Towards a Compiler for Business-IT Systems

Jana Koehler, Thomas Gschwind, Jochen Küster, Hagen Völzer, Olaf Zimmermann

Towards a Compiler for Business-IT Systems

A vision statement complemented with a research agenda

Jana Koehler, Thomas Gschwind,
Jochen Küster, Hagen Völzer, and Olaf Zimmermann

IBM Zurich Resarch Laboratory
Säumerstrasse 4, CH-8038 Rüschlikon, Switzerland

Abstract. Business information systems and enterprise applications have continuously evolved into Business-IT systems over the last decades, directly linking and integrating Business Process Management with recent technology evolutions such as Web services and Service-Oriented Architectures. Many of these technological evolutions include areas of past academic research: Business rules closely relate to expert systems, Semantic Web technology uses results from description logics, attempts have been made to compose Web services using intelligent planning techniques, and the analysis of business processes and Web service choreographies often relies on model checking. As such, many of the problems that arise with these new technologies have been solved at least in principle.
However, if we try to apply these "in principle" solutions, we are confronted with the failure of these solutions in practice: many proposed solution techniques do not scale to the real-world requirements or they rely on assumptions that are not satisfied by Business-IT systems.
As has been observed previously, research in this area is fragmented and does not follow a truly interdisciplinary approach. To overcome this fragmentation, we propose the vision of a compiler for Business-IT systems that takes business process specifications described at various degrees of detail as input and compiles them into executable IT systems. As any classical compiler, the parsing, analysis, optimization, code generation and linking phases are supported. We describe a set of ten research problems that we see as critical to bring our compiler vision to reality.

1 Introduction

Business processes comprise sequences of activities that bring people, IT systems, and other machines together to act on information and raw materials. They allow a business to produce goods and services and deliver them to its customers. A business is a collection of business processes and thus (re-)engineering business processes to be efficient is one of the primary functions of a company's management. This is one of the most fundamental mechanisms that drives advances in our society.

Business Process Management (BPM) is a structured way to manage the life cycle of business processes including their modeling (analysis and design), execution, monitoring, and optimization. Good tools exist that allow business processes to be analyzed and designed in an iterative process. Creating a model for a process provides insights

that allow the design of a process to be improved, in particular if the modeling tool supports process simulation.

A Service-Oriented Architecture (SOA), which is often implemented using Web services, is an architectural style that allows IT systems to be integrated in a standard way, which lends itself to the efficient implementation of business processes. SOA also enables efficient process re-engineering as the use of standard programming models and interfaces makes it much simpler to change the way in which the components of business processes are integrated. Once a process is implemented, it can be monitored, e.g., measured, and then further optimized to improve the quality, the performance, or some other aspect of the process.

Despite the progress that has been made recently in business process design and modeling on the one hand, and their execution and monitoring on the other, there is a significant gap in the overall BPM life cycle, which severely limits the ability of companies to realize the benefits of BPM. No complete solution exists to automate the translation from business process models to executable business processes. While partial solutions exist that allow some process models to be mapped to implementations (workflows), scalable and automated approaches do not exist that would support businesses in the full exploitation of the BPM life cycle to achieve rapid improvements of their business processes. Thus, many of the benefits of process modeling and SOA cannot be realized and effective BPM remains a vision.

As has been observed previously, research in this area is fragmented and does not follow a truly interdisciplinary approach. This lack of interdisciplinary research is seen as a major impediment that limits added economic growth through deployment and use of services technology [1]:

> *The subject of SOC*[1] *is vast and enormously complex, spanning many concepts and technologies that find their origins in diverse disciplines that are woven together in an intricate manner. In addition, there is a need to merge technology with an understanding of business processes and organizational structures, a combination of recognizing an enterprise's pain points and the potential solutions that can be applied to correct them. The material in research spans an immense and diverse spectrum of literature, in origin and in character. As a result research activities at both worldwide as well as at European level are very fragmented. This necessitates that a broader vision and perspective be established one that permeates and transforms the fundamental requirements of complex applications that require the use of the SOC paradigm.*

This paper presents a concrete technological vision and foundation to overcome the fragmentation of research in the BPM/SOA area. We propose the vision of a compiler for Business-IT systems that takes business process specifications described at various degrees of detail as input and compiles them into executable IT systems. This may sound rather adventurous, however, recall how the first compiler pioneers were questioned when they suggested to programmers that they should move from hand-written

[1] SOC stands for Service-Oriented Computing a term that is also used to denote research in the area of BPM and SOA. The still evolving terminology is a further indicator of the emerging nature of this new research field.

assembly code to abstract programming languages from which machine-generated code would then be produced. The emerging new process-oriented programming languages such as the Business Process Execution Language (BPEL) [2] or the Business Process Modeling Notation (BPMN) [3] are examples of languages that are input to such a compiler. In particular, the upcoming version 2 of the Business Process Modeling Notation (BPMN) [3] can be considered as a language that at the same time allows non-technical users to describe business processes in a graphical notation, while technical users can enrich these descriptions textually until the implementation of the business process is completely specified. With its formally defined execution semantics, BPMN 2.0 is directly executable. However, direct execution in the literal meaning of the words means to follow an interpreter-based approach with all its shortcomings, which cannot be considered as really desirable.

The envisioned compiler does not take a high-level business process model, magically adds all the missing information pieces, and then compiles it into executable code. The need to go from an analysis model to a design model in an iterative refinement process that involves human experts does not disappear. The compiler enables human experts to more easily check and validate their process model programs. The validation helps them in determining the sources of errors as well as the information that is missing. Only once all the required information is available, the code can be completely generated. The compiler approach also extends the Model-Driven Architecture (MDA) vision; beyond model transformations that provide a mapping between models with different abstractions, we combine code generation with powerful analytical techniques. Static analysis is performed yielding detailed diagnostic information and structural representations similar to the abstract syntax tree are used by the Business-IT systems compiler. This provides a more complete understanding of the process models, which is the basis for error handling, correct translation, and runtime execution.

