performance of a company.

The figure also contains the so-called Key indicator tables supporting the computation of KIs, in that they contain in a concise fashion, process instance by process instance (trace by trace), all the data values and their confidence levels and probabilities that we need to compute the indicators specified in the KI definitions document shown in Figure 5. The tables we highlight in Figure 6 are the ones we need to compute the TBAS indicator of our NIC: the RegDel table contains all the parameters regarding the Delegation process, and the DataMod table contains all the parameter regarding the Data Modification Request process. We only show the parameters necessary for the computation of the TBAS indicator: TReg is the registration time of the domain name, TRdev is the standard deviation, TRrep is the reputation, and ClientID is the identifier of the client; the parameters in DataMod are defined analogously. These tables are the output of the Key indicator tables creator in Figure 5. Next, we explain the logic of the Indicator calculator, that is, we show how we concretely compute uncertain indicators from the data warehouse. In order to compute the TBAS indicator from the auxiliary tables in Figure 6, we translate its mathematical formula into SQL statements that we can issue to the data warehouse. For example, the following statements compute TBAS for all customers who successfully registered a domain name in May 2009 (note that we TBAS does not contain probabilistic parameters, so we only compute its value and confidence interval):

```
IntervalSum = select sum(TSupp-TReg) from Rendell join
DataMod on RegDel.ClientID=DataMod.ClientID where
TReg>=20090501000000 and TReg<=20090531999999;

RegDelCount = select count(RegDel.ClientID) from RegDel join
DataMod on RegDel.ClientID=DataMod.ClientID where
TReg>=20090501000000 and TReg<=20090531999999;
```



RegDel

ID	...	TReg	TRdev	TRrep	ClientID	...
34	...	2009050 9144209	0000000 0050000	0.99	2354	...
35	...	2009050 9988343	0000000 0120000	0.99	985	...
36	...	2009051 0348954	0000000 0120000	0.95	1232	...
37	...	2009051 1896360	0000000 0120000	0.99	2289	...

⋮

DataMod

ID	...	TSupp	Sdev	Srep	ClientID	...
45	...	2009060 1003405	0000000 1120000	0.60	2354	...
46	...	2009070 9034034	0000000 1120000	0.70	2289	...
47	...	2009071 4523905	0000000 1120000	0.70	1232	...
48	...	2009080 9344633	0000000 1120000	0.65	985	...

⋮

Figure 6: Conceptual model of data warehouse with uncertain data, plus - in the foreground - the key indicator tables supporting the computation of the indicators.

```
TBAS_value = IntervalSum/RegDelCount;

TBAS_sigma = select sqrt(sum(power(TRrep,2) + power(Srep,2)))
from RegDel join DataMod on RegDel.ClientID=DataMod.ClientID
where TReg>=20090501000000 and TReg<=20090531999999;

TBAS_conf = select min(case when TRrep<=SRep then TRrep else
SRep end) from RegDel join DataMod on
RegDel.ClientID=DataMod.ClientID where TReg>=20090501000000
and TReg<=20090531999999;
```

The final indicator is therefore given by: $TBAS = \langle TBAS\_value, TBAS\_sigma, TBAS\_conf\rangle$. Although not shown in this paper, it is important to observe that the presence of probabilistic parameters in the computation of an indicator can be handled by suitably applying the group by SQL statement to the columns that contain multiple possible values for a given parameter. In this way, the output of the computation is no longer a single value, but a set of tuples describing all the legal alternatives for the indicator in terms of value and confidence.

Finally, as already discussed in [6], it is very important to convey the uncertainty that characterizes the indicator to the user of the Reporting dashboard, in order to create the necessary awareness of the data quality problem underlying the computation of the indicator and to enable better informed decisions. We are still working on this aspect, but we propose to provide first a very high-level view of the indicator through an intuitive, graphical visualization of all the indicators (and their alternatives) in the system and to explicitly mark those indicators that are uncertain. If a user wants to inspect the nature of the uncertainty, we will support a drill-down mechanism allowing the user to explore, for instance, the indicators' alternatives, their probability distribution, and the confidence we have in the indicator.

6 Related Work

Recently, there has been lots of interest in databases specifically designed to manage uncertain data [1][3][5][14]. In this case, data are coupled with a probability value indicating the degree of confidence to the accuracy of the data. These probabilities are then taken into account by the database management system when processing the data to produce answers to user queries. Most of the contributions deal with simple queries while only a few deals with the aggregation of uncertain data to produce the results of queries in which the aggregation functions (e.g., sum, count, avg) are used [9][13]. Anyway, the proposed systems however do not deal with the problems of deriving probabilities in more complex cases, such as when computing reports, and of extracting uncertain data from not trustable sources Furthermore, past contributions relate the value uncertainty only to the value correctness (i.e., accuracy) and do not consider the case in which meaningless and incomplete representation affect the databases.

In our case, we need to reason about data uncertainty caused by all the possible poor quality problems that could affect objects of different granularities (i.e., values, events), in order to characterize the reliability of the reports within their indicators. In an organization, KIs represent the links between available sources of performance data and organizational objectives [2]. KI measurement issues have been largely analyzed in the literature since sometimes decision support systems are not efficient as expected since they are based on erroneous KI values calculated on not reliable performance data. In fact, practically, any input data have uncertainty that can be caused by different issues such as inaccuracy of measuring and inaccuracy of rounding-up, scale restrictions, impossibility of measuring or definition of values with needed precision, hidden semantic uncertainty of qualitative data [11]. The number of data sources required to support the most common KIs measurement is large and thus the uncertainty issues cannot be neglected [10].

Previous work (e.g., [11]) focus on the evaluation of the KI measure starting from the assumption that the confidence of the obtained value depends on the assessment process. Here, the validity of the KI is defined as a property of a KI that makes it suitable as a basis for performance assessment. The generic attributes of valid performance measurements are straightforward: relevance, accuracy, timeliness, completeness, and clarity. In these contributions, the unavailability and the reliability of some data is not discussed since they assume that the internal operational systems provide all the needed information. In our approach, KI measurement relies on the available data obtainable from the companies involved in the cooperative process and takes into account the trustworthiness of all these sources. Thus, data availability and reputation are considered as variables to consider in data uncertainty evaluation. The evaluation of the quality of data received by other companies involved in a cooperative process is an issue that has been also analyzed in [12]. In [12] authors propose an architecture that evaluates the reputation of the different companies involved in the cooperative process on the basis of the quality of the information that they provide. In their evaluation, they did not consider uncertainty in data values but in order to evaluate data correctness they assume that is always possible to retrieve a certificate and correct value to assess the data provided by the different companies. This is an assumption that is difficult to validate in the real world since the availability of certificated values is scarcely guaranteed.

7 Conclusion and Future Work

Uncertainty is a real issue in modern data management. Indeed, the database community (both academia and industry) has already started investing huge efforts into research on uncertain and probabilistic databases, yet there is a lack of business intelligence applications that are able to profit from the results of such research. In this paper, we discuss how to compute uncertain key indicators from uncertain data and we provide a contribution toward uncertain business intelligence, that is, business intelligence that runs atop uncertain data and whose data analysis algorithms take into account uncertainty.

We characterized the problem of computing uncertain key indicators in the context of distributed, cooperative business scenarios that are characterized by different levels of reputation and different levels of visibility into the partners' business practices. We discussed and classified the typical data quality problems of that scenario and proposed both a conceptual and practical solution to the computation of key indicators, which, in general, we describe as a set of values, their standard deviations, their probabilities and an overall level of confidence (taking into account the reputation of the cooperating partners).

Next, we will apply the concepts and practices discussed in this paper to the case of compliance assessment. We will work on the correlation of uncertain key indicators, so as to identify correlations in the dynamics of two indicators over a predefined time span (e.g., KI_1 drops in average one business day after KI_2 drops). This will allow us to perform root-causes analyses or, if used to look into the future, to help in the prediction of future behaviours. In parallel, we will also work on the visualization of uncertain indicators inside reporting dashboards and test the solutions in the context of compliance assessment.

Acknowledgements. This work was supported by funds from the European Commission (contract N 216917 for the FP7-ICT-2007-1 project MASTER).

References

[1] Antova, L., Koch, C., Olteanu, D. 10ˆ(10ˆ6) Worlds and Beyond: Efficient Representation and Processing of Incomplete Information. Proceedings of ICDE'07, pp. 606-615.

[2] Baird, H. Validating performance measurements. Call Center Management Review (2005)

[3] Benjelloun, O., Das Sarma, A., Halevy, A., Thobald, M. and Widom, J. Databases with uncertainty and lineage. VLDB Journal, 17(2), pp. 243-264, 2008.

[4] Benjelloun, O., Garcia-Molina, Gong, H., Kawai, H., Larson, T., Menestrina, D., and Thavisomboon, S. D- Swoosh: A Family of Algorithms for Generic, Distributed Entity Resolution. Proceedings of the 27th IEEE International Conference on Distributed Computing Systems (ICDCS), 2007.

[5] Dalvi, N. and Suciu, D. Efficient query evaluation on probabilistic databases. VLDB Journal, 16(4), pp. 523- 544, 2007.

[6] Daniel, F., Casati, F., Palpanas, T., Chayka, O., Cappiello, C. Enabling Better Decisions through Quality-Aware Reports in Business Intelligence Applications. ICIQ'08, November 2008, Boston, pp. 310-324.

[7] Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J and Hong, W. Model-driven data acquisition in sensor networks. Proceedings of VLDB'04, 2004.

[8] Green, T., and Tannen, V. Models for Incomplete and Probabilistic Information. Data Engineering Bulletin, vol. 29, no. 1, 1996.

[9] Jayram, T.S., Kale, S., Vee, E. Efficient Aggregation Algorithms for Probabilistic Data. Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, 2007, pp. 346 355.

[10] Kechagioglou, I. Uncertainty and confidence in measurement. Proceedings of the 18th Panhellenic Conference on Statistics, 2005.

[11] Krissilov, D.S. Towards the problems of an evaluation of data uncertainty in decision support systems. Information Theories & Applications, 13, 2006, pp. 376-379.

[12] Mecella, M., Scannapieco, M., Virgillito, A., Baldoni, R., Catarci, T., Batini, C. The DaQuinCIS Broker: Querying Data and Their Quality in Cooperative Information Systems. J. Data Semantics, 1: 208-232 (2003).

[13] R, C. and Suciu, D. Efficient Evaluation of HAVING queries on a Probabilistic Database. Proceedings of the 11th International Symposium on Database Programming Languages, DBPL 2007

[14] Singh, S., Mayfield, C., Shah, R., Prabhakar, S., Hambrusch, S., Neville, J. and Cheng, R. Database Support for Probabilistic Attributes and Tuples. Proceedings of ICDE'08, pp. 1053-1061.

[15] Taylor, J. An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements. 2nd ed., ISBN 0-935702-42-3.

[16] Wand, Y. and Wang, R.Y. Anchoring Data Quality Dimensions in Ontological Foundations. Communications of the ACM, Vol. 39, No. 11, November 1996, pp. 86-95.

Appendix B

On the Design of Compliance Governance Dashboards for Effective Compliance and Audit Management

On the Design of Compliance Governance Dashboards for Effective Compliance and Audit Management *

Patricia Silveira Carlos Rodríguez Fabio Casati Florian Daniel
Vincenzo D'Andrea

Abstract

Assessing whether a company's business practices conform to laws and regulations and follow standards and best practices, i.e., compliance governance, is a complex and costly task. Few software tools aiding compliance governance exist; however, they typically do not really address the needs of who is actually in charge of assessing and controlling compliance, that is, compliance experts and auditors. We advocate the use of compliance governance dashboards, whose design and implementation is however challenging for at least three reasons: (i) it is fundamental to identify the right level of abstraction for the information to be shown; (ii) it is not trivial to visualize different analysis perspectives; and (iii) it is difficult to manage the large amount of involved concepts, instruments, and data. This paper shows how to address these issues, which concepts and models underlie the problem, and, eventually, how IT can effectively support compliance analysis in Service-Oriented Architectures.

1 Introduction

Compliance is a term generally used to refer to the conformance to a set of laws, regulations, policies, or best practices. Compliance governance refers to the set of procedures, methodologies, and technologies put in place by a corporation to carry out, monitor, and manage compliance.

Compliance governance is an important, expensive, and complex problem to deal with: It is important because there is increasing regulatory pressure on companies to meet a variety of policies and laws (e.g., Basel II, MiFID, SOX). This increase has been to a large extent fueled by high-profile bankruptcy cases

*The final publication is available at www.springerlink.com (http://link.springer.com/chapter/10.1007/978-3-642-16132-2_20)

(Parmalat, Enron, WorldCom, the recent crisis) or safety mishaps (the April 2009 earthquake in Italy has already led to stricter rules and certification procedures for buildings and construction companies). Failing to meet these regulations means safety risks, hefty penalties, loss of reputation, or even bankruptcy [11].

Managing and auditing/certifying compliance is a very expensive endeavor. A report by AMR Research [6] estimates that companies will spend USD 32B only on governance, compliance, and risk in 2008 and more than USD 33B in 2009. Audits are themselves expensive and invasive activities, costly not only in terms of auditors' salaries but also in terms of internal costs for preparing for and assisting the audit - not to mention the cost of non-compliance in terms of penalties and reputation.

Finally, the problem is complex because each corporation has to face a large set of compliance requirements in the various business segments, from how internal IT is managed to how personnel is trained, how product safety is ensured, or how (and how promptly) information is communicated to shareholders. Furthermore, rules are sometimes vague and informally specified. As a result, compliance governance requires understanding/interpreting requirements and implementing and managing a large number of control actions on a variety of procedures across the business units of a company. Each compliance regulation and procedure may require its own control mechanism and its own set of indicators to assess the compliance status of the procedure [1]. Today, compliance is to a large extent managed by the various business units in rather ad-hoc ways (each unit, line of business, or even each business process has its own methodology, policy, controls, and technology for managing compliance) [15]. As a result, today it is very hard for any CFO or CIO to answer questions such as: Which rules does my company have to comply with? Which processes should obey which rules? Which processes are following regulations? Where do violations occur? Which processes do we have under control? [19]. Even more, it is hard to do so from a perspective that not only satisfies the company but also the company's auditors, which is crucial as the auditors are the ones that certify compliance.

To address these and similar compliance problems, the EU has funded projects that bring together corporations, auditors, and researchers in conceptual modeling, process monitoring, business intelligence, and service computing. This paper is the result of a combined effort from two such projects (Compas [www.compas-ict.eu] and Master [www.master-fp7.eu]). It presents a conceptual model for compliance and for compliance governance dashboards (CGDs), along with a dashboard architecture and a prototype implementation. The aim of CGDs is to report on compliance, to create an awareness of possible problems or violations, and to facilitate the identification of root-causes for non-compliant situations.

The dashboard is targeted at several classes of users: chief officers of a company, line of business managers, internal auditors, and external auditors (certification agencies). These two latter typically focus on a fairly narrow set of processes and examine historical data to verify non-compliant situations and how they have been dealt with. Via the dashboard, they also have access to key compliance indicators (KCIs) defined for each process. Managers (especially high-level ones) are interested in a much broader set of compliance regulations and at quasi-real time compliance information that allows them to detect problems as they happen and identify the causes, so that they can correct them before they become (significant) violations. They have access and navigate through the entire set of regulations, business processes, and business units and also observe the overall compliance status (through aggregate KCIs). In addition, once problems are identified (unsatisfactory values for indicators) they drilldown to the root of the problem.

Technically, building a dashboard that shows a bunch of indicators and that allows drill-downs is easy. Indeed, the main challenges in this case are conceptual more than technological [18]. These challenges, which also correspond to the main contributions of this paper, are:

1. Provide a conceptual model for compliance and for compliance dashboards that covers a broad class of compliance issues. Identify the key abstractions and their relationships. Otherwise the dashboard loses its value of single entry point for compliance assessment.

2. Combine the above broadness with simplicity and effectiveness. The chal-

lenge here is to derive a model that, despite being broad, remains simple and useful/usable. If the abstractions are not carefully crafted and kept to a minimum, the dashboard will be too complex and remain unused. Models that are too generic are often too complex to use. As we have experienced, this problem may seem easy but is instead rather complex, up to the point that discussions on the conceptual model in the projects took well over a year. There is no clarity in this area, and this is demonstrated by the fact that while everybody talks about compliance, there are no generic but simple compliance models readily available.

3. Define, besides the conceptual abstractions, a user interaction and navigation model that captures the way the different kinds of users need to interact with the dashboard, to minimize the time to accesses spent in getting the information users need and to make sure that key problems do not remain unnoticed.

4. Derive a model that is in line with the criteria and approach that auditors have to verify compliance. In this paper, this last contribution is achieved “by design”, in that the model is derived also via a joint effort of two of the major auditing companies and reflects the desired method of understanding of and navigation among the various compliance concerns.

In the following, we first introduce our conceptual model for compliance and then the compliance management lifecycle. We then focus on the dashboard and present a structural and navigational model for compliance, describe the architecture and prototype, and then compare the work done with prior art and existing tools.

2 The Problem of Compliance Management

To characterize the compliance management problem intuitively introduced above, we now generalize the problem in terms of two models of its most important concepts, their relationships, and the dynamics that describe their adoption in practice.

2.1 Concepts and Terminology

Despite the increasing awareness of compliance issues in companies and the recognition that part of the compliance auditing task can be easily automated, i.e., assisted by means of software tools [11][14][15], there is still a lot of confusion around. This is especially true for the IT community, which would actually be in charge of aiding compliance governance with dedicated software. To help thinking in terms of auditing, in the following we aim to abstract a wide class of compliance problems into a few key concepts that are also the ones understood by auditors. The resulting model does not cover all possible compliance problems, but our goal is to strike a balance between coverage and simplicity. So far, we did not find any such model in literature. The model is illustrated in Fig. 1.

We read the model from the top-left corner: The Regulation entity generalizes all those documents that regulate or provide guidelines for the correct or good conduct of business in a given business domain. Common examples of regulations are legislations (e.g., MiFID, The Electronic Commerce Directive), laws (e.g., SOX, HIPAA), standards (e.g., CMMI, CoBIT, ISO-9001), and contracts or SLAs. Typically, a regulation defines a set of rules or principles in natural language, which constrain or guide the way business should be conducted. Complying with a regulation means satisfying its rules and principles. Yet, a company might be affected by only some of the rules or principles stated in a given regulation. The selection of the pertaining ones represents the requirements for compliance management, commonly expressed in terms of control objectives and control activities. A regulation expresses multiple requirements, and a requirement might relate to one or more regulations.

Assessing compliance demands for an interpretation and translation of the requirements provided in natural language in an actionable rule description (especially in the case of principle-based regulations) [9][10]. This is modeled by the Rule entity, which represents actionable rules expressed either in natural language (using the company's terminology and telling exactly how to perform work) or, as desirable in a formalism that facilitates its automated processing (e.g., Boolean expressions over events generated during business execution).

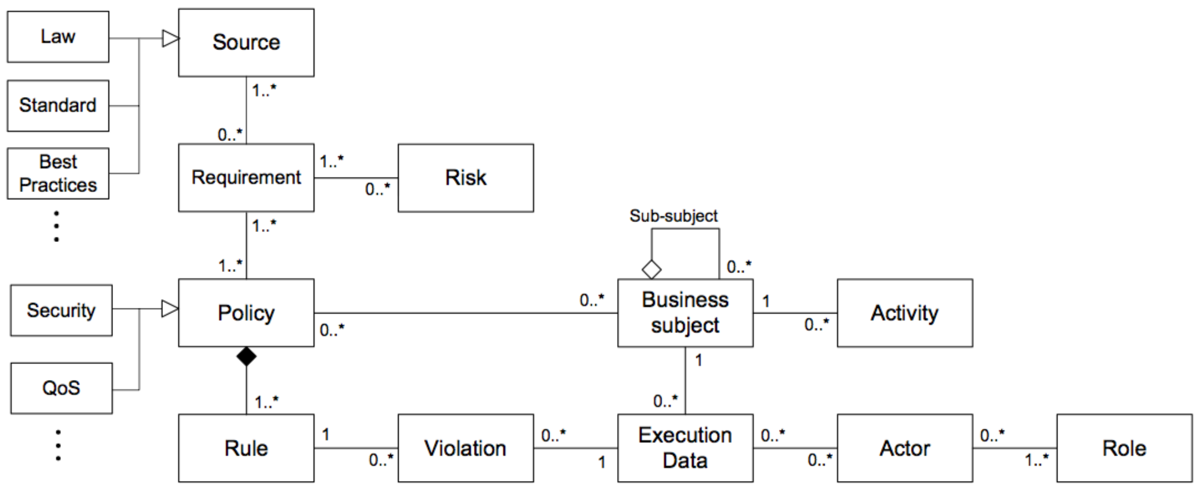


Figure 1: Conceptual model of the compliance management problem.

Rules are then grouped into policies, which are the company-internal documents that operatively describe how the company intends achieving compliance with the selected requirements. Typically, policies represent a grouping of the requirements into topics, e.g., security policies, QoS policies, and similar.

At a strategic level, compliance is naturally related to the concept of risk. Non-compliant situations expose a company to risks that might be mitigated. For example, a non-encrypted message that is sent through the network might violate a security compliance rule, which, in turn, might put at risk sensitive information. Risk mitigation is the actual driver for internal compliance auditing. The Risk entity represents the risks a company wants to monitor; risks are associated with compliance requirements. For the evaluation of whether business is executed in a compliant way or not, we must know which rules must be evaluated in which business context. We therefore assume that we can associate policies with specific business processes (though this can easily be generalized to the case of projects, products, and similar). Processes are composed of activities, which represent the atomic work items in a process.

The actual evaluation of compliance rules is not performed on business processes (that is, on their models) but on their concrete executions, i.e., their instances. Executing a business process means performing activities, invoking services, and tracking progression events and produced business data (captured by the Execution data entity). In addition, e.g., separation of duties, it is necessary to track the actors and roles of execution of activities. When evaluation of a rule for a process/activity instance is negative, it corresponds to violations, which are the core for the assessment of the level of compliance of a company and the computation of KCIs.

The model in Fig. 1 puts into context the most important concepts auditors are interested in when auditing a company. The actual auditing process, then, also looks at the dynamic aspect of the compliance management problem, that is, at how the company decides which regulations are pertaining, how it implements its business processes, how it checks for violations, and so on. In short, the auditing process is embedded in a so-called compliance management life cycle, which we discuss next.

2.2 The Compliance Management Life Cycle

In everyday business a company is subject to a variety of different regulations. It is up to the company to understand, select, and "internalize" them that affect its business, thus producing a set of internal policies (internalization phase in Fig. 2). The latter then drive the design of the company's business practices, yielding a set of business processes that are possibly designed compliantly (design phase), meaning that they are designed to respect the internal policies. To provide evidence of the (hopefully) compliant execution of designed business processes, the company also defines a set of events, often also called "controls" or "control points".

Process and event definitions are consumed in the business execution phase, where the company's employees perform the tasks and duties specified in the process models. Ideally (but not mandatorily), this execution is assisted by software tools such as workflow management or business process execution systems, also able to collect compliance-specific evidence and to generate respective execution events (the execution data), which can be stored in an audit trail or log file for evaluation.

The internal evaluation phase serves a twofold purpose: First, it is the point where collected data can be automatically analyzed to detect compliance violations. Indeed, designing compliant processes is not enough to assure compli-

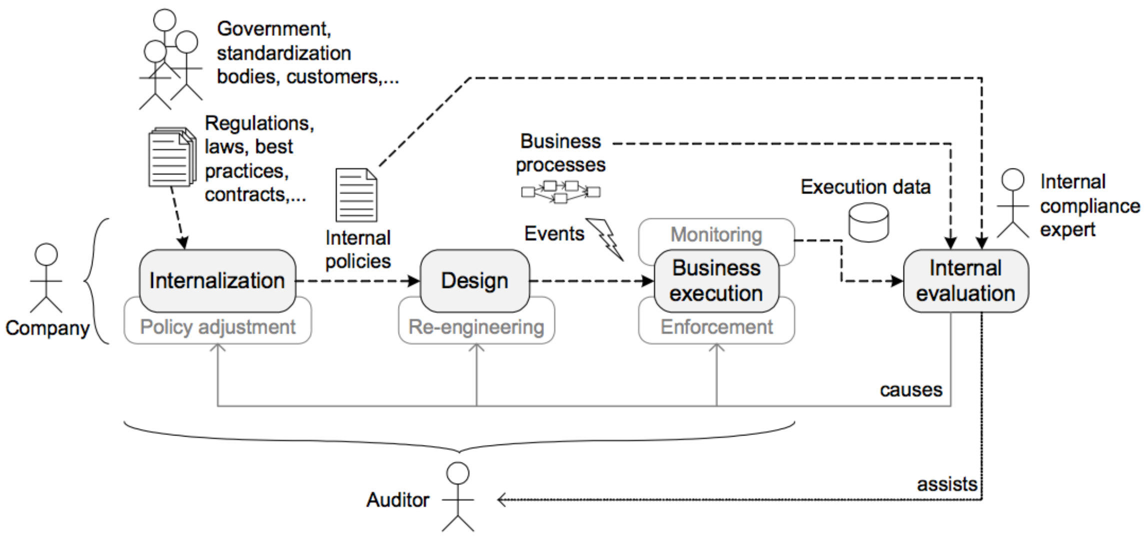


Figure 2: The compliance management life cycle with phases, products, and actors.

ance, as in practice there are a multitude of reasons for which deviations from an expected business process might happen (e.g., human factors, system downtimes). Some of such problems can be detected during runtime, resulting in the generation of respective events; some of them can only be detected after execution by means of, e.g., data mining or root-cause analysis techniques applied to tracked runtime data. Second, the internal evaluation is the moment where a company-internal expert (auditor) may inspect and interpret the tracked evidence to assess the company's level of compliance. The outcome of this internal evaluation might be the enforcement of corrective runtime actions (e.g., sending an alert), the re-engineering of process designs (e.g., to consider design flaws) or the adjustment of the internal policies (e.g., to cope with inconsistent policies).

Note that the internal evaluation does not yet certify a company's level of compliance; it rather represents an internal control mechanism by means of which the company is able to self-assess and govern its business. For the certification of compliance, an external auditor, e.g., a financial auditor, physically visits the company and controls whether (i) the company has correctly interpreted the existing regulations, (ii) business processes have been correctly implemented, and, finally, (iii) business processes have been executed according to the policies. In practice, external audits are based on statistical checks of physical documents. In addition to unavoidable statistical errors, a certified level of compliance is further subject to the auditor's assessment and, therefore, also contains a subjective component.

3 Designing Compliance Governance Dashboard (CGDs)

To aid the internal evaluation and to help a company pass external audits, a concise and intuitive visualization of its compliance state is paramount. To report on compliance, we advocate the use of a web-based CGDs, whose good design is not trivial [5][17]. It is important to understand: i) what the typical information auditors expect to find is; ii) how large amounts of data can be visualized in an effective manner, and how data can be meaningfully grouped and summarized; and iii) how to structure the available information into multiple pages, that is, how to interactively and intuitively guide the user through the wealth of information. Each page of the dashboard should be concise and intuitive, yet complete and expressive. It is important that users are immediately able to identify the key information in a page, but that there are also facilities to drill-down into details.

Designing CGDs requires mastering some new concepts in addition to those discussed above. Then, the new concepts must be equipped with a well-thought navigation structure to effectively convey the necessary information. Here, we do not focus on how data are stored and how rules are evaluated; several proposals and approaches have been conceived so far for that (see Section 5), and we build on top of them.

3.1 A Conceptual Model for CGDs

In Fig. 3 we extend the conceptual model (Fig. 1) to capture the necessary constructs for the development of a CGD (bold lines and labels represent new entities and their respective interrelations). The extensions aim at (i) providing different analysis perspectives (in terms of time, user roles, and organizational structures), (ii) summarizing data at different levels of abstraction, and (iii) enabling drill-down/roll-up features (from aggregated data to detailed data, and vice versa).

The Dashboard view entity represents individual views over the compliance status of the company. A view is characterized by the user role that accesses it, e.g., IT specialists, compliance experts, managers, or similar. Each of these roles has different needs and rights. For instance, managers are more interested in aggregated values, risk levels, and long time horizons (to take business decisions); IT personnel are rather interested in instance-level data and short time spans (to fix violations). A view is further characterized by the time interval considered for the visualization of data (e.g., day, week, month, or year), also providing for the historical analysis (e.g., last year) and supporting different reporting purposes (operative, tactical, strategic). Finally, a view might be restricted to only some of the business units in the company, based on the role of the user. Business units can be composed by other business units, forming a hierarchical organizational structure. In summary, views support different summarization levels of the overall available data, ranging over multiple granularity levels.

Effective summarization of data is one of the most challenging aspects in the design of CGDs, commonly instrumented by indicators [13]. An indicator is a quantitative summarization of a particular aspect of interest in the business, i.e., a metric of how well an objective is being reached. Typically, KPIs (key performance indicators), are used to summarize the level at which business objectives are reached. In our context, we speak about KCIs, referring to the achievement of the stated compliance objectives (e.g., the number of unauthorized accesses to our payroll data).

In general, indicators are computed out of a variety of data and functions; in the context of compliance assessment, however, indicators can typically be related to the ratio of encountered violations vs. compliant instances of a process or activity. As an abstraction of indicator values, we can define taxonomies (e.g., low, medium, high) and use colors (e.g., red, yellow, green) for their intuitive visualization. The same considerations hold for risk levels, which represent the level of summarization that is appropriate for long-term, strategic perspectives and are usually computed out of the values of indicators and additional

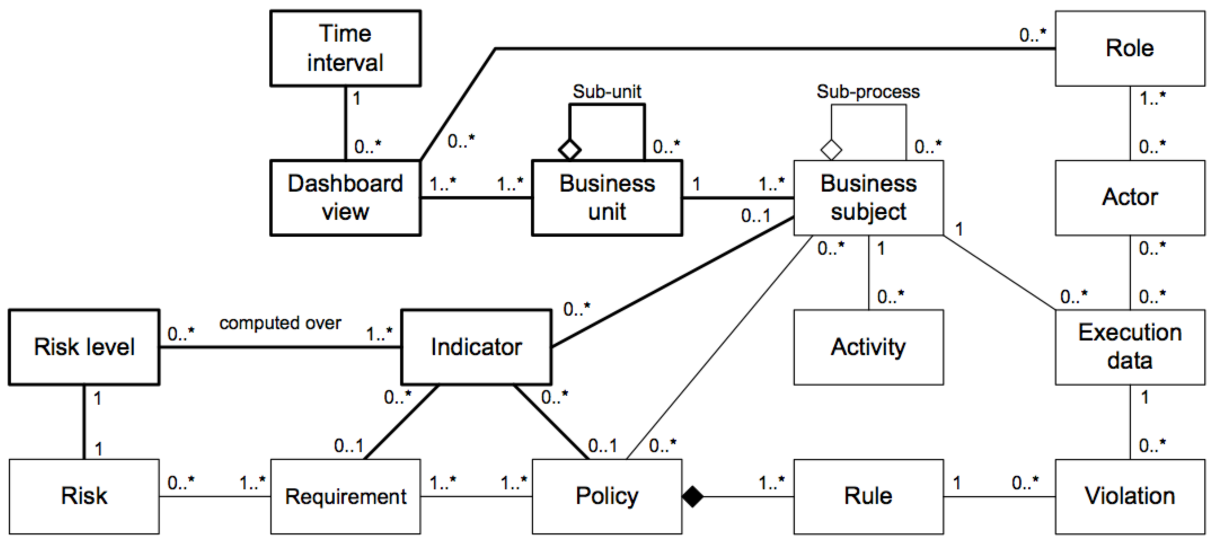


Figure 3: Conceptual model for CGDs (dashboard-specific constructs are highlighted in bold.

(external) data.

The described model extension aims at relating general compliance concepts with concepts that are specific to the design of dashboards. The model is general and extensible, so as to allow for the necessary flexibility to accommodate multiple concrete compliance scenarios.

3.2 Navigation Design

After discussing the static aspects of the design of CGDs, we now focus on the dynamic aspect, i.e., on how to structure the interaction of users with the dashboard, and on how users can explore the data underlying the dashboard application. Specifically, on top of the conceptual model for CGDs, we now describe how complex data can be organized into hypertext pages and which navigation paths are important.

For this purpose, we adopt the Web Modeling Language (WebML [3]), a conceptual modeling notation and methodology for the development of data-intensive web applications. We use the language for the purpose of illustration only (we show a simplified, not executable WebML schema) and intuitively introduce all the necessary constructs along with the description of the actual

CGD navigation structure.

The WebML hypertext schema (Fig. 4) describes the organization of our ideal web CGD. It consists of five pages (the boxes with the name labels in the upper left corner), ComplianceHome being the home page (note the H label). Each page contains a number of content units, which represent the publication of contents from the data schema in Fig. 3 (the selector condition below the units indicates the source data entity). Usually, there are many hyperlinks (the arrows) in a hypertext schema, representing the possible navigations a user might perform, but, for simplicity, we limit our explanation to only those links that represent the main navigation flow. Links carry parameters, which represent the selection performed by the user when activating a link (e.g., the selection of a process from a list). For the purpose of reporting on compliance, we define a new content unit (not part of the WebML), the compliance drill-down unit, which allows us to comfortably show compliance data in a table-like structure (see the legend in Fig. 4 and the examples in Fig. 5).

Let's examine the CGD's structure (Fig. 4): The home page of the CGD provides insight into the compliance state of the company at a glance. It shows the set of most important indicators (Main indicators multidata unit) and a set of indicators grouped by their policy (IndByPolicy hierarchical index unit). Then, we show the (BUnits/Regul.) unit that allows the user to drill-down from business units to processes and from regulations to policies. A click on one of: i) the processes leads the user to the Regulations by Activity page; ii) regulations leads her to the Rules by Business Units page; and iii) the cell of the table leads her to the Rules by Activity page. After the selection of a process, in the Regulations by Activity page the user can inspect the compliance state of each activity of the selected process with the given regulations and policies (RegByActivity), a set of related indicators (BPIndicators unit; the unit consumes the Process parameter), and the details of the selected process (Process data unit). Similar details are shown for policies in the Rules by BusinessUnits page, which allows the user to inspect the satisfaction of individual compliance rules at business unit or process level (RulesByBU). A further selection in the compliance drill-down units in these last two pages or the selection of a cell in the BUnits/Regul. unit in the home page leads the user to the Rules by Activity

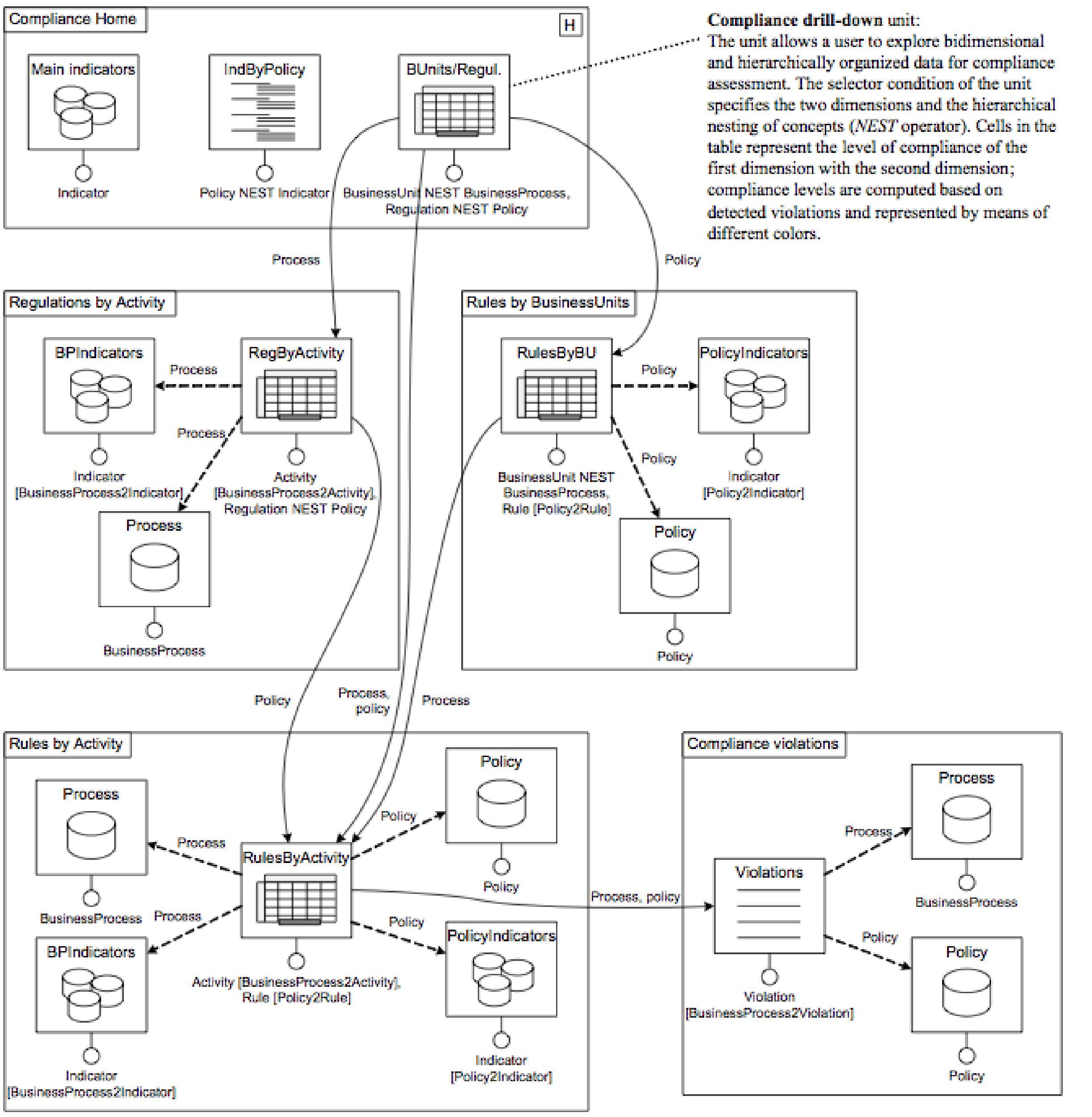

Compliance drill-down unit:
The unit allows a user to explore bidimensional and hierarchically organized data for compliance assessment. The selector condition of the unit specifies the two dimensions and the hierarchical nesting of concepts (*NEST* operator). Cells in the table represent the level of compliance of the first dimension with the second dimension; compliance levels are computed based on detected violations and represented by means of different colors.

Figure 4: WebML hypertext schema structuring the navigation of CGD concepts and data.

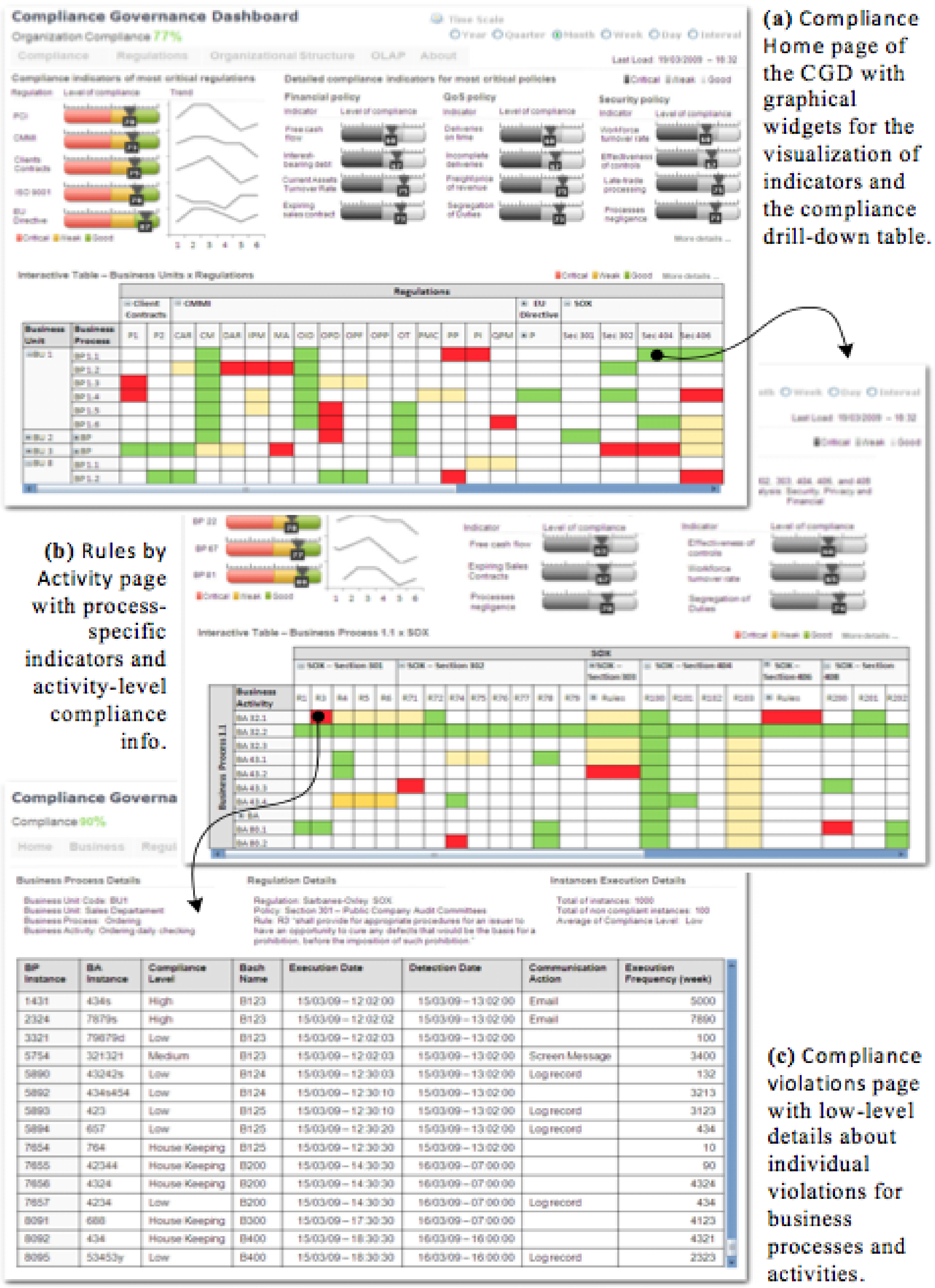
page, which provides the user with the lowest level of aggregated information. It visualizes the satisfaction of the compliance rules of the chosen policy by the individual activities of the chosen process (RulesByActivity), along with the details of the chosen policy and process and their respective indicators. A further selection in this page leads the user to the Compliance violations page, which shows the details of the violations related to the chosen process/policy combination at an instance level in the Violations index unit.

The navigation structure in Fig. 4 shows one of the possible views over the data in Fig. 3, e.g., the one of the internal compliance expert. Other views can easily be added by restraining access to data and defining alternative navigation structures. Each page provides a different level of summarization (overview, process-specific, policy-specific, process and policy-specific, violation instances), guiding the user from high-level information to low-level details. The time interval to be considered for the visualization can be chosen in each of the pages.

3.3 CGD in Practice

To provide the look-and-feel idea we have implemented, in Fig. 5 we illustrate screenshots from our prototype CGD. The screenshots show views that clarify and consistently present our ideal CGD. Fig. 5(a) shows the Compliance Home page (Fig. 4), Fig. 5(b) the Rules by Activity page, and Fig. 5(c) the Compliance violations page.

Compliance Home concentrates on the most important information at a glance, condensed into just one page (compare with Fig. 4). It represents the highest granularity of information. The five colored indicators (top left) are the most relevant, showing the most critical non compliant regulations. The gray indicators (right) report on the compliance with the three main policies. In the bottom, there is the interactive compliance drill-down table containing the compliance performance of business units and processes (rows) in relation to regulations and policies (columns). The user can easily reach lower levels of granularity by drilling down on the table or navigating to pages. For instance, the Rules by Activity page condenses lower level information concerning a com-


Compliance Governance Dashboard
Organization Compliance 77%
Compliance
Regulations
Organizational Structure
OLAP
About
Time Scale
Year
Quarter
Month
Week
Day
Interval
Compliance indicators of most critical regulations
Detailed compliance indicators for most critical policies
Financial policy
QoS policy
Security policy
Interactive Table – Business Units x Regulations
Regulations
Interactive Table – Business Process 1.1 x SOX
Compliance Governa
Compliance 90%
Home
Business
Regul

(a) Compliance Home page of the CGD with graphical widgets for the visualization of indicators and the compliance drill-down table.

(b) Rules by Activity page with process-specific indicators and activity-level compliance info.

Business Process Details

Business Unit Code: BU1
Business Unit: Sales Department
Business Process: Ordering
Business Activity: Ordering daily checking

Regulation Details

Regulation: Sarbanes-Oxley SOX
Policy: Section 301 – Public Company Audit Committees
Rule: R3 "shall provide for appropriate procedures for an issuer to have an opportunity to cure any defects that would be the basis for a prohibition, before the imposition of such prohibition."

Instances Execution Details

Total of instances: 1000
Total of non compliant instances: 100
Average of Compliance Level: Low

BP Instance	BA Instance	Compliance Level	Bach Name	Execution Date	Detection Date	Communication Action	Execution Frequency (week)
1431	434s	High	B123	15/03/09 – 12:02:00	15/03/09 – 13:02:00	Email	5000
2324	7879s	High	B123	15/03/09 – 12:02:02	15/03/09 – 13:02:00	Email	7890
3321	79879d	Low	B123	15/03/09 – 12:02:03	15/03/09 – 13:02:00		100
5754	321321	Medium	B123	15/03/09 – 12:02:03	15/03/09 – 13:02:00	Screen Message	3400
5890	43242s	Low	B124	15/03/09 – 12:30:03	15/03/09 – 13:02:00	Log record	132
5892	434s454	Low	B124	15/03/09 – 12:30:10	15/03/09 – 13:02:00		3213
5893	423	Low	B125	15/03/09 – 12:30:10	15/03/09 – 13:02:00	Log record	3123
5894	657	Low	B125	15/03/09 – 12:30:20	15/03/09 – 13:02:00	Log record	434
7654	764	House Keeping	B125	15/03/09 – 12:30:30	15/03/09 – 13:02:00		10
7655	42344	House Keeping	B200	15/03/09 – 14:30:30	16/03/09 – 07:00:00		90
7656	4324	House Keeping	B200	15/03/09 – 14:30:30	16/03/09 – 07:00:00		4324
7657	4234	Low	B200	15/03/09 – 14:30:30	16/03/09 – 07:00:00	Log record	434
8091	688	House Keeping	B300	15/03/09 – 17:30:30	16/03/09 – 07:00:00		4123
8092	434	House Keeping	B400	15/03/09 – 18:30:30	16/03/09 – 16:00:00		4321
8095	53453y	Low	B400	15/03/09 – 18:30:30	16/03/09 – 16:00:00	Log record	2323

(c) Compliance violations page with low-level details about individual violations for business processes and activities.

Figure 5: Example CGD screenshots of our prototype implementation.

bination of Business Process 1.1 and the company's SOX policy. The colors of the cells represent the compliance performance of each combination. For instance, the Business activity 32.1 presents a critical situation regarding Rule 3 of SOX Section 301 (red cell) and weak performance regarding Rule 5, and Rule 6 (yellow cells).

A drill-down on the red cell, for instance, leads us to the Compliance violations page, which provides the lowest level of abstraction in form of a table of concrete, registered violations of the selected rule. The page illustrates the main information that must be reported to assist internal and external auditors. The data in the particular page reports all violations of one activity in Business Process 1.1 of Business Unit 1, detected considering Rule 3 of SOX Section 301. Each row of the table represents a distinct violation and the columns contain the typical information required by auditors, e.g., responsible of activity, dates and times, mitigation action, outcome of mitigation action, type of applied control, cause of violation, frequency of control activity.

The amount and position of the graphical widgets for indicators, tables, summaries, and so on are chosen in accordance with our short-term memory and the convention of most western languages that are read from left to right and from top to bottom [5].

4 Implementation Usage

The above described concepts are a joint result of the Compas and Master projects, which involve Deloitte and PricewaterhouseCoopers (PwC) as industrial and auditing partners who participated in the design of and approved the models. Both projects share the same functional architecture from a reporting point of view (Fig.6). The CGD is set on the top of a data warehouse (optimized for reporting purposes) that implements the conceptual model described in Fig. 3. It is however important to recognize that this does not affect the logic behind the conceived navigation structure (Fig. 4), which represents a best practice for the rendering of compliance information to auditors, according to the experience by the industrial partners involved in the project.

