RZ 3705 (# 99715) 03/15/2008
Computer Science 18 pages

Research Report

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 Research GmbH
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

E-mail: koe@zurich.ibm.com

Towards a Compiler for Business-IT Systems
- A Vision Statement complemented with a Research Agenda -

Jana Koehler
Thomas Gschwind Jochen Küster Hagen Völzer Olaf Zimmermann

IBM Zurich Research Laboratory
8803 Rüschlikon, Switzerland
Research Report RZ 3705
March 2008

Summary: Business information systems or enterprise applications have continuously evolved into Business-IT systems over the last decades, directly linking and integrating Business Process Management (BPM) with recent technology evolutions such as Service-Oriented Architecture (SOA) concepts and Web services. Interestingly, many of these technological evolutions include revivals of areas that have been in the focus of academic research in the past. For example, 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 relies on techniques originating from model checking and protocol verification.

As such, many of the problems that arise with these new technologies have been solved at least in principle. Research in the BPM/SOA space, sometimes also called Service-Oriented Computing (SOC), has tackled many of the problems and proposed a variety of solutions. However, if we try to apply these “in principle” solutions, we are confronted with the failure of these solutions in practice. Either a solution technique does not scale to the real-world requirements or it relies on assumptions that are not satisfied by Business-IT systems. This failure of existing solutions is limiting a successful application of these new technologies, in particular for smaller business players.

As has been observed previously, research in this area is fragmented and does not follow a truly interdisciplinary approach. To overcome the fragmentation of the 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. As any classical compiler, the parsing, analysis, optimization, code generation and linking phases are supported and we describe a set of 10 research problems that we think must be solved in order to bring our compiler vision to reality. We argue that our vision provides a unique technological basis and foundation that enables a multi-disciplinary approach where existing techniques are combined with novel solutions filling in technology gaps. We position our vision within the life cycle of BPM and SOA applications to address the two interleaving trends of commoditization and innovation of business processes.

1. Introduction: Why a compiler?

Business processes are the set of activities including people, IT systems, and other machines acting on information and raw materials that 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 allows the process to be simulated.

Service-Oriented Architectures (SOA), which are often implemented as Web services, is an architectural style that allows IT systems to be integrated in a standard way, which lends itself to efficient implementation of business processes. SOA also enables efficient process re-engineering since the use of standard programming models and interfaces makes it much simpler to change the manner 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, notably the performance, or some other aspect of the process.

In spite of 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, automated approaches that allow the full exploitation of the BPM life cycle to make rapid improvements in business processes do not exist. 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 [32]:

"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

[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.

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 report presents a concrete technological vision and foundation to overcome the fragmentation of research in the BPM/SOA/SOC area. We propose the vision of a compiler for Business-IT systems that takes business process specifications described in 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 assembler to abstract programming languages from which machine-generated code would then be produced. The emerging new process-oriented programming languages are examples of languages that are input to such a compiler. In particular, BPMN 2.0 [33] is 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, a direct execution means to follow an interpreter-based approach with all its shortcomings.

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.

As any classical compiler, a compiler for Business-IT systems works in five phases. While we consider the parsing and lexical analysis phase as being solved by our previous and current work, we explore in this report 10 specific research problems that address key problems for the subsequent phases. The list below briefly summarizes the 10 problems. In the subsequent section, they will be discussed in more detail.

1. Lexical analysis and parsing

   We developed the Process Structure Tree (PST) as a unique decomposition of a workflow graph into a tree of fragments that can be computed in linear time. The PST plays the same role in the Business-IT systems compiler than 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

   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.
   The Business Process Modeling Notation (BPMN) as well as the Business Process Execution Language (BPEL) are the most relevant languages that the compiler must handle today. The precise and efficient translation from the unstructured BPMN language to the structured BPEL language is not yet completely solved and also requires the semantics of both languages to be formally defined.

4. Optimization

   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

   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. A compiler, however, should be able to perform the composition and linking of service components without requiring such annotations by relying on a semantic analysis of their descriptions.