The paper is organized as follows: In Section 2, we motivate the need for a Business-IT systems compiler by looking at BPM life cycle challenges. In Section 3, we summarize the ten research problems that we consider as particularly interesting and important to solve. In Section 4, we discuss the problems in more detail and review selected related work. Section 5 concludes the paper.

2 Life Cycle Challenges in Business-IT Systems

The IT Infrastructure underlying a business is a critical success factor. Even when IT is positioned as a commodity such as by Nicolas Carr in “IT doesn’t matter,” it is emphasized that a disruptive new technology has arrived, which requires companies to master the economic forces that the new technology is unleashing. In particular smaller companies are not so well positioned in this situation. The new technology in the form of Business Process Management (BPM) and Service-Oriented Architecture (SOA) is complex to use, still undergoing significant changes, and it is difficult even for the expert to distinguish hype from mature technology development.

There is wide agreement that business processes are the central focus area of the new technology wave. On the one hand, business processes are undergoing dramatic change made possible by the technology. On the other hand, increasing needs in making busi-

ness processes more flexible, while retaining their integrity and compliance with legal regulations continue to drive technology advances in this space. A prominent failure in process integrity is the financial crisis that emerged throughout 2008.

Figure 1 reflects our current view of the driving forces behind the life cycle of business processes. Two interleaved trends of commoditization and innovation have to be mastered that involve the solution of many technical problems. Let us spend some space discussing this picture to explain why this is a challenge for most businesses today and why underpinning business process innovation with compiler technology is essential to master the innovation challenge.

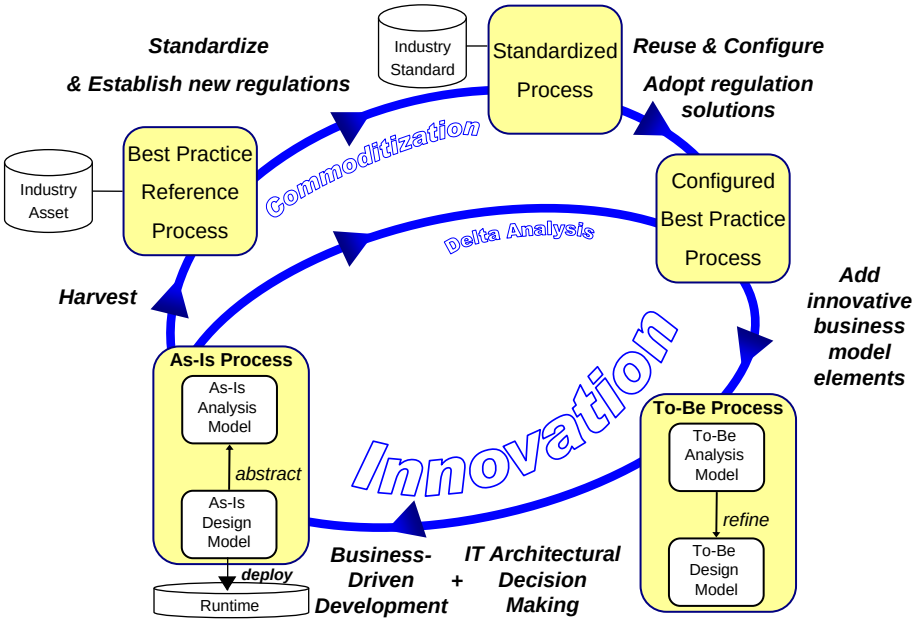

Fig. 1. Driving Forces behind the Life Cycle of Business Processes

Let us begin in the lower left corner with the As-Is Process box. This box describes the present situation of a business. Any business has many processes implemented, many of them run today in an IT-supported environment. As such they were derived from some As-Is design model and deployed. Sometimes, the design model is simply the code. The As-Is design model is linked to an As-Is analysis model. Here, we adopt the terminology from software modeling that distinguishes between an analysis model (in our case, the business view on the processes) and the design model (in our case, the implemented processes). The analysis model is usually an abstraction from the design model, i.e., a common view of the business on the implemented processes exists in

many companies.[2] The direct linkage between the business view on the processes and their implementation constitutes the Business-IT system.

The As-Is processes implemented by the players in an industry represent the state of the art of the Business-IT systems. Industries tend to develop a solid understanding of good and bad practices and often develop best practice reference solutions. Very often, consulting firms also specialize in helping businesses understanding and adopting these best practice processes. Today, one can even see a trend beyond adoption. For example, in the financial industry one can see first trends towards standardized processes that are closely linked to new regulations. This clearly creates a trend of commoditization forcing companies to adopt the new regulation solutions. With that we have arrived at the upper right corner of the picture.

The commoditization trend is pervasive in the economic model of the western society and it is as such not surprising that it now reaches into business processes. However, in a profit-driven economy, commoditization is not desirable as it erodes profit. Businesses are thus forced to escape the commoditization trap, which they mostly approach by either adopting new technologies or by inventing new business models. Both approaches directly lead to innovations in the business processes. In the picture above, the new business processes that result from the innovation trend are shown as the To-Be processes, which have to both accommodate commoditization requirements and as well as to include innovation elements at the same time. The To-Be innovation must also be evident with respect to the As-Is process and the best practice process, which is illustrated in the picture with the reference to a delta analysis involving the three process models. The To-Be process is usually (but not always) initiated at the analysis level, i.e., the business develops a need for change and begins to define this change. The To-Be analysis model must be refined into a To-Be design model and then taking through a business-driven development and IT-architectural decision process that is very complex today. With the successful completion of the development, the To-Be process becomes the new As-Is process. With that the trends of commoditization and innovation repeat within the life cycle of business processes [4, 5].

This paper focuses on the technological underpinnings for business process innovation, i.e., the adoption of best practice processes, their combination with innovative elements, and the replacement of the As-Is process by the To-Be process. We investigate these challenges from a strictly technological point of view and identify a number of specific problems that are yet unsolved, but have to be solved in order to support businesses in their innovation needs. Problems of process abstraction, harvesting, and standardization are also of general interest, but are outside the scope of this vision as is a study of the economic or social effects of what has been discussed above.