Both projects produce case studies that have been input - along with the

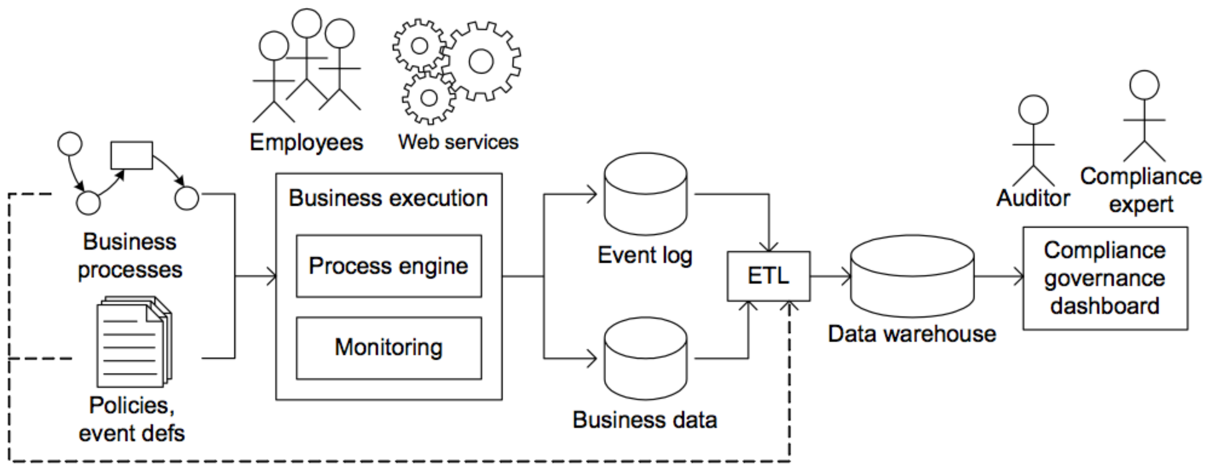


Figure 6: Functional architecture for logging business executions and reporting on compliance.

experience of Deloitte and PwC - to the design of the dashboard. As an example in this paper we briefly describe the WatchMe scenario, developed in the Compas project. This scenario deals with compliance regarding licenses and QoS polices for a mobile virtual network operator (MVNO) in the context of online sales of digital artifacts. The MVNO provides video and audio streams to its mobile phone clients based on pre-defined plans. For instance, the Per-view plan states that clients can acquire (invoke a service) only n streams at price p, while the Time-based plan states that clients can acquire any number of times any possible streams from StartDate till EndDate of the plan. In addition to those plans, the MVNO has also to comply with the licenses defined by each video provider. For instance, Video1 can be downloaded and played with any audio; instead, Video2 can only be used with one specific audio stream.

To govern compliance in this scenario, all compliance concerns are expressed in domain specific languages (DSLs), which are translated to Esper rules for complex event processing during runtime. Events and detected violations are logged and stored in a data warehouse to be used for the computation of KCIs (e.g., amount of violations, clients satisfaction index, average of unauthorized streaming downloads). Different summarization levels and perspectives of analysis are implemented according to the WebML schema in Fig. 4; compliance drill-down units and KCIs (cf. Fig. 5-a) are rendered according to the users' roles; e.g., violation details (low level) to internal auditors or IT personnel for root-cause analysis and main KCIs to external auditors as a start point for the auditing process.

The front-end of the dashboard is an interactive web application. The appealing graphical rendering of indicators is based on FusionCharts widgets, while the compliance drill-down table is AJAX-based. Queries over data are dynamically computed on the server and only rendered inside the client browser. The server-side support is based on Java and JSP. The data warehouse collects execution events, and indicators and process reconstructions are computed at ETL time (weekly or daily).

5 Related Work

Compliance has been investigated in several contexts yielding a variety of approaches. In the following, we discuss related work in three areas that fall in the context of this paper, namely, compliance modeling, compliance dashboards, and Business Activity Monitoring (BAM).

Most of the compliance modeling efforts have been done with the aim of checking compliance, and, therefore, the resulting models consist in formalisms for expressing low-level rules for the compliance requirements. For instance, in [7] the problem of static (i.e., before process execution) compliance checking of process models against compliance rules is addressed by expressing the models in pi-calculus and the corresponding rules in linear temporal logic; then, model checking techniques are used to determine whether a process model complies with the rules or not. In [2], policies are modeled and checked as deontic sentences (i.e., rules are of the form "it is obligatory that X..." or "it is permitted that Y...?"); then, a system can be compliant even if violations occur, in which case, a second-level set of rules might be applied, for which, again, compliance needs to be checked. A similar modeling technique is presented in [8], in which Format Contract Language (FCL), a combination of defeasible logic and deontic logic, is used to express normative specifications. Once the FCL specification is built, control tags can be derived from it and used to annotate the process model so that control concerns can be visualized in the process model space.

To the best of our knowledge, there are no works on dashboards that specifically address the problem of visualizing compliance concerns. However, there are some works that, in part, deal with the problems we address in this paper. For example, [1] studies the problem of designing visualizations (i.e., the representation of data through visual languages) for risk and compliance management. Specifically, the study is focused on capturing the exact information required by users and on providing visual metaphors for satisfying those requirements. In [4], the business performance reporting is provided in a model-driven fashion. The framework provides: data model, navigation model, report template model, and access control model, which jointly help designing a business performance dashboard. However, none of mentioned approaches provides suitable navigation models supporting different analysis perspectives, summarization levels, and user roles.

Business Activity Monitoring (BAM) has gained a lot of attention during the last decade, and many tools have been proposed to support it. BAM aims at providing aggregated information suitable for performing various types of analysis on data obtained from the execution of activities inside a business. For example, tools such as Oracle BAM, Nimbus and IBM Tivoli aim at providing its users with real-time visual information and alerts based on business events in a SOA environment. The information provided to users comes in the form of dashboards for reporting on KPIs and SLA violations. The compliance management part of these tools (if any) comes in the form of monitoring of SLA violations, which need the SLA formal specifications as one of its inputs. In our work, we take a more general view on compliance (beyond SLAs, which are a special case to us) and cover the whole lifecycle of compliance governance, including a suitable dashboard for reporting purposes.

It is important to notice that we do not provide any new compliance checking technique; we rather focus on how to make the most of existing approaches by putting on top of them a visualization logic that is validated by auditors themselves, an aspect that is at least as important as checking compliance. Our work mainly focuses on the case of compliance and provides a conceptual model for both compliance and dashboards, i.e., we present the relevant concepts regarding compliance and visualization and show the interplay of these two aspects. The purpose is that of providing compliance dashboard designers with a holistic and comprehensive view of the business and compliance aspects that characterized a good CGD.

6 Conclusions and Future Work

In this paper we have discussed a relevant aspect in modern business software systems, i.e., compliance governance. Increasingly, both industry and academia are investing money and efforts into the development of compliance governance solutions. Yet, we believe compliance governance dashboards in particular, probably the most effective means for visualizing and reporting on compliance, have mostly been neglected so far. It is important to implement sophisticated solutions to check compliance, but it is at least as important (if not even more) to effectively convey the results of the compliance checks to a variety of different actors, ranging from IT specialists to senior managers.

Our contribution is a conceptualization of the issues involved in the design of compliance governance dashboards in service and process-centric systems, the definition of a navigation structure that naturally supports drill-down and roll-up features at adequate levels of detail and complexity, and a set of concrete examples that demonstrate the concepts at work. Our aim was to devise a solution having in mind the real needs of auditors (internal and external ones) and - more importantly - with the help of people who are indeed involved every day in the auditing of companies.

As a continuation of this work, we are planning to perform extensive usage studies in the context of the projects mentioned earlier. First, such studies will allow us to assess the acceptance of the proposed CGD by auditors in their everyday work. Second, the studies will allow us to understand which support for actions for mitigating compliance problems or violations directly through the dashboard is desirable.

Acknowledgements: This work was supported by funds from the European Commission (contract N 216917 for the FP7-ICT-2007-1 project MASTER and contract N 215175 for the FP7-ICT-2007-1 project COMPAS).

References

[1] R. Bellamy, T. Erickson, B. Fuller, W. Kellogg, R. Rosenbaum, J. Thomas, T. Vetting Wolf. Seeing is believing: Designing visualizations for managing risk and compliance. IBM Systems Journal, 46(2), pp. 205-218, 2007.

[2] J. Brunel, F. Cuppens, N. Cuppens-Boulahia, T. Sans, J. Bodeveix. Security Policy Com- pliance with Violation Management. Proceedings of the 2007 ACM Workshop on Formal Methods in Security Engineering, pp. 31-40, 2007.

[3] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, M. Matera. Designing Data- Intensive Web Applications. Morgan Kaufmann Publishers Inc., USA, 2002.

[4] P. Chowdhary, T. Palpanas, F. Pinel, S.-K. Chen, F.Y. Wu. Model-driven Dashboards for Business Performance Reporting. Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference, pp. 374-386, 2006.

[5] S. Few. Information Dashboard Design: The Effective Visual Communication of Data. O'Reilly Media, Inc., p. 223, 2006.

[6] J. Hagerty, J. Hackbush, D. Gaughan, S. Jacobson. The Governance, Risk Management, and Compliance Spending Report, 2008-2009: Inside the $32B GRC Market. AMR Re- search, 2008.

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

[8] S. Saqid, G. Governatori, K. Naimiri. Modeling Control Objectives for Business Process- Compliance. Business Process Management, pp.149-164, 2007.

[9] C. Giblin, S. Mller, B. Pfitzmann. From Regulatory Policies to Event Monitoring Rules: Towards Model-Driven Compliance Automation. IBM Research Report RZ 3662, Zurich, October, 2006.

[10] K. Namiri, N. Stojanovic. A Semantic-based Approach for Compliance Management of Internal Controls in Business Processes. CAISE'07, pp. 61-64, 2007.

[11] H. Trent. Products for Managing Governance, Risk, and Compliance: Market Fluff or Relevant Stuff? In-Depth Research Report, Burton Group, 2008.

[12] J. Lam. Operational Risk Management ? Beyond Compliance to Value Creation. White Paper, Open Pages, 2007.

[13] L. Imrey. CIO Dashboards: Flying by Instrumentation. Journal of Information Technology Management, 19(4), pp. 31-35, 2006.

[14] G. Evans, S. Benton. The BT Risk Cockpit ? a visual approach to ORM. BT Technology Journal, 25(1), 2007.

[15] E. Sloane, E. Rosow, J. Adam, D. Shine. JEDI - An Executive Dashboard and Decision Support System for Lean Global Military Medical Resource and Logistics Management. Proceedings of the EMBS Annual International Conference, pp. 5440-5443, 2006.

[16] M.P. Papazoglou. Compliance Requirements for Business-process-driven SOAs. E- Government Ict Professionalism and Competences Service Science, July, Volume 280/2008. pp. 183-194, 2008.

[17] A. Read, A. Tarrel, A. Fruhling. Exploring User Preference for the Dashboard Menu De- sign. In Proceedings of the 42nd Hawaii International Conference on System Sciences, pp. 1-10, 2009.

[18] E. Allman. Complying with Compliance. ACM Queue, 4(7), pp. 18-21, September, 2006.

[19] J. Cannon, M. Byers. Compliance deconstructed. ACM Queue, 4(7), pp. 30-37, September, 2006.

[20] E. Oberortner, U. Zdun, and S. Dustdar: Tailoring a Model-Driven Quality-of-Service DSL for Various Stakeholders. Workshop on Modeling in Software Engineering (MiSE), 2009.

[21] F. Daniel, F. Casati, V. D'Andrea, S. Strauch, D. Schumm, F. Leymann, E. Mulo, U. Zdun, S. Dustdar, S. Sebahi, F. de Marchi, M. Hacid: Business Compliance Governance in Ser- vice-Oriented Architectures. Proceedings of AINA'09, IEEE Press, May 2009.

Appendix C

Toward Uncertain Business Intelligence: the Case of Key Indicators

Toward Uncertain Business Intelligence: the Case of Key Indicators *

Carlos Rodríguez Florian Daniel Fabio Casati Cinzia Cappiello

Abstract

Decision support systems and, in particular, business intelligence techniques are widely used by enterprises for monitoring and analyzing operations to understand in which aspects the business is not performing well and even how to improve it. These tools provide valuable results in the context of single departments and business processes, while they are often not suitable in scenarios driven by webenabled intercompany cooperation and IT outsourcing. In such contexts, the adoption of service-oriented company IT architectures and the use of external web services may prevent the comprehensive view over a distributed business process and raise doubts about the reliability of computed outputs. We analyze how these scenarios impact on information quality in business intelligence applications and lead to non-trivial research challenges. We propose the notions of uncertain events and uncertain key indicators, a model to express and store uncertainty, and a tool to compute with and visualize uncertainty.

1 Introduction

The increased usage of IT to support business operations and the advances in business intelligence (BI) techniques create the opportunity for monitoring and analyzing operations to understand in which aspects a business is not performing well and even how to improve it. This has been happening for a while in the context of single departments and business processes, but now it is extending to BI applications that integrate data from multiple departments and even multiple companies. Common examples are the now omnipresent Enterprise

*©2010 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. DOI: 10.1109/MIC.2010.59

Data Warehouse [1], which aggregates process data across departments and geographies; business process outsourcing scenarios, in which the execution of a process is delegated to other companies; or inter-company cooperation, where data and processes are shared across multiple companies.

While BI applications are often complex and comprise multiple kinds of analyses, one of the most widely used metaphors is that of Key Indicators (KI) [2], a set of values that summarize the performance of critical business operations. KIs are used to detect problems and trigger business decisions.

Despite the importance of KIs to business, little attention has been devoted to the expressiveness of KIs if they are computed out of low-quality data and to how possible uncertainties can be communicated to the BI analysts. Even in closed scenarios there are many possible sources of uncertainty in BI applications [4], and the problem is magnified when data comes from multiple sources and is collected with different methods and frequency by different departments, institutions, and geographies. In some cases, uncertainty can easily be predicted or detected (e.g., a partner does not send data on time or a source has an inherently unreliable data collection method), while in others the problems are occasional and harder to recognize. The goal of this article is to understand how to deal with the lack of a comprehensive knowledge about organizational business processes and how to compute meaningful indicators, despite uncertainty in the underlying data.

2 Motivation: Key Assurance Indicators in Healthcare

In the context of the EU project MASTER [1] (Managing Assurance, Security and Trust for sERvices; Euro 9.3M of funding) we are developing diagnostic algorithms to assess and report on compliance, even in presence of uncertain data. So-called Key Assurance Indicators (KAIs) are used to measure performance against compliance requirements, e.g., deriving from a privacy law. Algorithms are being tested in collaboration with Hospital San Raffaele (Mi-

[1] http://www.master-fp7.eu

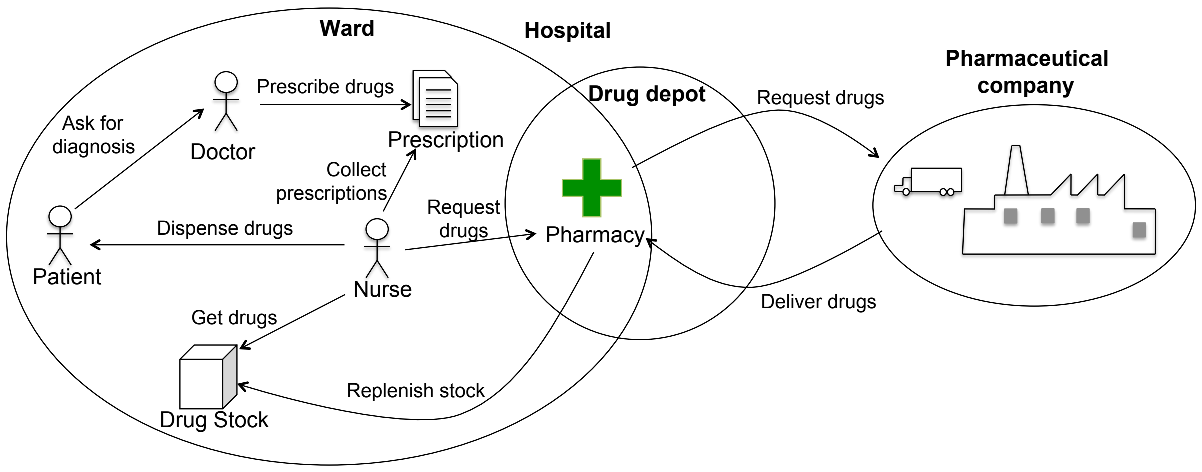


Figure 1: Outpatient drug dispensation in a hospital.

lano, Italy), which provides the necessary, distributed business context: their outpatient drug dispensation process. We summarize the process in Figure 1.

The process starts with the patients visit to the doctor in the hospitals ward. In the case any treatment is needed, the doctor sends an according prescription for drugs to the nurse, and the patient can ask the nurse for the dispensation of the drugs. The nurse collects all drug prescriptions and checks whether all necessary drug quantities are in stock. If yes, he/she can immediately dispense the drugs to the patient. If not, he/she must issue a drug request to the Pharmacy of the hospital, which is then in charge of providing the requested drugs. If, in turn, the Pharmacy is running out of stock, the personnel in charge issues a request to the Pharmaceutical Company that provides drugs to the Pharmacy. By law, the hospital must guarantee that all patient data are anonymized throughout the process, and the hospitals internal policy states that drug replenishment by the Pharmacy must occur within maximum two business days. In order to control, for instance, this latter aspect, the hospital wants to compute a KAI called Average Replenishment Duration (ARD), which allows the hospital to monitor the time it takes to refill the Wards drug stock.

From the IT point of view, the drug dispensation process is supported by several web service-based information systems that interact inside a service-oriented architecture (SOA). For instance, there are web services for issuing drug requests in the various dependencies of the institute, and the pharmaceutical companies the hospital cooperates with accept drug requests through web service interfaces. To retrieve the data requested by the hospitals BI application, during the execution of the process suitable events are generated, which can be logged and analyzed. In this article, we assume each arrow in Figure 1 corresponds to an event in form of a simple SOAP message.

3 Uncertain Events

The above process describes a BI scenario where data are sourced from multiple cooperating entities or companies. This kind of scenario is typically characterized by different levels of visibility into a partners business activities and by different levels of trust in the visible data that can be obtained from each partner.

In the case of cooperative processes (processes that span across organizational domains [3], e.g., the Ward, the Stock management, and the Pharmaceutical company), we can distinguish three kinds of business events: (i) Internal events that stem from the activities that are under the control of the company (the Ward) and consequently are completely visible and trustable. (ii) Shared events that are originated in the activities that are shared with the integrated partner (the Stock management); depending on the technical solution adopted for the implementation of the cooperative part of the process, the visibility into its internals (the events) might be lower than in the case of own activities; similarly, trust into events might be lower. (iii) External events that are part of the partners internal processes; these events are typically hidden to the company, and we cannot analyze them (e.g., we do not have access to the Stock managements internal processes). Similarly, we can associate visibility and trust levels to the case of outsourced processes (the production and shipment of drugs by the Pharmaceutical company). Yet, in this case both visibility and trust are typically lower than in the cooperative process scenario.

Trust and Reputation in Web-based Collaboration

Trust and reputation are concepts studied in different fields, e.g., economics, sociology, computer science, and biology. Although there is a growing literature on theory and applications of trust and reputation systems, definitions are not always coherent[1]. However, the concept of trust is undoubtedly associated with the concept of reliability[2]: trust is the subjective probability by which a party expects that another party performs a given action on which its welfare or business depends[3,1]; reputation is the general opinion about a person, a company, or an object. Therefore, while trust derives from personal and subjective phenomena, reputation can be considered as a collective measure of trustworthiness based on the referrals or ratings from members in a community.

To computer scientists, trust and reputation are particularly significant to support decisions in Internet-based service provisioning. Especially, reputation is able to drive the relationships of individuals and firms in online marketplaces[4,5]. For instance, collaborative filtering systems are used to judge the behavior of a party and to assist other parties in deciding whether or not to start business with that party. A reputation system collects, distributes, and aggregates feedbacks about participants past behavior and discourage unfair behavior[6]. The cross-analysis of different reputation systems enables the realization of mechanisms and methods for the online reputation monitoring and improvement[7].

References

1. D.H. McKnight and N.L. Chervany. The Meanings of Trust. Technical Report MISRC Working Paper Series 96-04, University of Minnesota, Management Information Systems Reseach Center, 1996.
2. A. Jsang R. Ismail C. Boyd. A Survey of Trust and Reputation Systems for Online Service Provision. Decision Support Systems, 43(2), 2007, pp.618-644.
3. D. Gambetta (1988) Trust: Making and Breaking Cooperative Relations, Oxford: Basil Blackwell.
4. M. Fan, Y. Tan, A. B. Whinston. Evaluation and Design of Online Cooperative Feedback Mechanisms for Reputation Management. IEEE TKDE 17(2), 2005, pp. 244-254.
5. G. Zacharia, P. Maes. Collaborative Reputation Mechanisms in Electronic Marketplaces. Proc. 32nd Hawaii International Conf on System Sciences, 1999.
6. P. Resnick, K. Kuwabara, R. Zeckhauser, E. Friedman (2000a). Reputation Systems. Communications of the ACM, 43(12): 45-48.
7. C. Ziegler, M. Skubacz. Towards Automated Reputation and Brand Monitoring on the Web. In Proceedings of the 2006 IEEE/WIC/ACM international Conference on Web intelligence, 2006.

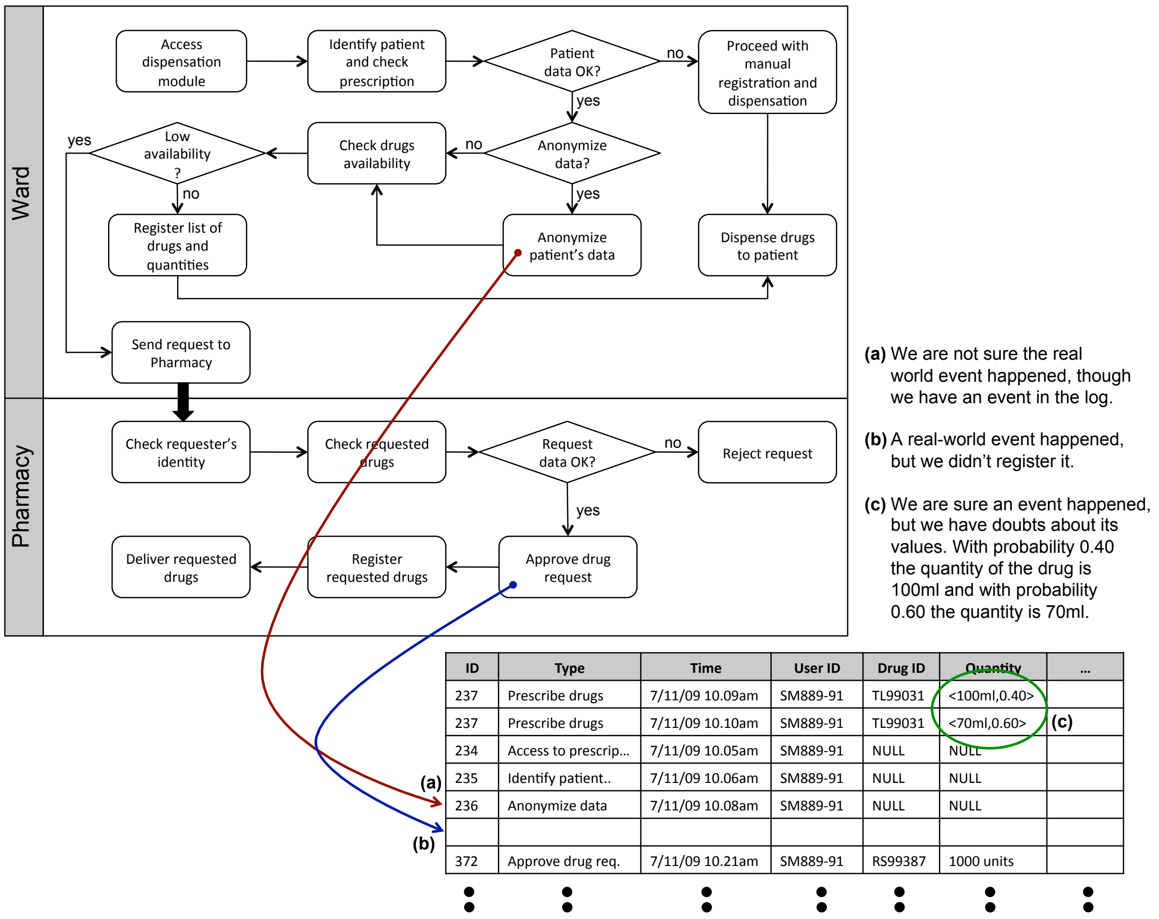
The visibility into shared or outsourced processes has typically structural or organizational roots (e.g., the use of incompatible IT systems or privacy restrictions) that do not frequently change over time. Trust in partners and the information they provide might instead vary with faster dynamics, e.g., based on trust assessment systems that automatically assess trust values for partners from past interactions (see sidebar 1 for more details).

With the use of web services and the SOA, cooperative processes moved to the Web. The consequent reliability problems raise information quality issues in the collection of the events upon which BI algorithms can perform their analyses. In this context, we identify some issues that are strongly related with the way events are collected (the situation is graphically represented in Figure 2):

- We registered an event in the log, yet we are not sure the corresponding real-world event really happened (case (a)). For instance, it may happen that the system is not able to successfully anonymize a patients data, e.g., due to a failure in the algorithm. If the failure is not registered properly, we register a wrong anonymization event.
- A real-world event happened, but we couldnt register it in the log (case (b)). In a running production system, large amounts of events may be published concurrently and, e.g., due to network overloads or system downtimes, events may get lost.
- A real-world event happened, but we have conflicting alternatives for it (case (c)). For instance, it may happen that a doctor prescribes a specific quantity of drug (e.g., 80 ml.), but there are only doses of 100 ml. or 70 ml. available. During data cleaning (before running the BI algorithms) the system may detect the mismatch and track it by keeping both options and associating probabilities to them, trying to reflect the doctors actual intention (see sidebar 2 for details on uncertain data management).

4 Dealing with Uncertainty in Event Logs

We have seen that the data underlying distributed BI is characterized by a number of data deficiencies, i.e., unconformities between the data we have in



ID	Type	Time	User ID	Drug ID	Quantity	...
237	Prescribe drugs	7/11/09 10.09am	SM889-91	TL99031	<100ml,0.40>	
237	Prescribe drugs	7/11/09 10.10am	SM889-91	TL99031	<70ml,0.60>	
234	Access to prescrip...	7/11/09 10.05am	SM889-91	NULL	NULL	
235	Identify patient..	7/11/09 10.06am	SM889-91	NULL	NULL	
236	Anonymize data	7/11/09 10.08am	SM889-91	NULL	NULL	
372	Approve drug req.	7/11/09 10.21am	SM889-91	RS99387	1000 units	

(a) We are not sure the real world event happened, though we have an event in the log.

(b) A real-world event happened, but we didn't register it.

(c) We are sure an event happened, but we have doubts about its values. With probability 0.40 the quantity of the drug is 100ml and with probability 0.60 the quantity is 70ml.

Figure 2: Typical data quality problems in web-based BI.

the event log and what happened in the real world [5]. The challenge is to deal with deficiencies in a way that allows us to perform meaningful analyses, despite the deficiencies. For this purpose, we propose a notion of uncertainty that is composed of three attributes: trust, completeness, and accuracy.

Uncertain/Probabilistic Data Management

In traditional data management, such as in relational databases, data items either exist or not in the database and data that exist are assumed true (they reflect reality) and correct (there are no errors). On the contrary, in Probabilistic/Uncertain Data Management (UDM) this is not taken for granted anymore, and the existence and values of data items are considered probabilistic events. As a consequence, also answering a query over these data becomes probabilistic.

UDM is motivated, among others, by the large number of applications that naturally need to take into consideration uncertainties emerging from the particular domain (e.g., sensor networks and risk analysis) and by the ever increasing speed at which data are automatically generated (e.g., in social networks and real-time systems). In this latter case, noise and incompleteness are ubiquitous because performing cleaning procedures at the same pace at which data is generated is simply impractical. Therefore, the need to manage and process uncertain data is real.

Research on UDM can be grouped into two big areas: uncertain data modeling[1] and query processing on uncertain data[2]. In the former area, the focus is on the modeling of uncertain data in such a way that data can be kept rich and useful for the applications that use them, while keeping the efficiency in terms of physical data management. The latter area addresses the problem of efficiently querying uncertain data, while providing rich semantics to both the definition of queries and the results coming from the query evaluation. Several tools for uncertain data management have been proposed, for instance, Mystic[3], Trio[4], Orion[5], and MayBMS[6].

References

1. T. Green, V. Tannen. Models for Incomplete and Probabilistic Information. Data Eng. Bull., vol. 29(1), 2006.
2. R. Cheng, D. Kalashnikov, S. Prabhakar. Querying Imprecise Data in Moving Object Environments. IEEE Transactions Knowledge and Data Engineering, vol. 16(9), 2004.
3. C. Re, D. Suciu. Managing Probabilistic Data with MystiQ: The Can-Do, the Could-Do, and the Cant-Do. Springer Verlag/Heidelberg, vol. 5291, 2008, pp. 5-18.
4. O. Benjelloun, A.D. Sarma, C. Hayworth, J. Widom. An Introduction to ULDBs and the Trio System. IEEE Data Eng. Bull., 29(1), 2006.
5. S. Singh, C. Mayfield, S. Mittal, S. Prabhakar, S. Hambrusch, R. Shah. Orion 2.0: native support for uncertain data. ACM SIGMOD08, Vancouver, Canada, 2008, pp. 1239-1242.
6. L. Antova, T. Jansen, C. Koch, D. Olteanu. Fast and Simple Relational Processing of Uncertain Data. ICDE08, Cancun, Mexico, 2008, pp. 983-992.

If we model the ideal, i.e., certain, event log as an ordered sequence of events $\overline{L} = [\overline{e}_i]$ (we use the bar to indicate certain data) and an event with k_i data parameters as $\overline{e}_i = \langle d_{i1}, ..., d_{ik_i} \rangle$, the three attributes allow us to deal with the deficiencies describe in Figure 2 as follows:

- Case (a) describes a *meaningless state*, i.e., an event that does not match with any real-world event. Without additional controls, e.g., additional events or certificates from cooperating partners, that specifically aim at identifying this kind of discrepancy, we cannot deal with this situation. What we can do, however, is leveraging on the trust we have in the partner that produced the event. That is, we use a trust measure $t_i \in [0..1]$ as an indicator of the probability with which an event registered in the log is true.

- Case (b) shows an *incomplete representation* of the real world, i.e., the lack of an event. This affects the completeness of the representation of the real-world process and refers to the whole event log. We know about missing events in the log since we know the models of the processes we monitor and the expected sets of events generated by them. In order to keep track of missing events, we associate a completeness measure $comp \in [0..1]$ to L. If we need to report or run algorithms only on subsets of L, e.g., by analyzing data from a given month or year, *comp* will refer to the particular subset.

- Case (c) proposes two different *alternatives* for the same real-world event. This leads to a problem with the accuracy of the event, since we cannot provide a single description but only a set of possible alternatives for the event. That is, each event may have a set of possible worlds (the alternatives) for its parameters $\{d_{ij1}, ..., d_{ijk_i}\}$, where the index j identifies each alternative. To keep track of the likelihood of each possible world, we associate to each world j a probability p_{ij}, where $\Sigma_{j=1}^{J_i} p_{ij} = 1$ and J_i is the number of alternatives. Each possible world has its own probability of being the right description of the real world.

In summary, we represent an uncertain event log as a tuple $L = \langle [e_i], comp \rangle$ (we omit the bar for uncertain data), with $[e_i]$ being the chronological sequence of uncertain events stemming from all the business processes we want to analyze and *comp* being the completeness of the log; and we model uncertain events as

$e_i = \{\langle d_{ij1}, ..., d_{ijk_i}, p_{ij}, t_i\rangle\}$, where the parameters d_{ijk} are the parameters of the events (e.g., the cost of a product) or event meta-data (e.g., the identifier of the event or its timestamp), p_{ij} are the probabilities of the possible worlds, and t_i is the trust level associated with the event.

In this article we do not focus on how the individual uncertainties for events are computed. We rather tackle the problem of how to represent uncertainty and how to compute with it.

5 Modeling, Computing and Visualizing Uncertain Key Indicators

KIs are typically associated with specific business processes, e.g., the execution time of a process or the delay between two activities. In order to specify a KI, we therefore imagine having a view over the event log that filters out the events of the process we are interested in and groups them according to executed process instances. The result is a set of event traces $\{t_l\} = \{[e_{l1}, ..., e_{ln_l}]\}$ with n_l being the number of events in each trace. This allows us to obtain KIs in the form of $\overline{KI}(\{\bar{t}_l\}) = v$ with $v \in \mathbb{R}$ being the scalar value of the indicator.

In the case of uncertain data, it is no longer appropriate to interpret KIs as simple, scalar values. We propose the idea of uncertain key indicator (UKI) as a means to convey to the business analyst both a value for the indicator and the uncertainty associated with it. A UKI can be defined as:

$$UKI(\{t_l\}) = \langle\{v_m, p_m\}, conf, comp\rangle \tag{1}$$

The set $\{\langle v_m, p_m\rangle\}$ represents the possible worlds for the values v_m of the indicator, and p_m is the probability for each of the alternatives. The number of possible worlds depends on the number of possible worlds of the events involved in the computation of the indicator. Specifically, the indicator will have $\prod_n J_n$ possible worlds, where J_n refers to the number of possible worlds of the event e_n in the event traces. The parameter $conf \in [0..1]$ represents the confidence we have in the correctness of the computed possible worlds; we compute this confidence by aggregating the trust levels of the events considered by the indicator. The parameter *comp* is the completeness of the data over which the UKI is computed.

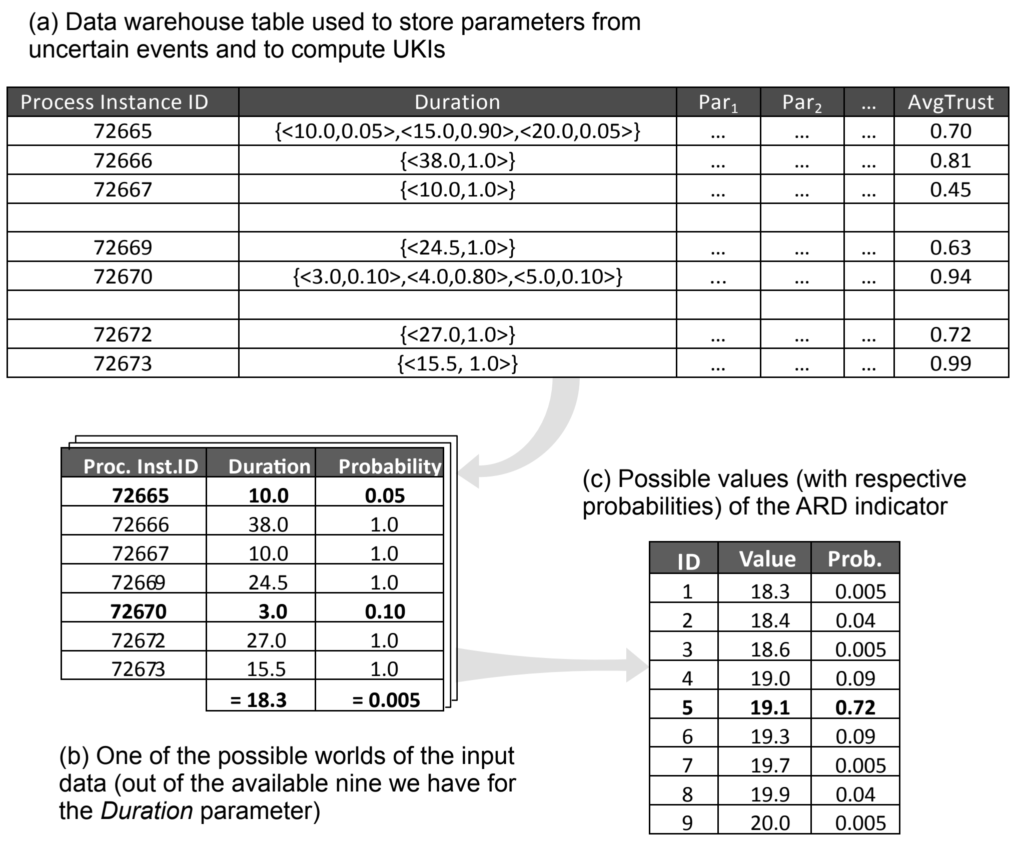
Let us consider the case of the ARD (Average Replenishment Duration) indicator, which is computed as the average time in hours needed to replenish drugs in the Wards drug stock. Figure 3(a) shows an excerpt of the data warehouse we use to store event data for reporting and analysis. Specifically, the table shows the parameters extracted from the event traces of the drug replenishment process (a sub-process of the drug dispensation process) that are used to compute indicators: each tuple corresponds to an executed process instance. The column Duration tells us how many hours each replenishment took; its values are expressed as a set of pairs $\{\langle durantion_{ij}, p_{ij}\rangle\}$ obtained during ETL and data cleansing. The column $AvgTrust$ contains the average of the trust values associated with the events in each trace.

In order to compute ARD, it is necessary to consider individually each possible world that emerges from the data in Figure 3(a). For instance, Figure 3(b) shows one possible world constructed by using the first alternatives for both tuples 72665 and 72670 and a first value for ARD ($v_1 = avg(Duration) = 18.3$) with its probability ($p_1 = \prod_{Proc.Inst.ID} = 0.01$). Applying the same logic to the other eight possible worlds allows us to compute all possible worlds of ARD as shown in Figure 3(c). The combination $\langle 19.1, 0.72\rangle$ is the most likely, though the other combinations cannot be excluded.

In order to obtain the overall confidence ($conf$) we have in the indicator as computed in Figure 3, we average the $AvgTrust$ values in Figure 3(a), which gives us a value of $conf = 0.75$. Finally, in Figure 3(a) we lack two tuples, i.e., process instances. The completeness for ARD is therefore $comp = 7/9 = 0.78$. Thus, the uncertain representation of ARD is:

$$ARD = \langle\{\langle 18.3, 0.01\rangle, \langle 18.4, 0.04\rangle, \langle 18.6, 0.01\rangle, \langle 19.0, 0.09\rangle, \langle 19.1, 0.72\rangle, \langle 19.3, 0.09\rangle, \langle 19.7, 0.01\rangle, \langle 19.9, 0.04\rangle, \langle 20.0, 0.01\rangle\}; 0.75; 0.78\rangle \quad (2)$$

But how do we compute and visualize UKIs in practice? Figure 4(a) shows a simplified version of the infrastructure being developed in the context of the MASTER project: process definitions instrumented with compliance an-



(a) Data warehouse table used to store parameters from uncertain events and to compute UKIs

Process Instance ID	Duration	Par$_1$	Par$_2$	…	AvgTrust
72665	{<10.0,0.05>,<15.0,0.90>,<20.0,0.05>}	…	…	…	0.70
72666	{<38.0,1.0>}	…	…	…	0.81
72667	{<10.0,1.0>}	…	…	…	0.45
72669	{<24.5,1.0>}	…	…	…	0.63
72670	{<3.0,0.10>,<4.0,0.80>,<5.0,0.10>}	…	…	…	0.94
72672	{<27.0,1.0>}	…	…	…	0.72
72673	{<15.5, 1.0>}	…	…	…	0.99

Proc. Inst.ID	Duration	Probability
72665	**10.0**	**0.05**
72666	38.0	1.0
72667	10.0	1.0
72669	24.5	1.0
72670	**3.0**	**0.10**
72672	27.0	1.0
72673	15.5	1.0
	= 18.3	**= 0.005**

(b) One of the possible worlds of the input data (out of the available nine we have for the *Duration* parameter)

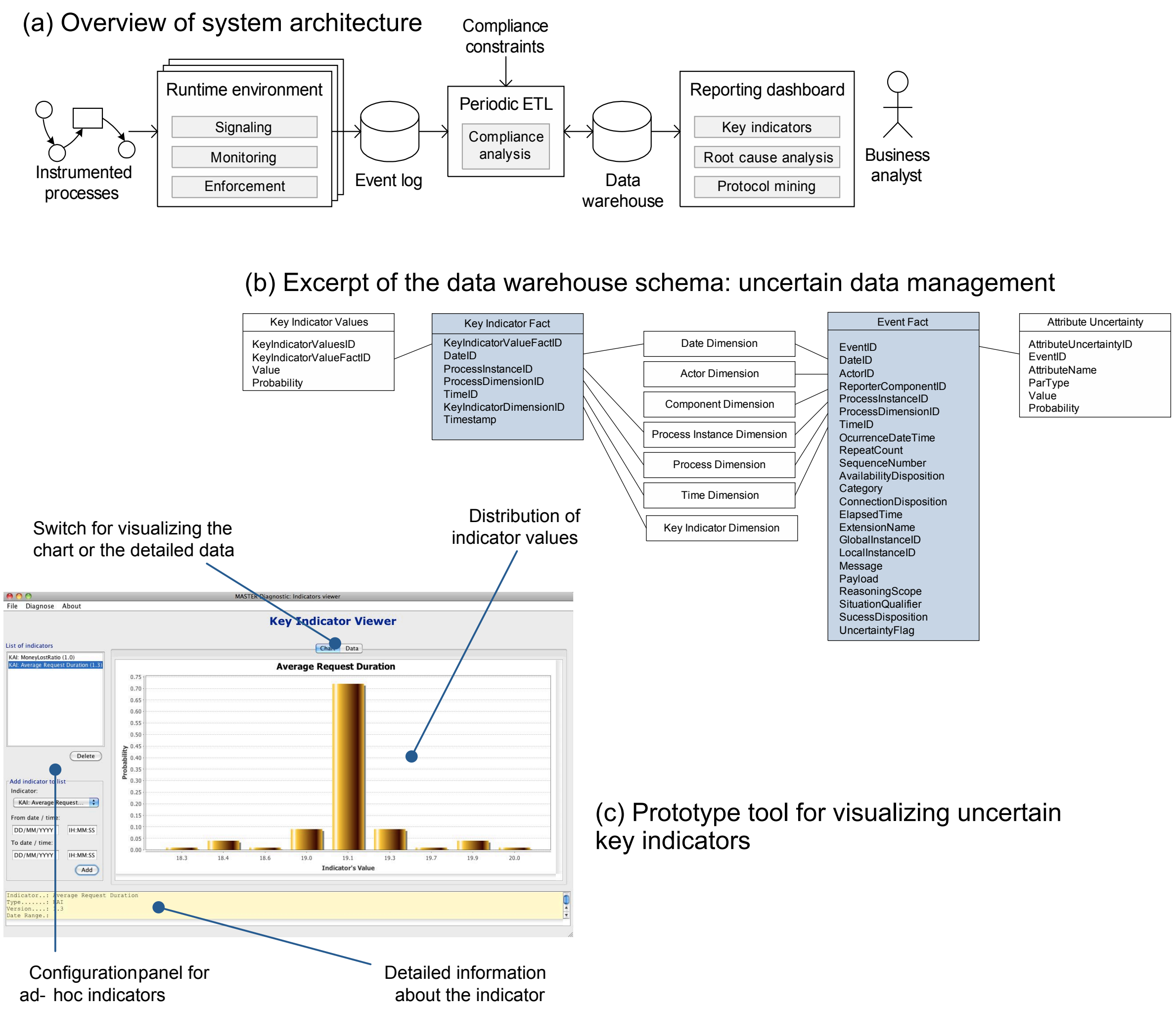
(c) Possible values (with respective probabilities) of the ARD indicator

ID	Value	Prob.
1	18.3	0.005
2	18.4	0.04
3	18.6	0.005
4	19.0	0.09
5	**19.1**	**0.72**
6	19.3	0.09
7	19.7	0.005
8	19.9	0.04
9	20.0	0.005

Figure 3: Example computation of the ARD indicator.

notations feed one or more runtime environments (e.g., operated by different partners) that execute the processes and signal, monitor, and enforce behaviors according to the annotations. Doing so produces events, which we log and periodically load into a data warehouse, where we also check the compliance of executed processes. We store all execution data for reporting (in the reporting dashboard) and analysis (key indicators, root cause analysis, protocol mining).

Figure 4(b) illustrates an excerpt of the dimensional data warehouse model [6], showing how we physically store uncertain data and uncertain key indicators in the warehouse. Fact tables are shaded gray, dimension and uncertainty metadata tables are white. The Event Fact table stores the events loaded from the event log. Dimensions that can be used to perform queries and multidimensional analysis are, e.g., Component Dimension, Process Instance Dimension, and Date Dimension. The auxiliary Attribute Uncertainty table stores uncertainty



```
Indicator..: Average Request Duration
Type.......: KAI
Version....: 1.3
Date Range.:
```

Figure 4: Storing events and computing and visualizing uncertain key indicators.

meta-data for the attributes of the Event Fact table. UKI values are stored in the Key Indicator Value Fact and Key Indicator Values tables. The former can be joined with the dimension tables it is associated with to support queries and multidimensional analysis. The latter is again an auxiliary table that stores the actual (uncertain) indicator values. The computation of an UKI therefore translates into a set of SQL statements evaluated over the data warehouse.

Finally, it is important to properly visualize UKIs in a dashboard, where the important aspects of the monitored business processes can be inspected at a glance. The challenge is to convey the uncertainty of UKIs to the business analysts, while keeping visual metaphors as simple and concise as possible. We approach this problem in a parallel line of research [7][8] where we work on the development of effective reporting dashboards. In Figure 4(c) we show a screenshot of our tool for the visualization of UKIs, which the business analyst can start by drilling down on uncertain indicators in the dashboard. The tool allows the analyst to inspect all uncertainty aspects introduced in this paper (possible worlds, confidence and completeness) and to write ad-hoc queries to better understand the nature of the underlying data.

6 Conclusion and Outlook

The discussion in this article follows in a way the footsteps of other areas of science, mainly in physics, where uncertainty has become a key ingredient when modeling reality. We believe the same should be done in information engineering, recognizing that our ability to observe reality is not as precise as we would like.

The result of the work presented here is a model for representing this imprecision in terms of uncertain events and uncertain indicators, an approach to store uncertainty metadata and compute uncertain indicators, and a tool to communicate uncertainty to users. While this is useful in its own right, the main contribution lies however in providing a basis for uncertainty in BI applications, as this is the branch that is concerned with understanding and analyzing the real world. Indicators are just one (although significant) aspect of BI applications, but what organizations aim at is understanding and improving their processes. On the understanding side, we are now adopting the uncertain data model introduced in this article in the context of process discovery from uncertain data. On the improvement side, we are applying the model to analyze the root causes of compliance violations, specifically working toward techniques like uncertain decision trees and correlation analysis of uncertain data. The computation model presented in this article is the conceptual basis for the outlined research and a first step toward a theory of uncertainty in business intelligence in general.

Acknoledgements. This work was supported by funds from the European Commission (contract N 216917 for the FP7-ICT-2007-1 project MASTER).

References

[1] B. H. Wixom, H. J. Watson An Empirical Investigation of the Factors Affecting Data Warehousing Success. MIS Quarterly, Vol. 25, No. 1 (Mar., 2001), pp. 17-41.

[2] R.S. Kaplan, D.P. Norton. The Balanced Scorecard: Translating Strategy into Action. Harvard Business School Press, USA (1996).

[3] M. Weske. Business Process Management: Concepts, Languages, Architectures. Springer, 2007.

[4] F. Daniel, F. Casati, T. Palpanas, O. Chayka. Managing Data Quality in Business Intelligence Applications. QDB'08, Auckland, New Zealand, 2008.

[5] Y. Wand and R.Y. Wang. Anchoring Data Quality Dimensions Ontological Foundations. Communications of the ACM, 39(11), November 1996.

[6] D. Hollingsworth. The Workflow Reference Model. TC00-1003, Workflow Management Coalition, January 1995.

[7] C. Rodríguez, F. Daniel, F. Casati, C. Cappiello. Computing Uncertain Key Indicators from Uncertain Data. ICIQ'09, Postdam, Germany, 2009.

[8] P.Silveira, C.Rodríguez, F.Casati, F. Daniel, V.D'Andrea, C.Worledge, Z.Taheri. On the Design of Compliance Governance Dashboards for Effective Compliance and Audit Management. NFPSLAM-SOC'09, Stockholm, Sweden, 2009.

Appendix D

Analyzing Compliance of Service-Based Business Processes for Root-Cause Analysis and Prediction

Analyzing Compliance of Service-Based Business Processes for Root-Cause Analysis and Prediction *

Carlos Rodríguez Patricia Silveira Florian Daniel Fabio Casati

Abstract

Automatically monitoring and enforcing compliance of service-based business processes with laws, regulations, standards, contracts, or policies is a hot issue in both industry and research. Little attention has however been paid to the problem of understanding non-compliance and improving business practices to prevent non-compliance in the future, a task that typically still requires human interpretation and intervention. Building upon work on automated detection of non-compliant situations, in this paper we propose a technique for the root-cause analysis of encountered problems and for the prediction of likely compliance states of running processes that leverages (i) on event-based service infrastructures, in order to collect execution evidence, and (ii) on the concept of key compliance indicator, in order to focus the analysis on the right data. We validate our ideas and algorithms on real data from an internal process of a hospital.