   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 10 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 10 research problems has significant impact on the integrity, improved agility and higher automation within BPM/SOA and Service-Oriented Computing.

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.

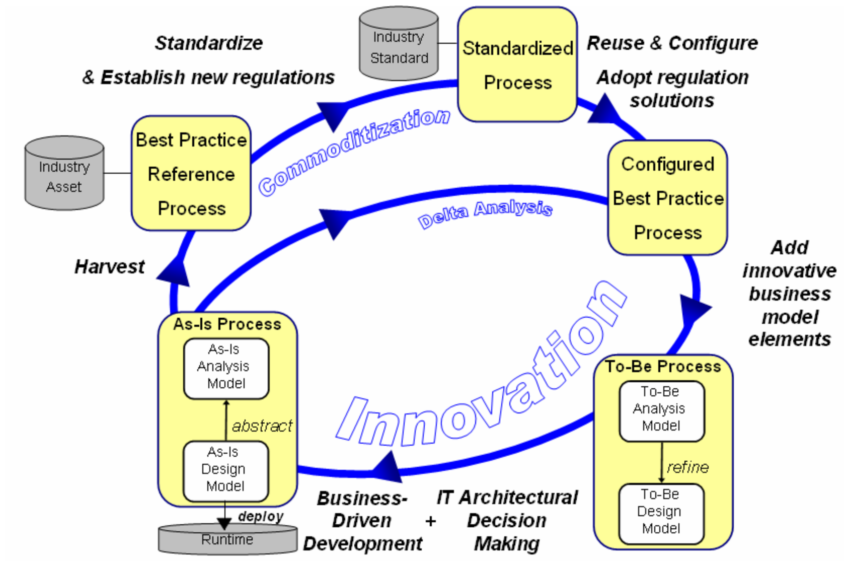
2. The Details of our Vision

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 business processes more flexible, while retaining their integrity and compliance with legal regulations continue to drive technology advances in this space. A recent prominent failure in process integrity led to the billion Euro loss at Société Générale.

Based on the development of new standards, algorithms, and tools, it will be possible to create well structured business process models that are free of errors and that can be simulated and studied to determine their performance, policy compliance and other characteristics. The aim of this vision is to contribute to these developments. We envision a compiler that enables users to specify process models at different levels of abstraction. The compiler analyses these models, provides detailed diagnostic information to the user helping them to correct and further refine the models until they can be compiled into executable code.

The figure below 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.

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 some industry represent the state of the art for 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 resulting from the innovation are shown as the To-Be processes, which have to both accommodate commoditization requirements and 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 [17,18].

This report 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. The report investigates these problems

[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 not in the focus of this document.

from a strict technology point of view and addresses 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 interesting, but are outside the scope of this vision as are a study of the economic or social effects of what has been discussed above.

The vision of a compiler for Business-IT systems 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 assembler 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) or the recent proposal by IBM, Oracle, and SAP for the Business Process Modeling Notation 2.0 (BPMN) are examples of languages that are input to such a compiler. In particular, BPMN 2.0 is 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. Due to its formally defined execution semantics and the fact that BPMN 2.0 fully subsumes BPEL, it is also directly executable. However, directly executing BPMN diagrams corresponds to a simple interpreting approach with all its limitations and thus, is not desirable.

The envisioned compiler does not take a high-level business process model, magically add all the missing information pieces, and then compile 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 from the formal description of the system, the code can be completely generated. This also applies to a classical compiler that can parse and analyze (and partially compile) an incomplete program that is in an intermediate state. In particular, the analytical information that a compiler returns to the users while there are developing the software, is extremely beneficial.

With the compiler approach, we also plan 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.