3 Compilation Phases and Associated Research Problems

Our main goal is to understand how a compiler for Business-IT systems works. At its core, we see the compilation of business process models that constitutes a well-defined

[2] By monitoring or mining the running processes or analyzing and abstracting the underlying design model or code in some form, an analysis model can also be produced in an automatic or semi-automatic manner, but this is beyond the focus of this paper.

problem. In the following, we relate the principal functionalities of a programming language compiler to the corresponding problems of compiling a business process model.

Following Muchnik [6] "compilers are tools that generate efficient mappings from programs to machines". Muchnik also points out that languages, machines, and target architectures continue to change and that the programs become ever more ambitious in their scale and complexity. In our understanding, languages such as BPMN are the new forms of programs and SOA is a new type of architecture that we have to tackle with compilers. A compiler-oriented approach helps to solve the business problems and to address the technical challenges around BPM/SOA. For example, verifying the compliance and integrity of a business with legal requirements must rely on a formal foundation. Furthermore, agility in responding to innovation requires a higher degree of automation. At a high-level, a compiler works in the following five phases:

1. Lexical analysis and parsing
2. Structural and semantic analysis
3. Translation and intermediate code generation
4. Optimization
5. Final assembly and linking and further optimization

We envision the compiler for Business-IT systems to work in the same five phases. While we consider the parsing and lexical analysis phase as essentially being solved by our previous and current work, we propose ten specific research problems that address key problems for the subsequent phases. The list below briefly summarizes the ten problems. In Section 4, they are discussed in more detail.

1. **Lexical analysis and parsing:**: We developed the Process Structure Tree (PST) as a unique decomposition of the workflow graph, which underlies any business process model, into a tree of fragments that can be computed in linear time. The PST plays the same role in the Business-IT systems compiler as the Abstract Syntax Tree (AST) in a classical compiler.
2. **Structural and semantic analysis:** We developed a control-flow analysis for workflow graphs that exploits the PST and demonstrates its usefulness, but which can still be significantly expanded in terms of the analysis results it delivers as well as the scope of models to which it can be applied.

   Problem 1: Clarify the role of orchestrations and choreographies in the compiler. Process models describe the flow of tasks for one partner (orchestration) as well as the communication between several partners (choreographies). Structural and semantic analysis must be extended to choreographies and orchestration models. Furthermore, it must be clarified which role choreography specifications play during the compilation process.

   Problem 2: Solve the flow separation problem for arbitrary process orchestrations. Process orchestrations can contain specifications of normal as well as error-handling flows. Both flows can be interwoven in an unstructured diagram, with their separation being a difficult, not yet well-understood problem.

   Problem 3: Transfer and extend data-flow analysis techniques from classical compilers to Business-IT systems compilers. Processes manipulate business data,

which is captured as data flow in process models. Successful techniques such as Concurrent Single Static Assignment (CSSA) must be transferred to the Business-IT systems compiler.

Problem 4: Solve the temporal projection problem for arbitrary process orchestrations. Process models are commonly annotated with information about states and events. This information is usually available at the level of a single task, but must be propagated over process fragments, which can exhibit a complex structure including cycles.

Problem 5: Develop scalable methods to verify the termination of a process choreography returning detailed diagnostic information in case of failure. Correctly specifying the interaction between partners that execute complex process orchestrations in a choreography model is a challenging modeling task for humans. In particular, determining whether the orchestration terminates is a fundamental analysis technique that the compiler must provide in a scalable manner.

3. **Translation and intermediate code generation:** Many attempts exist to translate business-level process languages such as BPMN to those languages used by the runtime such as BPEL. None of the proposed approaches is satisfying due to strong limitations in the subsets of the languages that can be handled and the quality of the generated code, which is often verbose. Major efforts have to be made to improve the current situation.

   Problem 6: Define a translation from BPMN to BPEL and precisely characterize the maximal set(s) of BPMN diagrams that are translatable to structured BPEL.[3]

4. **Optimization:** Code generated today or attempts to natively execute process-oriented languages are very limited with respect to the further optimization of the code. Specific characteristics of the target IT architecture are rarely taken into account.

   Problem 7: Define execution optimization techniques for the Business-IT systems compiler. Until today, business processes are usually optimized with respect to their costs. No optimization of a process with respect to the desired target platform happens automatically as it is available in a classical compiler. It is an open question which optimizations should be applied when processes are compiled for a Service-Oriented Architecture.

5. **Final assembly, linking and further optimization:** Assembly and linking problems in a Service-Oriented Architecture immediately define problems of Web service reuse and composition, for which no satisfying solutions have been found yet.

   Problem 8: Redefine the Web service composition problem such that it is grounded in realistic assumptions and delivers scalable solutions. Web service composition is studied today mostly from a Semantic Web perspective assuming that rich semantic annotations are available that are provided by humans. A compiler, however, should be able to perform the composition and linking of service components without requiring such annotations.

[3] We only define a single problem focusing on the challenge of BPMN-BPEL translation, although the question of an adequate BPEL-independent "byte-code" level for BPMN is also very interesting and deserves further study.

Problem 9: Redefine the adapter synthesis problem by taking into consideration constraints that occur in business scenarios. Incorrect choreographies have to be repaired. Often, this is achieved by not changing the processes that are involved in the choreography, but by synthesizing an adapter that allows the partners to successfully communicate with each other. Such an adapter often must include comprehensive protocol mediation capabilities. So far, no satisfying solutions have been found for this problem and we argue that it must be reformulated under realistic constraints.

Problem 10: Demonstrate how IT architectural knowledge and decisions are used within the compiler. The target platform for the Business-IT systems compiler is a Service-Oriented Architecture. Architectural decision making is increasingly done with tools that make architectural decisions explicit and manage their consistency. These decisions can thus become part of the compilation process, making it easier to compile processes for different back end systems.

The positioning of these ten problems within the various compilation phases makes it possible for researchers to tackle them systematically, study their interrelationships, and solve the problems under realistic boundary constraints. Our vision allows us to position problems in a consistent and comprehensive framework that have previously been tackled in isolation. This can lead to synergies between the various possible solution techniques and allows researchers to successfully transfer techniques that were successful in one problem space to another.