1 Introduction

Compliance means conformance with laws, regulations, standards, contracts, policies, or similar sources of requirements on how to run business. Effective compliance management, i.e., the practice of assuring compliance, is an increasingly more important concern in today's companies, since the set of compliance requirements a company has to implement grows fast and their effect on the traditional business practices in a company may be considerable. Despite its increasing importance, compliance is however to a large extent still managed in rather ad-hoc ways and with little or no IT support. As a result, today it is very hard for any CFO or CIO to answer questions like: Which requirements does my company have to comply with? Which processes should obey which

*The final publication is available at www.springerlink.com (http://link.springer.com/chapter/10.1007%2F978-3-642-16985-4_25) requirements? Which processes are following a given regulation? Where do violations occur? Which processes do we have under control? And so on. While IT has been supporting (in more or less automated fashions) the execution of business processes for long time now, in the past the adoption of ad-hoc and monolithic software solutions did not provide the necessary insight into how processes were executed and into their runtime state, preventing the adoption of IT also for compliance assessment. The advent of workflow management systems and, especially today, of web service-based business interactions and the service-oriented architecture (SOA) have changed this shortcoming, turning business processes into well-structured, modular, and distributed software artifacts that provide insight into their internals, e.g., in terms of execution events for tasks, service calls, exchanged SOAP messages, control flow decisions, or data flows. All these pieces of information can be used for online monitoring or enforcement of compliant process behaviors or they can be logged for later assessment. Unfortunately, however, the resulting amount of data may be huge (in large companies, hundreds of events may be generated per minute!), and especially in terms of reporting and analysis it is not trivial to understand which data to focus on and how to get useful information out of them. Doing so is challenging and requires answering questions like how to collect and store evidence for compliance assessment in service-based business processes, how to report on the compliance state, and how to support the analysis of non-compliant situations. But more than these, the challenges this paper aims to solve are how to collect evidence in a way that is as less intrusive as possible, how to devise solutions that are as useful as possible, yet at the same time as generic as possible and independent of the particular IT system to be analyzed, and, finally, how to provide compliance experts with information that is as useful and expressive as possible. In light of these challenges, this paper provides the following contributions:

- A method for the definition and a dashboard for the visualization of so-called Key Compliance Indicators (KCIs) for at-a-glance reporting on compliance;
- An algorithm and a tool for the mining of decision trees from process

execution logs that particularly look at data from the perspective of compliance;

- An application of the algorithm mining approach to real-world data stemming from a typical business process running in a large Italian hospital.

In the next section we provide the necessary details about this process and highlight its compliance requirements, so as to derive the requirements for this paper in Section 3. In Section 4 and 5, we then discuss how to report on compliance and how to analyze non-compliance, respectively. In Section 6 we discuss some related works, and in Section 7 we conclude the paper.

2 Scenario: Drug Reimbursement in Hospitals

Let us consider the case of a drug reimbursement process in the healthcare domain. The process is the case study in one of our EU projects, where we cooperate with Hospital San Raffaele (Milan, Italy), which runs the process shown in Figure 1. The overall purpose of this process, from the hospital's point of view, is to obtain reimbursements from the Italian Health Authority for the drugs dispensed to outpatients (i.e., patients that are not hospitalized). In order to obtain the reimbursement, there are many compliance requirements imposed by the Health Authority, among which we mention privacy preservation in personal information processing, separation of duties, and the adherence of standard template of dispensation reports.

The core process that generates the information that needs to be sent to the Health Authority occurs inside the Ward. The process starts when a patient visits the hospital's ward to consult a doctor. After diagnosing the patient, the doctor prepares a drug prescription that is delivered to a nurse, who is in charge of dispensing the prescribed drugs to the patient. If the amount of drugs is going below a certain threshold, the nurse issues a drug request to the central pharmacy of the hospital, which must replenish the ward's drug stock in no later than 48 hours. The execution of this process is fully supported by the ward's SOA-based information system, and all progress events generated during process executions are recorded in an event log for later inspection.

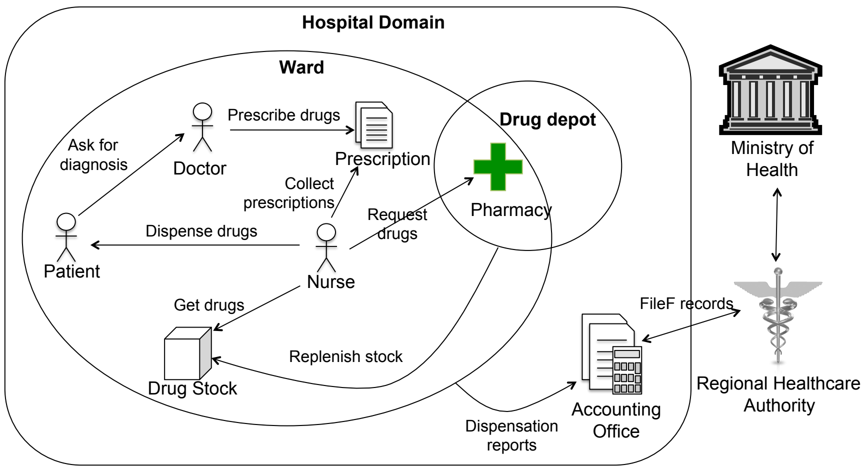


Figure 1: Summary of the direct drug reimbursement process.

While the process above is executed daily, the preparation of dispensation reports for drug reimbursement is a monthly task. That is, at the end of each month, the records of drug dispensations are collected from the various wards of the hospital and the corresponding dispensation reports to be sent to the Health Authority are created. These reports consist in simple text files (known as FileF) in which data about the dispensations are included. Examples of data included in these files are hospital identification, patient, doctor, dispensed drug and quantity, and amount in Euros. Whenever the report is ready it is sent to the Health Authority, which checks the quality of the report against some compliance requirements imposed on dispensation reports. For instance, one compliance requirement that decides whether a dispensation can be reimbursed or not regards the completeness and correctness of records: no null or incorrect data are tolerated in any field. If there are such problems in the report, the Health Authority sends a feedback to the hospital indicating the number and type of errors found for each record of the file, and, in turn, the hospital must correct them so as to get the reimbursement.

The complete reimbursement process is complex, and not complying with the applicable requirements can be costly. Therefore, in order to better control the compliance of the reimbursement process, the hospital wants to implement an early warning system that allows the hospital's compliance expert to have updated information on daily compliance issues, e.g., in form of indicators, reports, or predictions on the compliance of its processes. In addition, in case of repeated problems, it is important to understand why they happen and how they can be solved for the future. However, manually analyzing the data in the event log is time consuming and also error-prone but, still, the hospital wants to improve its compliance in order not to lose money for not reimbursed drug dispensations.

3 Service-Oriented Compliance Management: Requirements

The above scenario describes a service-based business process that is distributed over the hospital's ward and the drug depot and that asks for proper compliance management, that is, compliance assessment, reporting, and analysis.

As this paper has its roots in two EU FP7 research projects, i.e., Compas and Master, that both assist compliance assessment in the SOA, here we do not propose a new assessment technique and rather rely on the techniques proposed there: Compas (www.compas-ict.eu) strongly focuses on model-driven development of compliant processes and proposes a compliance checking approach that is based on (i) compliance requirements expressed in logical rules or process fragments and (ii) complex event processing (CEP) and business protocol monitoring to detect non-compliance with requirements. Master (www.master-fp7.eu), instead, specifically focuses on the security domain and proposes a two-layered approach to compliance assessment: first, it supports the CEP-based monitoring of running processes and the enforcement of individual rules; then, offline, it checks compliance of executed processes by assessing their conformance to a so-called ideal process model. Both approaches have in common the use of an instrumented service orchestration engine for the execution of business processes and the generation/logging of suitable execution events, starting from a signaling policy that specifies which events are necessary for compliance assessment.

Building on this background, reporting on the state of compliance requires being able to store process execution and compliance data and to develop a reporting dashboard on top, a task that we partly approached in [1]. But we also need to devise a method for the easy specification and, then, automated computation of key compliance indicators (KCIs), in order to visualize them in the dashboard. Next, the analysis of root-causes for non-compliance requires selecting a suitable analysis algorithm and more importantly understanding which data to look at, out of the huge amount of data that is available for this task, and to validate the algorithm in the context of the described scenario.

4 Reporting on Compliance

In order report on the compliance of business processes, the common approach is to visualize the compliance status at a high-level of abstraction, for instance, by means of KCIs that are graphically rendered in a compliance governance dashboard (CGD) [1]. KCIs support compliance experts with an overview of the compliance performance of business processes and can be seen as particular type of KPIs (key performance indicators) that specifically measures how compliant a process is with given requirements. A typical KCI may, for example, measure how many process instances, out of all the executed ones, satisfy a separation of duties requirement; but also a traditional QoS indicator (e.g., the average process execution time) can be seen as KCI, if we are subject to a compliance requirement regarding QoS (e.g., deriving from a contract with the customer). As we will see, KCIs also provide a starting point for finding the root-causes of non-compliance. This section explains how we store process execution data, specify and compute KCIs, and visualize them through effective visual metaphors.

4.1 Storing Process Execution and Compliance Data

The main sources of process execution and compliance data are the event logs generated by the execution of service-based business processes. Therefore, let us first conceptualize the key ingredients characterizing event logs, as we perceive them for our analysis. An event is a tuple $e = \langle t, s, ts, d, p_1, ..., p_n, B \rangle$, where t is the type of the event (e.g., ProcessStart, ActivityExecuted, Violation), s is the source that generates the event, ts is a timestamp, $p_1, ..., p_n$ is a set

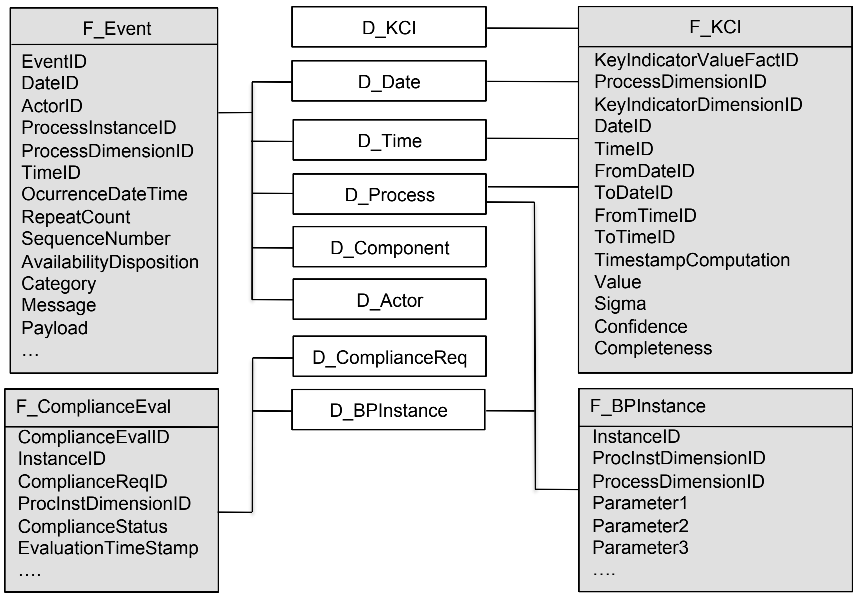
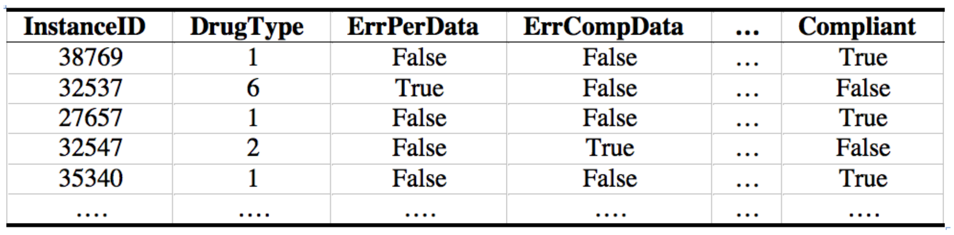


Figure 2: Simplified schema of the data warehouse model.

of properties (e.g., event message header properties such as correlation data, process instance identifier or similar), and B is the body of the event message (e.g., containing business data needed for the computation of an indicator). Using this data, events can be grouped together by their process instance and ordered by timestamp, forming this way traces. A trace is a sequence of events $T_i = \langle e_{i1}, e_{i2}, ..., e_{in} \rangle$, where i refers to a process instance identifier and n is the number of events that compose the process instance. This way, an event log can be expressed as a set of traces $L = \{T_1, T_2, , T_k\}$, where k is the total number of traces.

The events in the log are processed by Extract-Transform-Load (ETL) flows, in order to store them into a data warehouse (DW), which is modeled using a compliance-oriented dimensional data model. The reason for doing this is that we aim at leveraging the capability of dimensional models for keeping a conciliated view on the process execution and compliance data, and for supporting further analysis, e.g., by means of root-cause analysis algorithms or Online Analytical Processing (OLAP) tools. Figure 2 shows an excerpt of the schema of the DW. The tables in white are the dimensional tables that allow us to slice and dice through the fact tables (shaded in gray). The fact table F_Event stores the events as they come from the event log, F_KCI stores the



InstanceID	DrugType	ErrPerData	ErrCompData	...	Compliant
38769	1	False	False	...	True
32537	6	True	False	...	False
27657	1	False	False	...	True
32547	2	False	True	...	False
35340	1	False	False	...	True
....	....	....	....	...	....

Table 1: Example of a process instance table for the drug dispensation process.

computed values of indicators, $F\_BPInstance$, the instances of processes, and $F\_ComplianceEval$, the compliance status of process instances as computed, for instance, by the compliance checking algorithms adopted in the context of the Compas or Master projects.

The $F\_BPInstance$ table deserves a further explanation, as it constitutes an abstraction of the process execution data, and the basis for computing indicators and performing root-cause analysis. In our DW model, each business process BP has its own $F\_BPInstance$ table, or, as we call it, process instance table (e.g., in our scenario we have a $F\_DrugDispensationInstance$ table). In these tables, each row corresponds to an instance of the associated process, while columns (i.e., parameters of the process instance table) correspond to business data that are of interest for the analysis of each process. Table 1 shows a conceptual view on the process instance table for the drug dispensation process, where each row corresponds to a single drug dispensation. The DrugType column refers to the type of drug, ErrPerData indicates whether there was an error in the information about the patient, ErrCompData tells us if there was an error in any other complementary data, and Compliant tells us whether the dispensation was free of error. These parameters are obtained from the attributes of the events that are part of the event trace. Sometimes, the parameter values can be directly extracted from events without modifications (e.g., the DrugType parameter), while in other cases the values are obtained by performing aggregation/computations over a set of events and attributes of process instances (e.g., the Compliant parameter).

Finally, it is worth to mention that in order to populate the DW, the ETL usually needs to access other sources of data such as user management systems and human task managers, which are the main data providers for dimension tables, as opposed to event logs, which provide mostly the evidences of process executions.

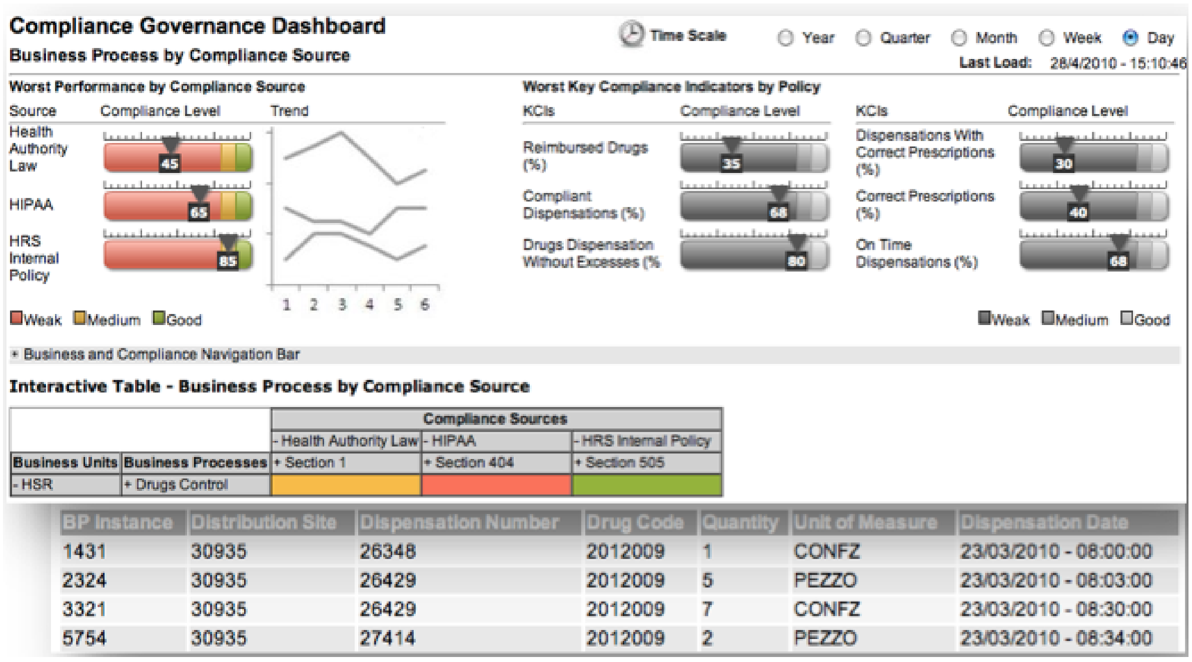
4.2 Specifying and Computing Key Compliance Indicators

Generally, indicators are computed out of a variety of data and by means of different functions, ranging from the lowest business data granularity to the highest business goals. In the context of compliance assessment, a KCI is a measure (i.e., a numeric value) that quantifies compliance performance against compliance targets in a pre-determined time interval. For instance, one of the compliance requirements imposed by the Healthcare Authority is that of sending drug dispensation reports without errors in the data about dispensed drugs and patients. Whenever there is an erroneous record of drug dispensation, the corresponding drug is not reimbursed to the hospital, and, thus, it is important for the hospital to keep an eye on the accomplishment of this compliance requirement. KCIs are therefore useful means to assist this task.

KCIs can be easily specified by using the available information in Table 1. For example, a KCI may be defined as the percentage of non-compliant process instances out of all instances in the DW (and the reporting time interval). More precisely, we can use the Compliant column of a process instance table to compute KCIs, and we can express their respective formulas using standard SQL queries. SQL has been designed also as a language for computing aggregates and is well known, understood, and supported, so there was no reason to come up with another language. Yet, the ease with which we are able to express KCIs stems from the abstraction we made on the process execution data by using the so called process instance tables.

4.3 Compliance Governance Dashboard

Finally, KCIs are rendered to the compliance experts by means of a CGD, such as the one depicted in Figure 3 [1]. The CGD features are a graphical representation of KCIs and serves as start point for further root-cause analysis. More specifically, the CGD creates an awareness of possible violations and concentrates the most important information to be evaluated at-a-glance. The


Compliance Governance Dashboard
Business Process by Compliance Source
Time Scale
Year
Quarter
Month
Week
Day
Last Load: 28/4/2010 - 15:10:46
Worst Performance by Compliance Source
Source
Compliance Level
Trend
Health Authority Law
45
HIPAA
65
HRS Internal Policy
85
1 2 3 4 5 6
Weak Medium Good
Worst Key Compliance Indicators by Policy
KCIs
Compliance Level
Reimbursed Drugs (%)
35
Compliant Dispensations (%)
68
Drugs Dispensation Without Excesses (%
80
Dispensations With Correct Prescriptions (%)
30
Correct Prescriptions (%)
40
On Time Dispensations (%)
68
Weak Medium Good
+ Business and Compliance Navigation Bar

Interactive Table - Business Process by Compliance Source

		Compliance Sources		
		- Health Authority Law	- HIPAA	- HRS Internal Policy
Business Units	Business Processes	+ Section 1	+ Section 404	+ Section 505
- HSR	+ Drugs Control	🟧	🟥	🟩

BP Instance	Distribution Site	Dispensation Number	Drug Code	Quantity	Unit of Measure	Dispensation Date
1431	30935	26348	2012009	1	CONFZ	23/03/2010 - 08:00:00
2324	30935	26429	2012009	5	PEZZO	23/03/2010 - 08:03:00
3321	30935	26429	2012009	7	CONFZ	23/03/2010 - 08:30:00
5754	30935	27414	2012009	2	PEZZO	23/03/2010 - 08:34:00

Figure 3: CGD with KCIs and the interactive table for drill-down and roll-up [1].

interactive table (at the front in Figure 3) provides a drill-down and roll-up mechanism for the compliance status, for example, for the different drug dispensation locations controlled by the hospital (i.e., clinics, laboratories, dispensaries), according to two main analysis perspectives (compliance performance vs. process performance), down to the individual event level (e.g., the list of incomplete records associated to a drug (background of Figure 3).

5 Analyzing Non-Compliance

While checking the compliance of business process instances means determining whether the process instances are compliant or not at the individual event trace level, analyzing non-compliance of business process executions, i.e., understanding and explaining the underlying reasons of non-compliance, needs to be performed over a set of traces in order to be able to derive meaningful knowledge that can be used to improve processes for future executions.

Incidentally, labeling event traces as compliant or non-compliant, which is the main goal of compliance checking, is very similar to classifying data tuples, a data mining practice that is well-studied in literature [20]. There are several algorithms that can help in performing this analysis, among which we choose decision trees, as they are good for knowledge discovery where neither complex settings nor assumptions are required [20], and they are easy to interpret and analyze. In this section, we discuss how we address the issue of compliance analysis through decision trees, going from data preparation to the actual building and interpretation of the decision tree.

5.1 Preparing the Analysis

In Section 4.1, we introduced our DW model, which constitutes the basis for our CGD and the root-cause analysis. Preparing the analysis therefore means selecting which data, out of the huge amount of events stored in the DW, are suitable for identifying root-causes for non-compliance. In the same section, we also introduced the idea of having process instance tables, one per process, in which we store those process parameters that are used for computing indicators. Recall that each tuple in a process instance table represents a particular instantiation of the process under consideration and that each instance comes with its compliance label. Now, considering that we are interested in analyzing non-compliance problems for process instances, it is interesting to note that the process instance tables initially conceived for the computation of indicators also contain the data we are searching for. In fact, by defining a set of indicators for each process (and the events and data attributes that are necessary to compute them), the compliance expert implicitly performs a pre-selection of the data that are most likely to be related with compliance issues. The availability of the compliance label for each instance indicates that the best choice for the root-cause analysis is to use the process instance tables to feed the decision tree mining algorithm, as their data naturally fits the typical input format of these kinds of algorithms.

For instance, considering again the process instance table shown in Table 1, one way of building the training tuples for the decision tree is to use the Compliant column as the class attribute (leaf nodes) for the decision tree, while ErrPerData and ErrCompData can be used as the attributes on which the

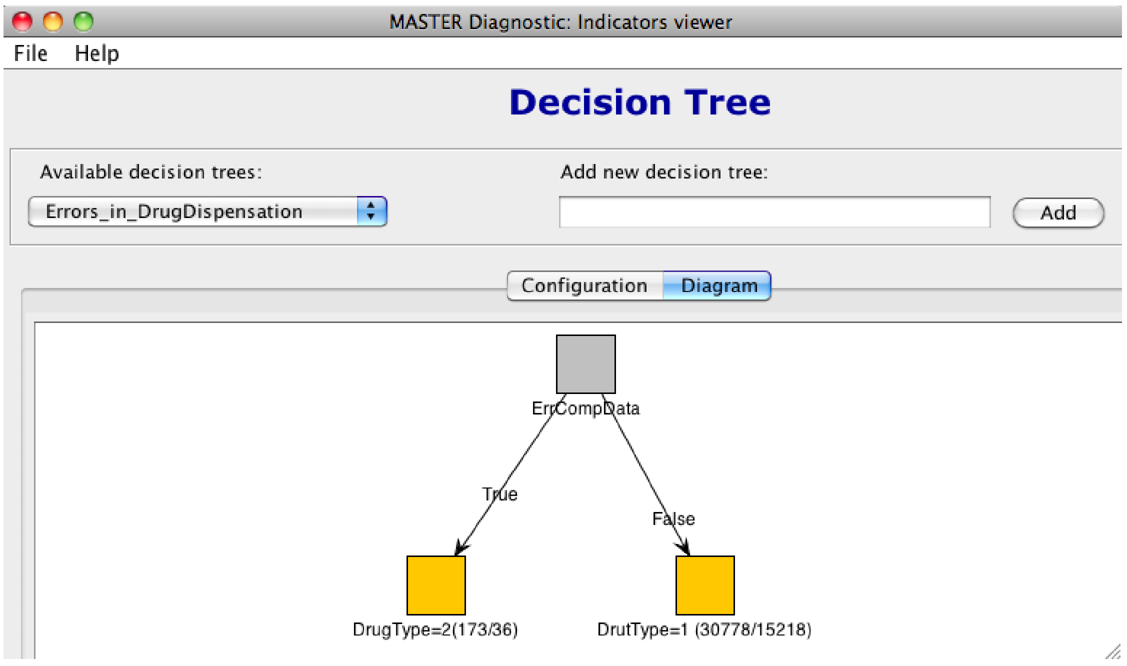


Figure 4: Decision tree computed over non-compliant instances of the drug dispensation process.

algorithm defines the split points (for internal nodes). This way, the training tuples can be represented as $\langle ErrPerData, ErrCompData, Compliant\rangle$.

The set of training tuples can be easily obtained through trivial SQL queries, and the retrieved result set can be used directly to feed the decision tree algorithm. Note that, as in the case of the specification and computation of the KCIs, the task of building the training tuples is greatly facilitated by the abstraction provided by the process instance tables.

5.2 Understanding Key Factors

The algorithm we use in our prototype implementation for building decision trees extends the C4.5 algorithm to handle uncertain data [21]. In this paper we do not discuss the uncertainty aspect in mining data. However, our prototypes are equipped to handle uncertainty in the event logs we use for analyzing business process executions (for details on how uncertainty in event logs can be handled, see [6]). Instead, here we focus more on the aspect of discovering and understanding the key factors that affect the compliance of business executions.

As in any decision tree, the internal nodes contain the criteria used for classifying tuples. The leaf nodes, instead, contain the classes to which tuples are classified. For instance, if we choose the Compliant column of Table 1 as the class attribute, we will obtain a decision tree where the leaf nodes contain the compliance outcomes for the paths drawn from the root of the tree. However, nothing prevents us from choosing any other parameter of the process instance table as the class attribute when searching for the root-causes of non-compliant process executions.

For instance, as part of the validation of this approach, we performed experiments on a dataset of more than 30000 drug dispensations performed between January and April of 2009 in the hospital described in the scenario (Section 2). To this end, a process instance table with around 25 relevant parameters was build for the drug dispensation process, among which the parameters shown in Table 1 were included. Since the dependence of the Compliance column on the ErrPerData and ErrCompData columns was fairly obvious (but still, proven with our tools), we narrowed our analysis by considering only those process instances that were not compliant. After exploring some combinations of parameters, we found out that there was a relation between the ErrCompData and DrugType parameters. More precisely, we found that 393 drugs dispensations out of around 30000 had some error, among which 173 had errors of the type ErrCompData and 220 errors of the type ErrPerData. While the decision tree was not able to tell us anything that was really significant about errors of the type ErrPerData, it was able to find something useful for the errors of the type ErrCompData, as shown in Figure 4. More precisely, the decision tree discovered that 137 out of 173 (79%) erroneous process instances corresponded to drugs of the type 2 (DrugType=2), which are drugs for ambulatory usage, while the rest (21%) corresponded to drugs of the type 6, 9 and 11.

Since the ErrCompData refers to error in the dispensation data (such as the drug code, quantity and unitary price), this may be an indication that, for example, this type of drugs is dispensed at ease, and thus, a better monitoring or compliance enforcement need to be carried out on the controls related to this compliance requirement.

5.3 Predicting Compliance States

While decision trees are generally perceived as simple classifiers, we however use them rather for discovering and understanding better the root-causes of undesirable behaviors. Furthermore, we advocate the use of decision trees also for predicting the potential outcomes of process instances that are still running. In fact, each decision point in a tree corresponds to an event (or better to an attribute of an event). So, if during process execution an event that corresponds to a decision point is generated, this allows performing predictions on the likely outcome (in terms of compliance) of the process instance: it suffices to inspect the path in the tree determined by the registered event to identify the instances' likely compliance label.

Thus, in the case of predictions of non-compliant behaviors, enforcement actions can be enacted in order to align process executions, whenever possible, to the corresponding compliance requirements. This is particularly useful in cases when the process has several tasks and long running times that span, e.g., over several hours. Also, the prediction is particularly useful in the case compliance is enforced manually, because it allows the compliance expert to better focus his effort on those process instances that are likely to be non-compliant, leaving out compliance ones.

6 Related Work

The major part of compliance management approaches focuses on the business process modeling aspect at design time [7-9]. Typically, they are based on formal languages to express compliance requirements (e.g., Business Property Specification Language, Linear Temporal Logic) and simulations to prevent errors at runtime (e.g., finite state machine, Petri nets). In this context, just few approaches address compliance monitoring at runtime. For instance, Trinh et. al. [10] monitor time constraints during the execution of process activities, using UML Timing Diagrams to specify constraints and Aspect Oriented Programming to control executions. Chung et. al. [11] check if the user-defined process is compliant to pre-defined ontology and a specific model, in which compliance requirements are described. An IBM research group [12] advocates the use of the REALM (Regulations Expressed As Logical Models) metamodel to define temporal compliance rules and the Active Correlation Technology to check them. That way, it can detect duplicate events or compute a user-definable function, which checks whether a function exceeds some threshold.

Concurrently, commercial Business Activity Monitoring (BAM) solutions have been developed to support compliance management (e.g., IBM Tivoli, HP Business Availability Center, Nimbus, Oracle Business Activity Monitoring). Although, such tools still do not have the capability to process and interpret generic events (e.g., user-defined business or compliance-related events). They only support the definition of thresholds for parameters or SLAs to be monitored. Also, the ability to compare monitored business process executions or, more in general, business patterns with expected execution behaviors is not supported. Regarding reporting on compliance and KCIs, few works address this aspect and they do it partially. For example, [18] studies the representation of data through visual languages for risk and compliance management. In [19], the authors purpose a model-driven fashion approach to report on business performance and design dashboards.

To the best of our knowledge, no mining approaches have been specifically proposed to understand the root-cause of the compliance violations. However, few related approaches for the mining of business processes are in place [3-5][14-16]. Similar to our solution, they adopted log files and a consolidated warehouse containing business and process historical data, from where data subsets are extracted and used as input to mining algorithms in order to predict or understand the origin of undesired business process execution behaviors.

Finally, we can conclude that Compas and Master have been done significant contributions in all the fields mentioned in this section, since they provide solutions to manage, monitor and report on compliance based on generic events. For instance, [2][13] provide approaches to the management of the compliance monitoring at runtime, [17] states how to compute uncertain key indicators from uncertain data, [1] presents CGD to report on compliance and this paper presents root-cause analysis based on data mining techniques to understand non-compliant business processes.

7 Conclusions and Future Work

In this paper, we leverage on automated compliance checking techniques and complement them with a tactical perspective that targets compliance experts, which are accountable for assuring and improving compliance. We assist them by automating the analysis of the huge amount of data that is produced during process execution and specifically provide (i) a reporting dashboard with KCIs and KPIs to assess the state of compliance, (ii) a root-cause analysis technique to understand non-compliance. Our experiments with real data from a major Italian hospital show that the developed dashboard is effective in highlighting encountered problems and that the proposed abstractions and selection of data indeed allow us to identify also unexpected causes for non-compliant situations out of a large amounts of data.

It is important to note that, although in this paper we focused on the case of compliance, the ideas and solutions we propose are of general nature and can, for instance, easily be applied to the computation and analysis of KPIs. Similarly, we are not limited to process engine event as only source of information; events may also stem from web services, human task managers, or similar if suitably instrumented.

References

[1] Silveira, P., Rodríguez, C., Casati, F., Daniel, F., D'Andrea, V., Worledge, C., Taheri, C.: On the Design of Compliance Governance Dashboards for Effective Compliance and Audit Management. In: NFPSLAM-SOC'09, Springer, Stockholm, Sweden (2009)

[2] Daniel, F., D'Andrea, V., Strauch, S., Schumm, D., Leymann, F., Mulo, E., Zdun, U., Dustdar, S., Sebahi, S., de Marchi, F., Hacid, M.: Business Compliance Governance in Service-Oriented Architectures. In: AINA'09, IEEE Press (2009)

[3] Rozinat, A., van der Aalst, W.M.P.: Decision Mining in Business Processes.

BETA Working Paper Series, WP 164, Eindhoven University of Technology, Eindhoven (2006)

[4] Grigori, D., Casati, F., Castellanos, M., Dayal, U., Sayal, M., Shan, M. Business Process Intelligence. Computers in Industry Journal, vol. 53/3, pp. 321-343 (2004)

[5] Seol, H., Choi, J., Park, G., Park, Y.: A framework for benchmarking service process using data envelopment analysis and decision tree. ESA, vol. 32, pp. 432-440 (2007)

[6] Rodríguez, C., Daniel, F., Casati, F., Cappiello, C.: Toward Uncertain Business Intelligence: the Case of Key Indicators. IEEE Internet Computing, vol. 14, no. 4 (2010)

[7] Liu, Y., Mueller, S., Xu, K.: A Static Compliance Checking Framework for Business Process Models. IBM Systems Journal, vol. 46, no. 2, pp. 335-361 (2007)

[8] Lu, R., Sadiq, S., Governatori, G.: Compliance Aware Business Process Design. In: BPM'05, pp. 120-131 (2007)

[9] Awad, A., Weske, M.: Visualization of Compliance Violation in Business Process Models. In: BPI'09, LNCS, vol. 43, pp. 182-193 (2009)

[10] Trinh, T., Do, T., Truong, N., Nguyen, V.: Checking the Compliance of Timing Constraints in Software Applications. In: KSE'09, pp. 220-225 (2009)

[11] Chung, P., Cheung, L., Machin, C.: Compliance Flow - Managing the compliance of dynamic and complex processes. Knowledge-Based Systems, vol. 21 no. 4, pp. 332-354 (2008)

[12] Giblin, C., Mller, S., Pfitzmann, B.: From Regulatory Policies to Event Monitoring Rules: Towards Model-Driven Compliance Automation. IBM Research Report RZ 3662 (2006)

[13] Mulo, E., Zdun, U., Dustdar, S.: Monitoring Web Service Event Trails for Business Compliance. In: SOCA'09, IEEE Computer Society Press (2009)

[14] Grigori, D., Casati, F., Dayal, U., Shan, M.: Improving business process quality through exception understanding, prediction, and prevention. In: VLDB'01, pp. 159-168 (2001)

[15] Apte, C., Bibelnieks, E., Natarajan, R., Pednault, E., Tipu, F., Campbell, D., Nelson, B.: Segmentation-Based Modeling for Advanced Targeted Marketing. In: Knowledge Discovery in Databases and Data Mining. ACM, pp. 408-413 (2001)

[16] Bibelnieks, E., Campbell, D.: Mail Stream Streamlining, Catalog Age, vol. 17, no. 12, pp. 118–120 (2000)

[17] Rodríguez, C., Daniel, F., Casati, F., Cappiello, C.: Computing Uncertain Key Indicators From Uncertain Data. In: ICIQ'09 (2009)

[18] Bellamy, R., Erickson, T., Fuller, B., Kellogg, W., Rosenbaum, R., Thomas, J., Wolf. T.: Seeing is believing: Designing visualizations for managing risk and compliance. IBM Systems Journal, vol. 46, no. 2, pp. 205-218 (2007)

[19] Chowdhary, P., Palpanas, T., Pinel, F., Chen, S., Wu, F.: Model-driven Dashboards for Business Performance Reporting. In: EDOC'06, pp. 374-386 (2006)

[20] Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann (2006)

[21] Tsang, S., Kao, B., Yip, K., Ho, W., Lee, S.: Decision Trees for Uncertain Data. In: ICDE'09. IEEE pp. 441-444 (2009)

Appendix E

Aiding Compliance Governance in Service-Based Business Processes

Aiding Compliance Governance in Service-Based Business Processes

Patricia Silveira Carlos Rodríguez Aliaksandr Birukou
Fabio Casati Florian Daniel Vincenzo D'Andrea Claire Worledge
Zouhair Taheri

Abstract

Assessing whether a companys business practices conform to laws and regulations and follow standards and SLAs, i.e., compliance management, is a complex and costly task. Few software tools aiding compliance management exist; yet, they typically do not address the needs of who is actually in charge of assessing and understanding compliance. We advocate the use of a compliance governance dashboard and suitable root cause analysis techniques that are specifically tailored to the needs of compliance experts and auditors. The design and implementation of these instruments are challenging for at least three reasons: (1) it is fundamental to identify the right level of abstraction for the information to be shown; (2) it is not trivial to visualize different analysis perspectives; and (3) it is difficult to manage and analyze the large amount of involved concepts, instruments, and data. This chapter shows how to address these issues, which concepts and models underlie the problem, and, eventually, how IT can effectively support compliance analysis in Service-Oriented Architectures (SOAs).

Note: We include only the abstract of this publication. The full publication is available at:

http://www.igi-global.com/chapter/aiding-compliance-governance-service-based/60900

Appendix F

Eventifier: Extracting Process Execution Logs from Operational Databases

Eventifier: Extracting Process Execution Logs from Operational Databases *

Carlos Rodríguez Robert Engel Galena Kostoska Florian Daniel
Fabio Casati Marco Aimar

Abstract
This demo introduces *Eventifier*, a tool that helps in reconstructing an event log from operational databases upon which process instances have been executed. The purpose of reconstructing such event log is that of discovering process models out of it, and, hence, the tool targets researches and practitioners interested in process mining. The aim of this demo is to convey to the participants both the conceptual and practical implications of identifying and extracting process execution events from such databases for reconstructing ready-to-use event logs for process discovery.

1 Introduction

Process discovery is the task of deriving a process model from process execution data that are typically stored in event logs, which in turn are generated by information systems that support the process execution [5]. Most of the approaches available in the state of the art assume the existence of an event log, where each event is assumed to have information, such as a process name, activity name, execution timestamp, event type (e.g., start or end), and process instance ID. In practice, most companies do not really have such an event log, either because they do not have a business process engine that is able to generate such logs or, if they do, the engine supports only parts of the process, e.g., because parts of the process are supported by legacy systems. In the second case, it may also happen that the engine does not generate an event log that can be used for process discovery, e.g., if the log contains only events regarding errors in the system.

*Copyright is held by the authors. The final publication is available at http://ceur-ws.org/Vol-940/

The information stored in an event log commonly provides a very narrow and focused view on the overall data produced by a process during its execution (e.g., focusing on errors for recovery or control flow decisions and actors for auditing). Typically, however, an information system also stores the full data produced by a process inside its ***operational databases*** (OD) (also known as *production databases*), where these data comprise process progression data, process state data, business data produced throughout the process, data related to the regular operations of an organization, as well as their related business facts and objects [2]. ODs therefore store more and richer data than event logs, but blur different aspects of data and neglect the event-based nature of process executions. For this reason, process discovery starts from event logs.

With this demo, we approach the problem of producing process execution events in a fundamentally different context, i.e., in a context where we do *not* have access to the information system running the process (hence we cannot instrument it) and where the only way of obtaining process execution events is deriving them from the OD of the information system *after* the actual process execution. We call this activity ***eventification*** of the OD and we perform it with the help of our tool ***Eventifier***. For the rest of the paper, we assume that the OD is a ***relational database*** [4].

Significance to the BPM field. Much attention has been paid so far to the problems of representing event logs [6], event correlation [3] and process discovery [5], while the problem of how to produce good events has been neglected by research. As explained above, ***Eventifier*** approaches an important issue in the field of process mining by providing an application that will help both researches and practitioners working in the field.

2 Eventification of the Operational Database

Let's start by giving some preliminary definitions. An ***event log*** can be seen as a sequence of events $E = [e_1, e_2, ..., e_m]$, where $e_i = \langle id, tname, pname, piid, ts, pl\rangle$ is an event of a process instance, with id being the identifier of the event, $tname$ being the name of the task the event is associated with, $pname$

being the name of the process type, *piid* being the process instance identifier, *ts* being the timestamp of the event, and *pl* being the payload of the event. Thus, an event log stores traces of process executions as atomic events that represent process progression information and that may carry business data in their payload.

Reconstructing an event log E with events e_i means deciding when to infer the existence of an event from the data in the OD and filling each of the attributes of the event structure with meaningful values. These values either stem from the data in the OD or they may be provided by a domain expert. Specifically, for the *id* attribute, assigning an identifier to an event means recognizing the *existence* of the event. Given that we do not have real events in the OD but other, indirect evidence of their occurrence, there is no "correct" or "original" event identifier to be discovered. The question here is what we consider *evidence* of an event. Similarly, in the case of *tname*, without the concept of task in the applications of the information system, there is no explicit *task naming* that can be discovered from the data. Thus, we need to find a way to label the boxes that will represent tasks in the discovered model. The value for the attribute *pname* (the *process name*) we can only get from the domain expert, who knows which process she is trying to discover. Then, the process instance identifier (*piid*) is needed to group events into process instances. The *piid* is derived by means of event correlation based on the values of the attributes of the identified events. The attribute *ts* is needed to *order* events chronologically, which is a requirement for process discovery. Therefore, we need to find evidences in the OD that help us in determining the ordering of events. Finally, the goal of choosing a payload *pl* for the purpose of eventification is not to reconstruct the complete *business data* that can be associated with a given task or event, but rather that of supporting the correlation of events into process instances. We can get this data from the rows that originate the events.

We call the assignment of values to *id*, *pname* and *tname* the *identification* of an event, to *ts* the *ordering* of events, to *pl data association*, and to *piid correlation*. These four activities together constitute the ***eventification*** process, and it is helped by heuristics in the form of ***eventification patterns***:

Event identification patterns. These patterns help in the identification of events from the OD. In these patterns, we assume that the existence of a row in a relation R indicates the presence of an event. We express these patterns as a function:

$$identify(R, pname, tname) \rightarrow e^0 = \langle id, pname, -, tname, -, t \rangle$$

where $pname$ and $tname$ are defined by the domain expert, and t is the tuple in R that originated e^0. In concrete, we rely on the following three patterns for the identification of events:

- *Single row, single event pattern* (Figure 1(a)). In this pattern, each row in a relation R indicates the existence of an event. R can be obtained with a simple SQL query as:

  SELECT * FROM $r_1, r_2, ..., r_n$

  WHERE [JOIN conditions for $r_1, r_2, ..., r_n$];

- *Single row, multiple event pattern* (Figure 1(b)). A tuple in R can evidence the existence of more than one event, such as when different values of the attributes A_i of R indicate different potential events. In this case, the relation R is built by applying filtering conditions in the WHERE clause so as to keep only the target events:

  SELECT * FROM $r_1, r_2, ..., r_n$

  WHERE [JOIN conditions for $r_1, r_2, ..., r_n$]

  AND [filtering conditions for the target event, e.g., $r_2.dispatched = yes$];

- *Multiple row, single event pattern* (Figure 1(c)). Multiple rows in a relation R indicate the presence of a single event. This last pattern is useful, for instance, when we deal with a *denormalized* relation that mixes data at different granularities, e.g., when in a single tuple we find both the header of an invoice and the item sold. The SQL for R has the following form,

  SELECT DISTINCT $A_1, A_2, ..., A_k$ FROM $r_1, r_2, ..., r_n$

  WHERE [JOIN conditions for $r_1, r_2, ..., r_n$] ;

where the attributes A_i should be the higher granularity attributes that would be typically used in a GROUP BY, SQL statement.

Event ordering pattern. The event ordering pattern aims at deriving the ordering of events from time-related information associated to the records stored in the OD, and is represented as:

$order(e^0) \rightarrow e^1 = \langle id, pname, -, tname, ts, t\rangle$

where e^1 is the result of attaching a timestamp value to ts, and ts is the projection of all timestamp or date attributes of $e^0.t$ generated by the previous pattern. If only one timestamp can be found, it is used straightaway. If there are more possible timestamps in pl, the domain expert chooses the one that best represents the execution time of the task.

Data association pattern. The data association pattern aims to select which data to assign to pl. In the above patterns, we have so far simply carried over the complete row t as payload of the event, while here we aim to select which attributes out of the ones in t are really relevant. Our assumption is that all necessary data is already present inside t, that is, we do not need to consult any additional tables of the OD to fill pl with meaningful data. Thus, in the event identification step, the necessary tables are joined, and t contains all potentially relevant data items. The data association pattern is represented as:

$getdata(e^1) \rightarrow e^2 = \langle id, pname, -, tname, ts, pl\rangle$

where e^1 is as defined before, and pl is the new payload computed by projecting attributes from t. In absence of any knowledge about the OD by the domain expert, the heuristic we apply is to copy into pl all attributes of t, except times-

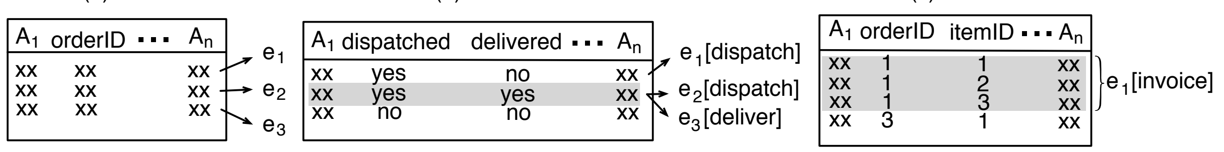


Figure 1: Types of event identification patterns: (a) single row, single event, (b) single row, multiple events, and (c) multiple row, single event pattern

tamps and auto-increment attributes, which by design cannot be used for correlation. The domain expert can of course also choose manually which attributes to include and which to exclude.

Event correlation patterns. Eventually, we are ready to correlate events and to compute the *piid* of the identified events. The goal of event correlation is to group events into process instances, which are the basis for process discovery. As explained above, we assume that after associating the final payloads to events all information we need to correlate events is present in the payload *pl* of the events in the form of attribute-value pairs. In practice, correlating events into traces means discovering the mathematical function over the attributes of *pl* that tells if an event belongs to a given process instance, identified by the output *piid* of the function. We represent this step as follows:

$$correlate(e^2) \rightarrow e = \langle id, pname, piid, tname, ts, pl \rangle$$

where e^2 is as defined above and e is the final version of the discovered event from the OD with the attribute *piid* filled with a suitable identifier of the process instance the event belongs to.

3 The Eventifier Environment

Figure 2 provides an architectural view on the resulting approach to eventification, which is a semi-automated process that requires the collaboration of a domain expert having some basic knowledge of the OD to be eventified. First, the domain expert identifies events in the OD, orders them, and associates data with them. All these activities are supported the the so-called *Event Extractor*, which supports the domain expert in an interactive and iterative fashion. The result of this first step is a set of events, which are however not yet correlated. Correlation is assisted via a dedicated *Event Correlator*, which again helps the domain expert to interactively identify the best attributes and conditions to reconstruct process traces. The result of the whole process is an event log that is ready for process discovery.

The Eventifier is implemented as an integrated platform that includes the

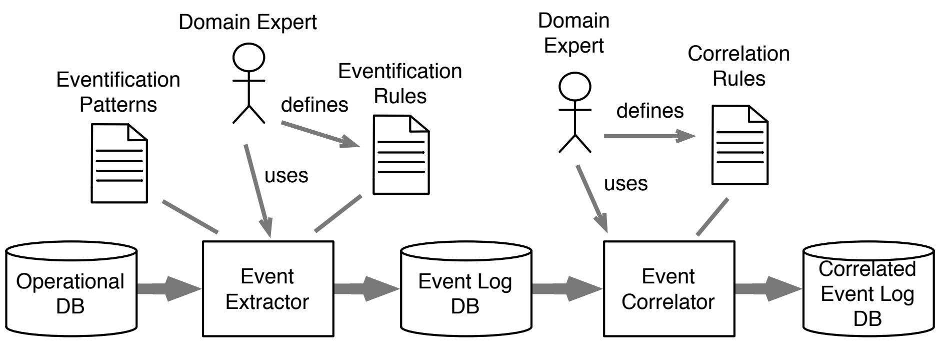


Figure 2: Overview of the database eventification prototype and approach.

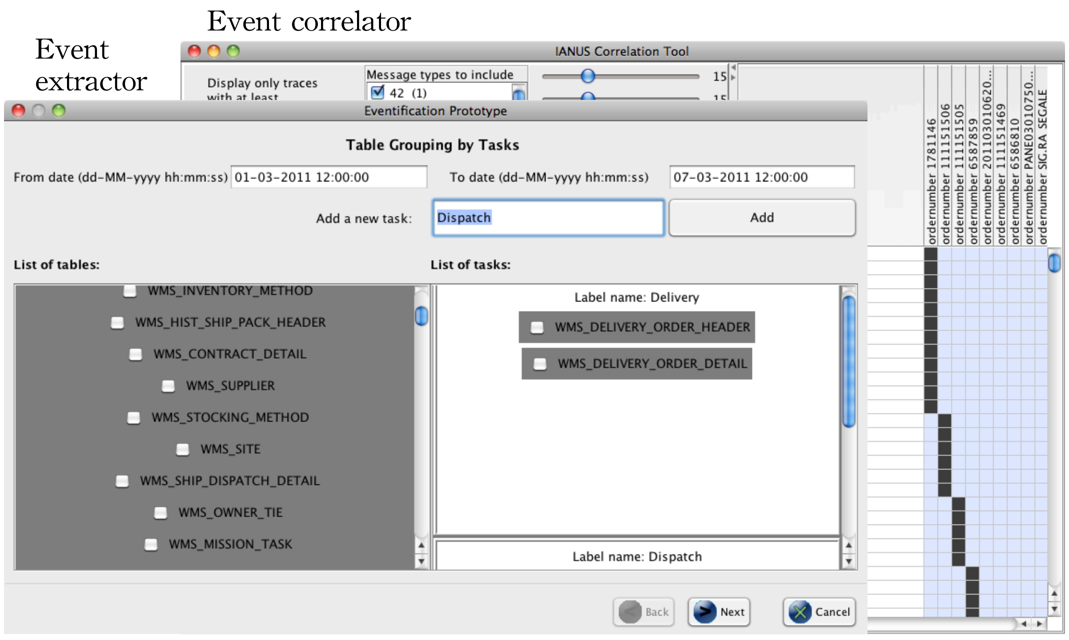


Table Grouping by Tasks

From date (dd-MM-yyyy hh:mm:ss) 01-03-2011 12:00:00

To date (dd-MM-yyyy hh:mm:ss) 07-03-2011 12:00:00

Add a new task: Dispatch

List of tables: WMS_INVENTORY_METHOD, WMS_HIST_SHIP_PACK_HEADER, WMS_CONTRACT_DETAIL, WMS_SUPPLIER, WMS_STOCKING_METHOD, WMS_SITE, WMS_SHIP_DISPATCH_DETAIL, WMS_OWNER_TIE, WMS_MISSION_TASK

List of tasks:

Label name: Delivery — WMS_DELIVERY_ORDER_HEADER, WMS_DELIVERY_ORDER_DETAIL

Label name: Dispatch

Display only traces with at least; Message types to include: 42 (1)

ordernumber 1781146, ordernumber 111151506, ordernumber 111151505, ordernumber 6587859, ordernumber 201103010620…, ordernumber 111151469, ordernumber 6586810, ordernumber PANE03010750…, ordernumber SIG.RA_SECALE

Figure 3: Screenshots of the components of our integrated platform for eventification.

components for eventification, correlation and process discovery. These components allow domain experts to interactively apply patterns and to navigate end-to-end from the OD to the discovered process model and back. Since our aim is not to make contributions on process discovery, we use existing process discovery algorithms implemented as plugins for the popular process mining suite ProM [6]. All components are implemented as Java desktop applications using standard libraries such as Swing. The implementation of the Event Correlator is partly based upon a software tool originally developed for the correlation of EDI messages [1]. For the creation of XES-conformant event logs [6] that are used in the interface to process discovery in ProM, we employ the OpenXES libraries (`http://www.xes-standard.org/openxes/start`). Figure 3 shows the screenshots of the Event Extractor and Correlator components.