A compiler can 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. Our main goal is to understand how such a compiler for Business-IT systems works. At its core, we see the compilation of business process models, which constitutes a well-defined 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 1997, "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 solving the business problems and addressing 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:

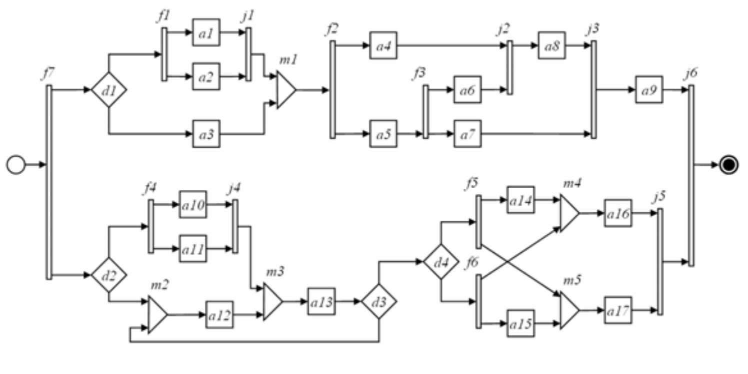
1. Lexical analysis and parsing
2. Structural and semantic analysis
3. Translation and intermediate code generation
4. Optimization
5. Final assembly and linking and further optimization

The core of our vision is to develop fundamental techniques for a Business-IT systems compiler. This includes contributions to the analysis, translation, optimization and assembly phases, while we consider the parsing problem as solved as is explained below.

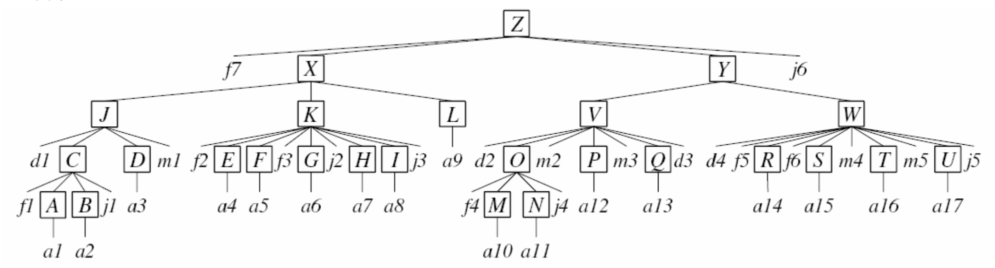
State-of-the-art and objectives: In the following, we review selected related work in the context of the five phases. The review will not be a comprehensive survey of the state of the art. We focus on where we stand in our research with respect to the compiler for Business-IT systems and point by example to existing work in various fields of computer science that is relevant for the five phases. Much other related work exists – it was not our goal to provide a comprehensive overview.

2.1 Parsing

The parsing problem for business process models has not yet been widely recognized as an important problem by the BPM community. The figure below shows a typical workflow graph underlying any business process model. It includes activities a_i, decisions d_i, merges m_i, forks f_i and joins j_i. 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.



In our own research, we developed the Process Structure Tree (PST) [1], 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. The figure below shows the PST for the process model above. By applying techniques from the analysis of program structure trees [5, 13] to business process models a unique decomposition of process models into a tree of fragments can be obtained.


Z
f7
X
Y
j6
J
K
L
V
W
d1 C D m1 f2 E F f3 G j2 H I j3 a9 d2 O m2 P m3 Q d3 d4 f5 R f6 S m4 T m5 U j5
f1 A B j1 a3 a4 a5 a6 a7 a8 f4 M N j4 a12 a13 a14 a15 a16 a17
a1 a2 a10 a11

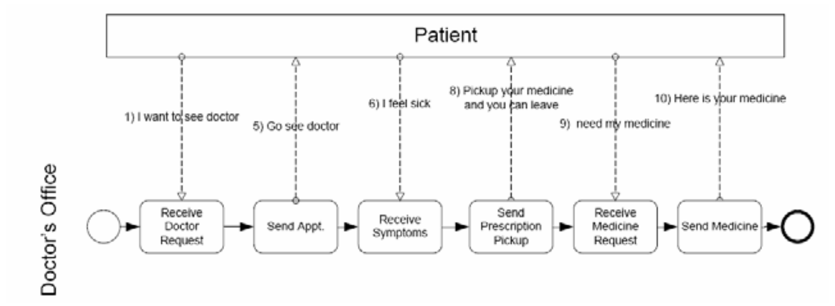
In our current work, we improve and generalize the fragment-tree computation further by combining methods from graph theory with those known from the theory of programs leading to a refined PST [30] that is built with more fine-grained fragments in linear time. This is a significant improvement compared to approaches that use graph grammars to parse the visual language, which is exponential in most cases [34].
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 it is more important to concentrate on the other remaining phases and validate that the PST is indeed as powerful as the AST.