Our vision provides researchers with continuity in the technological development, with compilers tackling increasingly complex languages and architectures. A solution of the ten research problems has significant impact on the integrity, improved agility and higher automation within BPM/SOA.

4 A Deeper Dive into the Research Problems

A compiler significantly increases the quality of the produced solution and provides clearer traceability. Approaches of manual translation are envisioned to be replaced by tool-supported refinement steps guided by detailed diagnostic information. The optimization of Business-IT systems with respect to their execution becomes possible, which can be expected to lead to systems with greater flexibility making it easier for businesses to follow the life cycle of process innovation.

A compiler can also help in automating many manual steps and be expected to produce higher-quality results than those that can be obtained by manual, unsupported refinement and implementation steps. With the compiler approach, we propose to go beyond the Model-Driven Architecture (MDA) vision that proposes models at different levels of abstraction and model transformations to go from a more abstract to a more refined model. Two problems prevent that MDA is fully workable for BPM/SOA. If used at all, model transformations are written mostly in an ad-hoc manner in industrial projects today. They rarely use powerful analytical techniques such as the static analysis performed by compilers, nor do they exploit structural representations similar to the

abstract syntax tree that a compiler builds for a program. Furthermore, too many different models result from the transformations with traceability between these models remaining an unsolved problem so far.

In the following, we review selected related work in the context of the five phases. The review will not be a comprehensive survey of the state of the art. We focus on where we stand in our own research with respect to the compiler for Business-IT systems and point by example to other existing work in various fields of computer science that we consider as relevant when tackling the problems that we define for the five phases.

4.1 Parsing

The parsing problem for business process models has not yet been widely recognized by the BPM community as an important problem. Figure 2 shows a typical workflow graph underlying any business process model. It includes activities a_i, decisions d_i, merges m_i (for alternative branching and joining) as well as forks f_i and joins j_i (for parallel branching and joining). Today, process-oriented tools treat such models as large, unstructured graphs. No data structure such as the Abstract Syntax Tree (AST) used by compilers is available in these tools.

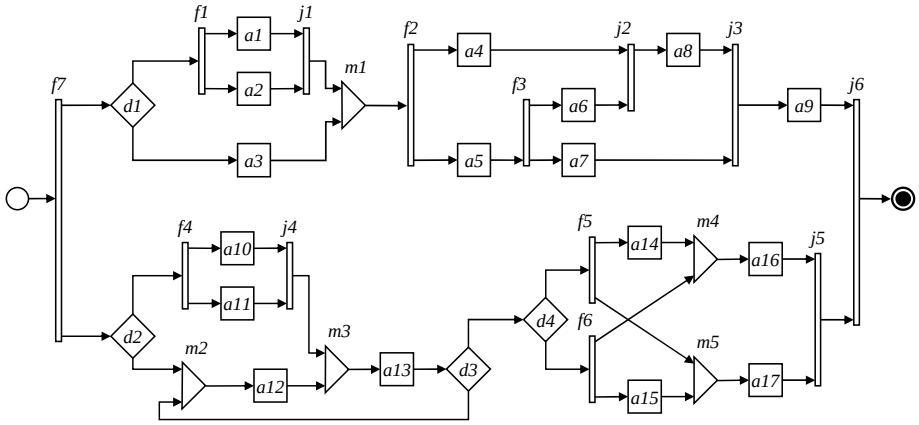

Fig. 2. Workflow graph of an example process model in a UML Activity Diagram-like notation.

In our own research, we developed the Process Structure Tree (PST) [7, 8], which we consider to be the AST analogy for Business-IT systems compilers. The PST is a fundamental data structure for all the subsequent phases of a compilation. By applying techniques from the analysis of program structure trees [9–12] to business process models a unique decomposition of process models into a tree of fragments can be computed using a linear time algorithm. The PST is a significant improvement compared to approaches that use graph grammars to parse the visual language, which is exponential in most cases [13]. Figure 3 shows the PST for the workflow graph of Figure 2.

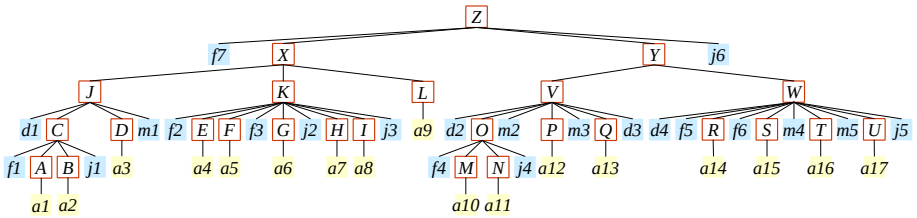

Fig. 3. Process Structure Tree (PST) for the workflow graph of Figure 2.

With this, we believe that the parsing problem for the Business-IT systems compiler is solved for the near future. Additional improvements can be imagined, but in the following we concentrate on the other phases of the compiler, which help validating that the PST is indeed as powerful as the AST.

4.2 Structural and Semantic Analysis

In our own research, we have developed two types of analysis based on the PST: a) a control-flow analysis [7, 14] and b) an approach to the structural comparison and difference analysis of process models [15]. Both demonstrated that the PST is an essential prerequisite and a powerful data structure to implement various forms of analyses. In the following, we shortly summarize our current insights into the analysis problem and identify a set of concrete problems that we consider as being especially relevant and interesting.

A business process model is also often referred to as a process orchestration. A process orchestration (the control- or sequence flow) describes how a single business process is composed out of process tasks and subprocesses. In a SOA implementation, each task or subprocess is implemented as a service, where services can also be complex computations encapsulating other process orchestrations. In contrast to an orchestration, a process choreography describes the communication and thus the dependencies between several process orchestrations. Note that the distinction between orchestration and choreography is a “soft” one and usually depends on the point of view of the modeler.