4 Demo scenario

A demo video of our eventification tool in action can be found at the website `http://sites.google.com/site/dbeventification`. The demo is in the form of a screencast and illustrates the main features of our tool using as scenario the case of an Italian logistics company for refrigerated goods. In this video we clearly show the two main tasks of our approach as outlined in Figure 2 and we also show the final outcome in terms of the process model discovered from the reconstructed event log.

Acknowledgements. This work was supported by the Ianus project funded by the Province of Trento (Italy) and Opera21 Group and by the Vienna Science and Technology Fund (WWTF) through project ICT10-010.

References

[1] R. Engel, W. van der Aalst, M. Zapletal, C. Pichler, and H. Werthner. Mining Inter-organizational Business Process Models from EDI Messages: A Case Study from the Automotive Sector. In *24th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2012)*, LNCS 7328, pp.222-237. Springer, 2012.

[2] R. Kimball and M. Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling.* Wiley, 2002.

[3] H. Montahari-Nezhad, R. Saint-Paul, F. Casati, and B. Benatallah. Event Correlation for Process Discovery from Web Service Interaction Logs. *VLDB Journal*, 20(3):417–444, 2011.

[4] R. Ramakrishnan and J. Gehrke. *Database Management Systems.* McGraw-Hill, 2007.

[5] W. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes.* Springer, 2011.

[6] H. Verbeek, J. Buijs, B. van Dongen, and W. van der Aalst. XES, XESame, and ProM 6. In *Information Systems Evolution*, volume 72, pages 60–75. 2011.

Appendix G

SOA-Enabled Compliance Management: Instrumenting, Assessing and Analyzing Service-Based Business Processes

SOA-Enabled Compliance Management: Instrumenting, Assessing and Analyzing Service-Based Business Processes *

Carlos Rodríguez Daniel Schleicher Florian Daniel Fabio Casati
Frank Leymann Sebastian Wagner

Abstract

Facilitating *compliance* management, i.e., assisting a company's management in conforming to laws, regulations, standards, contracts and policies, is a hot but non-trivial task. The service-oriented architecture (SOA) has evolved traditional, manual business practices into modern, service-based IT practices that ease part of the problem: the systematic definition and execution of business processes. This, in turn, facilitates the online monitoring of system behaviors and the enforcement of allowed behaviors – all ingredients that can be used to assist compliance management on the fly during process execution. In this paper, instead of focusing on monitoring and runtime enforcement of rules or constraints, we strive for an alternative approach to compliance management in SOAs that aims at *assessing and improving* compliance. We propose two ingredients: (i) a *model and tool* to design compliant service-based processes and to instrument them in order to generate evidence of how they are executed and (ii) a *reporting and analysis suite* to create awareness of a company's compliance state and to enable understanding *why* and *where* compliance violations have occurred. Together, these ingredients result in an approach that is close to how the real stakeholders – compliance experts and auditors – actually assess the state of compliance in practice and that is less intrusive than enforcing compliance.

1 Introduction

Compliance management [35] is an important, costly, and complex problem: It is *important* because there is increasing regulatory pressure on companies to meet a variety of requirements in terms of regulations, laws and similar (e.g.,

*The final publication is available at www.springerlink.com (http://link.springer.com/article/10.1007%2Fs11761-013-0129-3)

Basel II, MiFID, SOX). This increase has been to a large extent fueled by high-profile bankruptcy cases (e.g., Parmalat, Enron, WorldCom), safety mishaps (the April 2009 earthquake in L'Aquila, Italy, has already led to stricter rules and certification procedures for buildings and construction companies), or the recent financial crisis. Failing to meet these requirements may imply safety risks, hefty penalties, loss of reputation, or even bankruptcy or jail [36].

Managing and auditing/certifying compliance is a very *expensive* endeavor. In their 2008-2009 Governance, Risk Management, and Compliance Spending Report [13], AMR Research estimated that companies would spend US$ 32B only on governance, compliance, and risk in 2008 and more than US$ 33B in 2009. In addition, audits are themselves expensive and invasive activities, costly not only in terms of auditors' salaries but also in terms of internal costs for preparing for and assisting the audit.

Finally, the problem is *complex* because compliance requirements are often pervasive in that they span across many segments of a company, and many processes. They are also sometimes only vague and informally specified. Yet, compliance management requires understanding and interpreting requirements and then implementing and managing a typically large number of controls on a variety of procedures across the business units of a company. Each compliance requirement and procedure may demand for its own control mechanism and its own set of assessment metrics to adequately capture the state of compliance.

Today, compliance is to a large extent managed by the various business units in rather ad-hoc ways and with little or no IT support. As a result, today it is very hard for any CFO or CIO to answer questions such as: Which requirements does my company have to comply with? Which processes should obey which requirements? Which processes are following a given regulation? Where do violations occur? Which processes do we have under control? Even more, it is hard to do so from a perspective that not only satisfies the company but also the company's auditors, which is crucial as the auditors are the ones that certify the company's capabilities to control compliance.

Yet, business processes are indeed supported by IT. Technologies like web services and business process management systems have demonstrated, although more slowly than initially thought, their viability for organizing work and assisting and orchestrating also human actors involved in business processes. Interestingly, however, the automated operation of business processes has not yet lead to a significant facilitation of compliance management practices.

1.1 Reference Scenario: Outpatient Drug Dispensation in a Hospital

Let us consider, for example, the management and assessment of the outpatient *drug dispensation* process summarized in Figure 1. The process – and this paper – originates in the EU project MASTER (http://www.master-fp7.eu) where we cooperate on compliance management with Hospital San Raffaele (Milano, Italy), which runs the described distributed business process. The process is part of a bigger procedure known as the *outpatient drug reimbursement*, which implements the steps required for refunding hospitals for the drugs dispensed to patients that are not hospitalized. The overall process is regulated by the Italian Healthcare Authority, which dictates regulations on the dispensation and reporting requirements for the reimbursement of drug expenses, such as the ones concerning privacy in personal information processing.

The core process, shown in Figure 1, starts with the patient's visit to the doctor in the hospital's ward. Depending on the diagnosis, the doctor sends a prescription for drugs to the nurse, who dispenses the necessary drugs to the patient if the requested quantity is available. If the available drug quantity is

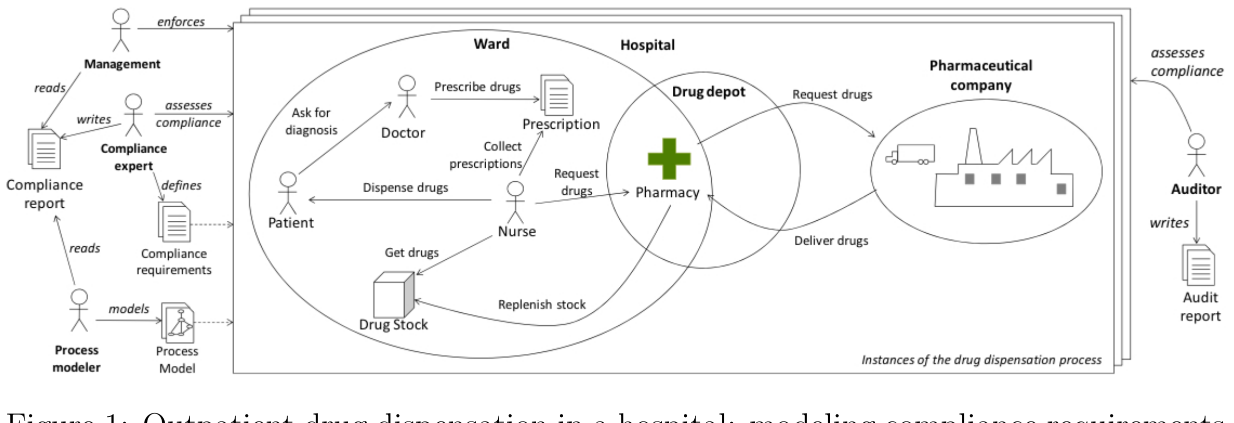


Figure 1: Outpatient drug dispensation in a hospital: modeling compliance requirements and assessing compliance

insufficient, she requests the drug to the hospital-internal pharmacy, which is then in charge of replenishing the nurse's drug store. If, in turn, the pharmacy is running out of stock, it orders the necessary drugs from the pharmaceutical company.

The drug dispensation process is supported by several web service-based information systems that interact inside a SOA that is distributed over the hospital, the pharmacy, and the pharmaceutical company. For instance, there are web services for issuing drug requests in the various dependencies of the institute, and the pharmaceutical company the hospital cooperates with accepts drug requests through web service interfaces. To retrieve the data necessary to assess the hospital's state of compliance, during the execution of the process suitable data (e.g., events) that can be logged and later analyzed are produced by all cooperating parties.

By law, the hospital must guarantee that all *patient data are anonymized* throughout the process (and in the generated events), and the hospital's internal policy states that *drug replenishment must occur within maximum two business days* and that the person who prescribes a drug cannot also dispense the drug (*separation of duties*). While this latter requirement is monitored internally by the hospital's own compliance expert, the former requirement is subject to yearly audit by an official security *auditor*, who certifies (or not) the hospital's compliance with the laws the hospital is subject to. Passing this audit is crucial for the hospital's business continuity.

The compliance requirements that apply to the hospital are identified and specified by the *compliance expert*, who knows about the applicable laws and regulations and about the internal policy. Typically, the compliance expert assists the *process modeler* in designing compliant processes, in order to prevent non-compliance by design. Yet, he also checks the execution of the designed processes, as at runtime non-compliant situations may occur despite a well-designed process model (e.g., due to system failures or manual intervention on a running process instance). Periodically, he then writes an internal compliance report, which is the basis (i) for the *management* to take decisions and enforce compliance and (ii) for the *process modeler* to understand violations and improve process models and controls.

Today, all these activities are typically performed *manually*, and compliance assessment is of *statistical* nature. That is, controls are added to processes in an ad-hoc and per-process fashion; process instances are checked by inspecting only a subset of physical documents or log files and estimating compliance levels; the compliance report is written by hand; and analyzing the root causes of violations is hard and time-consuming, given the large amount of data to be correlated. In addition to that, although the overall process is automatically orchestrated and activities have suitable IT support, in practice many tasks are still based on paper forms filled by doctors or nurses during their service and manually input only in a later stage. [1]

1.2 Contributions and Structure of the Paper

This paper describes an infrastructure and methodology that supports compliance management. Specifically, we provide the following contributions:

- We provide ***a model and a graphical modeling tool*** that eases building processes that are compliant with process-specific compliance requirements. The approach allows one to equip a common business process definition (e.g., BPEL process specification) with a definition of technical *compliance rules* and to *instrument* it in order to generate the necessary evidence for compliance assessment.
- In order to facilitate compliance assessment, we ***extend a state-of-the-art service orchestration engine*** with signaling capabilities that are able to generate compliance-related evidence on process executions.
- We provide a ***suite of reporting and analysis tools*** that facilitates the writing of the compliance report and helps the compliance expert and the process modeler to identify where and why compliance violations happened. The suite is based on a *compliance-oriented warehouse*, *key compliance indicators*, and *root cause analysis algorithms*.

[1]It is important to note here that we assume all the artifacts needed for compliance management are represented inside the information system. Also, we do not deal with the problem of fidelity regarding the computer representations of real world artifacts; this is a general modeling that requires adequate domain and modeling knowledge.

- We implement all reporting and analysis algorithms on top of a ***data model that supports compliance assessment***, which allows us to better reflect the *nature* of the data that is available for analysis and to enable *better business decisions*.

In summary, the main goal of this work is to enable humans to be aware of how compliant business processes are and to understand why problems happen, in order to improve compliance. We propose a methodology and tool for the definition and assessment of compliance rules. While compliance rules are typically domain-specific, our solution is generic and aims to support different regulations at a technical level, limited to those business processes of a company that are supported by web services and that are executed with the help of a business process engine.

The methodologies, prototypes and demos described in this work have been designed and evaluated with the help from audit experts of Deloitte, Paris, who deal with compliance in a variety of domains at a daily basis.

In the next section we introduce our approach to compliance management and show how the above contributions fit into an overall methodology. Then, in Sections 3-6 we describe the individual phases of the methodology, i.e., (i) design of processes and evidence, (ii) execution of processes and generation of evidence, (iii) assessment of compliance, and (iv) analysis of problems. In Section 7 we survey the most related works, and in Section 8 we recap the contributions of the paper.

2 Compliance Management in the SOA

2.1 Compliance Management Requirements

Compliance management should enable the company's *management* to know about the state of compliance, assess the risks associated with non-compliant situations, and take business decisions to correct them. Ideally, these decisions are based on up-to-date compliance reports, featuring a set of compliance-specific indicators that are easy to interpret and, hence, effective in communicating key information. The *compliance expert*, instead, is interested in knowing the individual compliance violations and understanding their causes, while the *process modeler* is rather interested in how to improve process models as well as control points for future executions.

Typically, this means that we need a dashboard for reporting on compliance that allows navigation across the company's processes and across compliance concerns and associated key indicators at different levels of aggregation and details. It also means that we need to provide a way to model concerns and indicators, and to collect evidence for their computation. In terms of modeling, we need a ***formalism and tool to express compliant behaviors***, e.g., in the form of process templates that specify compliance requirements and constrain the instantiation of the template. Once we have a definition of a compliant process trace, we can then verify if the actual execution is compliant. We also need a way to define and compute indicators which can typically be based on aggregating information over many process instances (e.g., the total amount of invoices that were handled in a non-compliant manner).

In this paper we consider as evidence of compliance and as source for the computation of reports the information and events related to the execution of processes. Some of these data and events (e.g., the start of an activity) are commonly produced by business process engines during runtime, but compliance assessment may ask for some specific execution evidence (e.g., a login event with actor information, or information about an invoice). Collecting proper evidence, typically within a data warehouse, requires the ***instrumentation*** of a process or service orchestration engine as well as a way to specify which events should be signaled by the process.

While processes, related evidence, compliance requirements, and indicators differ on a case by case basis, the challenge here is to adopt the same formalisms and computation model, otherwise the approach is not reusable and we would have to develop models and code for each new compliance requirement or new process.

In the case compliance violations have happened, it is of utmost importance to be able to ***understand why and where*** these violations occurred. Violations may stem from problems during process execution (*instance-level* violations) or from badly designed processes (*model-level* violations). To understand instance-level violations, we propose the use of classification by means of decision trees, which allows the correlation of a process instance's compliance state with its business data. In order to understand model-level violations, we propose the use of *protocol discovery*, which allows the comparison of a system's real behavior with its designed behavior. Finally, both instance-level and model-level violations manifest themselves also in compliance indicators. Correlating their values and dynamics over time may thus provide further indications on where violations occurred in a process model and which violations impact on which other violations.

2.2 System Architecture

Figure 2 illustrates how we approach the above requirements from a system architecture perspective. The architecture has been designed leveraging on events as concrete evidence of the runtime behavior of the system, where the necessary events can be either derived for free from service communications in the service-based environment or they can be obtained by instrumenting the system purposefully. Starting from business and compliance requirements, the compliance expert defines *compliance templates* (see Section 3.1) and *Key Compliance Indicators* (KCIs), i.e., indicators measuring the compliance of process instances (Section 3.2), with the help of a dedicated *compliance template editor*. A compliance template describes the compliant behavior of a business process, while the KCIs are key indicators that measure how compliant a company is with respect to its compliance requirements. Based on the compliance template and the specified KCIs, a so-called *signaling policy* is created, which states which execution events are needed to assess compliance.

The process modeler instantiates the templates, designs the *process models*, and generates executable process specifications (in our case, we generate BPEL), which can be inspected and fine-tuned with the help of a common *business process editor*. The engine takes process models as input and instantiates and runs them, establishing this way a communication between *web services*, *human users* (via dedicated user interfaces that allow them to in-

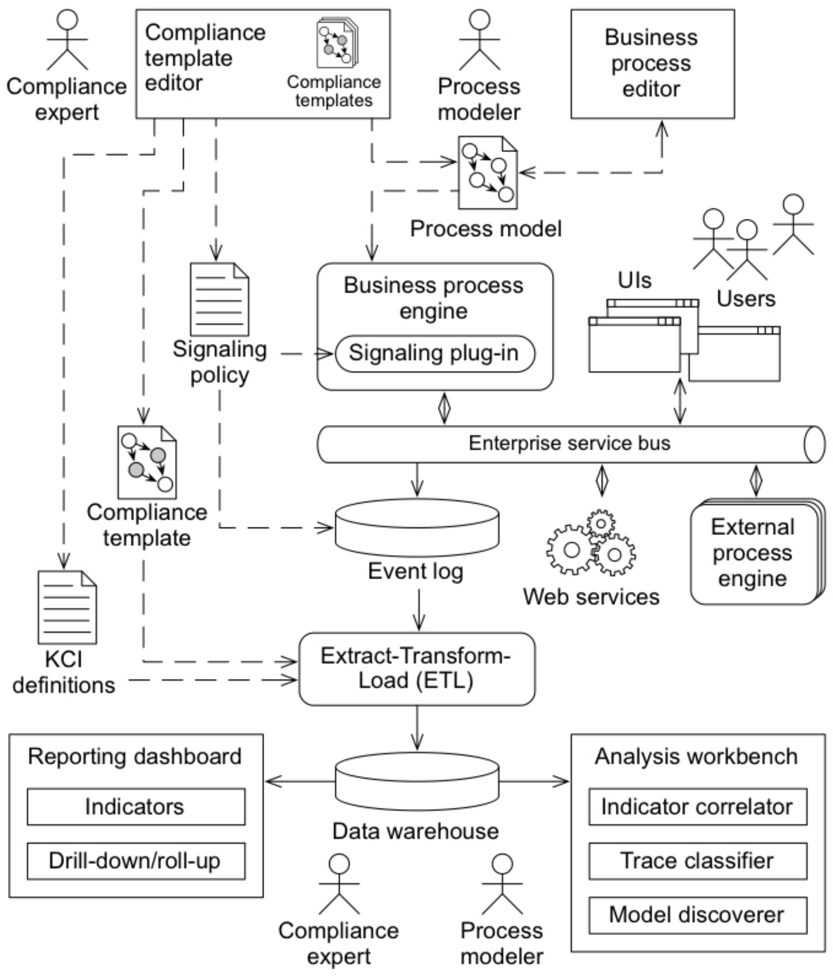


Figure 2: Event-based compliance management architecture

teract with the process), and possible *external business process engines* (in the case of distributed business processes). The *signaling policy* configures the *signaling plug-in* of the purposely extended *business process engine*. Communications and events are sent over a shared *enterprise service bus* (ESB), which allows the easy tracking of events in an *event log*. Out of all the messages that flow through the ESB, the event log only subscribes to the events defined in the signaling policy. Periodically (e.g., during the night), an ETL (*Extract-Transform-Load*) procedure loads tracked events into the *data warehouse* and computes compliance and KCIs. The data in the warehouse can then be inspected by the compliance expert in a *reporting dashboard* that visualizes indicators and supports the necessary drill-down (navigation to finer-grained details) and roll-up (aggregation) features. An *analysis workbench* provides for the analysis of compliance violations.

2.3 Compliance Management Methodology

Compliance management is *not* a simple issue, a property that manifests itself also in the complexity of the proposed system architecture (see Figure 2). Compliance management typically requires understanding multiple sources for regulatory compliance requirements (e.g., laws, standards or similar) and to translate the requirements that affect a given process into technical rules. We aim at supporting this latter aspect, which translates into the architecture and instruments in Figure 2. For a better understanding of how the joint use of these instruments can aid compliance management, we contextualize them in our ***assisted compliance management methodology***, which is based on the *Deming cycle* [38], known from business process improvement. The methodology consists of four phases, which we illustrate in Figure 3.

In the ***Plan*** phase, first we model a compliance template, which can then be instantiated into a process model. Given a process model, it is possible to specify which KCIs to compute for the process. Given the compliance template and the KCI definitions, the necessary signaling policy can be generated automatically. In the ***Do*** phase, processes and the signaling policy are executed, that is, processes are instantiated and run by the process engine, and specified events are generated and logged for later inspection. In the ***Check*** phase, the system periodically loads logged events into the data warehouse and labels event traces, i.e., process instances, as compliant or not. The so prepared data is used to compute indicators and to prepare the reports, which can then be inspected in order to understand the compliance state of the company. Depending on the encountered compliance violations, the management may enforce compliance (this step is not assisted by our system). Finally, in the ***Act*** phase, the compliance expert and process modeler try to understand the root causes of violations, so as to improve processes and policies by refining the respective models and specifications and, hence, restarting from the Plan phase.

In this paper, we do ***not*** propose the use of automatic techniques for the

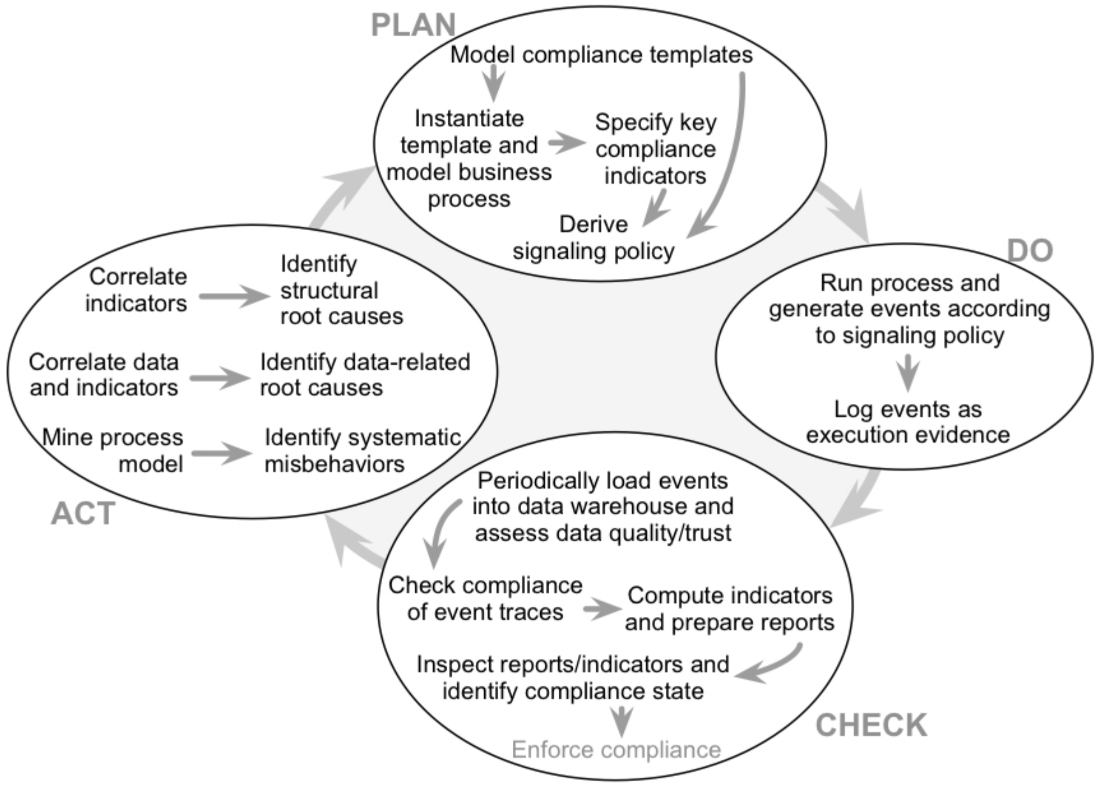

PLAN
Model compliance templates
Instantiate template and model business process
Specify key compliance indicators
Derive signaling policy
DO
Run process and generate events according to signaling policy
Log events as execution evidence
CHECK
Periodically load events into data warehouse and assess data quality/trust
Check compliance of event traces
Compute indicators and prepare reports
Inspect reports/indicators and identify compliance state
Enforce compliance
ACT
Correlate indicators
Identify structural root causes
Correlate data and indicators
Identify data-related root causes
Mine process model
Identify systematic misbehaviors

Figure 3: Assisted compliance management methodology

runtime enforcement of compliant behaviors in business processes. While such techniques undoubtedly allow a company to better control compliance requirements at a technical and operative level (e.g., at the level of individual events), compliance management is however still an organizational and tactical activity that most of the times requires human intervention and interpretation. The main goal of this work is therefore enabling *humans* to be aware of *how compliant* business processes are and to understand *why* problems happen, in order to incrementally improve compliance.

3 Plan – Designing Compliant Processes and Defining Evidence

For the purpose of designing compliant business processes, we complement traditional process modeling with three ingredients: (i) *compliance templates*, which define the compliance requirements of a process; (ii) a *signaling policy*, which specifies which events need to be generated, and (iii) a set of *KCIs*, which summarize events for reporting purposes.

3.1 Specifying Compliant Behaviors

To describe the compliant behavior a process should follow, we propose to use compliance templates. By using compliance templates we can for example define the acceptable order in which activities should be performed, which activities are allowed, and which constraints exist among them [31]. This approach has multiple benefits. First, compliance requirements that apply to a class of process models can be defined by individuals that are knowledgeable in their respective compliance domain, i.e., the compliance experts (typically members of the management or lawyers). Compliance experts are responsible for the compliance templates; they are the only ones that are allowed to authorise changes on them. Compliance experts are supported by business process experts when a compliance template must be changed, for example. Second, because templates, as the word denotes, are used as a starting point for defining the process itself by expanding and detailing them, following regulations is made easier during design time. In other words, templates are not only a compliance constraint, but also an aid to (compliant) process modeling. Finally, compliance templates can be stored (e.g., in a central repository) and reused in a variety of similar process models.

A ***compliance template*** comprises three parts, na-mely, an abstract business process, a compliance descriptor, and a variability descriptor.

The ***abstract business process*** defines the *compliant behavior* of a process in terms of its control flow and of allowed activities. It is called "abstract" because it lacks the necessary implementation details to be instantiated and run in a process engine. Only activities labeled *constrained region* can be customized by the process modeler in order to get an executable business process. Customizing a constrained region means inserting activities into it. Process modelers cannot change activities or control flows originally included in a compliance template, as this might lead to non-compliant processes.

As an example refer to Figure 4, which shows an abstract process model

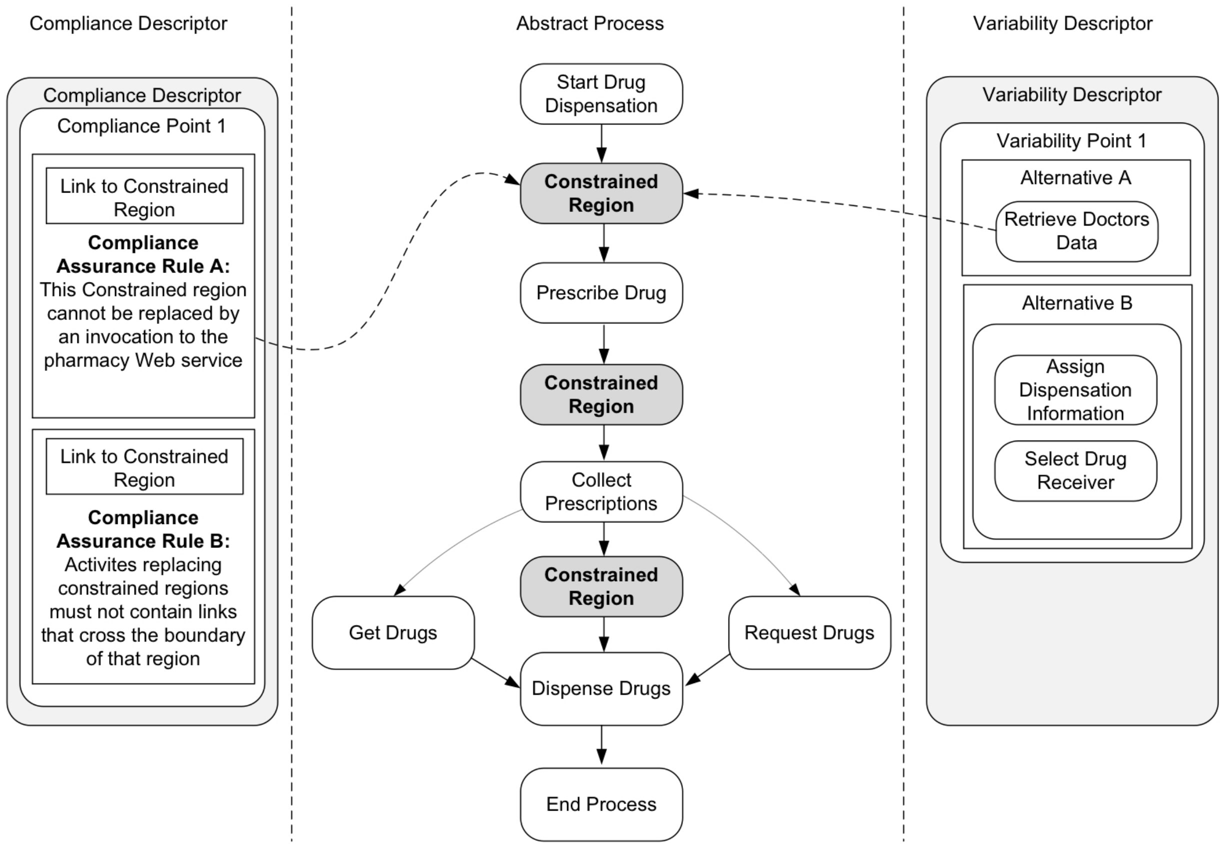


Figure 4: Compliance template for drug dispensations

(in the center of the figure) of the drug dispensation process sketched in Figure 1. We use a pseudo language in Figure 4 to specify the abstract process for reasons of simplicity. Any other process specification language may have been used to define this abstract process, because of the flexibility of the compliance template approach. The abstract process expresses a number of compliance constraints: activity *Prescribe Drug* must always be executed before activity *Collect Prescriptions*; or, after the activity *Collect Prescriptions* has been executed, the activities *Get Drugs* or *Request Drugs* can be executed. The separation of duties requirement for the *Prescribe Drug* and *Dispense Drugs* activities can also be expressed as compliance rules and associated with the respective web services, which must provide for the generation of the necessary evidence (the events) to assess the rules.

The ***compliance descriptor***, at the left of the abstract process model, allows the definition of *constraints* for the constrained regions. Compliance descriptors can be defined independently of the abstract process, and a single compliance descriptor can be re-used in more than one abstract process. The dashed arrow pointing from one compliance assurance rule (*Link to Constrained Region*) in Figure 4 to a constrained region shows which compliance assurance rule is applied to the first constrained region. Rules are expressed in first order logic. We chose first order logic because the class of compliance rules used with compliance templates deals with presence and absence of activities within a constrained region. These kinds of compliance rules can be expressed at best using the negation operator in front of a predicate to describe the absence of activities. Predicates without preceding operand are used to describe the presence of activities. The name of the used predicate maps to the name of the activity. One example for such a compliance rule is: *activity A and activity B must always be inserted together.* With first order logic, the example compliance rule above can easily be expressed as $A \wedge B$.

Compliance rules are evaluated at design time (in our graphical process development tool) every time the process modeler inserts an activity into a constrained region. The graphical tool notifies the process designer about which modifications are allowed and which modifications violate the implicit compliance of the abstract process. For example, an invocation of the *Pharmacy* web service in the first constrained region in Figure 4 would violate the compliance of the abstract process, because the activities *Prescribe Drug* and *Collect Prescriptions* would not yet have been executed.

The meta model of a compliance descriptor is shown in Figure 5. A compliance descriptor contains one or more compliance points comprising compliance rules. These compliance rules can be linked to the constrained regions in the abstract process of a compliance template.

The ***variability descriptor***, at the right of the abstract process model in Figure 4, contains variabilities that can be used to fill the constrained regions of the abstract process. The dashed arrow shows which variability descriptor is associated with which constrained region. A variability descriptor assists the process modeler by providing him/her with the *set of allowed activities* that can be used inside each constrained region; activities can again be compliance

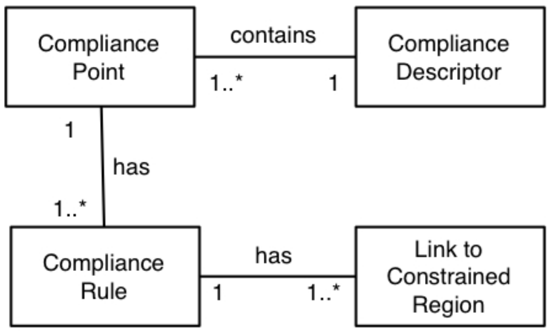


Figure 5: UML meta model of a compliance descriptor

templates containing constrained regions. For instance, we have used *Alternative A* in Figure 4 in the design of the compliant BPEL process. The activities in *Alternative B* are used in other constrained regions. Here it is, for example, important that the compliance expert only allows services (activities) in the variability descriptor that natively encrypt the data they exchange with other services, in order to provide for the anonymization of patients' data.

Compliance templates can be designed for robustness or for reusability. *Robustness* is achieved by adding detailed, domain-specific constraints that guide the process modeler through an only narrow scope of action during the instantiation of a compliance template. *Reu-sability* is achieved by keeping the template more general. It is up to the compliance expert to decide what is more important to him/her. In fact, while our approach facilitates the expression of compliance rules, it is still important for the human expert to have the right regulatory and domain knowledge in order to correctly interpret the company's compliance requirements and express them in terms of compliance rules.

3.1.1 Modeling Compliance Templates and Processes

In order to assist the compliance expert in defining compliance templates, we have extended the Oryx[2] BPMN editor. Oryx is a web-based BPMN editor that fully runs inside a web browser and does not require the installation of any additional software. Figure 6 shows a screen shot of Oryx at work. It mainly consists of three parts: the shape repository (labeled 1), the modeling canvas

[2]http://code.google.com/p/oryx-editor/

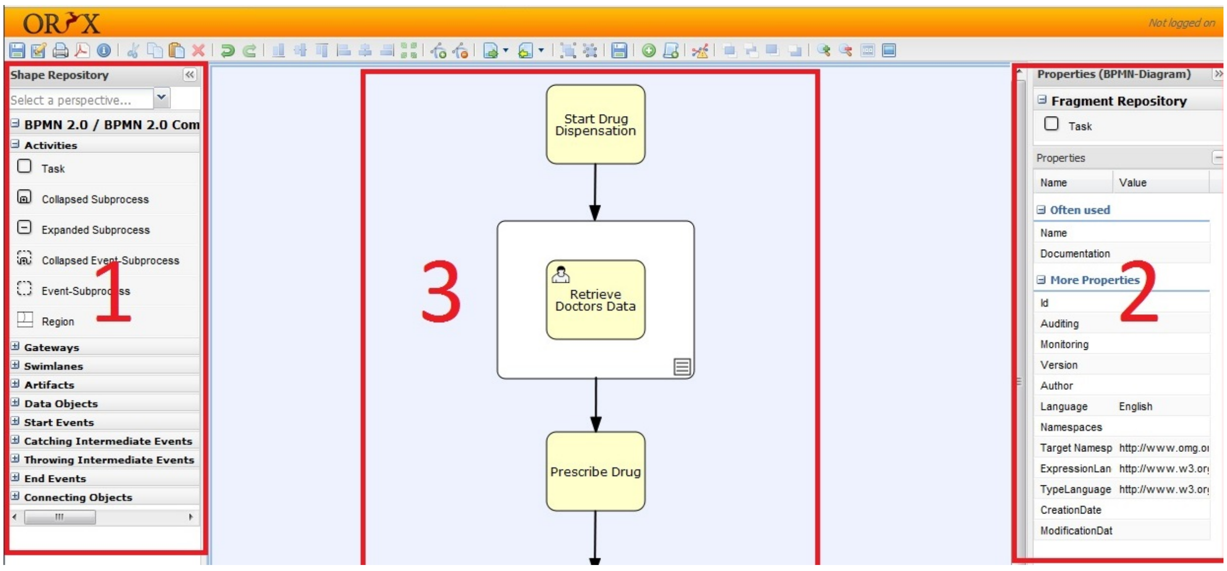


Figure 6: Oryx BPMN editor for compliance templates

(labeled 3), and a pane on the right hand side (labeled 2 and called Fragment Repository) containing the activities that compose the variability descriptor and a properties section.

To be able to create compliance templates we added a new activity type named *Region*. In Figure 6, the region activity type is shown in the shape repository and on the modeling canvas containing the task named Retrieve Doctors Data. To implement the compliance descriptor described above, we added a property named Compliance Descriptor to the region activity type. The Fragment Repository on the right implements the concepts of a variability descriptor as described before. Another addition we made is a compliance checker plug-in. This plug-in is used to check whether an activity inserted into a constrained region violates a constraint or not. The resulting BPMN 2.0 process model is transformed into a BPEL model that includes the mandatory activities imposed by the compliance template as highlighted in Figure 7.

3.1.2 Creating the Signaling Policy

To measure the compliance of a process, evidence on process execution must be generated, to be able to certify which activities have been executed by a given process instance and which have been skipped or which have generated

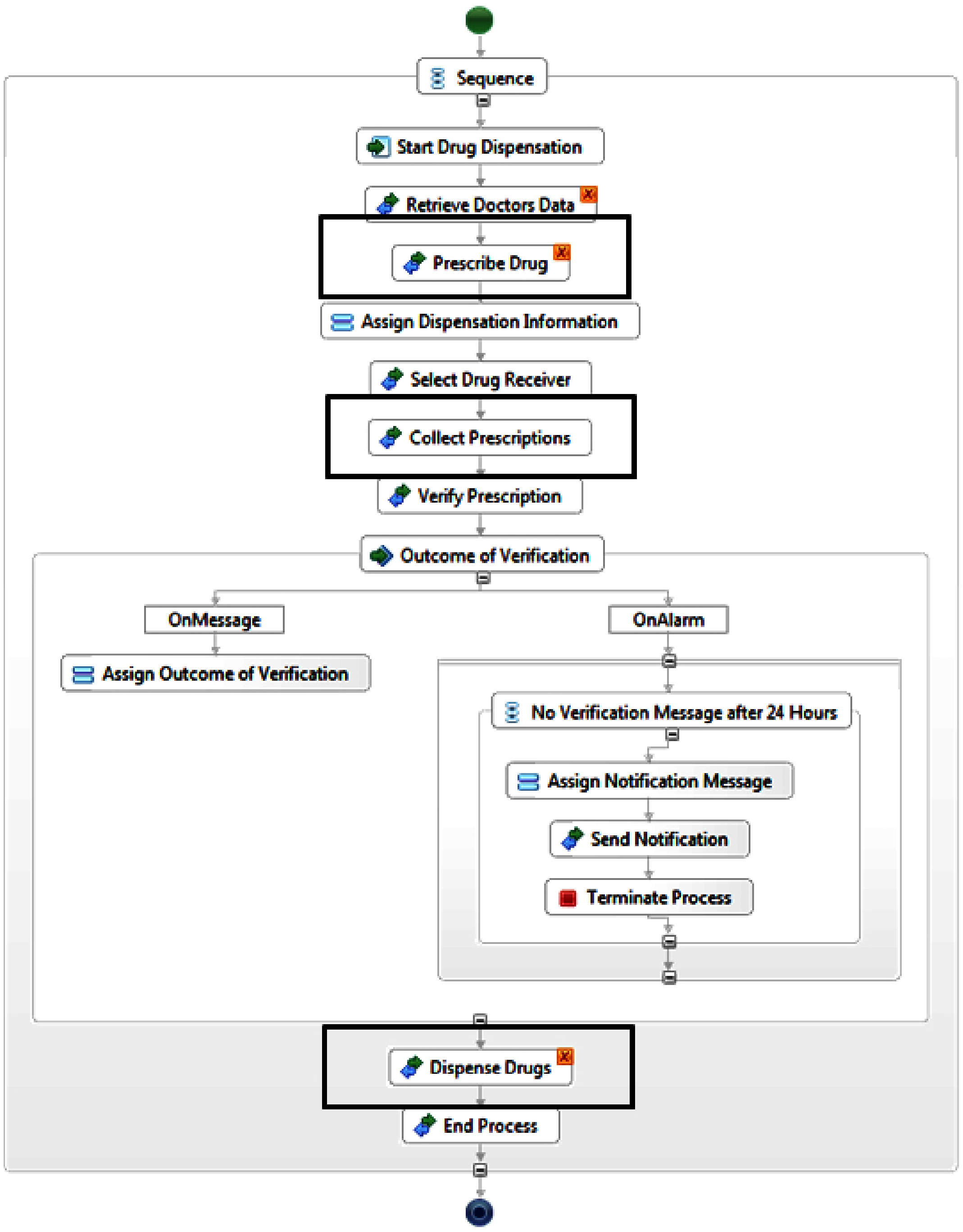


Figure 7: Executable drug dispensation process (for presentation purpose, we omit data assignments)

errors. This evidence is represented by *execution events*, which provide insight into the status of the process instance at the time of their generation. As we want to check compliance with the abstract process of a compliance template, it implicitly defines the minimum set of events (or the event traces) that characterize a compliant process instance. In order to check compliance, it therefore suffices to generate suitable *Start* and *End* events for each mandatory activity in the abstract process. Other execution events may be needed for computing indicators, e.g., if an indicator is to be computed over non-mandatory activities.

The exact set of events is specified in the so-called *signaling policy*, that is, the policy that tells the business process engine which events must be generated during process execution. The necessary events that need to be produced in order to check the compliance of the designed business process, can be chosen by compliance experts. A signaling policy can then be created with this information. In addition, the compliance expert can add properties to activities that hold any form of custom compliance policy beyond what can be expressed via the template. These are also checked in the assessment phase, discussed next.

3.2 Specifying Key Compliance Indicators

Business performance is commonly measured by means of key indicators, typically key performance indicators [21], which are metrics that summarize in a single number how well predefined business goals are being achieved. Similarly, we advocate the use of KCIs to measure *how compliant* a company is with its compliance requirements and to *better target* the company's efforts to check and improve compliance, lowering the overall complexity of compliance management.

KCIs can be computed out of the evidence collected from process executions. Given the huge quantity of available events and runtime data that are typically available for each single process instance, this can however be a very complex task both from the perspective of metaphors and languages for defining such indicators and from the perspective of performance.

We approach both issues by providing the compliance expert with a so-called ***process instance table*** for defining and computing indicators. This is an

Table 1: Excerpt of the process instance table for the drug dispensation process

PID	*TimeRequest*	*TimeReplenish*	*ReqWard*	*PendPresc*	*WrongDisp*	*Compliant*
72665	13-05-10 22:32	14-05-10 16:45	W5	1	2	true
72666	13-05-10 22:39	15-05-10 12:55	W3	3	1	false
72667	13-05-10 22:55	14-05-10 08:59	W3	3	1	true
72668	13-05-10 23:01	14-05-10 23:33	W7	25	4	false
72669	13-05-10 23:49	14-05-10 02:57	W6	2	0	true
....	...	...	...	...	...	...

abstract table that is specific to a given process model and contains one row per executed process instance. The attributes of the table are those process data items that the compliance expert needs for the definition of indicators, plus one or more Boolean attributes for each template to which the model must be compliant (if more than one template apply), reflecting the compliance requirements the process should satisfy. The values of the data items are carried by the events generated at runtime, while all necessary events are specified in the signaling policy and are either derived automatically from compliance templates or manually defined (if they are not yet part of the template). We will see in the assessment section how process instance tables are implemented and populated.

Given a process instance table, an indicator can now be defined as regular mathematical expression over the attributes and rows of the table (on paper by the compliance expert) and it can be implemented via standard SQL queries (by the process modeler). Although indicators typically come in the form of percentage values, averages, sums, or similar, the process instance table abstraction allows us to support the full expressive power of SQL in the computation of indicators. SQL has been designed also as a language for computing aggregates and is well known, understood, and supported, so there was no reason to come up with another language.

Table 1 shows, for instance, an excerpt from the process instance table of the drug dispensation process. The columns *TimeRequest* and *TimeReplenish* represent the time at which a drug request was issued and the time at which the request was fulfilled, respectively, while *PendPresc* and *WrongDisp* tell us the number of pending patient prescriptions and the number of wrong dispensations of drugs by the pharmacy (e.g., with a wrong drug type of quantity).

Notice that the number of wrong dispensations is computed when loading the data warehouse, and it can be done, e.g., by checking the records on the complains from patients about wrong dispensations. Finally, the column $ReqWard$ represents the identification of the ward that issued the request (we omit the other attributes). In the table we assume that the process should only follow one template, so there is only one compliance column.

Having in mind the structure of Table 1, the compliance expert can now, for instance, specify an indicator $KCI_{CompInst}$ to monitor compliance with the process' compliance template:

$$KCI_{CompInst} = \frac{|CompliantInstances|}{|AllInstances|}$$

The process modeler expresses the formula then as follows (we only show a simplified query, e.g., without time intervals; for more details please refer to [22]):

```
count_compliant_inst =
 select count(Compliant)
 from drug_dispensation_instance_table
 where Compliant = true;

count_all_inst =
 select count(Compliant)
 from drug_dispensation_instance_table;

KCI_CompInst =
 count_compliant_inst / count_all_inst;
```

The formula presented above is stored in the data-warehouse together with the definition of the indicator, from where the ETL procedure can retrieve periodically for the computation of indicators.

The specification and computation of the indicator presented in this example is rather trivial. The real challenges reside (i) in identifying which are the most effective indicators (and events) and (ii) in the transformation and correlation of raw events in order to create the process instance tables. In fact, the ease with which we specified and computed the above indicator is a consequence of this data preparation and one of the most important benefits of making this data arrangement.

Although the above examples associate KCIs with individual business processes, it is important to note that we can also have KCIs that measure properties of *multiple related processes* (e.g., a process and its sub-processes). Such kind of advanced KCIs can easily be specified by defining the indicator function over the join of the respective process instances tables, practically enabling the definition of arbitrarily complex indicators.

4 Do – Running Processes and Generating Evidence

Once business processes have been implemented according to their compliance templates and the signaling policy has been completed, processes can be executed and evidence can be collected. In case the process is implemented in BPEL, we also provide support to execute and most importantly to collect evidence (we support BPEL as it is a common situation; in case of ad hoc languages and infrastructures, we expect probes to be developed to generate the necessary events). We have chosen to extend the Apache ODE (http://ode.apache.org/) engine, although any other engine could be extended similarly.