2.2. Structural and semantic analysis

So far, we have built two applications: a) a control-flow analysis [1] and b) an approach to the structural comparison and difference analysis of process models [2]. 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 realization, 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 the Business Process Modeling Notation (BPMN) is shown in the figure below, taken from the BPMN 1.1 specification [26]. 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.[3]



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. This leads us to the formulation of our first research problem:

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

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., [4]. To the best of our knowledge, compiler techniques have not yet been considered so far.

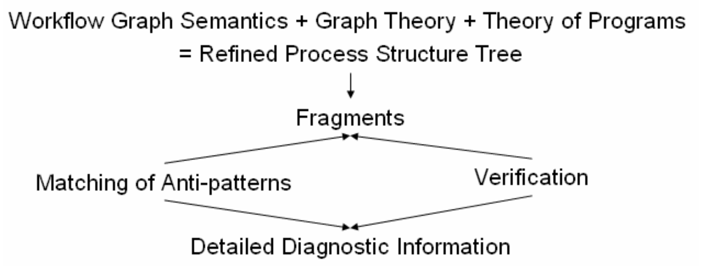
Verification of business processes is an area that has a long tradition. Locating errors in business processes is important in particular because of the side effects that processes have on data. Processes that do no terminate correctly because of deadlocks or processes that exhibit unintended execution traces due to a lack of synchronization often leave data in inconsistent states [35]. Common approaches to process verification usually take a business process model, translate it into a Petri-net or state-based encoding and then run a Petri-net analysis tool or model checker on the encoding. Examples are the Woflan tool [3] or the application of SMV or Spin to BPEL verification, e.g., [28, 29]. In principle, these approaches make it possible to detect errors in business processes. However, there are severe limitations that prevented the adoption of the proposed solutions in industrial tools:

- the encodings are of exponential size compared to the original process model,
- the verification tool does not give detailed enough diagnostic information in such a way that it allows an end user to easily correct errors,
- the approaches often make restricting assumptions on the subclass of process models that they can handle.

[3] Note that the clarification and formal definition of the semantics of BPMN is another focus area of our work. We contributed a formal definition of the execution semantics of BPMN to the submission by IBM, Oracle, SAP for the BPMN 2.0 Request for Proposals by the OMG. 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.

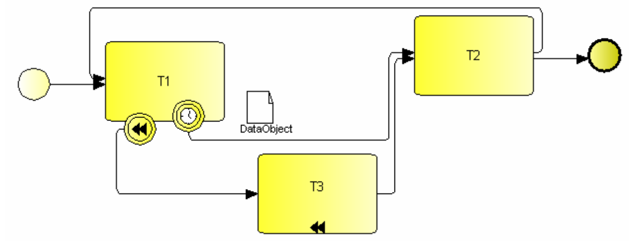
Consequently, the currently available solutions are only partially applicable in practice due to their long runtimes, the lack of suitable diagnostic information, and the restrictions of the defined encodings.

In our own work, we have followed a different approach. First, we analyzed hundreds of real-world business processes and identified commonly occurring anti-patterns [6]. Second, we developed the PST and showed how it can be used to speed up the verification of a process orchestration model [1]. 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 (or the presence of deadlocks or lack of synchronization errors) can be verified by matching them against patterns and anti-patterns. Only a small number of fragments remains, which requires the application of verification methods such as model checking. Furthermore, the size of the fragments 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 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. The following figure summarizes the approach.