An example of a simple process orchestration and choreography specification in BPMN is shown in Figure 4, taken from the BPMN 1.1 specification [3]. The figure shows an abstract process Patient and a concrete process Doctor’s office. The Doctor’s office process orchestration is a simple sequence of tasks. The dotted lines between the two processes represent an initial and incomplete description of the choreography by showing the messages flowing between the two processes.[4]

[4] Note that the clarification and formal definition of the semantics of BPMN is another focus area of our work. However, developing the fundamental techniques for a Business-IT compiler does not require BPMN as a prerequisite. Related well-defined languages such as Petri nets or workflow graphs can also be assumed. Nevertheless, we plan to apply our techniques to BPMN due to the growing practical relevance of the language.

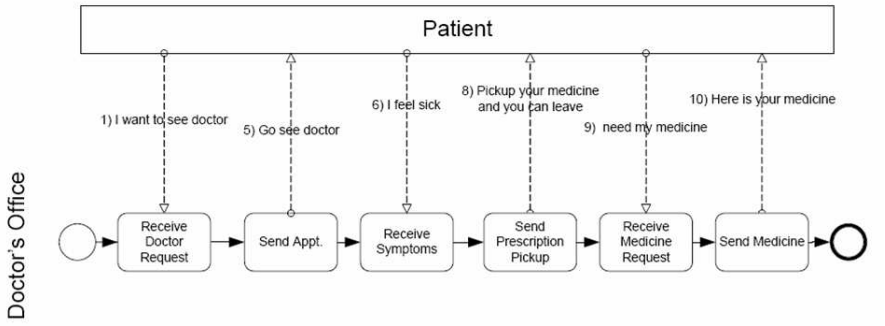

Fig. 4. Choreographies and orchestrations in the Doctor's Office example process.

Problem 1: Clarify the role of orchestrations and choreographies in the compiler.

Our compiler needs to be able to analyze orchestrations as well as choreographies. However, it is not fully clear at which phase choreography information is relevant for the compilation. It is clearly relevant in the assembly and linking phase when an entire Business-IT system is built, but one can also imagine that the optimization of an orchestration can be specific to a given choreography in order to better address the desired target architecture.

Another fundamental question for the analysis is the detection of control- and data-flow errors. In the context of a process orchestration, verification techniques have been widely used, e.g., [16] to find errors in the specified control flows. To the best of our knowledge, compiler techniques have not yet been considered so far.

Verification of business processes is an area of research that has established itself over the last decade. Locating errors in business processes is important in particular because of the side effects that processes have on data. Processes that do not terminate correctly because of deadlocks or processes that exhibit unintended execution traces due to a lack of synchronization often leave data in inconsistent states [17]. Common approaches to process verification usually take a business process model, translate a process model into a Petri net or another form of a state-based encoding and then run a Petri-net analysis tool or model checker on the encoding. Examples are the Woflan tool [18] or the application of SMV or Spin to BPEL verification, e.g., [19, 20]. In principle, these approaches make it possible to detect errors in business processes. However, there are severe limitations that so far prevented the adoption of the proposed solutions in industrial tools:

- Encodings are usually of exponential size compared to the original size of the process model.
- The verification tools in use do not give detailed enough diagnostic information in such a way that they allow an end user to easily correct errors—it has turned out in practice that counterexample traces are unfortunately only rarely pointing to the real cause of an error.

– The approaches often make restricting assumptions on the subclass of process models that they can handle.

Consequently, the currently available solutions are only partially applicable in practice due to their long runtimes, the lack of suitable diagnostic information, and the restrictions on the defined encodings.

In our own work, we have followed a different approach. First, we analyzed hundreds of real-world business processes and identified commonly occurring so-called anti-patterns [21]. Secondly, we used the PST as th unique parse tree of a process model to speed up the verification of the process [7, 14]. Each fragment in a PST can be analyzed in isolation because the tree decomposition ensures that a process model is sound if each fragment is sound. Many fragments exhibit a simplified structure and their soundness (i.e., the absence of deadlocks and lack of synchronization errors) can be verified by matching them against patterns and anti-patterns. Only a small number of fragments remains to which verification methods such as model checking must be applied. Furthermore, the size of a fragment is usually small in practice, which results in a significant state-space reduction. Consequently, the resulting combination of verification techniques with structural analysis leads to a complete verification method that is low polynomial in practice with worst-case instances only occurring rarely. As each error is local to a fragment, this method also returns precise diagnostic information.

Implementation of the work [14] showed that the soundness of even the largest business process models that we observed in practice can be completely analyzed within a few milliseconds. Consequently, the technology can be made available to users of modeling tools where they obtain instant feedback. Patterns and refactoring operations [22, 23] can be provided to users to help them correct the detected modeling errors easily. The patterns and refactoring operations take advantage of the fine-grained diagnostic information and the PST to support users in accomplishing complicated editing steps in a semi-automatic and correct manner.

With these results, a major step forward has been made. Still, two problems remain. First, the control-flow analysis must be extended to process models that are enriched with the description of error-handling or compensation flow as it is possible in BPMN. Secondly, no sufficient data-flow analysis techniques are yet available to analyze business processes. Figure 5 illustrates two more problems that we propose to investigate in more detail.

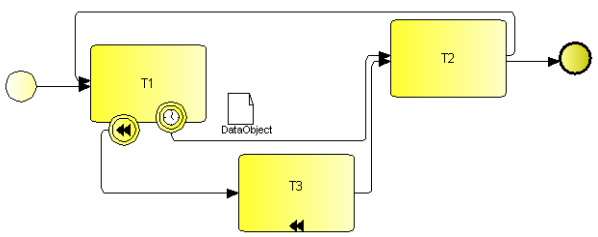

Fig. 5. Example of a business process model in BPMN showing an error-handling flow and data.

Problem 2: Solve the flow separation problem for arbitrary process orchestrations.