Apache ODE is equipped with a mechanism to issue events at certain state changes of a BPEL process during execution. These events are saved in an internal database, the audit-trail. The audit-trail can be queried via a web service interface to check the execution traces (the sequence of generated internal events) of processes that have been executed and of processes that are still in the executing phase. A drawback of this mechanism is that the audit-trail saves all events generated during process execution. In most cases, a third party is only interested in a subset of events, e.g., events indicating that the process took a certain branch. Thus, these particular events must be separated from the rest of the events in the audit-trail, which is not always an easy task. Also, if we think of distributed business processes with multiple cooperating parties (such as our reference scenario), for security reasons it is typically not possible to query a partner's audit trail. This is a major limitation for the assessment of the compliance of processes whose execution is distributed over multiple parties. To address these problems, we extended the Apache ODE BPEL engine to emit events to external subscribers, where the set of events and the allowed

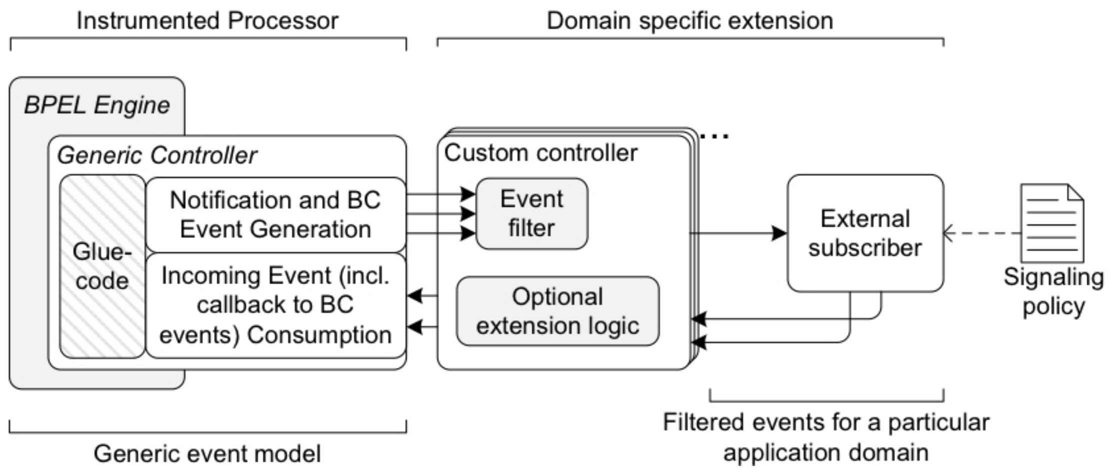

Instrumented Processor
Domain specific extension
BPEL Engine
Generic Controller
Glue-code
Notification and BC Event Generation
Incoming Event (incl. callback to BC events) Consumption
Custom controller
Event filter
Optional extension logic
External subscriber
Signaling policy
Generic event model
Filtered events for a particular application domain

Figure 8: Architectural overview of the core components of the signaling extension of Apache ODE [15]

subscribers can easily be configured (e.g., by means of the signaling policy) [16].

Figure 8 gives a schematic overview of the ***extensions*** made to ODE. On the left side, the instrumented BPEL engine is shown. We extended the BPEL engine with a so-called *generic controller*. It comprises the glue code connecting the process navigation parts of the BPEL engine to the event handling part in the generic controller. At certain points in this execution logic we throw events that are sent to one or more pluggable custom controllers, which correspond to the domain specific part of the signaling architecture (this part corresponds to the *Signaling plug-in* introduced in Figure 2). External stakeholders can write custom controllers to meet the requirements of their particular domain. All events occurring during the execution of a BPEL process are sent to every registered custom controller. In each custom controller, incoming events can be filtered and transformed. These filtered events can then be provided to external subscribers. The external subscribers can configure the filtering logic of the custom controllers. In our case, we use an external controller to parse the signaling policy and to instruct the custom controller to generate only those events that are required to assess compliance.

The ***signaling policy*** contains XPath expressions that point to the activity elements in a BPEL file, which is written in XML. We extended these XPath expressions with event indicators, since each BPEL activity may issue a number of different events. The XPath expressions in a signaling policy thus indicate which events of which activity need to be issued.

The underlying concept of the event subscription is resource-centric. We map process models, process instances, and activities deployed on a BPEL engine to resources and provide a suitable management API that allows one to access the resources. The API is exposed via web service interfaces.

Notice that the approach we present here is for offline assessment of compliance. This means that we log events that will only later be used for compliance assessment (e.g., during night hours). The performance issues for the generation of evidence regards more to the logging of event rather than the actual compliance assessment. Since logging performance is not the focus of this paper, and state of the art logging systems are capable of handling this issue very well, we do not discuss this concern further.

5 Check – Assessing Compliance

From an IT perspective, assessing compliance means developing an assessment engine that "executes" the specification discussed in Section 3 over the event log, which constitutes our "evidence". Specifically, the engine should *verify* that process execution comforms to the different process templates and *compute compliance indicators*. The challenge lies in how to do this in a way that minimizes the development work needed for each new process, new template, or new indicator, otherwise the system will not be easily maintainable. Given that changes are frequent (especially in regulations) this is an important aspect.

To compute ***conformance with templates*** we leverage on raw events. Although events in different processes may have different formats – as the process-specific data differ from process to process – what matters for verifying conformance is the process-independent part of events, that is, their type (Start or End), the activity that generates them, the process and instance in the context of which they are generated, and the occurrence time.

Reasoning in terms of language theory, process models are analogous to grammars and event traces are analogous to language strings. Therefore, computing whether a trace of events as described above conforms to a process model becomes analogous to checking if a string can be generated by a grammar. This is a well-known problem and therefore we do not discuss it further. The output of this procedure consists thus in giving a set of yes/no "labels" to each instance, one for each process template that was associated to the process model of that instance. Developing the conformance algorithm does not require any process-specific logic, which means that no new code is needed each time a new process or template is defined.

The case is different for computing ***indicators***. The metaphor we adopt for the indicator language implies that indicators can be arbitrarily complex queries over a dataset of process execution data with compliance information. This aspect, combined with the needs of providing efficient navigation and drill-down/roll-up (i.e., navigation through the dimension and fact tables of the data warehouse) over a complex dataset as well as the need for a more sophisticated root cause analysis suggests that a sensible approach to leverage is that of building a data warehouse of process data, oriented at computing and assessing compliance and key compliance indicators.

Figure 9 shows an excerpt of our ***dimensional data model***. In the model, described in detail in [22], the facts are essentially the events and the process instances, while dimensions are process models, activities, and actors. Because different processes have different data items, instances of different processes are stored in different fact tables, where the attributes correspond to those process variables that are considered useful for compliance analysis and for computing indicators. These constitute the physical representation of the process instance tables discussed in Section 3.

An alternative approach would have been that of storing all process instance data *vertically*, where each tuple contains instance ID, variable name, and value. The benefit of this approach is that the warehouse sche-ma does not change when new processes are defined. However, writing queries over vertical tables is more difficult and performance is lower, especially due to the high number of self-joins necessary.

The main data source of the warehouse is the event log. From there, the ETL procedure determines how to fill the process instance tables, based on *mapping specifications* done by the *compliance template editor* or the process

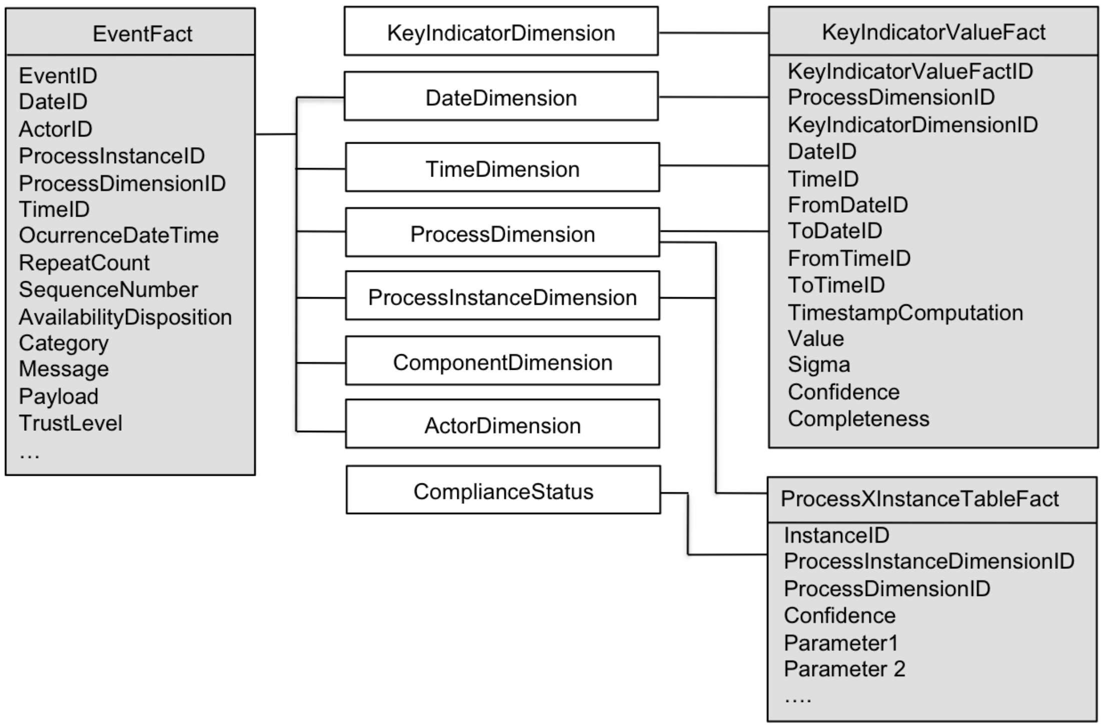


Figure 9: Excerpt of the DW model; fact tables are shaded in gray, dimension tables are white

modeler. In essence these mappings specify from which event and parameter the attributes in the table take their values. This is done via simple XPath statements. For instance, every time a new drug request is inserted into the system, an event of the type $NewDrugRequest$ is emitted that carries the number of pending prescriptions for that drug among its attributes. In order to obtain this information so that we can fill the $PendPresc$ attribute for each row in Table 1, we take all events of type $NewDrugRequest$ and access their $PendPresc$ parameters.

This means that for each new process model or for each change in an existing process model, the process modeler needs to (i) define the process instance table (this is also done in conjunction with the compliance analyst who defines which attributes are relevant for compliance analysis) and (ii) define the XPath expression that is used to populate the attributes for each given instance. Overall, this is something that can be done rather easily. The key problem is figuring out good indicators. Once that is done, the time taken to configure the process instance tables for their computation is small.

The ETL procedure that loads the warehouse incrementally and computes indicators runs *periodically*, e.g., each night or once a week. Only new and completed event traces (process instances) are loaded; running process instances are not considered. This assures that all events needed to assign the compliance labels are available (partial traces could be analyzed, e.g., to identify early violations; however, compliance can still only be ascertained after process termination). Once computed, also the values of indicators are stored in the warehouse (see the *KeyIndicatorValueFact* table in Figure 9) and made available for reporting and further analysis, such as correlation and risk analysis.

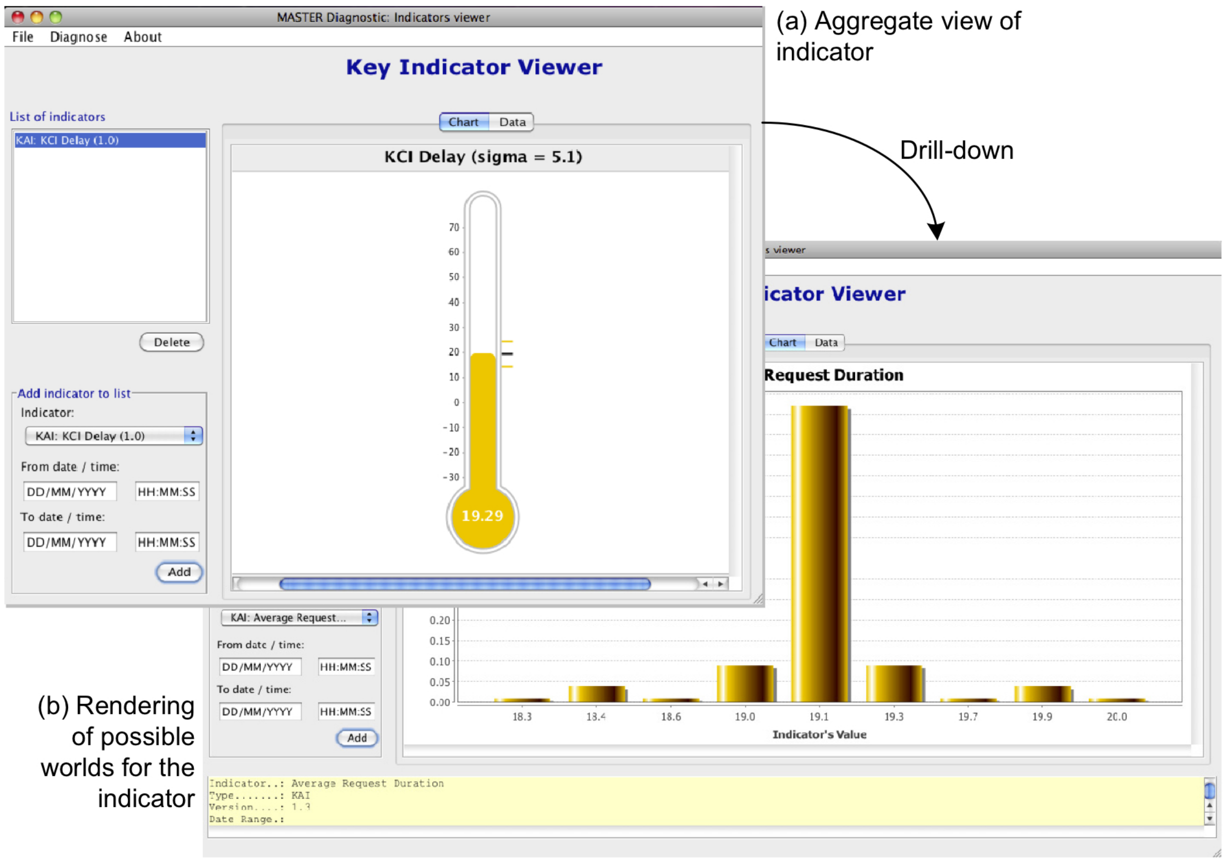
KCIs can then be rendered in the *reporting dashboard* as illustrated in Figure 10, which also takes into account data uncertainty when rendering indicator values to users. A description of the dashboard with details on how uncertainty is managed can be found in [24].

6 Act – Improving Processes and Compliance

This is the last phase of the Deming cycle that aims at *understanding* problems identified in the Check phase. While the cycle is closed by the compliance expert and process modeler by applying their findings in a new Plan phase, IT can significantly assist this phase and reduce the complexity of the analysis task. In the context of compliance management, IT can assist in (i) *identifying correlations among KCIs*, (ii) *identifying correlations among compliance states and business data*, and (iii) *reconstructing the actual behavior* of implemented processes.

6.1 Analyzing Correlations among Indicators

As explained in Section 3.2, indicators measure how well business processes conform to compliance requirements. In doing so, each indicator looks at a different aspect of a process, typically a different compliance requirement. Identifying correlations among indicators therefore allows us to identify relationships among compliance requirements. If we visualize all identified relationships in a graph,



```
Indicator..: Average Request Duration
Type.......: KAI
Version....: 1.3
Date Range.:
```

Figure 10: Visualizing key compliance indicators

this allows one to trace back from one indicator to another to understand its root causes. It is important to notice that correlation only indicates likely causal relationship, not certain causalities. The idea of using correlation is to help the human expert to spot places where to look at for root causes.

A particularly interesting analysis is that of *cross-correlating* multiple indicators over time: there may be situations in which changes in the values of an indicator KCI_1 is associated with changes in the values of another indicator KCI_2, but only after a time interval that also needs to be determined. The typical reason for this result is that KCI_1 is computed over events raised at an early stage of the process while KCI_2 is computed over events raised at a later stage. The dynamics of KCI_2 has therefore its likely roots in the part of the process measured by KCI_1.

Figure 11(a) shows the output of our *correlation analyzer* if applied to the

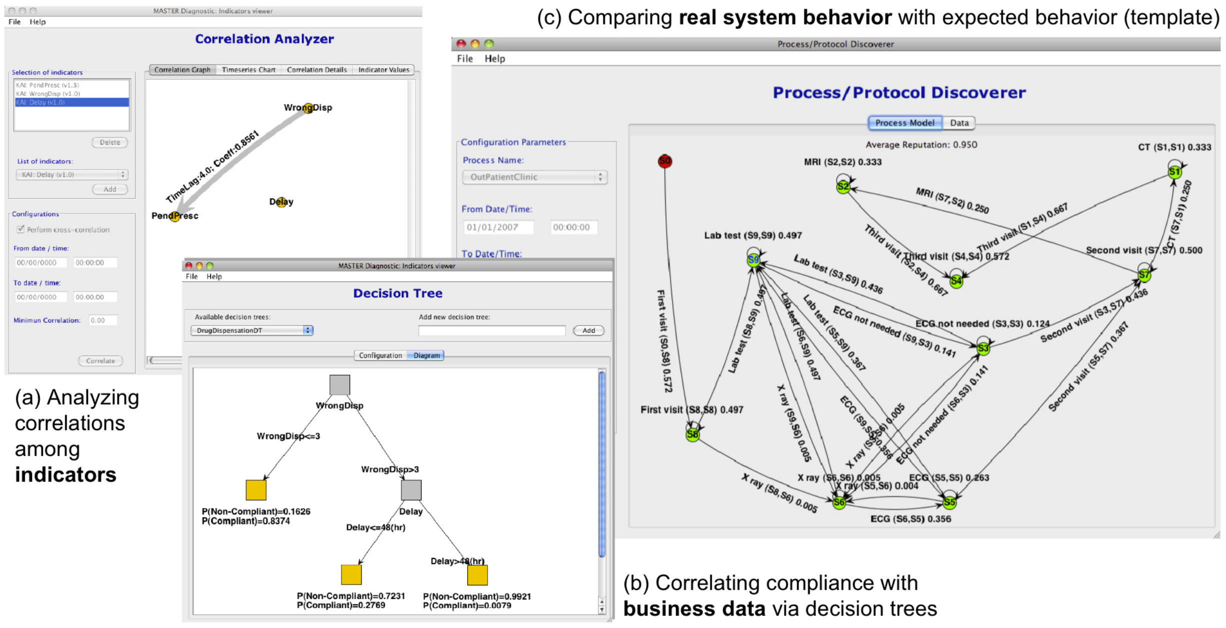

(c) Comparing **real system behavior** with expected behavior (template)
Correlation Analyzer
Correlation Graph
Timeseries Chart
Correlation Details
Indicator Values
WrongDisp
TimeLag:4.0; Coeff:0.8561
Delay
PendPresc
Process/Protocol Discoverer
Process Model
Data
Decision Tree
WrongDisp
WrongDisp<=3
WrongDisp>3
Delay
Delay<=48(hr)
Delay>48(hr)
P(Non-Compliant)=0.1626 P(Compliant)=0.8374
P(Non-Compliant)=0.7231 P(Compliant)=0.2769
P(Non-Compliant)=0.9921 P(Compliant)=0.0079
(a) Analyzing correlations among **indicators**
(b) Correlating compliance with **business data** via decision trees

Figure 11: Instruments of the analysis workbench: correlation analyzer, decision tree miner, protocol discoverer.

three indicators KCI_{Delay} (introduced in Section 3.2), $KCI_{PendPresc}$ (number of pending prescriptions), and $KCI_{WrongDisp}$ (number of erroneous dispensations, i.e., with wrong drug type or quantity). The correlations are based on the cross-corre-lation technique proposed in [6]. The graph shows a dependency ($coeff = 0.856$) among $KCI_{PendPresc}$ and
$KCI_{WrongDisp}$ with a time lag of 4 days (the arrow head of the correlation goes back in time), while there is no correlation with KCI_{Delay} ($coeff < 0.70$). That is, too many pending prescriptions in the system systematically lead to errors (e.g., wrong quantities or drugs dispensed) at dispensation time. More than a simple implementation issue, this correlation hints at an organizational problem in the drug dispensation (e.g., understaffed personnel).

6.2 Classifying Compliance Evaluations

We have shown earlier that we use process instance tables to store each process instance's event trace along with the data the events provide access to and that we associate compliance labels (i.e., *classes*) to each instance for each compliance template the process has to comply with. This conceptualization of the compliance problem allows us to apply standard classification algorithms to identify correlations between the compliance classes (compliant yes/no) and the process data, hopefully unveiling unknown dependencies. Indeed, the process instance table with the compliance "labeling" is the typical data format that can be fed to classification tools. We use *decision trees*, as they are simple and fast in classifying tuples and – more importantly – they are suitable for knowledge discovery without complex settings or assumptions and are easy to interpret and analyze.

Figure 11(b) shows, for instance, the decision tree built out of the data stored in Table 1. For this purpose, we have adopted the algorithm presented in [37]. As can be seen in the figure, the main decision point that affects compliance is $WrongDisp$: if $WrongDisp > 3$, non-compliance is very likely. Along this branch, the second decision point depends on the $Delay$ parameter: if it exceeds 48 hours, non-compliance is almost sure (99% of probability), yet also for lower values of $Delay$ non-compliance is the most likely (72% of probability) outcome.

Decision trees can also be used as a prediction (or risk detection) mechanism. For example, from Figure 11(b) we can derive the following rule:

if *WrongDisp > 3* **and** *Delay > 48hr* **then**
p(non-compliant) = 0.9921

This rule can be used to predict the compliance of process instances while they are still in execution, which allows a company to focus its attention to those process instances that are at risk.

6.3 Discovering Business Protocol Models

The use of the compliance templates introduced in Section 3.1 helps the process modeler to specify process models that are compliant by design with the stated requirements and the logic rules contained in the compliance descriptor. Yet, typically auditors do not assess compliance by looking at models only; rather, they look at how processes have been *executed concretely*. In fact, it is important to recognize that compliant models *do not* assure compliant execution. In practice, problems simply happen, for instance, due to *human factors* (e.g., untrained personnel or the process administrator explicitly changing a running instance or a deployed model without notifying the compliance expert), *misconfiguration* (e.g., wrong service endpoints), or *system failures* (e.g., a hard drive error). It is impossible to predict these kinds of problems and, therefore, it is even more important to identify them after they occurred.

We approach this need by means of protocol discovery, a problem for which there already exist valuable contributions. [18] presents a good overview on the protocol discovery problem and existing approaches to deal with it. Specifically, we have adapted the algorithm introduced in [17] as this algorithm supports the identification of models from service conversations that may be noisy (erroneously containing data from different conversations) and incomplete (missing part of the data produced in one conversation). The reason for this choice, instead of mainstream process and workflow mining techniques, is that we are interested in mining *events* as generated by the infrastructure, which might consist not only of events from the core business process but also of events generated by control processes put on top of it. Instead of focusing on message exchanges, i.e., SOAP messages, we feed the algorithm with events and we identify "conversations" by grouping events according to the process instance they stem from. Fig 11(c), for example, shows the output of the *protocol discoverer* if applied to data from the drug dispensation process. The tool uses finite state machines (FSMs) to graphically represent the reconstructed protocol model: nodes represent intermediate states of a process execution; edges represent events raised during the execution. Nodes are labeled with incremental numbers that serve simply as state IDs, edges with the name of the event they represent and the probability that the corresponding event took place [22].

6.4 User Study and Evaluation

Together with Hospital San Raffaele, we carried out an in-depth evaluation of the usability and understandability of methodology described in this paper. The evaluation involved the our *target users*, specifically the business process owner (the pharmacy), the process analyst/modeler, an internal auditor, a quality and accreditation expert, IT staff, and the CIO of the hospital, and took the form of a two-days evaluation workshop, which allowed us to collect feedback via questionnaires, interviews, and focus groups.

As *object of the evaluation* we used the prototypes and demos developed with the audit experts from Deloitte and described in this paper. The evaluation was performed using real data from the scenario described in this paper. The dataset consisted in 30000 drug dispensations done between January and April 2009. This dataset allowed us to make a realistic demo of our tool to showcase the indicator correlation and decision tree analysis as well as the business protocol discovery. The required size of the dataset for building good correlation and decision tree models depends strongly on the properties of the dataset (e.g., on whether indicators are computed for each process instance only weekly or monthly, or on the number of decision points inside a given process). The protocol discovery algorithm can instead infer a model already from a single process instance, capturing however only the behavior of this single process execution. The complexity of cross-correlation is linear in the number of KCIs by the number of considered data items by the number of time shifts [6]; the performance of decision tree computation and protocol discovery is discussed in [37] and [17], respectively.

According to the study, both the compliance templates and the Reporting Dashboard tool (for the Check phase) used to display indicators and navigate through the collected compliance data was perceived as very useful by all participants, while the process analyst, quality and accreditation expert, internal auditor, and business process owner particularly emphasized the usefulness of the correlation analyzer, decision tree miner, and business protocol miner. The overall judgment of the set of tools for the Check and Act phase reached an average score of 8 in an interval that ranges from 1 (very negative) to 10 (very positive).

The complete evaluation report D1.3.2 is available via the project web site (`http://www.project-master.eu`). Details on the implemented tools and a set of demonstration videos are available via `http://mashart.org/SOCA-Compliance`.

7 Related Work

We discuss the related work in five areas as related to our work, namely, IT governance, SOA governance, business process compliance, reporting on business performance and mining process execution logs.

IT governance. IT governance aims at ensuring that companies' IT systems sustain and extend the companies' strategies and objectives. Many frameworks have been proposed to approach IT governance, including COBIT, ITIL, ISO 2000 and ISO 17799. The focus of each of these varies from one another, from the alignment of business objectives to IT objectives (e.g., COBIT), to IT service management (e.g., ITIL), to IT security management (e.g., ISO 17799). While these frameworks typically provide general guidelines and best practices on how to govern IT, they provide no guidelines that are specific to compliance management. IT governance may act either as the source of compliance requirements or as a guide on how to instrument IT for compliance management. In the first case, for example, it may happen that a company must comply with one of these frameworks in order to provide services to a third party; in the second case, the framework itself can help enable compliance management. As such, IT governance and compliance management therefore complement each other.

SOA governance. SOA governance can be considered as a branch of IT governance where the focus is put on SOA-based systems. As in IT governance, many frameworks has been proposed to approach SOA governance. For example, Brauer and Kline 2005 [2] approach SOA governance in the area of business service life cycle through two key infrastructures: the business service registry and business service management. Software AG [34] proposes a maturity model with six levels: technology enablement, SOA enablement, SOA business services, SOA lifecycle management, SOA consistency and SOA optimization. It further describes the lifecycle of a service and SOA roles and provides a list of best practices and common mistakes to avoid. SAP AG [29] proposes a list of common guidelines and patterns for the modeling and implementation of enterprise services at different levels, including, map of process components and business objects, service interfaces and services operations per business objects, structure of message types, common set of reusable data types, transactional behavior and service implementation. Oracle's approach to SOA governance [19] proposes a framework, and a list of best practices that expands throughout the service lifecycle. It furthermore proposes a list of six steps to successful a successful SOA governance model, which aims at maturing the overall SOA and thereby its business goals. IBM [3] proposes an approach that relies on a lifecycle for SOA governance, which is distinguishable different from a service lifecycle that is governed. The SOA governance lifecycle consists of 4 phases: (i) in the plan phase, the governance focus is determined, (ii) in the define phase, the SOA governance model is defined, (iii) the enable phase, is where the SOA governance is implemented, and (iv) the measure phase, is where the governance model is measured and refined. All these frameworks deal with the governance of SOA-based systems to different degrees. Just like IT governance focus on managing the company's IT, SOA governance focuses on managing the overall lifecycle of SOA-based systems and the guidelines provided there are only at the high level and therefore they are not useful for compliance management as addressed in this paper.

Business process compliance. There is a considerable amount of work in the area of business process compliance. In [7] the authors describe, for instance, an algorithm to generate a BPMN model from a set of constraints written in deontic logic. In [9] deontic logic is also used to annotate business process models with compliance rules. Such annotations are then used to check compliance of the business process. Hoffmann et al. [14] instead use first order logic to annotate business process models with compliance constraints. The authors also show how to check compliance of a business process with these constraints. In [5] the authors propose the use of domain-specific languages to annotate processes with compliance constraints, and they equip their modeling tool with compliance-specific views on the process. Shadiq et al. [28] describe how control objectives can be modeled in formal contract language to annotate process models in the form of control tags that can be used by analysis tools to perform compliance checks on the business process model. Governatori et al. [8] advance this line of static compliance check of normative control objectives and provide status reports that highlight problematic cases together with the

control objectives that are violated.
Our approach is based on compliance templates that are the starting point for the development of a compliant business process. With this approach we cover the first phase of the compliance management life cycle. A compliance template implicitly defines the compliant behavior of the resulting business process. The variable parts of the compliance template are annotated with constraints written in first order logic. As opposed to lines of work like [28] and [8], we prefer to work with first order logic because it is a standard and well-understood formalism that suffices for our purposes and because compliance experts are more likely to be familiar with it. The so represented constraints prevent the compliance template from modifications that violate the compliance rules associated to the template. Yet, conceptually every formalism that allows us to express compliance rules over process events could be adopted in our system. From the modeling perspective, we advocate the use of these compliance templates because they are closer to compliance experts and process modelers. We further use compliance templates to provide process modelers with real-time conformance feedback during the instantiation of compliance templates (static compliance checks).

Reporting on business process performance. Several works focus on the reporting on business process performance. For instance, works like [30], [4], [33], [23], [11] and [24] focus on warehousing process execution data, so as to make these data available in a suitable schema for reporting and OLAP purposes. We face similar reporting issues in our dashboard, yet our aim is to analyze compliance of business processes not performance. This also leads us to the concept of KCIs as a special type of key performance indicator (KPI). In [21] the authors model KPIs and the relationships that exist among them. Our internal, XML-based representation of KCIs is very similar to the model proposed for KPIs, while, instead of modeling relationships, we discover them via cross-correlation for root causes analysis. Finally, there are many business process management commercial suites that include reporting on business process performance as part of the toolset, e.g., HP Business Process Monitor, IBM Business Process Manager, Oracle Business Process Management Suite, SAP Business Process Management and TIBCO Spotfire.

Mining process execution logs. Data mining techniques have been also used for analyzing business process execution data. As for the *root cause analysis*, Grigori et al. [12] [11] focus on understanding, predicting, and preventing exceptions in business executions by using decision trees built upon workflow log files. In the same line of thought, Rozinat and van der Aalst [26] mine event logs for decision point analysis, Apte et al. [1] focus on classification and prediction of customer behaviors, and Seol et al. [32] use the inputs and ouputs of each process to build decision trees for the analysis of the efficiency of processes. There are however no works that specifically address the problem of understanding compliance violations. In the context of *process and workflow mining*, several works have been proposed that aim at discovering process models and checking the conformance of process executions using process execution data. For instance, works like [17], [10], and [20] aim at discovering workflow/process models from execution logs with special focus on the behavioral/structural aspects of the process models. Rozinat and van der Aalst [27] [25] focus instead on the automatic verification of how well process executions conform with a predefined process model. We adopt algorithms of the first class for discovering protocol models; however, algorithms of the second class could be adopted for compliance assessment.

8 Conclusion

With this paper, we approach a relevant and critical issue in today's business reality, i.e., compliance management, and we do so by specifically taking into account the peculiarities of the service-oriented architecture and of distributed business contexts, two paradigms that heavily influence current and future business practices. Differently from many works in literature, we do not focus on monitoring and enforcement at the individual message level. We rather take the auditor's perspective and focus on the design of compliant processes and the assessment and improvement of their compliance. We assist these activities by means of (i) a *model and tool* to design compliant processes, (ii) an *extended service orchestration engine* to generate process execution evidence, and (iii) a *reporting and analysis suite* to report on compliance and support root cause analysis, in order to provide for better informed decision making. As such, the models and instruments we propose in this paper complement existing monitoring and enforcement approaches and provide for a comprehensive approach to service-based compliance management.

Our aim was to devise a solution having in mind the real needs of auditors (internal and external ones) and – more importantly – with the help of people who are involved every day in the auditing of companies (the dashboard [33] and solutions proposed in this paper have extensively been discussed with partners from Deloitte). While this paper specifically targets a company's internal compliance expert and process modeler, also the external auditor can benefit from the proposed system, e.g., by using the compliance reporting dashboard as a starting point for his analysis. This will not change the auditor's own auditing practice, yet the sole use of a systematic and assisted approach to compliance management will surely impact positively on the auditor's perception of the company's commitment to compliance.

Acknowledgements: This work was supported by funds from the European Commission (Contract Nbr. 216917 for the FP7-ICT-2007-1 project MASTER).

References

[1] Apte, C., Bibelnieks, E., Natarajan, R., Pednault, E., Tipu, F., Campbell, D., Nelson, B.: Segmentation-Based Modeling for Advanced Targeted Marketing. In: KDD'01, pp. 408–413 (2001)

[2] Brauer, B., Kline, S.: SOA Governance: A Key Ingredient of the Adaptive Enterprise. Tech. rep., Hewlett-Packard (2005). URL `http://goo.gl/WxTSe`

[3] Brown, W., Moore, G., Tegan, W.: SOA governance: IBM's approach. Tech. rep., IBM (2006). URL `http://goo.gl/q9Ini`

[4] Casati, F., Castellanos, M., Dayal, U., Salazar, N.: A Generic Solution for

Warehousing Business Process Data. In: VLDB'07, pp. 1128–1137. VLDB Endowment (2007)

[5] Daniel, F., Casati, F., D'Andrea, V., Strauch, S., Schumm, D., Leymann, F., Mulo, E., Zdun, U., Dustdar, S., Sebahi, S., de Marchi, F., Hacid, M.S.: Business Compliance Governance in Service-Oriented Architectures. In: AINA'09. IEEE Press (2009)

[6] Dunn, P.: Measurement and Data Analysis for Engineering and Science. McGraw-Hill Science (2004)

[7] Goedertier, S., Vanthienen, J.: Designing Compliant Business Processes from Obligations and Permission. In: BPM Workshops, vol. 4103, pp. 5–14. Springer (2006)

[8] Governatori, G., Hoffmann, J., Sadiq, S.W., Weber, I.: Detecting regulatory compliance for business process models through semantic annotations. In: BPM Workshops, pp. 5–17 (2008)

[9] Governatori, G., Sadiq, S.: The journey to business process compliance. Handbook of Research in BPM pp. 426–454 (2009)

[10] Greco, G., Guzzo, A., Pontieri, L., Sacca, D.: Discovering Expressive Process Models by Clustering Log Traces. IEEE TKDE **18**(8), 1010–1027 (2006)

[11] Grigori, D., Casati, F., Castellanos, M., Dayal, U., Sayal, M., Shan, M.: Business process intelligence. Computers in Industry **53**(3), 321–343 (2004)

[12] Grigori, D., Casati, F., Dayal, U., Shan, M.C.: Improving business process quality through exception understanding, prediction, and prevention. In: VLDB'01, pp. 159–168. San Francisco, CA, USA (2001)

[13] Hagerty, J., Hackbush, J., Gaughan, D., Jacobson, S.: The Governance, Risk Management, and Compliance Spending Report, 2008-2009: Inside the $32B GRC Market. Tech. rep., AMR Research (2008)

[14] Hoffmann, J., Weber, I., Governatori, G.: On compliance checking for clausal constraints in annotated process models. Inf. Sys. Frontiers **14**(2), 1–23 (2009)

[15] Khalaf, R., Karastoyanova, D., Leymann, F.: Pluggable Framework for Enabling the Execution of Extended BPEL Behavior. In: WESOA'07. Springer (2007)

[16] van Lessen, T., Leymann, F., Mietzner, R., Nitzsche, J., Schleicher, D.: A Management Framework for WS-BPEL. In: ECOWS'08, pp. 187–196. IEEE (2008)

[17] Motahari-Nezhad, H.R., Saint-Paul, R., Benatallah, B., asati, F.: Deriving Protocol Models from Imperfect Service Conversation Logs. IEEE Trans. on Knowl. and Data Eng. **20**(12), 1683–1698 (2008)

[18] Musaraj, K., Yoshida, T., Daniel, F., Hacid, M.S., Casati, F., Benatallah, B.: Message Correlation and Web Service Protocol Mining from Inaccurate Logs. In: Proceedings of ICWS'10 (2010)

[19] Oracle: SOA Governance: Framework and Best Practices. Tech. rep., Oracle (2007). URL `http://goo.gl/dtZjz`

[20] Pinter, S.S., Golani, M.: Discovering Workflow Models from Activities' Lifespans. Comput. Ind. **53**(3), 283–296 (2004)

[21] Popova, V., Sharpanskykh, A.: Modeling organizational performance indicators. Inf. Sys. **35**(4), 505–527 (2010)

[22] Rodriguez, C., Daniel, F., Casati, F., Anstett, T., Schleicher, D., Burri, S.: Warehouse Model and Diagnostic Algorithms. Deliverable d6.2.2, MASTER Project (2009). URL `http://www.master-fp7.eu/`

[23] Rodríguez, C., Daniel, F., Casati, F., Cappiello, C.: Computing uncertain key indicators from uncertain data. In: ICIQ'09, pp. 106–120 (2009)

[24] Rodríguez, C., Daniel, F., Casati, F., Cappiello, C.: Toward Uncertain Business Intelligence: the Case of Key Indicators. IEEE Internet Computing **14**(4), 32–40 (2010)

[25] Rozinat, A., Van der Aalst, W.: Conformance testing: Measuring the fit and appropriateness of event logs and process models. In: Business Process Management Workshops, pp. 163–176. Springer (2006)

[26] Rozinat, A., Aalst, W.: Decision mining in business processes. Beta, Research School for Operations Management and Logistics (2006)

[27] Rozinat, A., van der Aalst, W.M.P.: Conformance Checking of Processes Based on Monitoring Real Behavior. Inf. Syst. **33**(1), 64–95 (2008)

[28] Sadiq, S.W., Governatori, G., Namiri, K.: Modeling Control Objectives for Business Process Compliance. In: BPM'07, pp. 149–164 (2007)

[29] SAP AG: Governance for Modeling and Implementing Enterprise Services at SAP (2007). URL `http://goo.gl/kFAvS`

[30] Sayal, M., Casati, F., Dayal, U., Shan, M.C.: Business Process Cockpit. In: VLDB '02, pp. 880–883. VLDB Endowment (2002)

[31] Schleicher, D., Anstett, T., Leymann, F., Mietzner, R.: Maintaining Compliance in Customizable Process Models. In: CoopIS'09, *LNCS*, vol. 5870, pp. 60–75 (2009)

[32] Seol, H., Choi, J., Park, G., Park, Y.: A framework for benchmarking service process using data envelopment analysis and decision tree. Expert Systems with Applications **32**(2), 432–440 (2007)

[33] Silveira, P., Rodríguez, C., Casati, F., Daniel, F., D'Andrea, V., Worledge, C., Taheri, Z.: On the Design of Compliance Governance Dashboards for Effective Compliance and Audit Management. In: NFPSLAM-SOC'09. Springer (2009)

[34] Software AG: SOA Governance: "Rule your SOA". Tech. rep., Software AG (2007). URL `http://goo.gl/EtgEi`

[35] Tarantino, A.: Governance, Risk, and Compliance Handbook. John Wiley and Sons, Inc. (2008)

[36] Trent, H.: Products for Managing Governance, Risk, and Compliance: Market Fluff or Relevant Stuff? In-depth research report, Burton Group (2008)

[37] Tsang, S., Kao, B., Yip, K.Y., Ho, W.S., Lee, S.D.: Decision Trees for Uncertain Data. In: ICDE'09, pp. 441–444. IEEE (2009)

[38] Walton, M.: The Deming Management Method. Perigee Books (1988)

Part II

Process Design Perspective

Appendix H

Wisdom-aware Computing: On the Interactive Recommendation of Composition Knowledge

Wisdom-Aware Computing: On the Interactive Recommendation of Composition Knowledge *

Soudip Roy Chowdhury Carlos Rodríguez Florian Daniel
Fabio Casati

Abstract

We propose to enable and facilitate the development of service-based development by exploiting community composition knowledge, i.e., knowledge that can be harvested from existing, successful mashups or service compositions defined by other and possibly more skilled developers (the community or crowd) in a same domain. Such knowledge can be used to assist less skilled developers in defining a composition they need, allowing them to go beyond their individual capabilities. The assistance comes in the form of interactive advice, as we aim at supporting developers while they are defining their composition logic, and it adjusts to the skill level of the developer. In this paper we specifically focus on the case of process-oriented, mashup-like applications, yet the proposed concepts and approach can be generalized and also applied to generic algorithms and procedures.

1 Introduction

Although each of us develops and executes various procedures in our daily life (examples range from cooking recipes to low-level programming code), today very little is done to support others, possibly less skilled developers (or, in the extreme case, even end users) in developing their own. Basically, there are two main approaches to enable less skilled people to develop: either development is eased by simplifying it (e.g., by limiting the expressive power of a development language) or it is facilitated by reusing knowledge (e.g., by copying and pasting from existing algorithms).

Among the simplification approaches, the workflow and BPM community was one of the first to claim that the abstraction of business processes into

*The final publication is available at www.springerlink.com (http://link.springer.com/chapter/10.1007%2F978-3-642-%2019394-1%2015) tasks and control flows would allow also the less skilled users to define own processes, however with little success. Then, with the advent of web services and the service-oriented architecture (SOA), the web service community substituted tasks with services, yet it also didn't succeed in enabling less skilled developers to compose services. Recently, web mashups added user interfaces to the composition problem and again claimed to target also end users, but mashup development is still a challenge for skilled developers. While these attempts were aimed at simplifying technologies, the human computer interaction community has researched on end user development approaching the problem from the user interface perspective. The result is simple applications that are specific to a very limited domain, e.g., an interactive game for children, with typically little support for more complex applications.

As for what regards capturing and reusing knowledge, in IT reuse typically comes in the form of program libraries, services, or program templates (such as generics in Java or process templates in workflows). In essence, what is done today is either providing building blocks that can be composed to achieve a goal, or providing the entire composition (the algorithm possibly made generic if templates are used), which may or may not suit a developer's needs. In the nineties and early 2000s, AI planning [1] and automated, goal-oriented compositions (e.g., as in [2]) became popular in research. A typical goal there is to derive a service composition from a given goal and a set of components and composition rules. Despite the large body of interesting research, this thread failed to produce widely applicable results, likely because the goal is very ambitious and because assumptions on the semantic richness and consistency of component descriptions are rarely met in practice. Other attempts to extract knowledge are, for example, oriented at identifying social networks of people [3] or at providing rankings and recommendations of objects, from web pages (Google's Pagerank) to goods (Amazon's recommendations). An alternative approach is followed by expert recommender systems [4], which, instead of identifying knowledge, aim at identifying knowledge holders (the experts), based on their code production and social involvement.

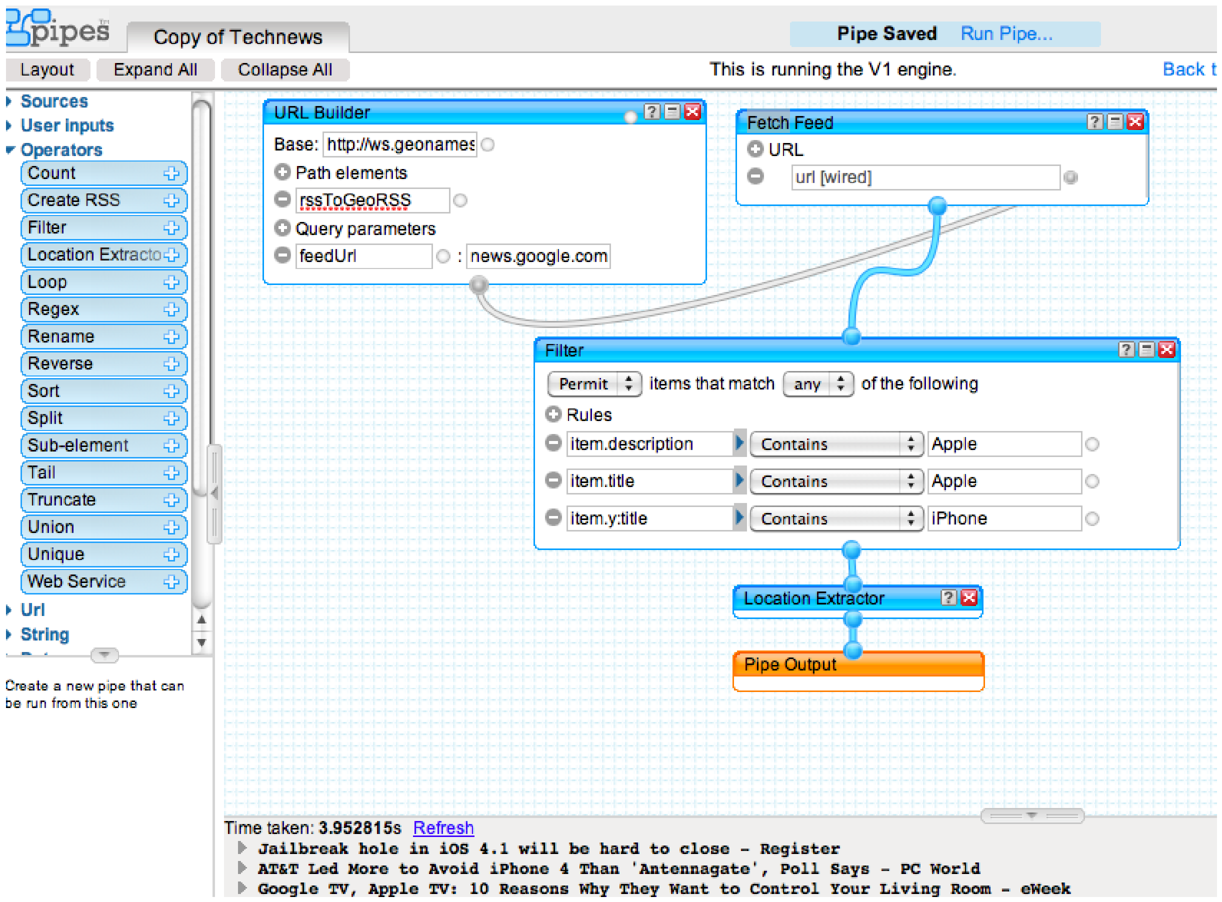
In this paper, we describe WIRE, a WIsdom-awaRE development environment we are currently developing in order to enable less skilled developers to perform also complex development tasks. We particularly target process-oriented, mashup-like applications, whose development and execution can be provided as a service via the Web and whose internals are characterized by relatively simple composition logic and relatively complex tasks or components. This class of programs seems to provide both the benefit of (relative) simplicity and a sufficient information base (thanks to the reuse of components) to learn and reuse programming/service composition knowledge. The idea is to learn from existing compositions (or, in general, computations) and to provide the learned knowledge in form of interactive advice to developers while they are composing their own application in a visual editor. The aim is both to allow developers to go beyond their own development capabilities and to speed up the overall development process, joining the benefits of both simplification and reuse.

Next, we discuss a state of the art composition scenario and we show that it is everything but trivial. In Section 3, we discuss the state of the art in assisted composition. In Section 4 and 5, we investigate the idea of composition advices and provide our first implementation ideas, respectively. Then we conclude the paper and outline our future work.

2 Example Scenario and Research Challenges

In order to better understand the problem we want to address, let's have a look at how a mashup is, for instance, composed in Yahoo! Pipes (http://pipes.yahoo.com/pipes/), one of the most well-known mashup platforms as of today. Let's assume we want to develop a simple pipe that sources a set of news from Google News, filters them according to a predefined filter condition (in our case, we want to search for news on products and services by a given vendor), and locates them on a Yahoo! Map.

The pipe that implements the required feature is illustrated in Figure 1. It is composed of five components: The URL Builder is needed to set up the remote Geo Names service, which takes a news RSS feed as an input, analyzes its content, and inserts geo-coordinates, i.e., longitude and latitude, into each news item (where possible). Doing so requires setting some parameters:



Base: http://ws.geonames
Path elements: rssToGeoRSS
Query parameters: feedUrl : news.google.com

URL: url [wired]

Permit items that match any of the following
Rules

item.description	Contains	Apple
item.title	Contains	Apple
item.y:title	Contains	iPhone

Time taken: 3.952815s Refresh

```
Jailbreak hole in iOS 4.1 will be hard to close - Register
AT&T Led More to Avoid iPhone 4 Than 'Antennagate', Poll Says - PC World
Google TV, Apple TV: 10 Reasons Why They Want to Control Your Living Room - eWeek
```

Figure 1: Implementation of the example scenario in Yahoo! Pipes.

Base=http://ws.geonames.org, Path elements=rssToGeoRSS, and Query parameters=FeedUrl:news.google.com/news?topic=t&output=rss&ned=us. The so created URL is fed into the Fetch Feed component, which loads the geo-enriched news feed. In order to filter out the news items we are really interested in, we need to use the Filter component, which requires the setting of proper filter conditions via the Rules input field. Feeding the filtered feed into the Location Extractor component causes Pipes to plot the news items on a Yahoo! Map. Finally, the Pipe Output component specifies the end of the pipe.

If we analyze the development steps above, we can easily understand that developing even such a simple composition is out of the reach of people without programming knowledge. Understanding which components are needed and how they are used is neither trivial nor intuitive. The URL Builder, for example, requires the setting of some complex parameters. Then, components need to be suitably connected, in order to support the data flow from one component to another, and output parameters must be mapped to input parameters. But more importantly, plotting news onto a map requires knowing that this can be done by first enriching a feed with geo-coordinates, then fetching the actual feed, and only then the map is ready to plot the items.

Enabling non-expert developers to compose a pipe like the above requires telling (or teaching) them the necessary knowledge. In WIRE, we aim to do so by providing non-expert developers with interactive development advices for composition, inside an assisted development environment. We want to obtain the knowledge to provide advices by extracting, abstracting, and reusing compositional knowledge from existing compositions (in the scenario above, pipes) that contain community knowledge, best practices, and proven patterns. That is, in WIRE we aim at bringing the wisdom of the crowd (possibly even a small crowd if we are reusing knowledge within a company) in defining compositions when they are both defined by an individual (where the crowd supports an individual) or by a community (where the crowd supports social computing, i.e., itself in defining its own algorithms). The final goal is to move towards a new frontier of knowledge reuse, i.e., reuse of computational knowledge.

Doing so requires approaching a set of challenges that are non-trivial:

1. First of all, identifying the types of advices that can be given and the right times when they can be given: depending on the complexity and expressive power of the composition language, there can be a huge variety of possible advices. Understanding which of them are useful is crucial to limit complexity.

2. Discovering computational knowledge: how do we harvest development knowledge from the crowd, that is, from a set of existing compositions? Knowledge may come in a variety of different forms: component or service compatibilities, data mappings, co-occurrence of components, design patterns, evolution operations, and so on.

3. Representing and storing knowledge: once identified, how do we represent

and store knowledge in a way that allows easy querying and retrieval for reuse?

4. Searching and retrieving knowledge: given a partial program specification under development, how do we enable the querying of the knowledge space and the identification of the most suitable and useful advice to provide to the developer, in order to really assist him?

5. Reusing knowledge: given an advice for development, how do we (re)use the identified knowledge in the program under development? We need to be able to weave it into the partial specification in a way that is correct and executable, so as to provide concrete benefits to the developer.

In this paper, we specifically focus on the first challenge and we provide our first ideas on the second challenge and on the assisted development environment.

3 State of the Art

In literature, there are approaches that aim at similar goals as WIRE, yet they mainly focus on the retrieval and reuse of composition knowledge. In [6], for instance, mashlets (the elements to be composed) are represented via their inputs and outputs, and glue patterns are represented as graphs of connections among them; reuse comes in the form of auto-completion of missing components and connections, selected by the user from a ranked list of top-k recommendations obtained starting from the mashlets used in the mashup. In [8], light-weight semantic annotations for services, feeds, and data flows are used to support a text-based search for data mashups, which are actually generated in an automated, goal-oriented fashion using AI planning (the search tags are the goals); generated data processing pipes can be used as is or further edited. The approach in [9] semantically annotates portlets, web apps, widgets, or Java beans and supports the search for functionally equivalent or matching components; reuse is supported by a semantics-aided, automated connection of components. Also the approach in [10] is based on a simple, semantic description of information sources (name, formal inputs [allowed ones], actual inputs [outputs consumed from other sources], outputs) and mashups (compositions of information sources), which can be queried with a partial mashup specification in order identify goals based on their likelihood to appear in the final mashup; goals are fed to a semantic matcher and an AI planner, which complete the partial mashup. This last approach is the only one that also automatically discovers some form of knowledge in terms of popularity of outputs in existing mashup specifications (used to compute the likelihoods of goals).

In the context of business process modeling, there are also some works with similar goals as ours. For instance, in [7], the authors more specifically focus on business processes represented as Petri nets with textual descriptions, which are processed (also leveraging WordNet) to derive a set of descriptive tags that can be used for search of processes or parts thereof; reuse is supported via copy and paste of results into the modeling canvas. The work presented in [13] proposes an approach for supporting process modeling through object-sensitive action patterns, where these patterns are derived from a repository of process models using techniques from association rule learning, taking into consideration not only actions (tasks), but also the business objects to which these actions are related. Finally, [14] presents a model for the reuse data mining processes by extending the CRISP-DM process [15]. The proposed model aims at including data mining process patterns into CRISP-DM and to guide the specialization and application of such patterns to concrete processes, rather than actually exploiting the community knowledge.

In general, the discovery of community composition knowledge is not approached by the works above (or they do it in a limited way, e.g., by deriving only behavioral patterns from process definitions). Typically, they start from an annotated representation of mashups and components and query them for functional compatibilities and data mappings, improving the quality of search results via semantics, which are explicit and predefined. WIRE, instead, specifically focuses on the elicitation and collection of crowd wisdom, i.e., composition knowledge that derives from the ways other people have solved similar composition problems in the past and that has a significant support in terms of number of times it has been adopted. This means that in order to create knowledge for WIRE, we do not need any expert developer or domain specialist that writes and maintains explicit composition rules or logics; knowledge is instead harvested from how people compose their very own applications, without requiring them to provide additional meta-data or descriptions (which typically doesn't work in practice).

4 Wisdom-Aware Development: Concepts and Principles

Identifying which advices can be provided and which advices do indeed have the potential to help less skilled developers to perform complex development tasks requires, first of all, understanding the expressive power of the composition language at hand. We approach this task next. Then we focus on the advices.

4.1 Expressiveness of the Composition Language

Let us consider again Yahoo! Pipes. The platform has a very advanced and pleasant user interface for drag-and-drop development of data mashups and supports the composition of also relatively complex processing logics. Yet, the strong point of Pipes is its data flow based composition paradigm, which is very effective and requires only a limited set of modeling constructs. As already explained in the introduction, constraining the expressive power of composition languages is one of the techniques to simplify development, and Pipes shares this characteristic with most of today's mashup platforms.

In order to better understand the expressiveness of Yahoo! Pipes, in Figure 2 we derived a meta-model for its composition language. A Pipe is composed of components and connectors. Components have a name and a description and may be grouped into categories (e.g., source components, user input components, etc.). Each pipe contains always one Pipe Output component, i.e., a special component that denotes the end of data flow logic or the end of the application. A component may be embedded into another component; for example components (except user inputs and operators) can be embedded inside a Loop Operator component. Components may also have a set of parameters. A Parameter has a name, a type, and may have a value assigned to it. There

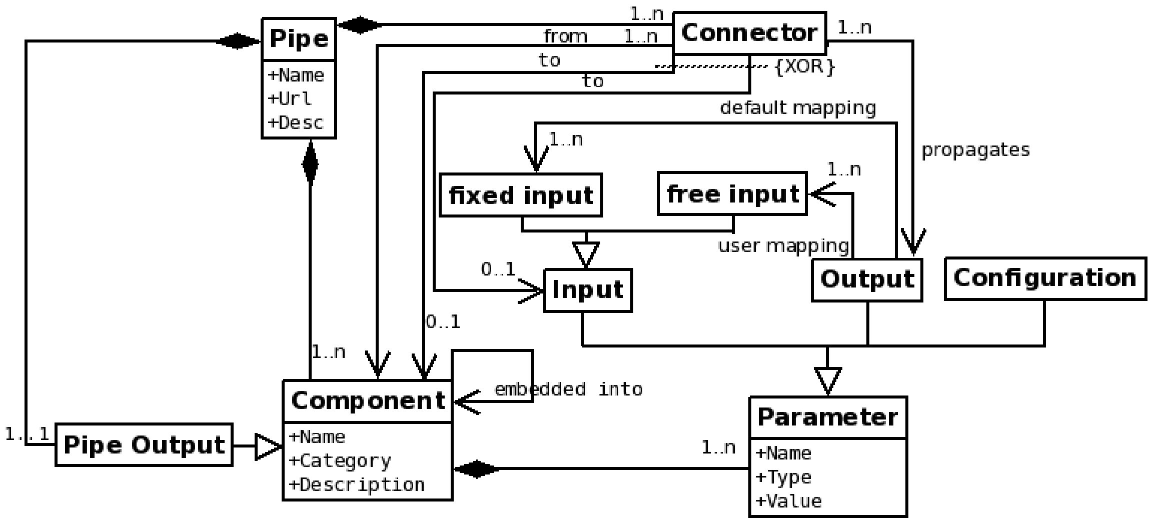


Figure 2: A meta-model for Yahoo! Pipes' composition language.

are basically three types of parameters: input parameters (accept data flows attributes), output parameters (produce data flow attributes), and configuration parameters (are manually set by the developer). For instance, in our example in Section 2, the URL parameter of the Fetch Feed component is an input parameter; the longitude and latitude attributes of the RSS feed fetched by the Fetch Feed component are output parameters; and the Base parameter of the URL Builder component is an example of configuration parameter.

Data flows in Pipes are modeled via dedicated connectors. A Connector propagates output parameters of one component (indicated in Figure 2 by the from relationship) to either another component or to an individual input field of another component. If a connector is connected to a whole component (e.g., in the case of the connector from the Fetch Feed component to the Filter component in Figure 1), all attributes of the RSS item flowing through the connector can be used to set the values of the target component's input parameters. If a connector is connected only to a single input parameter, the data flow's attributes are available only to set the value of the target input parameter. Input parameters are of two types: either they are fixed inputs, for which there are predefined default mappings, or they are free inputs, for which the user can provide a value or choose which flow attribute to use. That is, for free inputs it is possible to specify a simple attribute-parameter data mapping logic.

Figure 2 shows that Yahoo! Pipes' meta-model is indeed very simple: only 10 concepts suffice to model its composition features. Of course, the focus of Pipes is on data mashups, and there is no need for complex web services or user interfaces, two features that are instead present in our own mashup platform, i.e., mashArt [5]. Yet, despite these two additions, mashArt's meta-model only requires 13 concepts. If instead we look at the BPMN modeling notation for business processes [11], we already need more than 20 concepts to characterize its expressive power, and the meta-model of BPEL [12] has almost 60 concepts! Of course, the higher the complexity of the language, the more difficult it is to identify and reuse composition knowledge.

4.2 Advising Composition Knowledge

Given the meta-model of the composition language for which we want to provide composition advices, it is possible to identify which concrete compositional knowledge can be extracted from existing compositions (e.g., pipes). The gray boxes in the conceptual model in Figure 3 illustrate the result of our analysis. The figure identifies the key entities and relationships needed to provide composition advices.

An Advice provides composition knowledge in form of composition patterns. An advice can be to complete a given pattern (given it's partial implementation in the modeling canvas) or to substitute a pattern with a similar one, or the advice can highlight compatible elements in the modeling canvas or filter and rank advices.

Patterns represent the actual recommendation that we deliver to the user. They can be of five types (all these patterns can be identified in the model in Figure 3):

- Parameter Value Patterns: Possible values for a given parameter. For instance, in the URL Builder component the Base parameter value in a pattern can be set to http://ws.geonames.org, while the Path elements parameter value can be rssToGeoRSS, and feedUrl can be news.google.com/news?topic=t&output=rss&ned=us, as shown in our example scenario. Alternatively, we can have the URL Builder component with the Base parameter

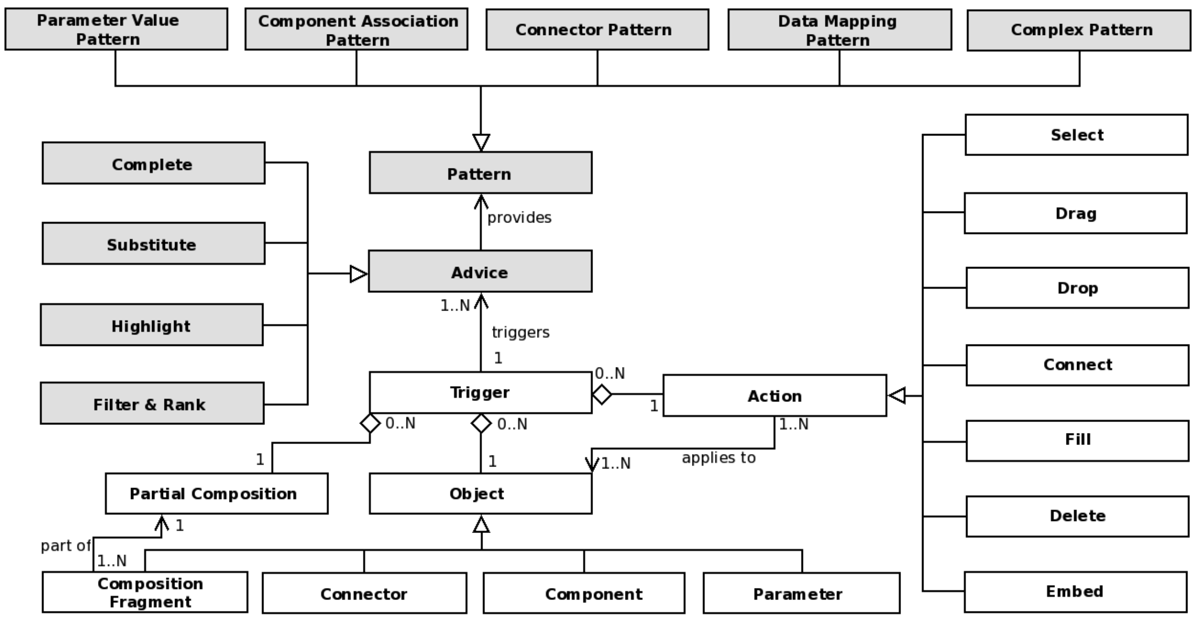


Figure 3: Conceptual model of WIRE's advice approach. Gray entities model the ingredients for advices; white boxes model the advice triggering logic inside the design environment.

set to news.google.com/news and the Query parameters set with different values.

- Component Association Patterns: Co-occurrence patterns for pairs of components. For instance, in our scenario, whenever a user drags and drops the URL Builder on the design canvas, a possible advice derived from a component association pattern can be to include in the composition the Fetch Feed component and connect it to the URL Builder.
- Connector Patterns: Component-component or component-input parameter patterns. This pattern captures the dataflow logic, i.e., how components are connected via connector elements. For example, URL Builder connector- Fetch Feed is a connector pattern in our example scenario.
- Data Mapping Patterns: Associations of outputs to inputs. In Figure 1, for instance, we map the description, title, and y:title attributes of the fetched feed to the first input field of the first, second, and third rule, respectively, telling the Filter component how we map the individual attributes in input

to the individual, free input parameters of the component.

- Complex Patterns: Partial compositions consisting of multiple components, connectors, and parameter settings. In our example scenario, different combinations of components and connectors, having their parameter values set and with proper data mappings, as a part and as a whole represent complex patterns. For example, the configuration URL Builder - Fetch Feed - Filter - Location Extractor, along with their settings, represents a complex pattern.

An Advice provides composition knowledge in form of composition patterns. An advice can be to complete a given pattern (given it's partial implementation in the modeling canvas) or to substitute a pattern with a similar one, or the advice can highlight compatible elements in the modeling canvas or filter and rank advices.

Patterns represent the actual recommendation that we would like to deliver to the user. They can be of five different types: Complex Patterns (partial compositions possibly consisting of multiple components, connectors, and parameter settings), Parameter Value Patterns (possible values for a given parameter), Component Association Patterns (co-occurrence patterns for pairs of components), Connector Patterns (component-component or component-input parameter patterns), and Data Mapping Patterns (associations of outputs to inputs).

Now, let's discuss the white part of the model. This part represents the entities that jointly define the conditions under which advices can be triggered. A Trigger for an advice is defined by an object, an action of the user in the modeling canvas, and the state of the current composition, i.e., the partial composition in the modeling canvas. This association can be thought of as a triplet that defines the triggering condition. The Objects a user may operate are Composition Fragments (e.g., a selection of a subset of the pipe in the canvas), individual Components, Connectors, or Parameters (by interacting with the respective graphical input fields). The Action represents the action that the user may perform on an object during composition. We identify seven actions: Select (e.g., a composition fragment or a connector), Drag (e.g., a component or a connector endpoint), Drop, Connect, Fill (a parameter value), Delete, and Embed (one component into another). Finally, the Partial Composition represents the status of the current overall composition.

While the object therefore identifies which advice may be of interest to the user, the action decides when the advice can be given, and the state filters out advices that are not compatible with the current partial composition (e.g., if the Location Extractor component has already been used, recommending its use becomes useless).

Regarding the model in Figure 3, not all associations may be needed in practice. For instance, not all components are compatible with the embed action. Yet, the model identifies precisely which advices can be given and when.

5 The WIRE Platform

Figure 4 illustrates the high-level architecture of the assisted development environment with which we aim at supporting wisdom-aware development according to the model described in the previous section: developers can design their applications in a wisdom-aware development environment, which is composed of an interactive recommender (for development advice) and an offline recommender as well as the wisdom-aware editor implementing the interactive development paradigm. Compositions or mashups are stored in a compositions repository and can be executed in a dedicated runtime environment, which generates execution data. Compositions and execution data are the input for the knowledge/advice extractor, which finds the repeated and useful patterns in them and stores them as development and evolution advice in the advice repository. Then, the recommenders provide them as interactive advices through its query interface upon the current context and triggers of the user's development environment. Here, we specifically focused on development advices related to composition; we will approach evolution advices in our future work (evolution advices will, for instance, take into account performance criteria or evolutions applied by developers over time on their own mashups).

We realize that each domain will have suitable languages and execution

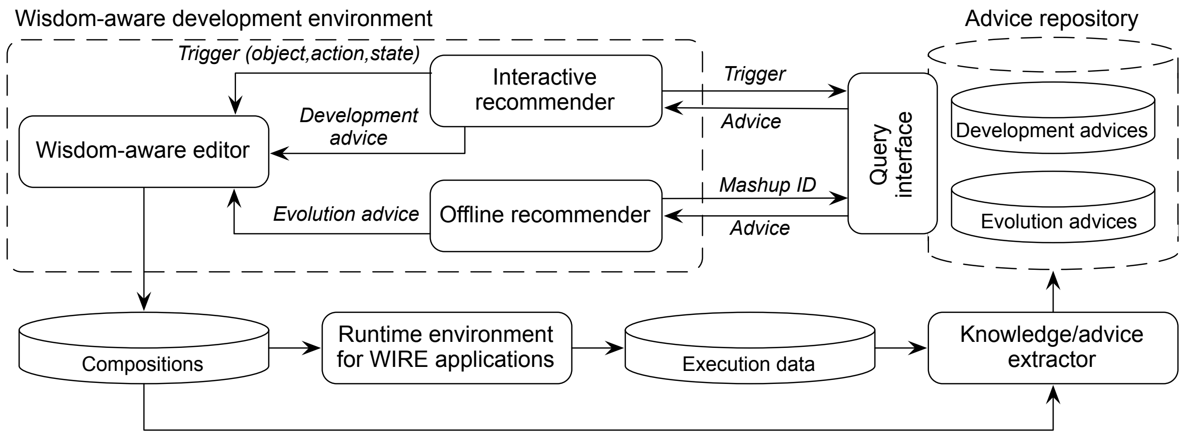


Figure 4: High-level architecture of the envisioned system for wisdom-aware development.

engines, such as a mashup engine or a scientific workflow engine. Our goal is not to compete with these, but to define mechanism to WIRE these languages and tools with the ability to extract knowledge and provide advice. For this reason, in this paper we started with studying the case of Yahoo! Pipes, which is well known and allows us to easily explain our ideas. We however intend to apply the wisdom-aware development paradigm to our own mashup editor, mashArt [5], which features a universal composition paradigm user interface components, application logic, and data web services, a development paradigm that is similar in complexity to that of Pipes.

As for the reuse of knowledge, the WIRE approach is not based on semantic annotations, matching, or AI planning techniques, nor do we aim at automated or goal-driven composition or at identifying semantic similarity among services. We also do not aim at having developers tag components or add metadata to let others better reuse services, processes, or fragments. In other words, we aim at collecting knowledge implicitly, as we believe that otherwise we would face an easier wisdom extraction problem but end up with a solution that in practice does not work because people do not bother to add the necessary metadata. WIRE will rather leverage on statistical data analysis techniques and data mining as means to extract knowledge from the available information space. To do so, we propose the following core steps:

- Cleaning, integration, and transformation: We take as input previous compositions and execution data and prepare them for the analysis.

- Statistical data analysis and data mining: On the resulting data, we apply statistical data analysis and data mining techniques, which may include mining of frequent patterns, association rules, correlations, classification and cluster analysis. The results of this step are used to create the composition patterns.
- Evaluation and ranking of advices (knowledge): Once we have discovered the potential advices, we evaluate and rank them using standard interestingness measures (e.g., support and confidence) and ranking algorithms.
- Presentation of advices: The advices are presented to the user through intuitive visual metaphors that are suitable to the context and purpose of the advice.
- Gathering of user feedback: The popularity of advices is gathered and measured in order to better rank them.

Among the techniques we are applying for the discovery tasks, we are specifically leveraging on data mining approaches, such as frequent itemset mining, association rules learning, sequential pattern mining, graph mining, and link mining. Each of these techniques can be used to discover a different type of advice:

- Frequent itemset mining: The objective of this technique is to find the co-occurrence of items in a dataset of transactions. The co-occurrence is considered frequent whenever its support equals or exceeds a given threshold. This technique can be used as a support for discovering any of the advices introduced before. For instance, in the case of discovering Component Association Patterns we can this technique.
- Association rules: This technique aims at finding rules of the form $A \to B$, where A and B are disjoint sets of items. This technique can be applied to help in the discovery of any of the proposed advices. For instance, in the case of the Parameter Value Pattern, given the value of two parameters of

a component, we can find an association rule that suggests us the value for a third parameter.

- Sequential pattern mining: Given a dataset of sequences, the objective of sequential pattern mining is to find all sequences that have a support equal or greater than a given threshold. This technique can be applied to discover Complex Patterns, Component Association Patterns, and Connector Patterns. For instance, in the case of the Connector Pattern, we can use this technique to extract patterns that can be then used for suggesting connectors among components placed on the design canvas.
- Graph mining: given a set of graphs, the goal of graph mining is to find all subgraphs such that their support is equal or greater than a given threshold. For our purpose, we can use graph mining for discovering Complex Patterns and Connector Patterns. For instance, for Complex Patterns we can suggest a list of existing ready compositions based on the partial composition the user has in the canvas, whenever this partial composition is deemed as frequent.
- Link mining: rather than a technique, link mining refers to a set of techniques for mining data sets where objects are linked with rich structures. Link mining can be applied to support the discovery of any of the proposed advices. For example, in the case of Data Mapping Patterns, we can discover patterns for mapping the parameters of two components, based on the types these parameters.

Once community composition knowledge has been identified, we store the extracted knowledge in the advice repository in the form of directed graphs. In our advice repository, elements in the patterns, e.g., a component or a connector, are represented as nodes of the graph, and relationships among them, e.g., a component has a parameter, are represented as edges between those nodes. We also store a set of rules in our advice repository, which represent the trigger conditions under which a specific knowledge can be provided as an advice. Based upon this information, through our query interface we can match knowledge with the current composition context and retrieves relevant advices from the advice repository. Retrieved advices are filtered, ranked, and delivered based on user profile data (e.g., the programming expertise of the user or his/her preferences over advice types).

6 Conclusion

In this paper we propose the idea of wisdom-aware computing, a computing paradigm that aims at reusing community composition knowledge (the wisdom) to provide interactive development advice to less skilled developers. If successful, WIRE can extend the developer base in each domain where reuse of algorithmic knowledge is possible and it can facilitate progressive learning and knowledge transfer.

Unlike other approaches in literature, which typically focus on structural and semantic similarities, we specifically focus on the elicitation of composition knowledge that derives from the expertise of people and that is expressed in the compositions they develop. If, for instance, two components have been used together successfully multiple times, very likely their joint use is both syntactically and semantically meaningful. There is no need to further model complex ontologies or composition rules.

In order to provide identified patterns with the necessary semantics, we advocate the application of the WIRE paradigm to composition environments that focus on specific domains. Inside a given domain, component names are self-explaining and patterns can easily be understood. In the Omelette (http://www.ict-omelette.eu/) and the LiquidPub (http://liquidpub.org/) projects, we are, for instance, working on two domain-specific mashup platforms for telco and research evaluation, respectively.

For illustration purposes, in this paper we used Yahoo! Pipes as reference mashup platform, as Pipes is very similar in complexity to our own mashArt platform [5] but better known. In order to have access to the compositions that actually hold the knowledge we want to harvest, we will of course apply WIRE to mashArt.

Acknoledgements. This work was supported by funds from the European Commission (project OMELETTE, contract no. 257635).

References

[1] H. Geffner. Perspectives on artificial intelligence planning. AAAI'02, pp.1013-1023.

[2] D. Roman, J. de Bruijn, A. Mocan, H. Lausen, J. Domingue, C. Bussler, D. Fensel. WWW: WSMO, WSML, and WSMX in a Nutshell, ASWC'06, pp. 516-522.

[3] A. Koschmider, M. Song, H.A. Reijers. Social Software for Modeling Business Processes. BPM'08 Workshops, pp. 642-653.

[4] T. Reichling, M. Veith, V. Wulf. Expert Recommender: Designing for a Network Organization. Computer Supported Cooperative Work, vol. 16, no. 4-5, pp. 431-465, Oct. 2007.

[5] F. Daniel, F. Casati, B. Benatallah, M.-C. Shan. Hosted Universal Composition: Models, Languages and Infrastructure in mashArt. ER'09, pp. 428-443.

[6] O. Greenshpan, T. Milo, N. Polyzotis. Autocompletion for mashups. VLDB'09, pp.538-549.

[7] T. Hornung, A. Koschmider, G. Lausen. Rommendation Based Process Modeling Support: Method and User Experience. ER'08, pp. 265-278.

[8] A.V. Riabov, E. Bouillet, M.D. Feblowitz, Z. Liu, A. Ranganathan. Wishful Search: Interactive Composition of Data Mashups. WWW'08, pp. 775-784.

[9] A.H.H. Ngu, M. P. Carlson, Q.Z. Sheng. Semantic-Based Mashup of Composite Applications. IEEE Transactions on Services Computing, vol. 3, no. 1, Jan-Mar 2010.

[10] H. Elmeleegy, A. Ivan, R. Akkiraju, R. Goodwin. MashupAdvisor: A Recommendation Tool for Mashup Development. ICWS'08, pp. 337-344.

[11] OMG. Business Process Model and Notation (BPMN) - Version 1.2, January 2009. [Online] http://www.omg.org/spec/BPMN/1.2

[12] OASIS. Web Services Business Process Execution Language Version 2.0, April 2007. [Online]. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[13] S. Smirnov, M. Weidlich, J. Mendling, M. Weske. Object-Sensitive Action Patterns in Process Model Repositories. BPM'10 Workshops, NJ, USA, September 2010.

[14] D. Wegener, S. Rueping. On Reusing Data Mining in Business Processes A Pattern-based Approach. BPM'10 Workshops, NJ, USA, September 2010.

[15] C. Shearer. The CRISP-DM model: the new blueprint for data mining. Journal of Data Warehousing, Vol. 5, Nr. 4, pp. 1322, 2000.

Appendix I

End-User Requirements for Wisdom-Aware EUD

End-User Requirements for Wisdom-Aware EUD *

Antonella De Angeli   Alberto Battocchi   Soudip Roy Chowdhury
Carlos Rodríguez   Florian Daniel   Fabio Casati

Abstract

This paper presents requirements elicitation study for a EUD tool for composing service-based applications. WIRE aims at enabling EUD by harvesting and recommending community composition knowledge (the wisdom), thus facilitating knowledge transfer from developers to end-users. The idea was evaluated with 10 contextual interviews to accountants, eliciting a rich set of information, which can lead to requirements for Wisdom-Aware EUD.

1 Introduction

There are two common approaches to enable less skilled users to develop software artifacts. Development can be eased by simplifying it or by reusing knowledge. Among the simplification approaches, the business process management and service computing communities have focused on abstracting process development and service composition into activities, as well as control and data flows. However, these are still challenging tasks even for expert developers [1,2]. Traditional reuse approaches, in the form of program libraries, services, or templates (such as generics in Java or process templates in workflows) have targeted developers rather than end-users. Recently, some effort has been invested into knowledge reuse techniques for end-users. In programming by demonstration [3], the system auto-completes a process definition, starting from a set of examples chosen by the user. Goal-oriented approaches [5] assist the users by automatically composing solutions that satisfy user-specified goals. Pattern-based

*The final publication is available at www.springerlink.com (http://link.springer.com/chapter/10.1007%2F978-3-642-21530-8_21) development [4] proposes the use of libraries of patterns provided by experts to represent good development practices, yet patterns, such as the glue patterns in [7], may also be derived from existing compositions. Syntactic approaches [11], for instance, suggest operators based on syntactic similarity (comparing output and input data types), while semantic-based approaches [6] annotate ingredients to support the retrieval of semantically matching elements.

While some of these approaches support end-users with reusable knowledge, they all suffer from some shortcomings. Programming by demonstration and goal-based approaches propose best, complete solutions, not allowing the user to control which exact ingredients the solution should contain. Pattern and semantics-based approaches are hard to maintain, in that they require explicit input from human experts.

In this paper we present the results of a requirement study for WIRE (WIsdom-awaRE development environment) a EUD tool to exploit the benefits of simplification and reuse. WIRE targets process-oriented, mashup-like applications that are characterized by relatively simple composition logics and complex tasks or components. This class of programs provides the benefit of simplicity (composition, not coding) and a sufficient information base (the compositions themselves). The idea is to learn from existing compositions developed by expert IT developers and provide learned knowledge in the form of interactive recommendations to facilitate EUD.

2 WIRE

The motivation behind the idea of WIRE has derived by the analysis of the shortcomings of existing mashup development tools. To exemplify this claim, let us consider a simple application created by Yahoo! Pipes, which retrieves news feeds from a specified website, filters the content based on user-specified criteria, and publishes the filtered content for viewing (Fig. 1). Such a simple application requires 5 components. The user has to set the value of the configuration parameters of a component (e.g., the URL Parameter of the Fetch Feed component) and define the dataflow logic between components. Assuming that an end-user has this kind of technical knowledge is not realistic.

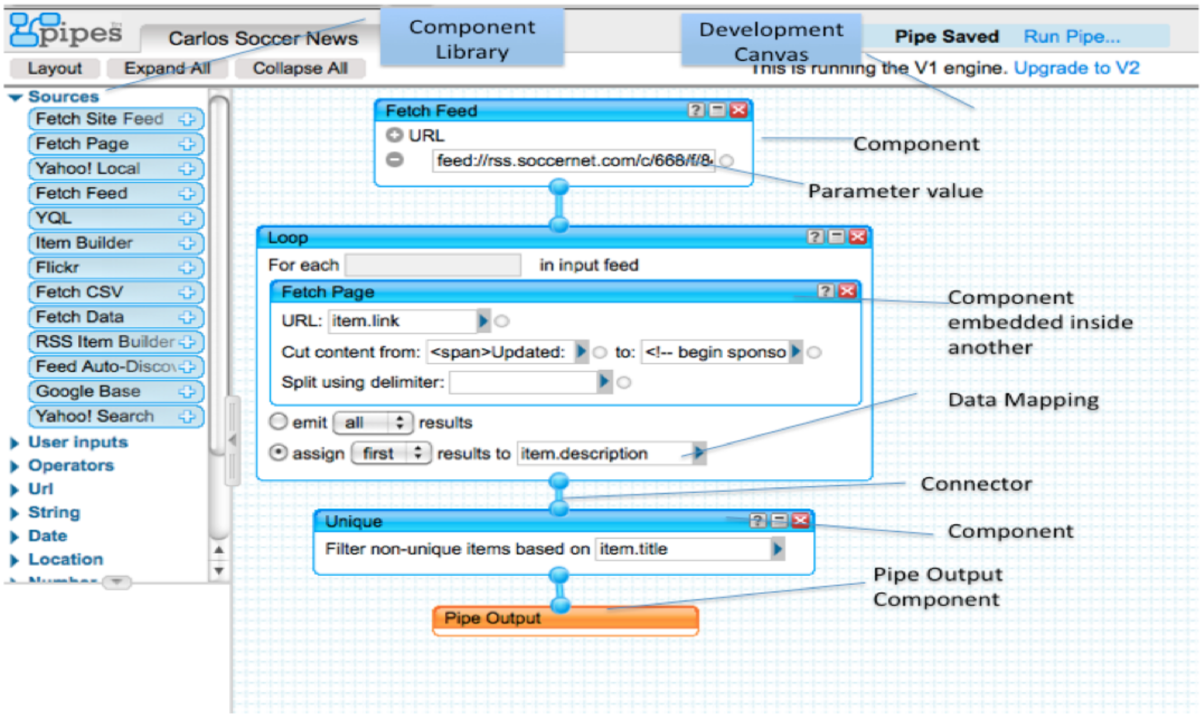


Figure 1: Implementation of the example scenario in Yahoo! Pipes.

WIRE is aimed at discovering technical knowledge by analyzing existing, successful applications, storing knowledge as development advice (community composition knowledge[8]), and delivering it in the form of contextual interactive recommendations to the end-user. The intuition is that this knowledge can be captured through composition patterns and reused as recommendations. The patterns we identified include Parameter Values (e.g., values for the URL parameter in the Fetch Feed component), Component Associations (e.g., suggest that a Loop component should be added together with a Fetch Feed component), Connectors (e.g., possible connections between components), Data Mapping (e.g., suggest that the item.link element coming from the Fetch Feed component should be mapped to the URL parameter of the Fetch Page component), or Complex patterns (e.g., suggest adding components based on a Component Association pattern together with the wiring among them based on a Connector pattern). A detailed explanation of the conceptual model and architecture of WIRE is presented in [8].

3 User study

An evaluation of the conceptual design of WIRE was run in order to address benefits and limitations of the proposal and elicit user requirements [12]. The evaluation was based on contextual interviews to 10 University accountants (7 F, 3 M; mean age = 37 years of age), which lasted approximately one hour. None of them had a background in computer science. Participation was rewarded with 15 Euros. The interview addressed two main topics. Section A targeted the strategies that people use for overcoming the difficulties that emerge while using computers during day-to-day work, and their attitudes towards computer-provided help and advice with particular focus on the comparison between automatic/contextual and on-demand help. Participants were shown a slideshow of familiar examples of automatic/contextual advice (i.e., word completion in the Google search box, friend suggestion in Facebook, book suggestions in Amazon, passwords auto-save in web-browsers, pop-up reminder on calendars, related videos sidebar on YouTube), invited to comment on each example, and report their understanding on how the advice was created.

Section B collected opinions and suggestions about WIRE by a plus and a minus scenario [9] reporting on an accountant who is using WIRE for automating the process of management of travel reimbursement. Both scenarios described the effects brought forward by WIRE on a new user. These effects were taken to the positive or negative extreme to help users to think what consequences the approach could have in their work practices. In the Positive Scenario, the accountant had a successful experience, which helped him to save time and speed up repetitive work leading to adoption. In the Negative Scenario, the accountant encountered serious difficulties and eventually decided to go back to his traditional work procedures. Scenarios were presented with a counterbalanced order. Interviewees were asked specific questions addressing their willingness to use the system, advantages and drawbacks, preference for contextual or on-request help, and for the way the help was presented.

4 Results

Asking to colleagues and technicians represented the most common option used by half of the interviewees to seek for help and advice. The person to whom they asked for help was usually chosen on the basis of his/her level of expertise or on friendship/acquaintanceship. Google represented the first choice of help for four of the participants and the second choice for those participants who could not find a solution to their problems asking colleagues or technicians. Participants reported using online help and help menus rarely, and this was the first choice only for one interviewee. When asked which source of help was the most effective, eight participants indicated colleagues and technicians. Their choice was motivated by the fact that technicians are professional and helpful, and that providing support is part of their job. One participant indicated Google as the best source of information *"because you can use it at any time, also when you are at home"* (P10).

Seven participants reported a preference for automatic/contextual help rather than help on-request, but two of them also specified that this method works better for new or simple applications. Participants suggested that the automatic/help function should be customizable in order to be really useful. One participant provided an interesting observation about the function of automatic/contextual help:

> *"Automatic/contextual help has a double function: it appears when you need help and reminds you of potential errors; help on request covers only the first function"* (P10).

Participants provided valuable comments on the effectiveness and usefulness of common examples of contextual advice. People favoured less intrusive contextual advice, that do not try to guess the user's preferences or opinions, and that do not present risks for data security, such as Google's automatic word completion, pop-up reminders in Google Calendar, and the related videos sidebar in YouTube. Contextual advice was valued mainly in the case of objective suggestions (e.g., YouTube) but perceived as less accurate when it tries to enter users' private space (e.g., Facebook). When asked to formulate their nave theories about contextual help is generated, all the participants reported that they are created on the basis of the inserted keywords. One participant also made a distinction between general, or simple, and particular, or complex, suggestions:

> *"For simple queries, the system works on simple analogies with the inserted keywords; for more complex issues, the system does a matching with your personal characteristics (provided while registering to a service"* (P8).

Participants provided useful information about their attitude toward WIRE. When reading the positive scenario, participants recognized several similarities with their work practices and perceived the system as potentially very useful. Two participants expressed a common concern about the introduction of WIRE into their work practices and suggested that, in order to benefit of its potentialities, the use of WIRE should totally replace previous practices, without leaving space for overlapping. The Negative Scenario was also perceived as very plausible as it described well fears and frustrations that may emerge when something goes wrong dealing with new systems or procedures. In particular the interviewees stressed the need for a system which is well designed and thoroughly tested before being introduced into the work practice:

> *"I gave for granted that this technology was previously tested and approved by the central administration office. [...]. In the case of dealing with sensitive or financial matters, I would trust the system only if I am 100% sure that it is effective and functional"* (P7).

Participants were asked if they would be interested in using WIRE. Nine of the interviewees responded positively and one was openly skeptical stating that using WIRE would take the same time it takes doing the procedure manually (P1). Anyway, formal training was indicated by two participants as a fundamental prerequisite for adoption. Drivers to adoption were identified in better organization of work, optimization of time, reduction of errors, and sharing of procedures and methodologies with colleagues. Major obstacles were connected to loss of control over work processes, in the case that these were entirely completed in an automatic way:

> *"I would like to keep track of each step of the process; if everything is made automatically, the users misses the logic that stays behind the process"* (P4).

All the participants declared that the advice provided by WIRE in the scenarios would be effective in meeting their needs, as they were generated on the basis of past experience of colleagues that share the same work procedures and possibly the same difficulties. Interviewees showed a marked preference for contextual help (8 participants) over help on-request; two of them added that the possibility of personalizing the way suggestions are provided would be a very important feature in order to make help messages really effective. Help messages provided during the task were preferred to messages provided before the task by nine of the interviewees. One participant suggested that the two modalities could be combined:

> *"I can see the two modalities as complementary. At the beginning of the activity the system asks what your needs are in general; during the activity, pop-up windows provide you solutions when the system feels that you are stuck"* (P8).

5 Conclusion

End-users acknowledged that the idea of WIRE for providing assistance, which was derived from the experience of colleagues working in a similar context, was useful. However, issues related to trust, timing and usefulness of the advice still remained as concerns to the users. During the design of WIRE we will need to find new strategies to make its operations transparent e.g. showing users how the advice is generated and why a particular advice is suggested in a given context. Transparency will also help to build up the trust of the users to use a recommendation tool like WIRE. This is particularly important when people deal with the sensitive and financial issues. Personalization is another desired feature, which enables users to receive optimized advice based upon their expertise level. Helping users with the personalized advice can certainly reduce the barrier for adopting this kind of EUD tools to a larger end-user community.

The study provides support to the proposal of collaborative tailoring, discussed in [10], as often participants mentioned that their willingness to engage in EUD was mediated by having support from other people and technical help easily available to them. This help was meant not only to alleviate some of the technical difficulties they had to face during development but also to take the responsibility out of their hands, making them less accountable in case of software failures. Issues related to organizational regulations and corporate processes also emerged as barriers to general EUD uptake, as people often mentioned the need to have explicit approval from their manager as a fundamental step towards making them willing to explore new techniques and tools to automatize their work practices.

Acknowledgement. This work was partially supported by funds from the European Commission (project OMELETTE, contract no. 257635; project SERVFACE, contract no. 216699).

References

[1] Namoun, A., Nestler, T., De Angeli, A.: Conceptual and Usability Issues in the Composable Web of Software Services. Current Trends in Web Engineering - ICWE 2010 Workshops. LNCS, vol. 6385, pp. 396–407, Springer, Heidelberg (2010)

[2] Namoun A., Nestler T., De Angeli, A.: Service Composition for Non Programmers: Prospects, Problems, and Design Recommendations. ECOWS (2010)

[3] Cypher, I., Halbert, D.C., Kurlander, D., Lieberman, H., Maulsby, D., Myers, B.A., Turransky, A. (eds.): Watch what I do: Programming by Demonstration. MIT Press, Cambridge, MA, USA (1993)

[4] Van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow Patterns. Distributed and Parallel Databases. 14(3), 5–51 (July2003)

[5] Henneberger, M., Heinrich, B., Lautenbacher, F., Bauer, B.: Semantic-based Planning Process Models. In: MKWI'08, Munich, Germany (2008)

[6] Ngu, A., Carlson, M., Sheng, Q., Paik, H.: Semantic-Based Mashup of Composite Applications. IEEE Transactions on Services Computing. 3(1), 2–15 (2010)

[7] Greenshpan, O., Milo, T., Polyzotis, N.: Autocompletion for Mashups. In: Proc. VLDB Endow. 2, no. 1, pp. 538–549 (2009)

[8] Roy Chowdhury, S., Rodríguez, C., Daniel, F., Casati, F.: Wisdom-Aware Computing: On the Interactive Recommendation of Composition Knowledge. Proc. of WESOA (2010)

[9] Bdker, S.: Scenarios in user-centred design - setting the stage for reflection and action. Interacting with Computers. 13, 61–75 (2000)

[10] Wulf, V., Pipek, V., and Won, M.: Component-based tailorability: enabling highly flexible software applications, Int. Journal of Human-Computer Studies. 66(1), 1–22 (2008)

[11] Jeffrey Wong and Jason I. Hong. 2007. Making mashups with marmite: towards end-user programming for the web. In Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '07). ACM, New York, NY, USA, 1435–1444.

[12] De Angeli, A., Battocchi, A., Roy Chowdhury, S., Rodríguez, C., Daniel, F. and Casati, Fabio Conceptual Design and Evaluation of WIRE: A Wisdom-Aware EUD Tool. Technical Report DISI-11-353, Ingegneria e Scienza dell'Informazione, University of Trento.

Appendix J

Baya: Assisted Mashup Development as a Service

Baya: Assisted Mashup Development as a Service *

Soudip Roy Chowdhury Carlos Rodríguez Florian Daniel
Fabio Casati

Abstract

In this demonstration, we describe *Baya*, an extension of Yahoo! Pipes that *guides* and *speeds up* development by interactively recommending composition knowledge harvested from a repository of existing pipes. Composition knowledge is delivered in the form of *reusable mashup patterns*, which are retrieved and ranked on the fly while the developer models his own pipe (the mashup) and that are automatically weaved into his pipe model upon selection. Baya mines candidate patterns from pipe models available online and thereby leverages on the *knowledge of the crowd*, i.e., of other developers. Baya is an extension for the Firefox browser that seamlessly integrates with Pipes. It enhances Pipes with a powerful new feature for both *expert developers* and *beginners*, speeding up the former and enabling the latter. The discovery of composition knowledge is provided *as a service* and can easily be extended toward other modeling environments.

1 Introduction

Mashup tools, such as Yahoo! Pipes (`http://pipes.yahoo.com/pipes/`) or JackBe Presto Wires (`http://www.jackbe.com`), simplify the development of composite applications by means of easy development paradigms (e.g., using visual programming metaphors) and hosted runtime environments that do not require the installation of any client-side software. Yet, despite the initial goal of enabling end users to develop own applications and the advances in simplifying technology, mashup development is still a *complex task* that can only be managed by skilled developers.

*This is an electronic version of an Article published in WWW 2012 Companion, April 1620, 2012, Lyon, France. ACM 978-1-4503-1230-1/12/04. 2012 International World Wide Web Conference Committee

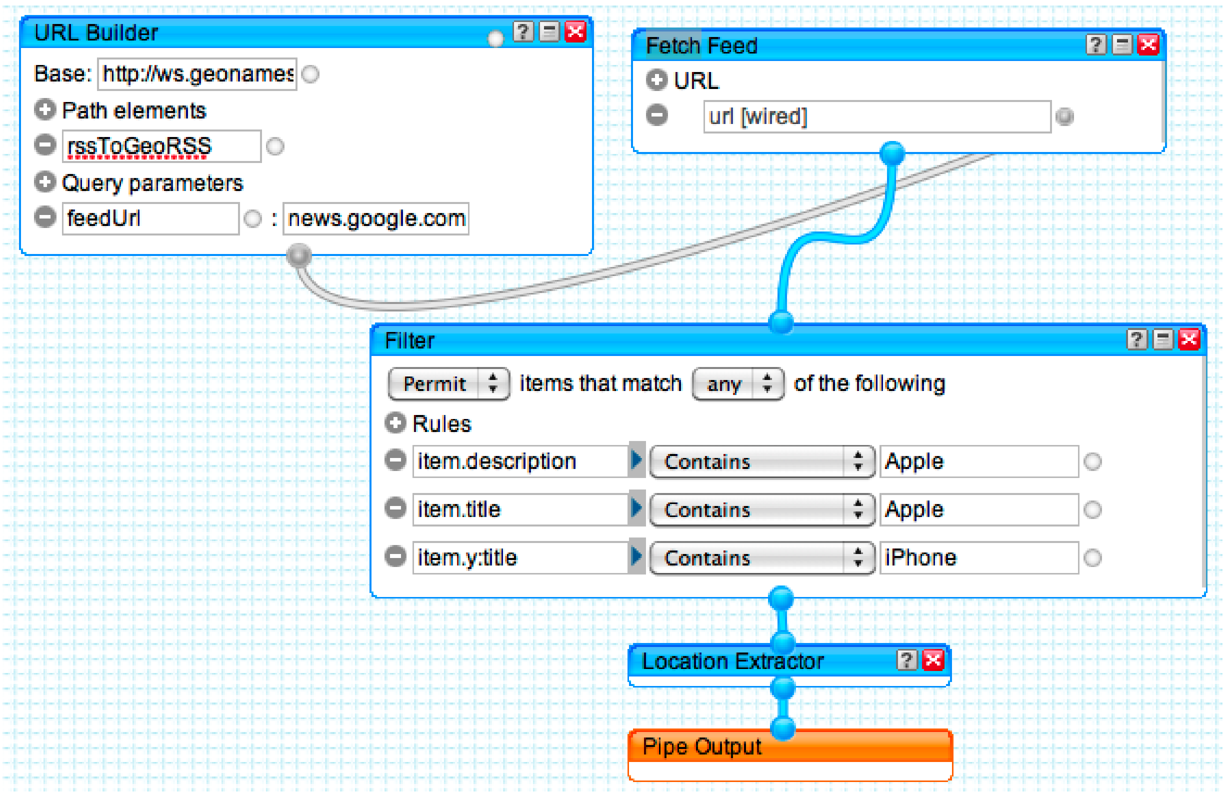

URL Builder
Base: http://ws.geonames
Path elements
rssToGeoRSS
Query parameters
feedUrl : news.google.com
Fetch Feed
URL
url [wired]
Filter
Permit items that match any of the following
Rules
item.description Contains Apple
item.title Contains Apple
item.y:title Contains iPhone
Location Extractor
Pipe Output

Figure 1: A typical pattern in Yahoo! Pipes

For instance, Figure 1 illustrates a Yahoo! Pipes model that encodes how to plot news items on a map. The example shows that understanding and modeling the logic for building such a mashup is ***neither trivial nor intuitive***. Firstly, we need to enrich the news feed with geo-coordinates, then, we must fetch the actual news items, and only then we can plot the items on a map. If modeling difficulties arise, it is common practice to manually search the Web for *examples* or *help* on which components to use, on how to fill the respective parameter fields, or on how to propagate data.

In order to aid less skilled developers in the design of mashups like the one above, in programming by demonstration [1], for instance, the system aims to auto-complete a process definition, starting from a set of user-selected model examples. Goal-oriented approaches [4] aim to assist the user by automatically deriving compositions that satisfy user-specified goals. Pattern-based development [3] aims at recommending connector patterns (so-called glue patterns) in response to user selected components (so-called mashlets) in order to autocomplete the partial mashup. Syntactic approaches [7] suggest modeling constructs based on syntactic similarity (comparing output and input data types), while semantic approaches [5] annotate constructs to support suggestions based on the meaning of constructs. The limitations in these approaches lie in the fact that they overlooked the perspectives for end user development, as they either still require advanced modeling skills (which users don't have), or they expect the user to specify complex rules for defining goals (which they are not able to), or they expect domain experts to specify and maintain the semantics of the modeling constructs (which they don't do).

Driven by a user study on how end users would like to be assisted during mashup development [2], we have developed ***Baya***, a plug-in for Yahoo! Pipes that provides *interactive, contextual recommendations of reusable composition knowledge*. The knowledge Baya recommends is re-usable *composition patterns*, i.e., model fragments that bear knowledge about how to compose mashups, such as the one in Figure 1. For instance, Baya may suggest a candidate next component or a whole chain of constructs. Upon selection of a recommendation, Baya weaves the respective pattern automatically into the current model in the modeling canvas[1]. Baya mines community composition knowledge from existing mashup models publicly available in the online Yahoo! Pipes repository and provides the respective patterns as a service to client-side modeling environments.

In this demo paper, we describe Baya, outline the concepts and architecture behind its simple user interface, and provide insight into its implementation and future evolution.

2 The Baya Approach

Baya aims to seamlessly extend existing mashup or composition instruments with advanced knowledge reuse capabilities. It targets both expert developers and beginners and aims to speed up the former and to enable the latter.

The ***design goals*** behind Baya can be summarized as follows: We didn't

[1]This is also the capability that inspired the name of the tool: the *Baya weaver* is a so-called weaverbird that weaves its nest with long strips of leaves.

want to develop *yet another* mashup environment; so we opted for an extension of existing and working solutions (in this demo, we focus on Yahoo! Pipes; other tools will follow). We wanted to *reuse* composition knowledge that has proven successful in the past; mining modeling patterns from existing mashups allows us to identify exactly this, i.e., recurrent modeling practice. We wanted to support a variety of *different* mashup tools, not just one; as we will see, the sensible design of a so-called canonical mashup model serves exactly this purpose. Modelers should not be required to *ask* for help; we therefore proactively and interactively recommend contextual composition patterns. We did not want the *reuse* to be limited to simple copy/paste of patterns, but knowledge should be *actionable*, and therefore, Baya features the automated weaving of patterns.