Figure 5 shows a repetitive process where a task $T1$ is executed followed by a task $T2$. $T1$ has some data object as output. During the execution of $T1$, some compensation event can occur that requires task $T3$ to execute. When the compensation is finished, the process continues with $T2$. BPMN allows business users to freely draw "normal" flows as well as error-handling flows within the same process model. An error-handling flow can branch off in some task interrupting the normal flow and then merge back later into the normal flow. For a process without cycles, it is relatively easy to tell from the process model where normal and error-handling flows begin and end. For processes with cycles, this is much more complicated and constitutes an unsolved problem that we denote as the "flow separation problem." A solution to this problem requires the definition of the semantics of error-handling flows. Furthermore, an error-handling flow must always be properly linked to a well-defined part of the normal flow, which is usually called the scope. Computing the scope of an error handling flow from an unstructured process model is an open problem.

Problem 3: Transfer and extend data-flow analysis techniques from classical compilers to Business-IT systems compilers.

Data-flow analysis for unstructured business process models is also a largely unsolved problem. Figure 5 shows some data object as an output of task $T1$. Large diagrams often refer to many different types of data objects as the inputs and outputs of tasks. Furthermore, decision conditions in the branching points of process flows often refer to data objects. Users who work with process models are interested in answering many questions around data such as whether data input is available for a task, whether data can be simultaneously accessed by tasks running in parallel in a process, or whether certain decision conditions can ever become true given certain data, i.e., whether there are flows in the process that can never execute.

An immediate candidate is the Concurrent Single Static Assignment approach [24] that we have begun to explore. Data-flow analysis is also a prerequisite to answer questions such as whether a compensation flow really compensates for the effects of a failed normal flow.

Problem 4: Solve the temporal projection problem for arbitrary process orchestrations.

Recently, additional knowledge about the process behavior in the form of semantic annotations is added to process models. These annotations take the form of formally specified pre- and postconditions or simple attribute-value pairs. A tool should be able to reason about these semantic annotations, for example to conclude what pre- and postconditions hold for a complex process fragment containing cycles when the control flow is specified and the pre- and postconditions of the individual tasks are known. This problem of computing the consequences of a set of events has been studied as the so-called Temporal Projection problem in the area of Artificial Intelligence (AI)

planning [25] and regressing and progression techniques have been developed. Unfortunately, AI plans exhibit a much simpler structure than process models, in particular they are acyclic, i.e., the existing techniques are not directly applicable. A solution to the temporal projection problem is important for the analysis of data flows as well as for the composition of processes (and services).

These four research problems address major challenges for the analysis phase of the compiler when investigating a process orchestration, i.e., a single process model. For process choreographies that describe the interaction and communication between several processes, we are mostly interested in termination problems. Can two processes successfully communicate with each other such that both terminate?

Problem 5: Develop scalable methods to verify the termination of a process choreography returning detailed diagnostic information in case of failure.

If a process choreography is fully specified, this question can be precisely answered. Even in the case of abstract models and underspecified choreographies such as in the example of Figure 4, interesting questions can be asked and answered. For example, which flow constraints must the abstract Patient process satisfy such that a successful communication with the Doctor's office is possible? Previous work, notably the research on operating guidelines [26–28] has provided an initial answer to these questions. The proposed analysis techniques are based on Petri nets, but do not yet scale sufficiently well. Similar to the case of process orchestrations, we are also interested in precise diagnostic information when verifying choreographies.

4.3 Translation and intermediate code generation

For the translation phase, we consider one problem as especially important and want to restrict us to this problem, namely the translation from unstructured BPMN to structured BPEL [2].

Problem 6: Define a translation from BPMN to BPEL and precisely characterize the maximal set of BPMN diagrams that are translatable to structured BPEL.

An example of relevant related work is [29]. The approach exploits a form of structural decomposition, but not as rigorous as the PST and therefore leads to non-uniform translation results, i.e., the order of application of the translation rules determines the translation output. It is also important to further improve on the initial insights into the classes of BPMN diagrams that are translatable into structured BPEL. Beyond BPEL, one can also imagine that the translation of BPMN to other runtimes, e.g., ones that use communicating state machines is of major practical relevance.

4.4 Optimization

The optimization phase for the Business-IT systems compiler is a completely unaddressed research area so far.

Problem 7: Define execution optimization techniques for the Business-IT systems compiler.

Classical process optimization, which is mostly performed during a Business Analysis phase, usually focuses on cost minimization. For the compiler, we are envisioning an optimization of processes with respect to their execution on the planned target architecture, but not so much a cost optimization of the process itself. One advantage of compilers is their ability to support multiple platforms. Different architectures, including different styles of SOA, require and enable differences in the process implementation. Optimizations such as load balancing or clustering have for example been studied in the context of J2EE applications [30]. We have some initial insights, but first of all the main goal must be to clarify what can and should happen during the optimization phase.

4.5 Final assembly and linking further optimization

For the assembly and linking phase, we see two problem areas that are of particular interest. First, we propose to further study several well-defined synthesis problems. In the literature, two instances of process synthesis problems have been investigated so far: On the one hand, there is the Web service composition problem that is mostly tackled using AI planning techniques [31, 32].

Problem 8: Redefine the Web service composition problem such that it is grounded in realistic assumptions and delivers scalable solutions.

Web service composition tries to assemble a process orchestration from a predefined set of services. It is commonly assumed that the goal for the composition is explicitly given and that services are annotated with pre- and postconditions. Unfortunately, both assumptions are rarely satisfied in practice. In particular, business users usually have a rather implicit understanding of their composition goals. We cannot expect these users to explicitly formulate their goals in some formal language. Furthermore, the processes returned by the proposed methods for service composition are very simple and resemble more those partially-ordered plans as studied by the AI planning community than those processes modeled by BPMN diagrams.

The second problem is the adapter synthesis problem, which is addressed by combining model checking techniques with more or less intelligent "guess" algorithms [33, 34]. Adapter synthesis tries to resolve problems in a faulty choreography by generating an additional process that allows existing partners to successfully communicate.

Problem 9: Redefine the adapter synthesis problem by taking into consideration constraints that occur in business scenarios.

The problem is inherently difficult in particular due to the unconstrained formulation in which it is studied. Usually, the goal is to generate "some" adapter without formulating any further constraints. Consequently, an infinite search space is opened up and the methods are inherently incomplete. In addition, the synthesized adapters must

be verified, because the correctness of the synthesis algorithms is usually not guaranteed.