2.1 Composition Knowledge

Considering the typical actions performed by a developer in a graphical modeling environment (e.g., filling input fields, connecting components, copying/pasting model fragments), Baya specifically supports the following set of ***pattern types***:

- ***Parameter value pattern.*** The parameter value pattern represents a set of recurrent value assignments for the input parameters of a component. This pattern helps filling input parameters of a component that require explicit user input.
- ***Connector pattern.*** The connector pattern represents a recurrent connector between a pair of components, along with the data mapping of the target component. The pattern helps connecting a newly placed component to the partial mashup model in the canvas.
- ***Connector co-occurrence pattern.*** The connector co-occurrence pattern captures which connectors occur together. The pattern also includes the associated data mappings. This pattern is particularly valuable in those cases where people, rather than developing their mashup model

in an incremental but connected fashion, first select the desired functionalities (the components) and only then connect them.

- ***Component co-occurrence pattern.*** Similarly, the component co-occurrence pattern captures couples of components that occur together. It comes with the two associated components as well as with their connector, parameter values, and data mapping logic. The pattern helps developing mashups incrementally in a connected fashion.
- ***Component embedding pattern.*** The component embedding pattern captures which component is typically embedded into which other component, both being preceded by another component. The pattern helps, for instance, modeling *loops*, a task that is usually not trivial to non-experts.
- ***Multi-component pattern.*** The multi-component pattern represents recurrent model fragments that are composed of multiple components. It represents more complex patterns, such as the one in Figure 1, that are not yet captured by the other pattern types.

This list of pattern types is extensible and will evolve over time. However, this set of pattern types at the same time leverages on the interactive modeling paradigm of the mashup tools (the patterns represent modeling actions that could also be performed by the developer) and provides as much information as possible.

2.2 Discovery, Recommendation and Weaving

Figure 2 details the internals of the Baya architecture. The overall architecture is devided into two blocks, namely, the *recommendation server* and the *client-side extension* of the chosen mashup tool, i.e., Yahoo! Pipes.

The ***Baya recommendation server*** (at the left in Figure 2) is in charge of discovering and harvesting composition knowledge patterns from existing mashup compositions. The first step for discovering composition patterns consists in taking the *native models* of the target mashup tools from a repository

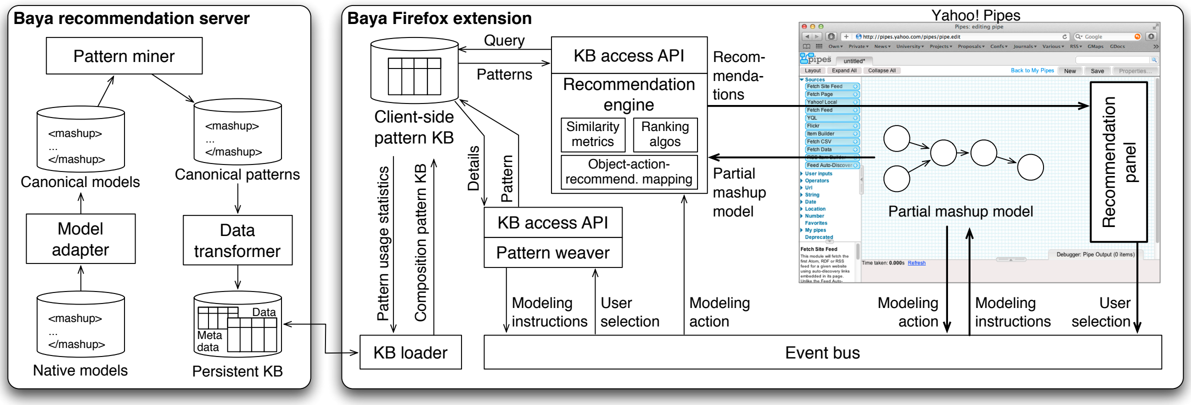


Figure 2: The internals of Baya: functional architecture for pattern discovery, recommendation and weaving

of existing compositions and translating them into a *canonical mashup model*, a step that is performed by a dedicated *model adapter*. The canonical model is able to represent a variety of similar mashup languages and allows the development of more generic mining algorithms. The *pattern miner* runs a set of *pattern mining algorithms* on the data in the canonical model and discovers the above introduced patterns. Discovered patterns are stored back into a database of *canonical patterns*, transformed by the *data transformer*, and loaded into the *persistent knowledge base* (KB). The persistent KB consists in a database that is structured in such a way that patterns can be efficiently queried and retrieved by the *client-side browser extension* for interactive recommendation.

The ***Baya Firefox extension*** consists of two main components: a recommendation engine and a pattern weaver. In the client, we have the actual interactive modeling environment (Pipes), in which the developer can visually compose components by dragging and dropping them from a component tool bar and connecting them together in the canvas. The developer therefore performs composition *actions* (e.g., select, drag, drop, connect, delete, fill, map,...), where the *action* is performed on a modeling construct in the modeling canvas; we call this construct the *object* of the action. For instance, we can *drop* a component onto the canvas, or we can *select* a parameter to fill it with a value, and so on. Upon each interaction, the *action* and its *object* are published on a browser-internal *event bus*, which forwards them to the ***recommendation engine***. Given a modeling *action*, the *object* it has been applied to, and the partial mashup model, the engine queries the *client-side pattern KB* via the *KB access API* for recommendations (pattern representations) and gets a list of candidate patterns. Baya uses both *exact* and *approximate* pattern matching algorithms [6] to determine the final candidate set of recommendations that also match the current composition context, ranks them in order of their similarity and popularity, and finally renders them in the *recommendation panel*.

Upon the selection of a pattern from the recommendation panel, the ***pattern weaver*** weaves it into the partial mashup model in the modeling canvas. For each supported pattern type, Baya retrieves a *basic weaving strategy* (a static set of modeling instructions; see `http://goo.gl/Xk7VF`), which is independent of the partial mashup model, and derives a *contextual weaving strategy*, which applies the basic strategy to the partial model at runtime. Applying the mashup operations in the basic strategy may require the resolution of possible *conflicts* among the constructs of the partial model and those of the pattern to be weaved. For instance, if we want to add a new component of type *ctype* but the mashup already contains an instance of type *ctype*, say *comp*, we are in the presence of a conflict: either we decide that we reuse *comp*, which is already there, or we decide to create a new instance of *ctype*. In order to choose how to proceed, Baya allows one to choose among different policies (see `http://goo.gl/9jJtK`). Given a final, contextual strategy, the pattern weaver applies the respective modeling actions to the partial mashup model.

Upon successful weaving of a recommended pattern into the partial composition, the *usage statistics* of the selected pattern in the client-side KB get updated, and simultaneously this information is sent to the server-side *persistent KB* via the *KB loader*. This updated metadata is used for future recommendation filtering and ranking. In the Baya client side, we also consider the option for saving patterns, in which users can select and store to the pattern KB new user-defined patterns from their current composition. This feature is part of our on-going development and will be available in future versions of Baya.

3 Implementation

Baya is implemented as Mozilla Firefox (`http://mozilla.com/firefox`) extension for Yahoo! Pipes, adding an interactive recommendation panel at the right of its modeling canvas. Baya implementation is based on JavaScript for the business logic (e.g., the algorithms) and XUL (XML User Interface Language, `https://developer.mozilla.org/En/XUL`) for UI development. The use of JavaScript in Firefox Extension development framework eases the interaction with the HTML DOM elements in the browser window and the implementation of dedicated listeners to intercept modeling events on elements in the DOM tree (e.g., model constructs in the Pipes modeling canvas). A screenshot of Baya in action is shown in Figure 3.

The server side is implemented in Java. This comprises the model adapter (cf. Figure 2), which is able to convert Yahoo! Pipes' internal JSON representation of mashups into our canonical mashup model as well as the necessary mining algorithms for the discovery of the patters (a description of the algorithms can be found at `http://goo.gl/Dis5V`). Parts of our mining algorithms make use of frequent itemset mining, for which we used the tool ARMiner (`http://www.cs.umb.edu/~laur/ARMiner/`).

Discovered patterns are transformed and stored in a knowledge base that is optimized for fast pattern retrieval at runtime. The implementation of the persistent pattern KB at server side, is based on MySQL (`http://www.mysql.com/`). Via a dedicated Java RESTful API, at startup of the recommendation panel the KB loader synchronizes the server-side KB with the client-side KB, which instead is based on SQLite (`http://www.sqlite.org`). The pattern matching and retrieval algorithms are implemented in JavaScript and triggered by events generated by the event listeners monitoring the DOM modifications related to the mashup model.

The weaving algorithms are also implemented in JavaScript. Upon the selection of a recommendation from the panel, they derive the contextual weaving strategy that is necessary to weave the respective pattern into the partial mashup model. Each of the instructions in the weaving strategy refers to a modeling action, where modeling actions are implemented as JavaScript ma-

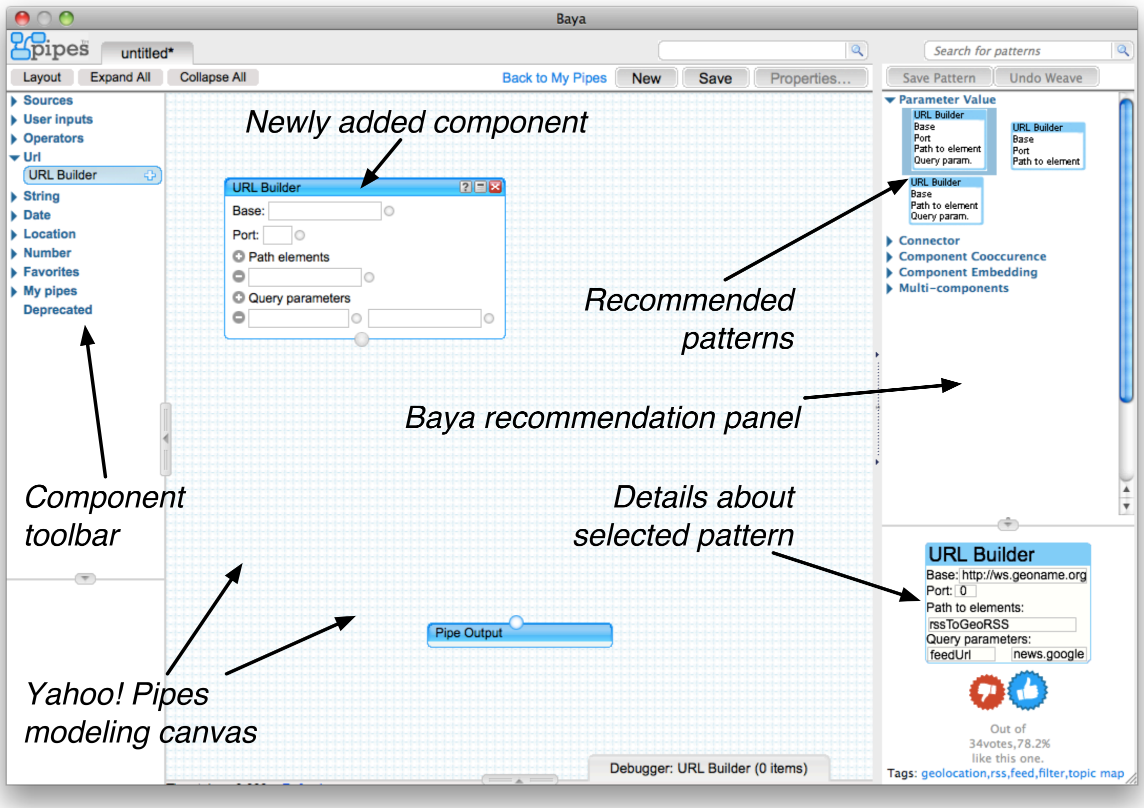


Figure 3: Screenshot of Baya in action.

nipulations of the mashup model's DOM elements. Both the weaving strategies (basic and contextual) are encoded as JSON arrays, which enable us to use the native `eval()` command for fast and easy parsing of the weaving logic.

For our experiments we extracted 303 pipes definitions from the repository of Pipes. The average numbers of components, connectors and input parameters were 12.7, 13.2 and 3.1, respectively, indicating fairly complex mashups. We were able to identify patterns of all the types described above. For example, the minimum/maximum support for the *connector patterns* was 0.0759/0.3234, while the one for the *component co-occurrence patterns* was 0.0769/0.2308. We used these patterns to populate our KB and generated additional synthetic patterns to test the performance of the recommendation engine (the sizes of the KBs ranged from 10, 30, 100, 300, 1000 multi-component patterns) [6]. The complexity of the patterns ranged from $3 - 9$ components per pattern, and we used queries with $1 - 7$ components. In the worst case scenario (KB of

1000 patterns, approximate similarity matching of patterns), the recommendation engine could retrieve relevant patterns within 608 millisecond – everything entirely inside the client browser.

4 Demonstration Storyboard

During the live demonstration, we will showcase Baya at work and take our audience through the theoretical as well as the usage aspects of the tool, using a mix of slides and hands-on examples. In particular, we intend to organize the demonstration as follows:

1. ***Introduction***: A short intro to the goals and key concepts of Baya.
2. ***Example***: A simple example developed by us with the use of the interactive recommendations.
3. ***Non-assisted development by audience***: A similar modeling exercise for a member of the audience, however without the help of the interactive recommender.
4. ***Assisted development by audience***: The same modeling scenario as in 3, this time however with the help of the interactive recommender.
5. ***Patterns and discovery***: An explanation of the pattern types supported by Baya, along with the mining approach underlying the pattern knowledge base.
6. ***Architecture and internals***: Explanation of the internal architecture of Baya and of the recommendation and weaving algorithms working behind the scenes.
7. ***Conclusion***: Lessons learned and outline of future works and the evolution of Baya.

This process will allow us to introduce the audience to Baya and help us evaluate the efficacy and usability of the tool. We hope we will get valuable feedback from the audience, in order to further fine-tune Baya's UI and algorithms.

An introduction to and a screencast of Baya is available at `http://www.youtube.com/watch?v=RNRAsX1CXtE`.

5 Status and Lessons Learned

Baya was born in the context of the EU research project OMELETTE, in order to assist mashup development inside the project's own mashup editors. Soon, however, we recognized that the kind of knowledge discovery algorithms we were working on and the conceptual approach to pattern recommendation and weaving are generic enough to be applied in the context of many other modeling or mashup tools. As a proof of concept, we therefore developed Baya, an apparently simple, yet effective tool. The idea of composition knowledge as a service makes it unique among other assisted development approaches, and a-priori definition of pattern structures allows us to extract meaningful knowledge also from single mashup models.

Next, we will extend the mining algorithms to other composition paradigms and develop dedicated clients for different composition tools. The idea is to make Baya publicly available and to study how effectively pattern recommendation and weaving can help users to develop own mashups.

Acknowledgment. This work was supported by the European Commission (project OMELETTE, contract 257635).

References

[1] A. Cypher, D. C. Halbert, D. Kurlander, H. Lieberman, D. Maulsby, B. A. Myers, and A. Turransky, editors. *Watch what I do: programming by demonstration.* MIT Press, Cambridge, MA, USA, 1993.

[2] A. De Angeli, A. Battocchi, S. Roy Chowdhury, C. Rodríguez, F. Daniel, and F. Casati. End-User Requirements for Wisdom-Aware EUD. In *IS-EUD'11*, pages 245–250.

[3] O. Greenshpan, T. Milo, and N. Polyzotis. Autocompletion for mashups. *VLDB'09*, 2:538–549.

[4] M. Henneberger, B. Heinrich, F. Lautenbacher, and B. Bauer. Semantic-Based Planning of Process Models. In *Multikonferenz Wirtschaftsinformatik'08*, 2008.

[5] A. Ngu, M. Carlson, Q. Sheng, and H. young Paik. Semantic-based mashup of composite applications. *IEEE TSC*, 3(1):2 –15, 2010.

[6] S. Roy Chowdhury, F. Daniel, and F. Casati. Efficient, Interactive Recommendation of Mashup Composition Knowledge. In *ICSOC'11*, pages 374–388, 2011.

[7] J. Wong and J. I. Hong. Making mashups with marmite: towards end-user programming for the web. In *CHI'07*, pages 1435–1444.

Appendix K

Discovery and Reuse of Composition Knowledge for Assisted Mashup Development

Discovery and Reuse of Composition Knowledge for Assisted Mashup Development *

Florian Daniel Carlos Rodríguez Soudip Roy Chowdhury
Hamid R. Motahari Nezhad Fabio Casati

Abstract

Despite the emergence of mashup tools like Yahoo! Pipes or JackBe Presto Wires, developing mashups is still *non-trivial* and requires intimate knowledge about the functionality of web APIs and services, their interfaces, parameter settings, data mappings, and so on. We aim to *assist the mashup process* and to turn it into an interactive co-creation process, in which one part of the solution comes from the developer and the other part from *reusable composition knowledge* that has proven successful in the past. We harvest composition knowledge from a repository of existing mashup models by mining a set of reusable *composition patterns*, which we then use to interactively provide *composition recommendations* to developers while they model their own mashup. Upon acceptance of a recommendation, the purposeful design of the respective pattern types allows us to *automatically weave* the chosen pattern into a partial mashup model, in practice performing a set of modeling actions on behalf of the developer. The experimental evaluation of our prototype implementation demonstrates that it is indeed possible to harvest meaningful, reusable knowledge from existing mashups, and that even complex recommendations can be efficiently queried and weaved also inside the client browser.

1 Introduction

Mashup tools, such as Yahoo! Pipes (`http://pipes.yahoo.com/pipes/`) or JackBe Presto Wires (`http://www.jackbe.com`), generally promise easy development tools and lightweight runtime environments, both typically running inside the client browser. By now, mashup tools undoubtedly simplified some complex composition tasks, such as the integration of web services or user

*Copyright is held by the author/owner(s). WWW 2012 Companion, April 1620, 2012, Lyon, France. ACM 978-1-4503-1230-1/12/04.

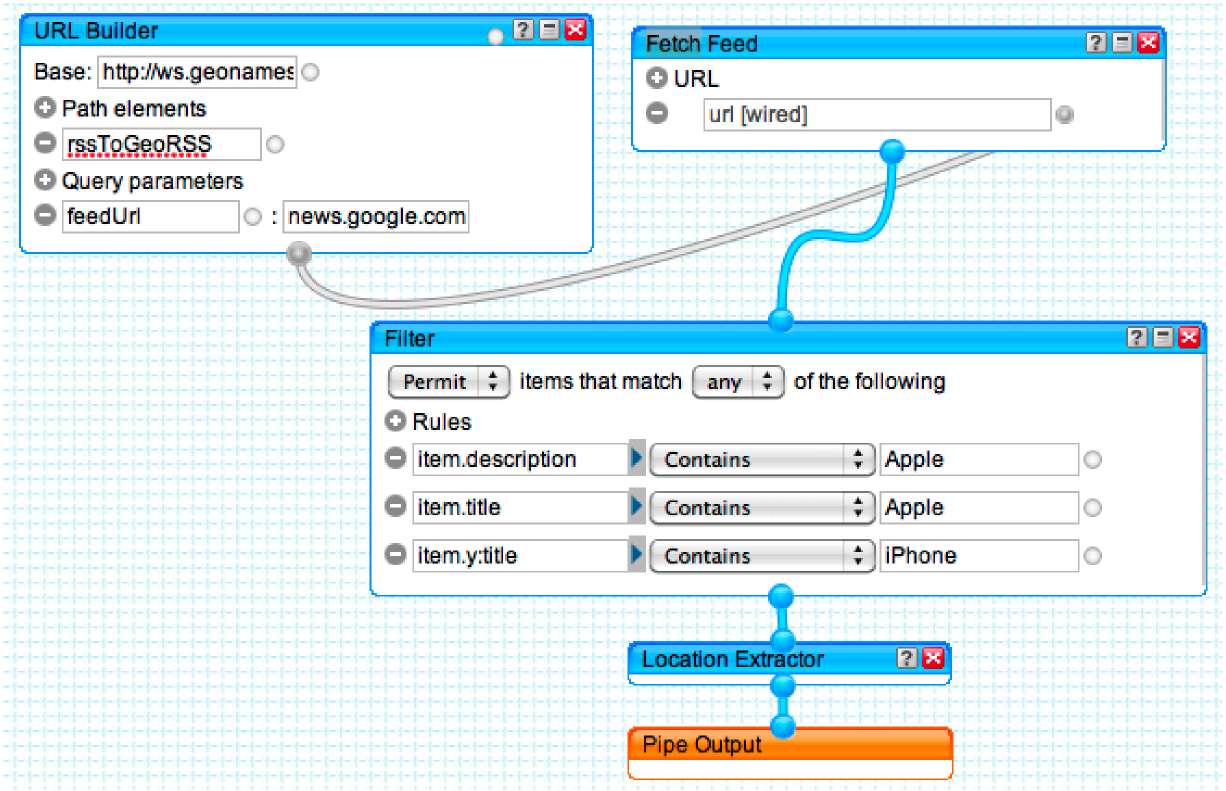

URL Builder
Base: http://ws.geonames
Path elements
rssToGeoRSS
Query parameters
feedUrl : news.google.com
Fetch Feed
URL
url [wired]
Filter
Permit items that match any of the following
Rules
item.description Contains Apple
item.title Contains Apple
item.y:title Contains iPhone
Location Extractor
Pipe Output

Figure 1: A typical pattern in Yahoo! Pipes

interfaces. Yet, despite these advances in simplifying technology, mashup development is still a *complex task* that can only be managed by skilled developers.

Figure 1 illustrates a Yahoo! Pipes model that encodes how to plot news items on a map. The lesson that can be learned from it is that plotting news onto a map requires enriching the news feed with geo-coordinates, fetching the actual news items, and handing the items over to the map. Understanding this logic is neither trivial nor intuitive.

In order to aid less skilled developers in the design of mashups like the one above, Carlson et al. [1], for instance, leverage on semantic annotations of components to recommend compatible components, given a component in the canvas. Greenshpan et al. [3] recommend components and connectors (so-called glue patterns) in response to the user providing a set of desired components. Elmeleegy et al. [2] recommend a set of components related to a component in the canvas, leveraging on conditional co-occurrence and semantic matching, and automatically plan how to connect selected components to the partial mashup.

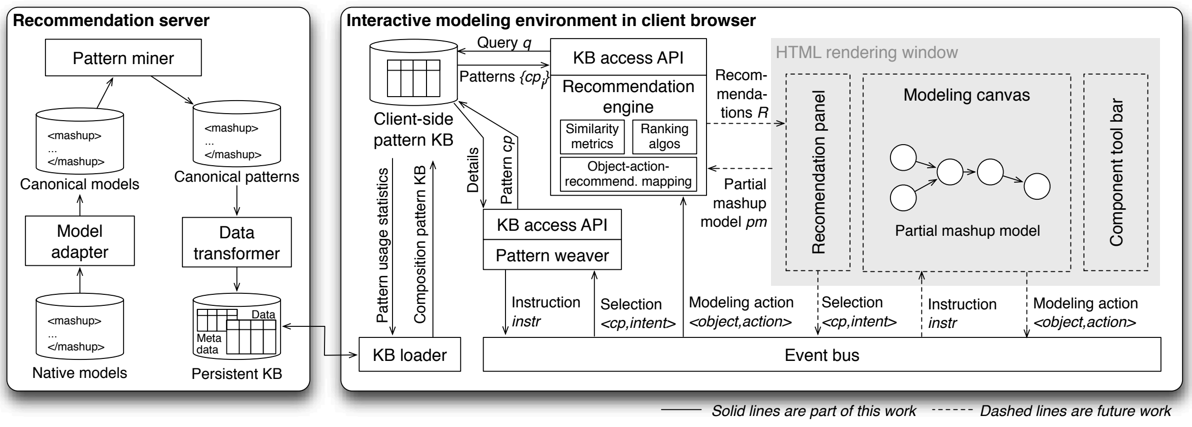


Figure 2: Functional architecture of the composition knowledge discovery and recommendation approach

Riabov et al. [4] allow users to express goals as keywords, in order to feed an automated planner that derives candidate mashups.

We assist the modeler in each step of his development task by means of ***interactive, contextual recommendations of composition knowledge***. The knowledge is re-usable *composition patterns*, i.e., fragments of mashup models. Such knowledge may come from a variety of possible sources; we specifically focus on community composition knowledge (recurrent model fragments in a mashup model repository). In this poster, we describe (i) how we mine *mashup composition patterns*, (ii) the *architecture* of our knowledge recommender, (iii) its *recommendation algorithms*, and (iv) its *pattern weaving algorithms* (automatically applying patterns to mashup models).

2 The Recommendation Platform

Figure 2 details our knowledge discovery and recommendation prototype. The ***pattern discovery*** logic is located in the server. After converting mashup models into a canonical format, the *pattern miner* extracts patterns, which we store into a knowledge base (KB) that is structured to minimize pattern retrieval at runtime. We support six composition pattern types: *parameter value*, *connector*, *connector co-occurrence*, *component co-occurrence* , *component embedding*, and *multi-component* patterns (cf. Figure 1).

The *interactive modeling environment* runs in the client. It is here where the ***pattern recommendation*** logic reacts to modeling *actions* performed by the modeler on a construct (the *object* of the action) in the canvas. For instance, we can *drop* a component onto the canvas, or we can *select* a parameter. Upon each interaction, the *action* and its *object* are published on a browser-internal *event bus*, which forwards them to the *recommendation engine*. With this information and the partial mashup model *pm* the engine queries the *client-side KB* for recommendations, where an *object-action-recommendation mapping* tells the engine which types of recommendations are to be retrieved. The list of patterns retrieved from the KB are then ranked and rendered in the *recommendation panel*.

Upon the selection of a pattern from the recommendation panel, the ***pattern weaver*** weaves it into the partial mashup model. The *pattern weaver* first retrieves a *basic weaving strategy* (a set of model-agnostic mashup instructions) and then derives a *contextual weaving strategy* (a set of model-specific instructions), which is used to weave the pattern. Deriving the contextual strategy from the basic one may require the resolution of possible *conflicts* among the constructs of the partial model and those of the pattern to be weaved. The pattern weaver resolves them according to a configurable conflict resolution policy.

Our ***prototype*** is a Mozilla Firefox extension for Yahoo! Pipes [6], with the recommendation and weaving algorithms implemented in JavaScript. Event listeners listen for DOM modifications, in order to identify mashup modeling actions inside the modeling canvas. The instructions in the weaving strategies refers to modeling actions, which are implemented as JavaScript manipulations of the mashup model's DOM elements. The server-side part is implemented in Java.

3 Evaluation

For our experiments we extracted 303 pipes definitions from the repository of Pipes. The average numbers of components, connectors and input parameters were 12.7, 13.2 and 3.1, respectively, indicating fairly complex mashups. We were able to identify patterns of all the types described above. For example, the minimum/maximum support for the *connector patterns* was 0.0759/0.3234, while the one for the *component co-occurrence patterns* was 0.0769/0.2308. We used these patterns to populate our KB and generated additional synthetic patterns to test the performance of the recommendation engine (the sizes of the KBs ranged from 10, 30, 100, 300, 1000 multi-component patterns) [5]. The complexity of the patterns ranged from $3 - 9$ components per pattern, and we used queries with $1 - 7$ components. In the worst case scenario (KB of 1000 patterns, approximate similarity matching of patterns), the recommendation engine could retrieve relevant patterns within 608 millisecond – everything entirely inside the client browser. The next step is going online and performing users studies.

Acknowledgment. This work was supported by the European Commission (project OMELETTE, contract 257635).

References

[1] M. P. Carlson, A. H. Ngu, R. Podorozhny, and L. Zeng. Automatic mash up of composite applications. In *ICSOC'08*, pages 317–330, 2008.

[2] H. Elmeleegy, A. Ivan, R. Akkiraju, and R. Goodwin. Mashup advisor: A recommendation tool for mashup development. In *ICWS'08*, pages 337–344, 2008.

[3] O. Greenshpan, T. Milo, and N. Polyzotis. Autocom- pletion for mashups. *VLDB'09*, 2:538–549, 2009.

[4] A. V. Riabov, E. Boillet, M. D. Feblowitz, Z. Liu, and A. Ranganathan. Wishful search: interactive composi- tion of data mashups. In *WWW'08*, pages 775–784, 2008.

[5] S. Roy Chowdhury, F. Daniel, and F. Casati. Efficient, Interactive Recommendation of Mashup Composition Knowledge. In *ICSOC'11*, pages 374–388, 2011.

[6] S. Roy Chowdhury, C. Rodríguez, F. Daniel, and F. Casati. Baya: Assisted Mashup Development as a Service. In *WWW'12*, 2012.

Appendix L

Assisted Mashup Development: On the Discovery and Recommendation of Mashup Composition Knowledge

Assisted Mashup Development: On the Discovery and Recommendation of Mashup Composition Knowledge *

Carlos Rodríguez Soudip Roy Chowdhury Florian Daniel
Hamid R. Motahari Nezhad Fabio Casati

Abstract

Over the past few years, mashup development has been made more accessible with tools such as Yahoo! Pipes that help in making the development task simpler through simplifying technologies. However, mashup development is still a difficult task that requires knowledge about the functionality of web APIs, parameter settings, data mappings, among other development efforts. In this work, we aim at assisting users in the mashup process by recommending development knowledge that comes in the form of *reusable composition knowledge*. This composition knowledge is harvested from a repository of existing mashup models by mining a set of *composition patterns*, which are then used for interactively providing composition recommendations while developing the mashup. When the user accepts a recommendation, it is automatically woven into the partial mashup model by applying modeling actions as if they were performed by the user. In order to demonstrate our approach we have implemented *Baya*, a Firefox plugin for Yahoo! Pipes that shows that it is indeed possible to harvest useful composition patterns from existing mashups, and that we are able to provide complex recommendations that can be automatically woven inside Yahoo! Pipes' web-based mashup editor.

1 Introduction

Mashup tools, such as Yahoo! Pipes (`http://pipes.yahoo.com/pipes/`) or JackBe Presto Wires (`http://www.jackbe.com`), generally promise easy development tools and lightweight runtime environments, both typically running inside the client browser. By now, mashup tools undoubtedly simplified some complex composition tasks, such as the integration of web services or user

*The final publication will be available at www.springerlink.com.

interfaces. Yet, despite these advances in simplifying technology, mashup development is still a *complex task* that can only be managed by skilled developers.

People without the necessary programming experience may not be able to profitably use mashup tools like Pipes — to their dissatisfaction. For instance, we think of ***tech-savvy people***, who like exploring software features, authoring and sharing own content on the Web, that would like to mash up other contents in new ways, but that don't have programming skills. They might lack appropriate awareness of which composable elements a tool provides, of their specific functionality, of how to combine them, of how to propagate data, and so on. In short, these are people that do not have software development knowledge. The problem is analogous in the context of web service composition (e.g., with BPEL) or business process modeling (e.g., with BPMN), where modelers are typically more skilled, but still may not know all the features or typical modeling patterns of their tools.

What people (also programmers) typically do when they don't know how to solve a tricky modeling problem is searching for ***help***, e.g., by asking more skilled friends or by querying the Web for solutions to analogous problems. In this latter case, examples of ready mashup models are one of the most effective pieces of information – provided that suitable examples can be found, i.e., examples that have an analogy with the modeling situation faced by the modeler. Yet, searching for help does not always lead to success, and retrieved information is only seldom immediately usable as is, since the retrieved pieces of information are not contextual, i.e., immediately applicable to the given modeling problem.

For instance, Figure 1 illustrates a Yahoo! Pipes model that encodes how to plot news items on a map. Besides showing how to connect components and fill parameters, the key lesson that can be learned from this pipe is that plotting news onto a map requires first enriching the news feed with geo-coordinates, then fetching the actual news items, and only then handing the items over to the map. Understanding this logic is neither trivial nor intuitive.

Driven by a user study on how end users imagine assistance during mashup development [4], we aim to automatically offer them help pro-actively and interactively. Specifically, we are working toward the ***interactive, contextual***

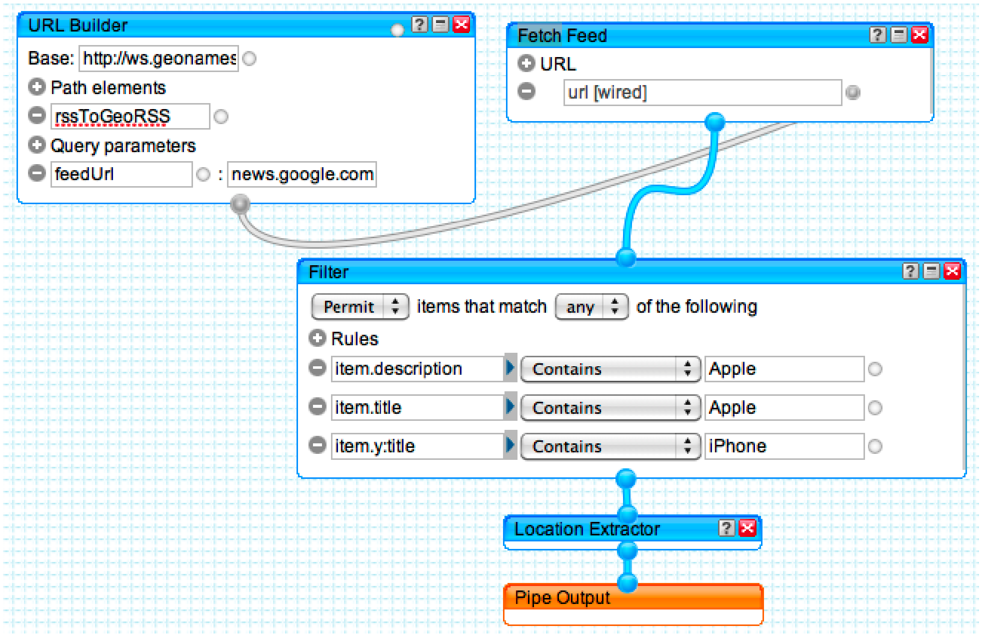


Figure 1: A typical pattern in Yahoo! Pipes

recommendation of reusable composition knowledge, in order to assist the modeler in each step of his development task, e.g., by suggesting a candidate next component or a whole chain of tasks. The knowledge we want to recommend is re-usable *composition patterns*, i.e., model fragments that bear knowledge about how to compose mashups, such as the pattern in Figure 1. Such knowledge may come from a variety of possible sources. In this work, we specifically focus on community composition knowledge and mine recurrent model fragments from a repository of given mashup models.

The ***vision*** is that of enabling the development of assisted, web-based mashup environments that deliver composition knowledge much like Google's Instant feature delivers search results already while still typing keywords into the search field.

In this chapter, we approach two core ***challenges*** of this vision, i.e., the *discovery* of reusable composition knowledge from a repository of ready mashup models and the *reuse* of such knowledge inside mashup tools, a feature that we call *weaving*. Together with the ability to search and retrieve composition patterns contextually when modeling a new mashup, a problem we approached in [10] and that we summarize in this chapter, these two features represent the key enablers of the vision of assisted development. We specifically provide the following ***contributions***:

- We describe a *canonical mashup model* that is able to represent in a single modeling formalism a variety of data flow mashup languages. The goal is to mine composition knowledge from multiple source languages by implementing the necessary algorithms only once.
- Based on our canonical mashup model, we define a set of *mashup pattern types* that resemble the modeling actions of typical mashup environments.
- We describe an *architecture* of our knowledge recommender that can be used to equip any mashup environment with interactive assistance for its developers.
- We develop a set of *data mining algorithms* that discover composition knowledge in the form of reusable mashup patterns from a repository of mashup models.
- We present our *pattern recommendation* and *pattern weaving* algorithms. The former aims at recommending composition patterns based on the user actions on the design canvas. The later aims at automatically appying patterns to mashup models, allowing the developer to progress in his development task.

In the next section, we start by introducing the canonical mashup model, which will help us to formulate our problem statement, define mashup pattern types and describe our pattern mining algorithms. Section 3 is where we describe the types of mashup patterns we are interested in and the architecture of our recommendation platform. In Sections 4, 5 and 6 we, respectively, describe in details the mining, recommendation, and weaving algorithms. Section 7 presents the details of the implementation of our approach. In Section 8 we overview related work. Then, with Section 9, we conclude the chapter.

2 Preliminaries and Problem

The development of a data mining algorithm strongly depends on the data to be mined. The data in our case are the mashup models. Since in our work we do not only aim at the reuse of knowledge but also at the reuse of our algorithms across different platforms, we strive for the development of algorithms that are able to accommodate different mashup models in input. Next, we therefore describe a ***canonical mashup model*** that allows us to concisely express multiple data mashup models and to implement mining algorithms that intrinsically support multiple mashup platforms. The canonical model is not meant to be executed; it rather serves as description format.

As a first step toward generic modeling environments, in this chapter we focus on data flow based mashup models. Although relatively simple, they are the basis of a significant number of mashup environments, and the approach can easily be extended toward other mashup environments.

2.1 A Canonical Mashup Model

Let CT be a set of ***component types*** of the form $ctype = \langle type, IP, IN, OP, OUT, is\_embedding\rangle$, where $type$ identifies the type of component (e.g., RSS feed, filter, or similar), IP is the set of input ports of the component type (for the specification of data flows), IN is the set of input parameters of the component type, OP is the set of output ports, OUT is the set of output attributes[1], and $is\_embedding \in \{yes, no\}$ tells whether the component type allows the embedding of components or not (e.g., to model a loop). We distinguish three types of components:

- *Source* components fetch data from the web (e.g., from an RSS feed) or the local machine (e.g., from a spreadsheet), or they collect user inputs at runtime. They don't have input ports, i.e., $IP = \emptyset$.
- *Data processing* components consume data in input and produce processed data in output. Therefore: IP, $OP \neq \emptyset$. Filter components,

[1]We use the term *attribute* to denote data attributes produced as output by a component or flowing through a data flow connector and the term *parameter* to denote input parameters of a component.

operators, and data transformers are examples of data processing components.

- *Sink* components publish the output of a mashup, e.g., by printing it onto the screen (e.g., a pie chart) or providing an API toward it, such as an RSS or RESTful resource. Sinks don't have outputs, i.e., $OP = \emptyset$.

Given a set of component types, we are able to instantiate components in a modeling canvas and to compose mashups. We express the respective ***canonical mashup model*** as a tuple $m = \langle name, id, src, C, GP, DF, RES \rangle$, where $name$ is the name of the mashup in the canonical representation, id a unique identifier, $src \in \{$"$Pipes$", "$Wires$", "$myCocktail$", ...$\}$ keeps track of the source platform of the mashup, C is the set of components, GP is a set of global parameters, DF is a set of data flow connectors propagating data among components, and RES is a set of result parameters of the mashup. Specifically:

- $GP = \{gp_i | gp_i = \langle name_i, value_i \rangle\}$ is a set of ***global parameters*** that can be consumed by components, $name_i$ is the name of a given parameter, $value_i \in (STR \cup NUM \cup \{null\})$ is its value, with STR and NUM representing the sets of possible string or numeric values, respectively. The use of global parameters inside data flow languages is not very common, yet tools like Presto Wires or myCocktail (`http://www.ict-romulus.eu/web/mycocktail`) support the design-time definition of globally reusable variables.

- $DF = \{df_j | df_j = \langle srccid_j, srcop_j, tgtcid_j, tgtip_j \rangle\}$ is a set of ***data flow connectors*** that, each, assign the output port $srcop_j$ of a source component with identifier $srccid_j$ to an input port $tgtip_j$ of a target component identified by $tgtcid_j$, such that $srccid \neq tgtcid$. Source components don't have connectors in input; sink components don't have connectors in output.

- $C = \{c_k | c_k = \langle name_k, id_k, type_k, IP_k, IN_k, DM_k, VA_k, OP_k, OUT_k, E_k \rangle\}$ is the set of ***components***, such that $c_k =$

$instanceOf(ctype)$[2], $ctype \in CT$ and $name_k$ is the name of the component in the mashup (e.g., its label), id_k uniquely identifies the component, $type_k = ctype.type$[3], $IP_k = ctype.IP$, $IN_k = ctype.IN$, $OP_k = ctype.OP$, $OUT_k = ctype.OUT$, and:

- $DM_k \subseteq IN_k \times (\bigcup_{ip \in IP_k} ip.source.OUT)$ is the set of ***data mappings*** that map attributes of the input data flows of c_k to input parameters of c_k.
- $VA_k \subseteq IN_k \times (STR \cup NUM \cup GP)$ is the set of ***value assignments*** for the input parameters of c_k; values are either filled manually or taken from global parameters.
- $E_k = \{cid_{kl}\}$ is the set of identifiers of the ***embedded components***. If the component does not support embedded components, $E_k = \emptyset$.

- $RES \subseteq \bigcup_{c \in C} c.OUT$ is the set of ***mashup outputs*** computed by the mashup.

Without loss of generality, throughout this chapter we exemplify our ideas and solutions in the context of Yahoo! Pipes, which is well known and comes with a large body of readily available mashup models that we can analyze. Pipes is very similar to our canonical mashup model, with two key differences: it does not have global parameters, and the outputs of the mashup are specified by using a dedicated *Pipe Output* component (see Figure 1). Hence, $GP, RES = \emptyset$ and a pipe corresponds to a restricted canonical mashup of the form $m = \langle name, id, \text{“}Pipes\text{”}, C, \emptyset, DF, \emptyset\rangle$ with the attributes as specified above. In general, we refer to the generic canonical model; we explicitly state where instead we use the restricted Pipes model.

[2]To keep models and algorithms simple, we opt for a *self-describing* instance model for components, which presents both type and instance properties.

[3]We use a *dot notation* to refer to sub-elements of structured elements; *ctype.type* therefore refers to the *type* attribute of the component type *ctype*.

2.2 Problem Statement

Given the above canonical mashup model, the problem we want to address in this chapter is understanding (i) which *kind of knowledge* can be extracted from the canonical mashup model so as to automatically assist users in developing their mashups, (ii) what *algorithms* we need to develop in order to be able to discover such knowledge from existing mashup models, (iii) how to *interactively recommend* discovered patterns inside mashup tools in order to guide users with the next modeling step/s and (iv) how to automatically apply (*weave*) the selected recommendation inside the current mashup design.

3 Approach

The current trend in modeling environments in general, and in mashup tools in particular, is toward intuitive, web-based solutions. The key principles of our work are therefore to conceive solutions that *resemble the modeling paradigm* of graphical modeling tools, to develop them so that they can *run inside the client browser*, and to specifically *tune their performance* so that they do not annoy the developer while modeling. These principles affect the nature of the knowledge we are interested in and the architecture and implementation of the respective recommendation infrastructure.

3.1 Composition Knowledge Patterns

Starting from the canonical mashup model, we define composition knowledge as reusable ***composition patterns*** for mashups of type m, i.e., model fragments that provide insight into how to solve specific modeling problems, such as the one illustrated in Figure 1. In general, we are in the presence of a set of composition pattern types PT, where each pattern type is of the form $ptype = \langle C, GP, DF, RES \rangle$, where C, GP, DF, RES are as defined for m.

The size of a pattern may vary from a single component with a value assignment for one input parameter to an entire, executable mashup. The most ***basic patterns*** are those that represent a co-occurrence of two elements out of C, GP, DF or RES. For instance, two components that recur often together form a basic pattern; given one of the components, we are able to recommend the other component. Similarly, an input parameter plus its value form a basic pattern, given the parameter, we can recommend a possible value for it. As such, the most basic patterns are similar to *association rules*, which, given one piece of information, are able to suggest another piece of information.

Aiming, however, to help a developer refine his mashup model step by step with as less own effort as possible, we are able to identify a set of pattern types that allow the developer to obtain more practical and meaningful composition knowledge. Such knowledge is represented by sensible combinations of basic patterns, i.e., by ***composite patterns***.

Considering the typical modeling steps performed by a developer (e.g., filling input fields, connecting components, copying/pasting model fragments), we specifically identify the following set PT of ***pattern types***:

Parameter value pattern. The parameter value pattern represents a set of recurrent value assignments VA for the input fields IN of a component c:

$ptype^{par} = \langle\{c\}, GP, \emptyset, \emptyset\rangle$;
$c = \langle name, 0, type, \emptyset, IN, \emptyset, \emptyset, VA, \emptyset, \emptyset\rangle$[4];
$GP \neq \emptyset$ if VA also assigns global parameters to IN;
$GP = \emptyset$ if VA assigns only strings or numeric constants.

This pattern helps filling input fields of a component that require explicit user input.

Connector pattern. The connector pattern represents a recurrent connector df_{xy}, given two components c_x and c_y, along with the respective data mapping DM_y of the output attributes OUT_x to the input parameters IN_y:

$ptype^{con} = \langle\{c_x, c_y\}, \emptyset, \{df_{xy}\}, \emptyset\rangle$;
$c_x = \langle name_x, 0, type_x, \emptyset, \emptyset, \emptyset, \emptyset, \{op_x\}, OUT_x, \emptyset\rangle$;
$c_y = \langle name_y, 1, type_y, \{ip_y\}, IN_y, DM_y, \emptyset, \emptyset, \emptyset, \emptyset\rangle$.

This pattern helps connecting a newly placed component to the partial mashup model in the canvas.

[4]The identifier $c.id = 0$ does not represent recurrent information. Identifiers in patterns rather represent internal, system-generated information that is necessary to correctly maintain the structure of patterns. When mining patterns, the actual identifiers are lost; when weaving patterns, they need to be re-generated in the target mashup model.

Connector co-occurrence pattern. The connector co-occurrence pattern captures which connectors df_{xy} and df_{yz} occur together, also including their data mappings:

$ptype^{coo} = \langle \{c_x, c_y, c_z\}, \emptyset, \{df_{xy}, df_{yz}\}, \emptyset \rangle;$
$c_x = \langle name_x, 0, type_x, \emptyset, \emptyset, \emptyset, \emptyset, \{op_x\}, OUT_x, \emptyset \rangle;$
$c_y = \langle name_y, 1, type_y, \{ip_y\}, IN_y, DM_y, \emptyset, \{op_y\},$
$OUT_y, \emptyset \rangle$
$c_z = \langle name_z, 2, type_z, \{ip_z\}, IN_z, DM_z, \emptyset, \emptyset, \emptyset, \emptyset \rangle.$

This pattern helps connecting components. It is particularly valuable in those cases where people, rather than developing their mashup model in an incremental but connected fashion, proceed by first selecting the desired functionalities (the components) and only then by connecting them.

Component co-occurrence pattern. Similarly, the component co-occurrence pattern captures couples of components that occur together. It comes with two components c_x and c_y as well as with their connector, global parameters, parameter values, and c_y's data mapping logic:

$ptype^{com} = \langle \{c_x, c_y\}, GP, \{df_{xy}\}, \emptyset \rangle;$
$c_x = \langle name_x, 0, type_x, \emptyset, IN_x, \{op_x\}, OUT_x, VA_x, \emptyset, \emptyset \rangle;$
$c_y = \langle name_y, 1, type_y, \{ip_y\}, IN_y, DM_y, VA_y, \emptyset, \emptyset, \emptyset \rangle.$

This pattern helps developing a mashup model incrementally, producing at each step a connected mashup model.

Component embedding pattern. The component embedding pattern captures which component c_z is typically embedded into a component c_y preceded by a component c_x. The pattern has three components, in that both the embedded and the embedding component have access to the outputs of the preceding component. How these outputs are jointly used is valuable information. The pattern, hence, contains the three components with their connectors, data mappings, global parameters, and parameter values:

$ptype^{emb} = \langle \{c_x, c_y, c_z\}, GP, \{df_{xy}, df_{xz}, df_{zy}\}, \emptyset \rangle;$
$c_x = \langle name_x, 0, type_x, \emptyset, \emptyset, \{op_x\}, OUT_x, \emptyset, \emptyset, \emptyset \rangle;$
$c_y = \langle name_y, 1, type_y, \{ip_y\}, IN_y, DM_y, VA_y, \emptyset, \emptyset, \emptyset \rangle;$
$c_z = \langle name_z, 2, type_z, \{ip_z\}, IN_z, DM_z, VA_z, \{op_z\},$

$OUT_z, \emptyset\rangle$.

This pattern helps, for instance, modeling cycles, a task that is usually not trivial to non-experts.

Multi-component pattern. The multi-component pattern represents recurrent model fragments that are generically composed of multiple components. It represents more complex patterns, such as the one in Figure 1, that are not yet captured by the other pattern types alone. It allows us to obtain a full model fragment, given any of its sub-elements, typically, a set of components or connectors:

$ptype^{mul} = \langle C, GP, DF, RES\rangle$;
$C = \{c_i | c_i.id = i; i = 0, 1, 2, ...\}$.

Besides providing significant modeling support, this pattern helps understanding domain knowledge and best practices as well as keeping agreed-upon modeling conventions.

This list of pattern types is extensible, and what actually matters is the way we specify and process them. However, this set of pattern types, at the same time, leverages on the interactive modeling paradigm of the mashup tools (the patterns represent modeling actions that could also be performed by the developer) and provides as much information as possible (we do not only tell simple associations of constructs, but also show how these are used together in terms of connectors, parameter values, and data mappings).

Given a set of pattern types, an actual pattern can therefore be seen as an ***instance*** of any of these types. We model a composition pattern as $cp = instanceOf(ptype)$, $ptype \in PT$, where $cp = \langle type, src, C, GP, DF, RES, usage, date\rangle$, $type \in \{$"Par", "Con", "Coo", "Com", "Emb", "Mul"$\}$, $src \in \{$"$Pipes$", "$Wires$", "$myCockail$", ...$\}$ specifies the target platform of the pattern, C, GP, DF, RES, src are as defined for the pattern's *ptype*, *usage* counts how many times the pattern has been used (e.g., to compute rankings), and *date* is the creation date of the pattern.

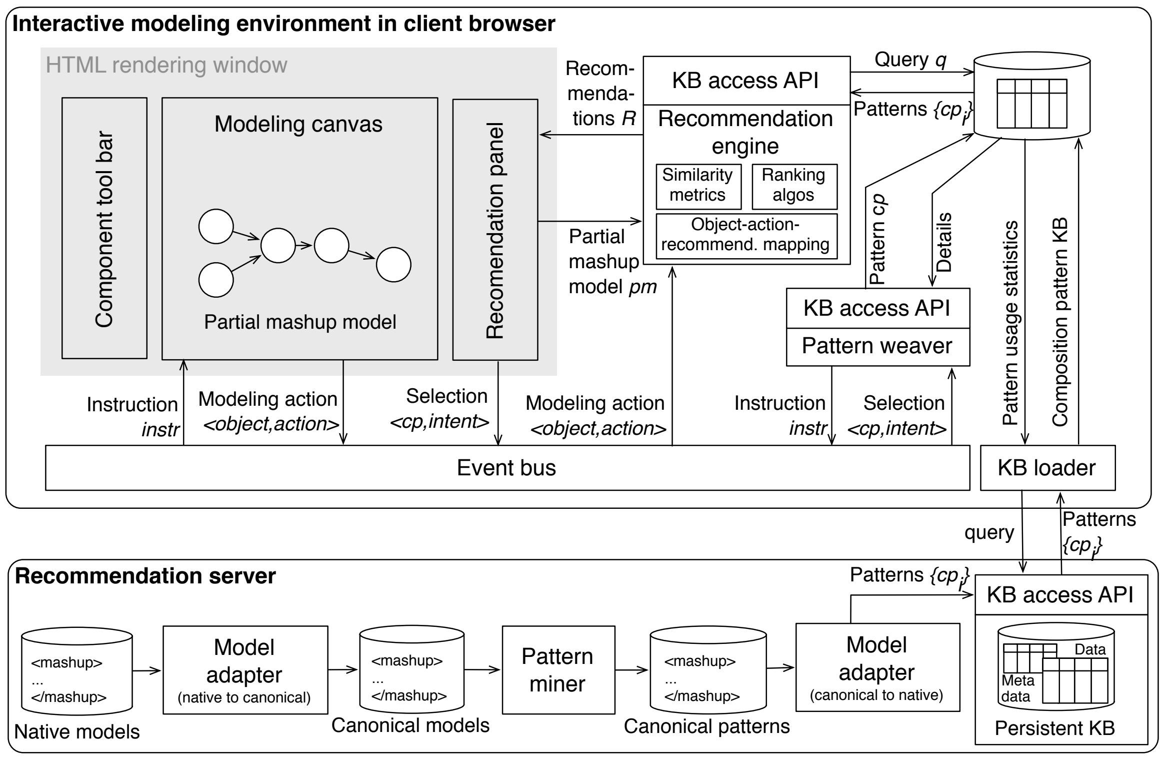


Figure 2: Functional architecture of the composition knowledge discovery and recommendation approach.

3.2 Architecture

Figure 2 details the internals of our knowledge discovery and recommendation prototype. We distinguish between client and server side, where the discovery logic is located in the server and the recommendation and weaving logic resides in the client. In the *recommendation server*, a *model adapter* imports the native mashup models into the canonical format. The *pattern miner* then extracts reusable composition knowledge in the form of composition patterns, which is then handed to a second *model adapter* to convert the canonical patterns into native patterns and load them into a knowledge base (KB). This KB is structured to maximize the performance of pattern retrieval at runtime.

In the client, we have the *interactive modeling environment*, in which the developer can visually compose components (in the *modeling canvas*) taken from the *component tool bar*. It is here where patterns are queried for and delivered in response to modeling actions performed by the modeler in the modeling canvas. In visual modeling environments, we typically have $action \in \{``select", ``drag", ``drop", ``connect", ``delete", ``fill", ``map", ...\}$, where the *action* is performed on a modeling construct in the canvas; we call this construct the *object* of the action. For instance, we can *drop* a component onto the canvas, or we can *select* a parameter to fill it with a value, we can *connect* a data flow with a target component, or we can *select* a set of components and connectors. Upon each interaction, the *action* and its *object* are published on a browser-internal *event bus*, which forwards them to the *recommendation engine*. Given a modeling *action*, the *object* it has been applied to, and the partial mashup model *pm*, the engine queries the *client-side pattern KB* via the *KB access API* for recommendations (pattern representations). An *object-action-recommendation mapping* (*OAR*) tells the engine which types of recommendations are to be retrieved for each modeling action on a given object (for example, when selecting an input field, only recommending possible values makes sense). The client-side KB is filled at startup by the *KB loader*, which loads the available patterns into the client environment, decoupling the knowledge recommender from the server side.

The list of patterns retrieved from the KB (either via regular queries or by applying dedicated similarity criteria) are then ranked by the engine and rendered in the *recommendation panel*, which renders the recommendations to the developer for inspection. Selecting a recommendation enacts the *pattern weaver*, which queries the KB for the usage details of the pattern (data mappings and value assignments) and generates a set of modeling instructions that emulate user interactions inside the modeling canvas and thereby weave the pattern into the partial mashup model.

4 Discovering Patterns

The first step in the information flow described in the above architecture is the discovery of mashup patterns from canonical mashup models. To this end, we look into the details of each individual pattern and implement dedicated mining algorithms for each of them, which allow us to fine-tune each mashup-specific characteristic (e.g., to treat threshold values for parameter value assignments and data mappings differently). The pattern mining algorithms make use of standard statistics as well as frequent itemset and subgraph mining algorithms [13].

4.1 Mining algorithms

For each of the pattern types identified in Section 3.1, we have implemented a respective pattern mining algorithm, the details of which we provide in the following.

Parameter value pattern. In the case of the parameter value pattern, we are interested in finding suitable values for the input fields in a given component. Most of the components in mashup compositions contain more than one parameter and more often than not the values of these parameters are related to one another and therefore we need take into account the co-occurrence of parameter values. In order to discover such co-occurrences, we map this problem to the well-known problem of itemset mining [13]. Algorithm 1 outlines the approach for finding parameter value patterns. Here, we first get all component instances from the mashups in the mashup repository (line 2) and group them together by their type (line 5-6) and then perform the parameter value pattern mining by component type (line 7). Finally, we construct the actual set of patterns that consists in tuples $\langle ct, VA\rangle$, where ct represents a component type and VA represents the value assignment for its parameters.

Algorithm 1: mineParameterValues

```
Data: repository of mashup compositions $M$ and minimun support ($minsupp_{par}$) for the frequent itemset mining
Result: set of parameter value patterns $\langle ct, VA\rangle$.
1  $Patterns$ = set();
2  $C$ = set of component instances in $M$;
3  $CT$ = array();
4  $Patterns$ = set();
5  foreach type of component ct in C do
6      $CT[ct] = c_x.VA$ with $c_x \in C$ such that $c_x.type = ct$ ;    // get all the parameter value assignments of
       component instances of type ct
7      $FI$ = mineFrequentItemsets($CT[ct], minsupp_{par}$);
8      foreach $VA \in FI$ do
9          $Patterns = Patterns \cup \{\langle ct, VA\rangle\}$;
10 return $Patterns$;
```

Connector pattern. A connector pattern is composed of two components, the source component c_x and the target component c_y, their data flow connector df_{xy}, and the data mapping DM_y of the target component. Given a repository of mashup models $M = \{m_i\}$ and the minimum support levels for the data flow connectors and data mappings, the pseudo-code in Algorithm 2 shows how we mine connector patterns.

We start the mining task by getting the list of all recurrent connectors in M (line 1). The respective function *getRecurrentConnectors* is explained in Algorithm 3; in essence, it computes a recurrence distribution for all connectors and returns only those that exceed the threshold $minsupp_{df}$. The function returns a set of connector types without repetitions and without information about the instances that generated them. Given this set, we construct a database of concrete instances of each connector type (using the *getConnectorInstances* function in line 5 and described in Algorithm 4) and, for each connector type, derive a database of the data mappings for the connectors' target component c_y (lines 7-9). We feed the so constructed database into a standard *mineFrequentItemsets* function [13], in order to obtain a set of recurrent data mappings for each connector type. Finally, for each identified data mapping DM_y, we construct a tuple $\langle df_{xy}, DM_y\rangle$ (lines 11-12), which concisely represents the connector pattern structure introduced in Section 3.1; the rest of the pattern comes from the component definitions.

Connector co-occurrence pattern. The *connector pattern* introduced previously is about how pairs of components are connected together. *The connector co-occurrence pattern* goes a step further: it tells how connectors between different pairs of components co-occur together in compositions and how data mappings are defined for them. Algorithm 5 presents the logic for computing *connector co-occurrence patterns*. The main difference with respect to Algorithm 2 is that, instead of computing the frequency of individual dataflow connectors between pairs of components, we compute frequent itemsets of dataflow connectors (lines 2-4).

Component co-occurrence pattern. The component co-occurrence pattern is an extension of the connector pattern; in addition to the connectors and data mappings, it also contains the parameter value assignments of the

Algorithm 2: mineConnectors

```
Data: repository of mashup models $M$, minimum support of data flow connectors ($minsupp_{df}$) and data mappings
      ($minsupp_{dm}$)
Result: set of connectors with their corresponding data mappings $\{\langle df_{xy,i}, DM_{y,i}\rangle\}$
1  $F_{df}$ = getRecurrentConnectors($M, minsupp_{df}$);
2  $DB$ = array();                                   // database of recurrent connector instances
3  $Patterns$ = set();                                            // set of connector patterns
4  foreach $df_{xy} \in F_{df}$ do
5      $DB[df_{xy}]$ = getConnectorInstances($M, df_{xy}$);
       // create database for frequent itemset mining
6      $DBDM_y$ = array():
7      foreach $dfi_{xy} \in DB[df_{xy}]$ do
8          $c_y$ = target component of $dfi_{xy}$;
9          append($DBDM_y$, $c_y.DM$);
10     $FI_{dy}$ = mineFrequentItemsets($DBDM_y$, $minsupp_{dm}$);
       // construct the connector patterns
11     foreach $DM_y \in FI_{dy}$ do
12         $Patterns = Patterns \cup \{\langle df_{xy}, DM_y\rangle\}$;
13 return $Patterns$;
```

Algorithm 3: getRecurrentConnectors

```
Data: repository of mashup models $M$, minimum support of data flow connectors ($minsupp_{df}$)
Result: set of recurrent connectors $F_{df}$
1  $DB_{df}$ = array();                              // database of data flow connector instances
2  foreach $m_i \in M$ do
3      append($DB_{df}, m_i.DF$) ;                                           // fill with instances
4  $F_{df}$ = set();                                       // set of recurrent data flow connectors
5  foreach $df_{xy} \in DB_{df}$ do
6      if computeSupport($df_{xy}, DB_{df}$) $\geq minsupp_{df}$ then
7          $F_{df} = F_{df} \cup \{df_{xy}\}$;
8  return $F_{df}$;
```

two components involved in the connector. As shown in Algorithm 6, the respective mining logic is similar to the one of the connector pattern, with two major differences: in lines 6-17 we also mine the recurrent parameter value assignments of c_x and c_y, and in lines 18-21 we consider only those combinations of VA_x, VA_y and DM_y that co-occur in mashup instances for the given connector. Notice that, for the purpose of explaining this algorithm, we perform a cartesian product of VA_x,, VA_y and DM_y in line 22. Doing this can be computational expensive if implemented as-is. In practice, the implementation of this algorithm is performed in such a way that we do not have to explore the whole search space. This comment also applies to the rest of the algorithms presented in this section.

Algorithm 4: getConnectorInstances

```
Data: repository of mashup models $M$, reference connector $df_{xy}$
Result: array of connector instances $DB_{xy}$
1 $DB_{xy}$ = array();                                  // database of data flow connector instances
2 foreach $m_i \in M$ do
3     append($DB_{xy}$], $m_i.DF \cap \{df_{xy}\}$) ;    // fill with instances of the reference connector type
4 return $DB_{xy}$;
```

Component embedding pattern. Mashup composition tools typically allow for the embedding of components inside other components. However, not all components present this capability. A common example is the *loop* component: it takes as input a set of data items and then loops over them executing the operations provided by the embedded component (e.g., a *filter* component). Embedding one component into another is not a trivial task, as there may be complex dataflow connectors and data mappings between the outer and inner component as well as between the last two and the component that proceeds the outer component in the composition flow. Algorithm 8 shows the logic for mining component embedding patterns. First, we get the instances of component embeddings from the mashup repository and then we keep only those that have a support greater or equal to $minsupp_{em}$ (lines 2-10). Using these *frequent embeddings*, we look for *frequent dataflows* that involve these embeddings (lines 11 to 17). For these patterns, we are also interested in finding data mapping and parameter value patterns and thus we proceed as in the previous algorithms to mine them (lines 18-31). In the last part of the algorithm (lines 32-37), we proceed with building the actual patterns with tuples $\langle\{c_x, c_y, c_z\}, DF, DM, VA\rangle$ that include information about the components involved in the pattern as well as the dataflow connectors, data mappings and parameter value assignments.

Multi-component pattern. The multi-component pattern represents recurrent model fragments that are composed of multiple components. It represents more complex patterns, which are not yet captured by the other pattern types alone. This pattern helps understanding domain knowledge and best practices as well as keeping modeling conventions. Multi-component patterns consists in a combination of the patterns we have introduced before. Algorithm 7 provides the details of the mining algorithm. We start by obtaining

Algorithm 5: mineConnectorCooccurrences

```
Data: repository of mashup compositions $M$, minimun support for dataflow connectors ($minsupp_{df}$) and data
      mappings ($minsupp_{dm}$)
Result: list of connector patterns with their corresponding data mappings $\langle DF_{xy}, DM_y \rangle$
// find the co-occurrence of dataflow connectors
1  $DB_{df}$ = array();
2  foreach $m_i \in M$ do
3      append($DB_{df}$, $m_i.DF$);
4  $F_{df}$ = mineFrequentItemsets($DB_{df}, minsupp_{df}$);
5  $DB_{ci}$ = array();
6  foreach $m_i \in M$ do
7      foreach $DF_{xy} \in F_{df}$ do
8          if $DF_{xy} \cap m_i.DF = DF_{xy}$ then
9              foreach $dfi_{xy} \in DF_{xy}$ do
10                 append($DB_{ci}[DF_{xy}]$, getConnectorInstances($\{m_i\}, dfi_{xy}$);
// find data mappings for the frequent dataflow connectors obtained above
11 $DBDM_y$ = array();
12 foreach $DF_{xy} \in DB_{ci}$ do
13     foreach $dfi_{xy} \in DF_{xy}$ do
14         $c_y$ = target component of $dfi_{xy}$;
15         append($DBDM_y$, $c_y.DM$);
16 $FI_{dy}$ = mineFrequentItemsets($DBDM_y$, $minsupp_{dm}$);
// construct the connector patterns
17 $Patterns$ = set();
18 foreach $DM_y \in FI_{dy}$ do
19     $Patterns = Patterns \cup \{\langle DF_{xy}, DM_y \rangle\}$;
20 return $Patterns$;
```

the graph representation of the mashups in the repository and mining frequent sub-graphs out of them (lines 2-5). For the sub-graph mining we can choose among the state of the art sub-graph mining algorithms [13]. Then, we get from the mashup repository the list of mashup fragments that match the frequent sub-graphs mined in the previous step (lines 6-11). We do this, so that next we can mine both the parameter value and data mapping patterns using again standard itemset mining algorithms (lines 13-21). Finally, we build the actual multicomponent patterns by going through the mashup repository and keeping only those combinations of patterns that co-occur in the mashup instances (lines 22-25).

```
Algorithm 6: mineComponentCooccurrences
Data: repository of mashup models $M$, minimum support of data flow connectors ($minsupp_{df}$), data mappings
      ($minsupp_{dm}$), parameter value assignments ($minsupp_{va}$) and pattern co-occurrence ($minsupp_{pc}$).
Result: set of component co-occurrence patterns with their corresponding dataflow connectors, data mappings and
        parameter values $\{\langle df_{xy,i}, VA_{x,i}, VA_{y,i}, DM_{y,i}\rangle\}$
$F_{df}$ = getRecurrentConnectors($M, minsupp_{df}$);
$DB$ = array();                                  // database of recurrent connector instances
$Patterns$ = set();                              // set of component co-occurrence patterns
foreach $df_{xy} \in F_{df}$ do
    $DB[df_{xy}]$ = getConnectorInstances($M, df_{xy}$);
    // create databases for frequent itemset mining
    $DBVA_x$ = array();
    $DBVA_y$ = array();
    $DBDM_y$ = array();
    foreach $dfi_{xy}$ in $DB[df_{xy}]$ do
        $c_x$ = source component of $dfi_{xy}$;
        $c_y$ = target component of $dfi_{xy}$;
        append($DBVA_x$, $c_x.VA$);
        append($DBVA_y$, $c_y.VA$);
        append($DBDM_y$, $c_y.DM$);
    $FI_{vx}$ = mineFrequentItemsets($DBVA_x$, $minsupp_{par}$);
    $FI_{vy}$ = mineFrequentItemsets($DBVA_y$, $minsupp_{par}$);
    $FI_{dy}$ = mineFrequentItemsets($DBDM_y$, $minsupp_{dm}$);
    // keep only those combinations of value assignments and data mappings that occur together in
       mashup instances
    $Coo$ = set();
    foreach $\langle VA_x, VA_y, DM_y\rangle \in FI_{vx} \times FI_{vy} \times FI_{dy}$ do
        if $computeSupport(\langle VA_x, VA_y, DM_y\rangle, DB[df_{xy}]) \geq minsupp_{pc}$ then
            $Coo = Coo \cup \{\langle VA_x, VA_y, DM_y\rangle\}$;
    // construct the component co-occurrence patterns
    foreach $\langle VA_x, VA_y, DM_y\rangle \in Coo$ do
        $Patterns = Patterns \cup \{\langle df_{xy}, VA_x, VA_y, DM_y\rangle\}$;
return $Patterns$;
```

5 Recommending Patterns

Recommending patterns is non-trivial, in that the size of the knowledge base may be large, and the search for composition patterns may be complex; yet, recommendations are to be delivered at high speed, without slowing down the modeler's composition pace. Recommending patterns is platform-specific. The following explanations therefore refer to the specific case of Pipes-like mashup models. In [10], we show all the details of our approach; in the following we summarize its key aspects.

Algorithm 7: mineMulticomponentPatterns

```
Data: repository of mashup compositions $M$ and minimun support for multi-components ($minsupp_{mc}$), parameter
      value ($minsupp_{par}$) and data mapping ($minsupp_{dm}$) patterns.
Result: set of multi-component patterns $\langle mf.C, mf.DF, VA, DM\rangle$.
$DB_g$ = array() ;                                   // database of graph representations of mashups
foreach $m_i \in M$ do
    // get a graph representation of mashup $m_i$ where the nodes represent components and arcs
       represent dataflows; here, the arcs are labeled with the output and input ports involved
       in the dataflow
    $g_i$ = getGraphRepresentation($m_i$);
    append($DB_g, g_i$);
$FG$ = mineFrequentSubraphs($DB_g, minsupp_{mc}$);
$DB_{mc}$ = array();
foreach $m_i \in M$ do
    foreach $fg_i \in FG$ do
        if getGraphRepresentation($m_i$) contains $fg_i$ then
            // get the fragment $mf$ from mashup instance $m_i$ that matches $fg_i$; notice that $mf$ is
               represented as a canonical mashup model
            $mf$ = getSubgraphInstance($m_i, fg_i$);
            append($DB_{mc}[fg_i], mf$)
$Patterns$ = set();
foreach $MC \in DB_{mc}$ do
    // get parameter values and data mappings and compute the corresponding frequent itemsets
    $DBVA$ = array();
    $DBDM$ = array();
    foreach $mf \in MC$ do
        foreach $c_x \in mf.C$ do
            append($DBVA, c_x.VA$);
            append($DBDM, c_x.DM$);
    $FI_{va}$ = mineFrequentItemsets($DBVA, minsupp_{par}$);
    $FI_{dm}$ = mineFrequentItemsets($DBDM, minsupp_{dm}$);
    // construct the multi-component pattern
    foreach $\langle VA, DM\rangle \in FI_{va} \times FI_{dm}$ do
        foreach $mf \in MC$ do
            if $\langle VA, DM\rangle \in mf$ then
                $Patterns = Patterns \cup \{\langle mf.C, mf.DF, VA, DM\rangle\}$ ;    // using $mf$, build the patterns
                with its components ($mf.C$), dataflows ($mf.DF$), value assignments ($mf.VA$) and
                data mappings ($mf.DM$)
return $Patterns$;
```

Algorithm 8: mineComponentEmbeddings

Data: repository of mashup compositions M, minimum supports for component embeddings ($minsupp_{em}$), data flows ($minsupp_{df}$), data mappings ($minsupp_{dm}$), parameter value ($minsupp_{par}$) and pattern co-occurrence ($minsupp_{pc}$)

Result: list of component embedding patterns with their corresponding components, dataflow connectors, data mappings and parameter value assignments $\langle\{c_x, c_y, c_z\}, DF, DM, VA\rangle$

```
// get the list of component embeddings
1  $DB_{em}$ = array();
2  foreach $m_i \in M$ do
3      foreach $\langle c_x, c_y, c_z\rangle \in m_i.C \times m_i.C$ do
4          if ($c_x$ preceeds $c_y$) and ($c_y$ embeds $c_z$) then
5              $em_{xyz} = \langle c_x, c_y, c_z\rangle$;
6              append($DB_{em}, em_{xyz}$);

// find the frequent component embeddings
7  $F_{em}$ = set();
8  foreach $em_{xyz} \in DB_{em}$ do
9      if computeSupport($em_{xyz}, DB_{em}$) $\geq minsupp_{em}$ then
10         append($F_{em}, em_{xyz}$);

// get dataflows involving the frequent component embeddings
11 $DB_{df}$ = array();
12 $F_{df}$ = array();
13 foreach $m_i \in M$ do
14     foreach $em_{xyz} \in F_{em}$ do
15         if $em_{xyz} \in m_i$ then
16             append($DB_{df}[em_{xyz}], \langle m_i.df_{xy}, m_i.df_{xz}, m_i.df_{yz}\rangle$);
17     $F_{df}$ = mineFrequentItemsets($DB_{df}, minsupp_{df}$);

// get parameter value and data mapping instances and compute the corresponding frequent itemsets
18 $DB_{va}$ = array(); $DB_{dm}$ = array();
19 foreach $m_i \in M$ do
20     foreach $\langle df_{xy}, df_{xz}, df_{yz}\rangle \in F_{df}$ do
21         if $\langle df_{xy}, df_{xz}, df_{yz}\rangle \in m_i$ then
22             $c_x$ = component instance $c_x \in m_i$ corresponding to $df_{xy}$;
23             $c_y$ = component instance $c_y \in m_i$ corresponding to $df_{xy}$;
24             $c_z$ = component instance $c_z \in m_i$ corresponding to $df_{yz}$;
25             $VA_x = c_x.VA$; $DM_x = c_x.DM$;
26             $VA_y = c_y.VA$; $DM_y = c_y.DM$;
27             $VA_z = c_z.VA$; $DM_z = c_z.DM$;
28             append($DB_{va}, VA_x \cup VA_y \cup VA_z$);
29             append($DB_{dm}, DM_x \cup DM_y \cup DM_z$);
30 $F_{va}$ = mineFrequentItemsets($DB_{va}, minsupp_{par}$);
31 $F_{dm}$ = mineFrequentItemsets($DB_{dm}, minsupp_{dm}$);
// construct the component embedding pattern
32 $Patterns$ = set();
33 foreach $\langle EM, DF, DM, VA\rangle \in F_{em} \times F_{df} \times F_{dm} \times F_{va}$ do
34     if computeSupport($\langle EM, DF, DM, VA\rangle, M$) $\geq minsupp_{pc}$ then
35         $c_x, c_y, c_z$ = components corresponding to the dataflows $df \in DF$;
36         $Patterns = Patterns \cup \{\langle\{c_x, c_y, c_z\}, DF, DM, VA\rangle\}$;
37 return $Patterns$;
```

5.1 Pattern Knowledge Base

The core of the interactive recommender is the pattern KB. In order to enable the incremental and fast recommendation of patterns, we *decompose* them into their constituent parts and focus only on those aspects that are necessary to convey the meaning of a pattern. That is, we leverage on the observation that, in order to convey the structure of a pattern, already its components and connectors enable the developer to choose in an informed fashion. Data mappings and value assignments, unless explicitly requested by the developer, are then delivered only during the weaving phase upon the selection of a specific pattern by the developer.

This strategy leads us to the ***KB*** illustrated in Figure 3, whose structure enables the retrieval of each of the patterns introduced in Section 3.1 with a one-shot query over a single table. For instance, let's focus on the component co-occurrence pattern: to retrieve its representation, it is enough to query the *ComponentCooccur* entity for the *SourceComponent* and the *TargetComponent* attributes. The query is assembled automatically upon interactions in the modeling canvas and is of the form $q = \langle object, action, pm \rangle$. Only weaving the pattern into the mashup model requires querying $ComponentCooccur \bowtie Connectors \bowtie DataMapping$ and $ComponentCooccur \bowtie ParameterValues$.

5.2 Exact and Approximate Pattern Matching

The described KB supports both *exact queries* for the patterns with predefined structure and *approximate matching* for multi-component patterns whose structure is not known a priori. Patterns are queried for or matched against the *object* of the query, i.e., the last modeling construct manipulated by the developer. Conceptually, all recommendations could be retrieved via similarity search, but for performance reasons we apply it only when strictly necessary.

Algorithm 9 details this ***strategy*** and summarizes the logic implemented by the recommendation engine. In line 3, we retrieve the types of recommendations that can be given (*getSuitableRecTypes* function), given an *object-action* combination. Then, for each recommendation type, we either query for patterns

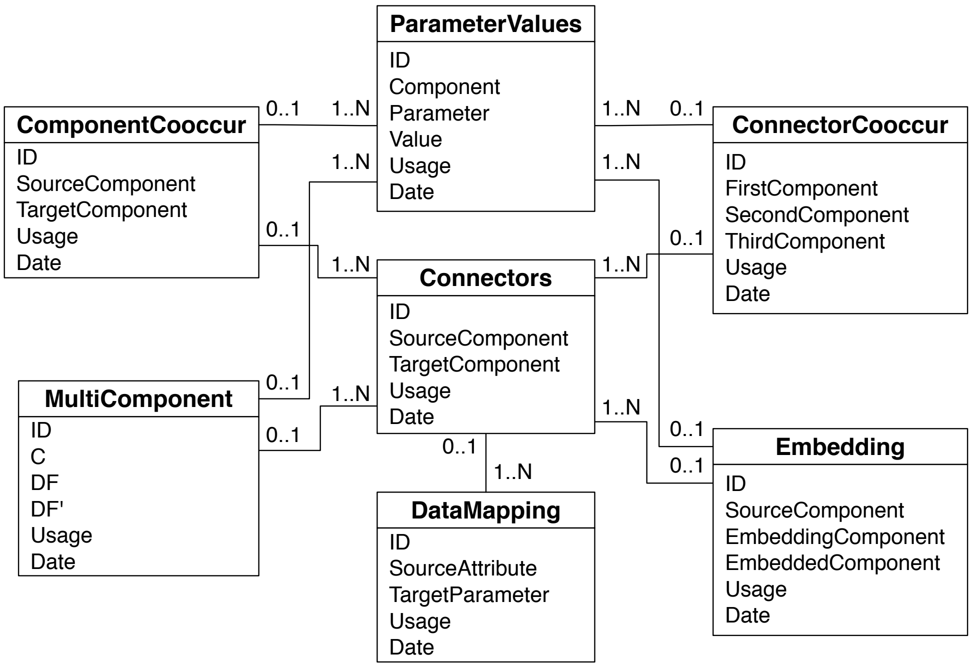


Figure 3: KB structure optimized for Pipes

(the *queryPatterns* function can be seen like a traditional SQL query) or we do a similarity search (*getSimilarPatterns* function). For each retrieved pattern, we compute a rank, e.g., based on the pattern description (e.g., containing *usage* and *date*), the computed similarity, and the usefulness of the pattern inside the partial mashup, order and group the recommendations by type, and filter out the best *n* patterns for each recommendation type.

As for the retrieval of ***similar patterns***, we give preference to exact matches of components and connectors in *object* and allow candidate patterns to differ for the insertion, deletion, or substitution of at most one component in a given path in *object*. Among the non-matching components, we give preference to functionally similar components (e.g., it may be reasonable to allow a Yahoo! Map instead of a Google Map); we track this similarity in a dedicated *CompSim* matrix. For the detailed explanation of the approximate matching logic we refer the reader to [10].

Algorithm 9: getRecommendations

```
Data: query $q = \langle object, action, pm \rangle$, knowledge base $KB$, object-action-recommendation mapping $OAR$,
      component similarity matrix $CompSim$, similarity threshold $T_{sim}$, ranking threshold $T_{rank}$, number $n$ of
      recommendations per recommendation type
Result: recommendations $R = [\langle cp_i, rank_i \rangle]$
$R$ = array();
$Patterns$ = set();
$recTypeToBeGiven$ = getRecTypes($object, action, OAR$);
foreach $recType \in recTypeToBeGiven$ do
    if $recType \neq$ "$Mul$" then
        $Patterns = Patterns \cup$ queryPatterns($object, KB, recType$) ;    // exact query
    else
        $Patterns = Patterns \cup$ getSimilarPatterns($object,$
        $KB, CompSim, T_{sim}$) ;    // similarity search
foreach $pat \in Patterns$ do
    if $rank(pat.cp, pat.sim, pm) \geq T_{rank}$ then
        append($R, \langle pat.cp, rank(pat.cp, pat.sim, pm) \rangle$) ;    // rank, threshold, remember
orderByRank($R$);
groupByType($R$);
truncateByGroup($R, n$);
return $R$;
```

6 Weaving Patterns

Weaving a given composition pattern *cp* into a partial mashup model *pm* is not straightforward and requires a thorough analysis of both *cp* and *pm*, in order to understand how to connect the pattern to the constructs already present in *pm*. In essence, weaving a pattern means emulating developer interactions inside the modeling canvas, so as to connect a pattern to the partial mashup. The problem is not as simple as just copying and pasting the pattern, in that new identifiers of all constructs of *cp* need to be generated, connectors must be rewritten based on the new identifiers, and connections with existing constructs may be required.

We approach the problem of pattern weaving by first defining a *basic weaving strategy* that is independent of *pm* and then deriving a *contextual weaving strategy* that instead takes into account the structure of *pm*.

6.1 Basic Weaving Strategy

Given an *object* and a pattern *cp* of a recommendation, the ***basic weaving strategy*** *BS* provides the sequence of mashup operations that are necessary to weave *cp* into the *object*. The basic weaving strategy does not use *pm*; it tells how to expand *object* into *cp* (*object* being a part of *cp*). This basic strategy is *static* for each pattern type and it consists a set of ***mashup operations*** that resemble the operations a developer can typically perform manually in the modeling canvas. Typical examples of mashup operations are *addComponent* that corresponds to adding a new component to *pm*, *addConnector* that corresponds to adding a connector between two selected components in *pm*, *assignValues* that corresponds to assigning values to configuration parameters of a component, and similar. Mashup operations are applied on the partial mashup *pm* and result in an updated pm'. All operations assume that the *pm* is globally accessible. The internal logic of these operations are highly platform-specific, in that they need to operate inside the target modeling environment.

For instance, the basic weaving strategy for a component co-occurrence pattern of type $ptype^{comp}$ is as follows (we assume $object = comp$ with $comp.type = c_x.type$, c_x being one of the components of the pattern):

```
1 $\$newcid$[5]=addComponent($c_y.type$);
2 addConnector($\langle comp.id, c_x.op, \$newcid, c_y.ip \rangle$);
3 assignDataMapping($\$newcid, c_y.DM$);
4 assignValues($comp.id, c_x.VA$);
5 assignValues($\$newcid, c_y.VA$);
```

That is, given a component c_x, we add the other component c_y (line 1) as mentioned in the selected pattern to the *pm*, connect c_x and c_y together (line 2) and then apply the respective data mappings (line 3) and value assignments (line 4 and line 5). Note that, the basic strategy is not yet applied to *pm*; it represents an array of basic modeling operations to be further processed before being able to weave the pattern.

6.2 Contextual Weaving Strategy

Given an object *object*, a pattern *cp*, and a partial mashup *pm*, the ***contextual weaving strategy*** WS is derived by applying the mashup operations in the *basic weaving strategy* to the current partial mashup model and thus

[5]We highlight identifier place holders (variables) that can only be resolved when executing the operation with a "$" prefix.

by weaving the selected *cp* into *pm*. The WS is *dynamically* built at runtime by taking into consideration the structure of the partial mashup (the context).

Applying the mashup operations in the basic weaving strategy may require the resolution of possible ***conflicts*** among the constructs of *pm* and those of *cp*. For instance, if we want to add a new component of type *ctype* to *pm* but *pm* already contains an instance of type *ctype*, say *comp*, we are in the presence of a conflict: either we decide that we reuse *comp*, which is already there, or we decide to create a new instance of *ctype*. In the former case, we say we apply a *soft* conflict resolution policy, in the latter case a *hard* policy:

Soft: substitute("\$$var$=addComponent(*ctype*)") with "\$$var$ = *comp.id*"

Hard: substitute("\$$var$=addComponent(*ctype*)") with "\$$var$= addComponent(*ctype*)"

Formally, the conflict resolution policy corresponds to a function **resolve-Conflict**$(pm, instr) \rightarrow CtxInstr$, where $instr$ is the mashup operation to be applied to *pm* and $CtxInstr$ is the set of instructions that replace $instr$. Only in the case of a conflict, $instr$ is replaced; otherwise the function returns $instr$ again.

In Algorithm 10 we describe the logic of our pattern weaver. First, it derives a basic strategy BS for the given composition pattern *cp* and the $object$ from *pm* (line 2). Then, for each of the mashup operations $instr$ in the basic strategy, it checks for possible conflicts with the current modeling context *pm* (line 4). In case of a conflict, the function resolveConflict$(pm, instr)$ derives the corresponding contextual weaving instructions $CtxInstr$ replacing the conflicting, basic operation $instr$. $CtxInstr$ is then applied to the current *pm* to compute the updated mashup model pm' (line 5), which is then used as basis for weaving the next $instr$ of BS. The contextual weaving structure WS is constructed as concatenation of all conflict-free instructions $CtxInstr$.

Note that Algorithm 10 returns both the list of contextual weaving instructions WS and the final updated mashup model pm'. The former can be used to interactively weave *cp* into *pm*, the latter to convert pm' into native formats.

Algorithm 10: getWeavingStrategy

```
Data: partial mashup model $pm$, composition pattern $cp$, object $object$ that triggered the recommendation
Result: weaving strategy $WS$, i.e., a sequence of abstract mashup operations; updated mashup model $pm'$
1 $WS$ = array();
2 $BS$ = getBasicStrategy($cp, object$);
3 foreach $instr \in BS$ do
4     $CtxInstr$ = resolveConflict($pm, instr$);
5     $pm$ = apply($pm, CtxInstr$);
6     append($WS, CtxInstr$);
7 return $\langle WS, pm \rangle$;
```

7 Implementation and Evaluation

We have implemented our prototype system, *Baya* [11], as Mozilla Firefox (`http://mozilla.com/firefox`) extension for Yahoo! Pipes to demonstrate the viability of our interactive recommendation approach. The ***design goals*** behind Baya can be summarized as follows: We didn't want to develop *yet another* mashup environment; so we opted for an extension of existing and working solutions (so far, we focused on Yahoo! Pipes; other tools will follow). Modelers should not be required to *ask* for help; we therefore pro-actively and interactively recommend contextual composition patterns. We did not want the *reuse* to be limited to simple copy/paste of patterns, but knowledge should be *actionable*, and therefore, Baya automatically weaves patterns.

In Baya we have implemented the *model adapters* (see Figure 2) in Java (1.6), which are able to convert Yahoo! Pipes's JSON representation into our canonical mashup model and back. All the mining algorithms are also implemented in Java. For the frequent itemset mining we used the tool Carpenter (`http://www.borgelt.net/carpenter.html`), while for graph mining we used the tool MoSS (`http://www.borgelt.net/moss.html`). The resulting patterns are expressed in terms of canonical mashup models, which are then converted to native models (in this case, Yahoo! Pipes JSON representations) by our canonical-to-native model adapter and loaded into the pattern KB.

For testing our mining algorithms, we used a dataset of 970 pipes definitions from Yahoo! Pipes that were retrieved using YQL Console (`http://developer.yahoo.com/yql/console/`). We selected pipes from the list of "most popular" pipes, as popular pipes are more likely to be functioning and useful. The average numbers of components, connectors and input parameters are 11.1, 11.0 and 4.1, respectively, which is an indication that we are dealing with fairly complex pipes.

The results obtained from running our algorithms on the selected dataset show that we are able to discover recurrent practices for building mashups. Table 1 reports on the list of pattern types and their Upper Threshold for *minsupp* (UTm). The UTm tells us what is the upper threshold for the *minsupp* values at which we start finding patterns of a given type and for a given dataset. In the cases where we use more than one type of *minsupp* (such as in the component co-occurrence pattern where we use $minsupp_{df}$, $minsupp_{dm}$ and $minsupp_{par}$), the *minsupp* we consider is the one corresponding to the pattern that is first computed in the algorithm. For our dataset, in Table 1 we can see that we are always able to find parameter value patterns for some component types. For example, this is the case of Yahoo! Pipes' component *YQL* that has the parameter *raw* with a default value *Results only* that is always kept as-is by the users. From the table we can also notice that the connector and component co-occurrence patterns have the same UTm value. This is because in both cases their corresponding algorithms compute first the frequent dataflow connectors and thus the reference minimum support for the UTm is $minsupp_{df}$. Finally, for the Multi-component pattern we have a UTm of 0.021, a relatively low value, when we consider patterns with at least 4 components. However, considering that here we are talking about complex patterns with at least 4 components that, furthermore, include dataflow connectors, data mappings and parameter value assignments, we can say that, even with a relatively low support value, these patterns still captures recurrent modeling practices for fairly complex settings.

The discovered patterns are transformed and stored in a knowledge base that is optimized for fast pattern retrieval at runtime. The implementation of the persistent pattern KB at server side, is based on MySQL (`http://www.mysql.com/`). Via a dedicated Java RESTful API, at startup of the recommendation panel the KB loader synchronizes the server-side KB with the client-side KB, which instead is based on SQLite (`http://www.sqlite.org`). The pattern matching and retrieval algorithms are implemented in JavaScript and triggered

Pattern type	**UTm**
Parameter value pattern	1
Connector pattern	0.257
Connector co-occurrence pattern	0.072
Component co-occurrence pattern	0.257
Component embedding pattern	0.124
Multi-component pattern	0.021

Table 1: Summary of pattern types with their corresponding UTm.

by events generated by the event listeners monitoring the DOM changes related to the mashup model.

The weaving algorithms are also implemented in JavaScript. Upon the selection of a recommendation from the panel, they derive the contextual weaving strategy that is necessary to weave the respective pattern into the partial mashup model. Each of the instructions in the weaving strategy refers to a modeling action, where modeling actions are implemented as JavaScript manipulations of the mashup model's JSON represenation. Both the weaving strategies (basic and contextual) are encoded as JSON arrays, which enables us to use the native `eval()` command for fast and easy parsing of the weaving logic.

Figure 4 illustrates the performance of the interactive recommendation algorithm of Baya as described in Algorithm 9 in response to the user placing a new component into the canvas, a typical modeling situation. Based on the object-action-recommendation mapping, the algorithm retrieves parameter value, connector, component co-occurrence, and multi-component patterns. As expected, the response times of the simple queries can be neglected compared to the one of the similarity search for multi-component patterns, which basically dominates the whole recommendation performance. During the performance evaluation for Baya, we have also observed that the time required for weaving a pattern is negligible with respect to the total time required for the pattern recommendation and weaving.

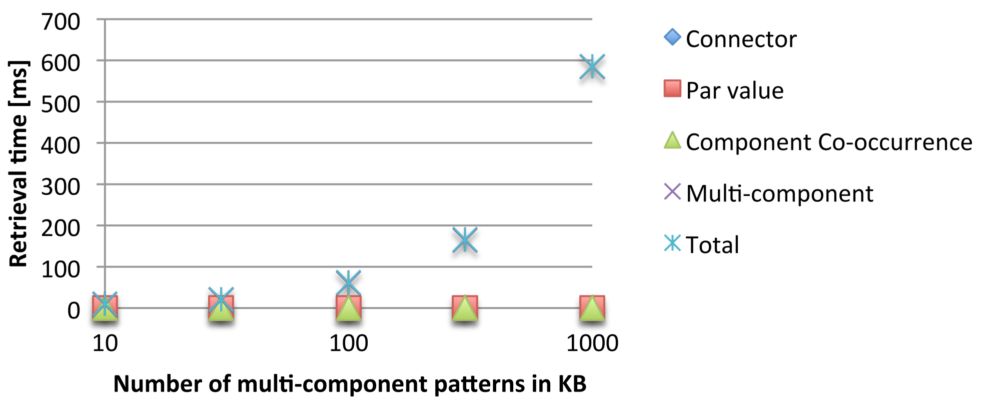


Figure 4: Recommendation types and times in response to a new component added to the canvas

8 Related work

Traditionally, ***recommender systems*** focus on the retrieval of information of likely interest to a given user, e.g., newspaper articles or books. The likelihood of interest is typically computed based on a *user profile* containing the user's areas of interest, and retrieved results may be further refined with collaborative filtering techniques. In our work, as for now we focus less on the user and more on the partial mashup under development (we will take user preferences into account in a later stage), that is, recommendations must match the partial mashup model and the object the user is focusing on, not his interests. The approach is related to the one followed by research on *automatic service selection*, e.g., in the context of QoS- or reputation-aware service selection, or adaptive or self-healing service compositions. Yet, while these techniques typically approach the problem of selecting a concrete service for an abstract activity at runtime, we aim at interactively assisting developers at design time with domain knowledge in the form of modeling patterns.

In the context of ***web mashups***, Carlson et al. [2], for instance, react to a user's selection of a component with a recommendation for the next component to be used; the approach is based on semantic annotations of component descriptors and makes use of WordNet for disambiguation. Greenshpan et al. [6] propose an auto-completion approach that recommends components and connectors (so-called glue patterns) in response to the user providing a set of desired components; the approach computes top-k recommendations out of a graph-structured knowledge base containing components and glue patterns (the nodes) and their relationships (the arcs). While in this approach the actual structure (the graph) of the knowledge base is hidden to the user, Chen et al. [3] allow the user to mashup components by navigating a graph of components and connectors; the graph is generated in response to the user's query in form of descriptive keywords. Riabov et al. [9] also follow a keyword-based approach to express user goals, which they use to feed an automated planner that derives candidate mashups; according to the authors, obtaining a plan may require several seconds. Elmeleegy et al. [5] propose MashupAdvisor, a system that, starting from a component placed by the user, recommends a set of related components (based on conditional co-occurrence probabilities and semantic matching); upon selection of a component, MashupAdvisor uses automatic planning to derive how to connect the selected component with the partial mashup, a process that may also take more than one minute. Beauche and Poizat [1] use automatic planning in ***service composition***. The planner generates a candidate composition starting from a user task and a set of user-specified services.

The ***business process management*** (BPM) community more strongly focuses on patterns as a means of knowledge reuse. For instance, Smirnov et al. [12] provide so-called co-occurrence action patterns in response to action/task specifications by the user; recommendations are provided based on label similarity, and also come with the necessary control flow logic to connect the suggested action. Hornung et al. [8] provide users with a keyword search facility that allows them to retrieve process models whose labels are related to the provided keywords; the algorithm applies the traditional TF-IDF technique from information retrieval to process models, turning the repository of process models into a keyword vector space. Gschwind et al. [7] allow users to use the control flow patterns introduced by Van der Aalst et al. [14], just like other modeling elements. The system does not provide interactive recommendations and rather focuses on the correct insertion of patterns.

In summary, assisted mashup and service composition approaches either focus on single components or connectors, or they aim to auto-complete compositions starting from user goals by using AI Planning techniques. The BPM approaches do focus on patterns, but most of the times pattern similarity is based on label/text similarity, not on structural compatibility. In our work, we consider that if components have been used together successfully multiple times, very likely their joint use is both syntactically and semantically meaningful. Hence, there is no need to further model complex ontologies or composition rules. Another key difference is that we leverage on the *interactive recommendation* of composition patterns to assists users step-by-step based on their actions on the design canvas. We do not only tell users which patterns may be applied to progress in the mashup composition process, but we also *automatically weave* recommended patterns on behalf of the users.

9 Conclusions

With this work, we aim to pave the road for assisted development in web-based composition environments. We represent *reusable knowledge* as patterns, explain how to automatically *discover* patterns from existing mashup models, describe how to *recommend* patterns fast, and how to *weave* them into partial mashup models. We therefore provide the *basic technology* for assisted development, demonstrating that the solutions proposed indeed work in practice.

As for the discovery of patterns, it is important to note that even patterns with very low support carry valuable information. Of course, they do not represent generally valid solutions or complex best practices in a given domain, but still they show *how* its constructs have been used in the past. This property is a positive side-effect of the sensible, a-priori design of the pattern structures we are looking for. Without that, discovered patterns would require much higher support values, so as to provide evidence that also their pattern structure is meaningful. Our analysis of the patterns discovered by our algorithms shows that, in order to get the best out them, domain knowledge inside the mashup models is crucial. Domain-specific mashups, in which composition elements and constructs have specific domain semantics, are a thread of research we are already following. As a next step, we will also extend the canonical model toward more generic mashup languages, e.g., including UI synchronization.

The results of our tests of the pattern recommendation approach even out-perform our own expectations, also for large numbers of patterns. In practice, however, the number of really meaningful patterns in a given modeling domain will only unlikely grow beyond several dozens. The described recommending approach will therefore work well also in the context of other browser-based modeling tools, e.g., business process or service composition instruments (which are also model-based and of similar complexity), while very likely it will perform even better in desktop-based modeling tools like the various Eclipse-based visual editors. Recommendation retrieval times of fractions of seconds and negligible pattern weaving times will definitely allow us – and others – to develop more sophisticated, assisted composition environments. This is, of course, our goal for the future – next to going back to the users of our initial study and testing the effectiveness of assisted development in practice.

Acknowledgment. This work was supported by the European Commission (project OMELETTE, contract 257635).

References

[1] S. Beauche and P. Poizat. Automated service composition with adaptive planning. In *ICSOC'08*, pages 530–537. Springer-Verlag, 2008.

[2] M. P. Carlson, A. H. Ngu, R. Podorozhny, and L. Zeng. Automatic mash up of composite applications. In *ICSOC'08*, pages 317–330. Springer, 2008.

[3] H. Chen, B. Lu, Y. Ni, G. Xie, C. Zhou, J. Mi, and Z. Wu. Mashup by surfing a web of data apis. *VLDB'09*, 2:1602–1605, August 2009.

[4] A. De Angeli, A. Battocchi, S. Roy Chowdhury, C. Rodríguez, F. Daniel, and F. Casati. End-user requirements for wisdom-aware eud. In *IS-EUD'11*. Springer, 2011.

[5] H. Elmeleegy, A. Ivan, R. Akkiraju, and R. Goodwin. Mashup advisor: A recommendation tool for mashup development. In *ICWS'08*, pages 337–344. IEEE Computer Society, 2008.

[6] O. Greenshpan, T. Milo, and N. Polyzotis. Autocompletion for mashups. *VLDB'09*, 2:538–549, August 2009.

[7] T. Gschwind, J. Koehler, and J. Wong. Applying patterns during business process modeling. In *BPM'08*, pages 4–19. Springer, 2008.

[8] T. Hornung, A. Koschmider, and G. Lausen. Recommendation based process modeling support: Method and user experience. In *ER'08*, pages 265–278. Springer, 2008.

[9] A. V. Riabov, E. Boillet, M. D. Feblowitz, Z. Liu, and A. Ranganathan. Wishful search: interactive composition of data mashups. In *WWW'08*, pages 775–784. ACM, 2008.

[10] S. Roy Chowdhury, F. Daniel, and F. Casati. Efficient, Interactive Recommendation of Mashup Composition Knowledge. In *ICSOC'11*, pages 374–388. Springer, 2011.

[11] S. Roy Chowdhury, C. Rodríguez, F. Daniel, and F. Casati. Baya: Assisted Mashup Development as a Service. In *WWW'12*, 2012.

[12] S. Smirnov, M. Weidlich, J. Mendling, and M. Weske. Action patterns in business process models. In *ICSOC-ServiceWave'09*, pages 115–129. Springer-Verlag, 2009.

[13] P. Tan, S. M, and K. V. *Introduction to Data Mining*. Addison-Wesley, 2005.

[14] W. M. P. Van Der Aalst, A. H. M. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distrib. Parallel Databases*, 14:5–51, July 2003.