There is thus a wide gap between the currently proposed techniques and the needs of a practically relevant solution. A first goal must therefore be to formulate practically relevant variants of the service composition and adapter synthesis problems. Secondly, solutions to these problems must be worked out that make realistic assumptions, scale to real-world problems and are accepted by the commercial as well as the academic world.

An initial goal for these last two research problems is thus to identify realistic problem formulations. For the web composition problem, this means to replace the assumptions of explicit goals and pre- and postconditions by the information that is available in real-world use cases of service composition. Furthermore, the composition methods must be embedded into an approach based on iterative process modeling where a human user is involved, similar to what has been studied by the AI planning community under the term of so-called mixed-initiative approaches. It also seems to be a promising approach to combine such approaches with pattern-based authoring methods similar in spirit to those known from the object-oriented software engineering community [35], i.e., to provide users with predefined composition problems and proven solutions in the form of composition patterns that they "only" need to instantiate and apply to their problems.

The second problem area for the assembly and linking phase focuses on those architectural design decisions that must be taken when compiling business processes to IT systems.

Problem 10: Demonstrate how IT architectural knowledge and decisions are used within the compiler.

Today, these decisions are taken by IT architects mostly working with paper and pen. Decisions are not formally represented in tools and no decision-making support is available. Consequently, architectural decisions are not available in a form that they can really be used by the Business-IT systems compiler. Recent work by others and us has shown that architecturally decision making can be systematically supported and that decision alternatives, drivers and dependencies can be explicitly captured in tools and injected into a code-generating process [36–39]. By separating and validating the architectural decisions, design flaws can be more easily detected and a recompilation of a system for a different architecture is becoming more feasible.

With this list of ten specific research problems, the vision of a compiler for Business-IT systems is broken down into a specific set of key problems. We believe that a solution of these problems constitutes the essential cornerstones for such a compiler. The positioning of the problems within the various compilation phases makes it possible to tackle them systematically as well as study their relationships and dependencies, and to solve the problems under realistic boundary constraints.

We believe that the compiler vision is a key to overcome the most urgent problems in the BPM and SOA space. Today, BPM and SOA applications are built from business process models that were drawn in modeling tools that offer little analytical or pattern-based support. From the process analysis models, design models are created

by hand by manually translating and refining the information contained in the analysis model. Usually, the direct linkage between analysis and design gets lost during this step. Changes made at the design level are rarely reflected back at the analysis level. Commonly, the business processes are modeled in isolation. Their interdependencies and communication, their distributed side effects on shared data are rarely captured in models, but remain hidden in hand-written code. Thus, building the applications is expensive, resource-intensive, and often ad-hoc. The resulting BPM and SOA systems are hard to test, to maintain, and to change.

A compiler significantly increases the quality of the produced solution and provides clearer traceability. Approaches of manual translation are replaced by tool-supported refinement steps guided by detailed diagnostic information. When embedding the compiler into a development environment supporting the life cycle of process models in horizontal (distributed modeling) and vertical (refinement) scenarios, versions of the process models can be tagged, compared, and merged. Alternative views on the processes for different purposes can also be more easily provided.

The optimization of Business-IT systems with respect to their execution becomes possible, which can be expected to lead to systems with greater flexibility making it easier for businesses to follow the life cycle of process innovation.

5 Conclusion

In this paper, we proposed the vision of a compiler for Business-IT systems that takes business process specifications described at various degrees of detail as input and compiles them into executable IT systems. We defined ten research problems that have to be solved towards creating a compiler for Business-IT systems. Our vision allows us to position problems in a consistent and comprehensive framework that have previously been tackled in isolation. None of the presented research problems is new. In fact, many research projects have been initiated around them. However, as we tried to outline in the previous discussion, none of these projects has been truly successful, because the developed solutions commonly fail in practice: they do either not scale to the size of real-world examples, they do not provide users with the information that they need, or they rely on assumptions that do not hold in practice. However, many of these research projects have delivered interesting partial solutions that are worth to be preserved and integrated into a compiler for Business-IT systems. Consequently, many of these results have to be combined with novel "gap-closing" technology that still has to be developed and placed within the vision of the compiler. In many cases, the gap is in fact quite wide, requiring researchers to leave established solution approaches and develop much more than a small delta on top of existing research results.

The ten research problems have been defined at different levels of abstraction. Some are concrete, while others first have to be addressed at the conceptual level before they can be refined into a concrete set of problems. We believe that this mix makes the proposed problems particularly interesting and will enable researchers to drive progress in complementary strands of work. The positioning of these ten problems within the various compilation phases makes it possible for researchers to tackle them systemat-

ically, study their interrelationships, and solve the problems under realistic boundary constraints.

References

1. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F., Krämer, B.J.: Service-oriented computing: A research roadmap. In: Dagstuhl Seminar Proceedings on Service-Oriented Computing. (2006)
2. Jordan, D., et al.: Web services business process execution language (WSBPEL) 2.0. htpp://www.oasis-open.org/committees/wsbpel/ (2007)
3. OMG: Business Process Modeling Notation Specification. (2007) Version 1.1.
4. Reichert, M., Dadam, P.: ADEPT_Flex - supporting dynamic changes of workflows without losing control. Journal of Intelligent Information Systems **10**(2) (1998) 93–129
5. Rinderle, S., Reichert, M., Dadam, P.: Disjoint and overlapping dynamic changes: Challenges, solutions, applications. In: Int. Conference on Cooperation Information Systems (CoopIS-04). Volume 3290., Springer (2004) 101–120
6. Muchnik, S.: Advanced Compiler Design and Implementation. Morgan Kaufmann (1997)
7. Vanhatalo, J., Völzer, H., Leymann, F.: Faster and more focused control-flow analysis for business process models though SESE decomposition. In: 5th Int. Conference on Service Oriented Computing (ICSOC-07). Volume 4749 of LNCS., Springer (2007) 43–55
8. Vanhatalo, J., Völzer, H., Koehler, J.: The Refined Process Structure Tree. In: 6th Int. Conference on Business Process Management (BPM-08). Volume 5240 of LNCS., Springer (2008) 100–115
9. Johnson, R., Pearson, D., Pingali, K.: The program structure tree: Computing control regions in linear time. In: ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI-94). (1994) 171–185
10. Ananian, C.S.: The static single information form. Master's thesis, Massachusetts Institute of Technology (September 1999)
11. Valdes-Ayesta, J.: Parsing Flowcharts and series-parallel graphs. PhD thesis, Stanford University (1978)
12. Tarjan, R.E., Valdes, J.: Prime subprogram parsing of a program. In: 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, ACM (1980) 95–105
13. Ehrig, H., Engels, G., Kreowski, H.J., Rozenberg, G.: Handbook of Graph Grammars and Computing by Graph Transformation. Volume 2. World Scientific (1999)
14. Favre, C.: Algorithmic verification of business process models. Master's thesis, École Polytechnique Fédérale de Lausanne (August 2008)
15. Küster, J.M., Gerth, C., Förster, A., Engels., G.: Detecting and resolving process model differences in the absence of a change log. In: 6th Int. Conference on Business Process Management (BPM-08). Volume 5240 of LNCS., Springer (2008) 244–260
16. Baresi, L., Nitto, E.D., eds.: Test and Analysis of Web Services. Springer (2007)
17. Leymann, F., Roller, D.: Production Workflow. Prentice Hall (2000)
18. Verbeek, H., Basten, T., van der Aalst, W.: Diagnosing workflow processes using WOFLAN. The Computer Journal **44**(4) (2001) 246–279
19. Fu, X., Bultan, T., Su, J.: Analysis of interacting BPEL Web Services. In: 13th Int. Conference on the World Wide Web (WWW-04), ACM (2004) 621–630
20. Trainotti, M., Pistore, M., Calabrese, G., Zacco, G., Lucchese, G., Barbon, F., Bertoli, P., Traverso, P.: ASTRO: Supporting composition and execution of Web Services. In: 3rd Int. Conference on Service Oriented Computing (ICSOC-05). Volume 3826., Springer (2005) 495–501

21. Koehler, J., Vanhatalo, J.: Process anti-patterns: How to avoid the common traps of business process modeling. IBM WebSphere Developer Technical Journal **10**(2+4) (2007)
22. Koehler, J., Gschwind, T., Küster, J., Pautasso, C., Ryndina, K., Vanhatalo, J., Völzer, H.: Combining quality assurance and model transformations in business-driven development. In: 3rd Int. Symposium on Applications of Graph Transformations with Industrial Relevance (AGTIVE-07). Volume 5088 of LNCS., Springer (2008) 1–16
23. Gschwind, T., Koehler, J., Wong, J.: Applying patterns during business process modeling. In: 6th Int. Conference on Business Process Management (BPM-08). Volume 5240 of LNCS., Springer (2008) 4–19
24. Lee, J., Midkiff, S.P., Padua, D.A.: Concurrent static single assignment form and const propagation for explicitly parallel programs. In: 10th Int. Workshop on Languages and Compilers for Parallel Computing (LCPC-97). Volume 1366 of LNCS., Springer (1997) 114–130
25. Nebel, B., Bäckström, C.: On the computational complexity of temporal projection, planning, and plan validation. Artificial Intelligence **66**(1) (1994) 125–160
26. Massuthe, P., Reisig, W., Schmidt, K.: An operating guideline approach to the soa. AMCT **1**(3) (2005) 35–43
27. Lohmann, N., Massuthe, P., Wolf, K.: Operating guidelines for finite-state services. In: 28th Int. Conference on Application and Theory of Petri Nets (ICATPN-07). Volume 4546 of LNCS., Springer (2007) 321–341
28. Massuthe, P., Wolf, K.: An algorithm for matching nondeterministic services with operating guidelines. Int. Journal of Business Process Integration and Management **2**(2) (2007) 81–90
29. Ouyang, C., Dumas, M., ter Hofstede, A.H.M., van der Aalst, W.M.P.: From BPMN process models to BPEL Web Services. In: IEEE Int. Conference on Web Services (ICWS-06). (2006) 285–292
30. Sriganesh, R.P., Bose, G., Silvermanohn, M.: Mastering Enterprise JavaBeans 3.0. John Wiley (2006)
31. Rao, J., Su, X.: A survey of automated web service composition methods. In: 1st Int. Workshop on Semantic Web Services and Web Process Composition. Volume 3387 of LNCS., Springer (2004) 43–54
32. Hoffmann, J., Bertoli, P., Pistore, M.: Web service composition planning, revisted: In between background theories and initial state uncertainty. In: 22nd AAAI Conference on Artificial Intelligence (AAAI-07). (2007) 1013–1018
33. Bertoli, P., Hoffmann, J., Lécué, F., Pistore, M.: Integrating discovery and automated composition: From semantic requirements to executable code. In: IEEE Int. Conference on Web Services (ICWS-07), IEEE (2007) 815–822
34. Brogi, A., Popescu, R.: Automated generation of bpel adapters. In: 4th Int. Conference on Service-Oriented Computing (ICSOC-06). Volume 4294 of LNCS., Springer (2006) 27–39
35. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (1995)
36. Kruchten, P., Lago, P., van Vliet, H.: Building up and reasoning about architectural knowledge. In: 2nd Int. Conference on the Quality of Software Architectures (QoSA-06). Volume 4214 of LNCS., Springer (2006) 43–58
37. Jansen, A., Bosch, J.: Software architecture as a set of architectural design choices. In: 5th IFIP Conference on Software Architecture (WICSA-05), IEEE (2005) 109–120
38. Tyree, J., Akerman, A.: Architecture decisions: Demystifying architecture. IEEE Software **22**(2) (2005) 19–27
39. Zimmermann, O., Zduhn, U., Gschwind, T., Leymann, F.: Combining pattern languages and architectural decision models into a comprehensive and comprehensible design method. In: 8th IFIP Conference on Software Architecture (WICSA-08), IEEE (2008) 157–166