Workflow Graph Semantics + Graph Theory + Theory of Programs
= Refined Process Structure Tree
Fragments
Matching of Anti-patterns
Verification
Detailed Diagnostic Information

Implementation of the work showed that the soundness of large business process models can be completely analyzed in within a few seconds – as such, the technology can be made available to users of modeling tools where they obtain instant feedback. Smart editing macros [7] can be provided to users to help them correct the detected modeling errors easily. These macros take advantage of the fine-grained diagnostic information and the PST to support users in accomplishing complicated editing steps in a semi-automatic 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 orchestrations that are enriched with the description of error-handling or compensation flow. Second, no sufficient data-flow analysis techniques are yet available to analyze business processes. The next figure illustrates two more problems that we want to look at in more detail.



A repetitive process comprising task T1 followed by task T2 is executed. T1 changes the state of some data object. 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 later merge back 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. This leads us to the formulation of the second research problem:

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

Data-flow analysis for unstructured business process models is a largely unsolved problem. The figure above 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. This leads us to the formulation of the third research problem:

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

An immediate candidate is Concurrent Single Static Assignment [31] 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, because the side effects of the flows on the data must be investigated.

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 [8] 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). Thus, we formulate it as the fourth research problem:

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

With this, we have the research problems defined that we want to tackle with respect to process orchestrations for the analysis phase.

For process choreographies, we are mostly interested in termination problems. Can two processes successfully communicate with each other such that both terminate? 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 our simple example above, interesting questions can be asked and answered. For example, which flow constraints must the abstract Patient process satisfy such that a successful communication is possible?

Previous work, notably the research on operating guidelines [9, 11-12] is trying to answer such 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 detailed diagnostic information when verifying choreographies. This leads us to the formulation of our fifth research problem:

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

2.3. Translation and intermediate code generation

In the translation phase, we want to essentially tackle one especially challenging problem, namely the translation from unstructured BPMN to structured BPEL. An example of relevant related work is presented in [10]. It exploits a form of structural decomposition that resembles some ideas similar to the PST, but is not as well-defined. With our contribution of the execution semantics for BPMN 2.0, we also have developed initial insights into the classes of BPMN diagrams that are translatable into structured BPEL that we want to work out further. This leads us to the definition of our sixth research problem.

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

2.4 Optimization

The optimization phase for the Business-IT systems compiler is a completely unaddressed research area so far. Classical process optimization from BPM only 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 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. Such optimizations have for example been studied in the context of J2EE applications [14]. We have some initial insights, but our main goal here is to clarify what can and should happen during the optimization phase. This leads us to the definition of our seventh research problem.

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

2.5. Final assembly and linking and further optimization

For the assembly and linking phase, two problem areas are within the focus of our interest. First, we are interested in studying some well-defined synthesis problems. In the literature, two instances of process synthesis problems have been investigated so far: First, the Web service composition problem, which is mostly tried to be solved using AI planning techniques [15, 16]. 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 not really satisfied in the real world. 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 attempted by combining model checking techniques with more or less intelligent “guess” algorithms [19, 20]. Adapter synthesis tries to resolve problems in a faulty choreography by generating an additional process that allows existing partners to successfully communicate. 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. As such an infinite search space is opened up and the methods are inherently incomplete. In addition, the synthesized adapters must be verified, because the synthesis algorithms can usually not be guaranteed to be correct. There is thus a wide gap between the currently proposed techniques and the needs of a practically relevant solution.

A first aim must therefore be to formulate practically relevant variants of the service composition and adapter synthesis problem and then work out solutions to these problems that make realistic assumptions, scale to real-world problems and are accepted by the commercial as well as the academic world. This leads us to the formulation of our eighths and ninth research problems:

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

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

An initial goal for these 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 [27], i.e. 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 the architectural design decisions that must be made when compiling business processes to IT systems. 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, [21-25]. 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. This leads us to the definition of our tenth and last research problem:

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

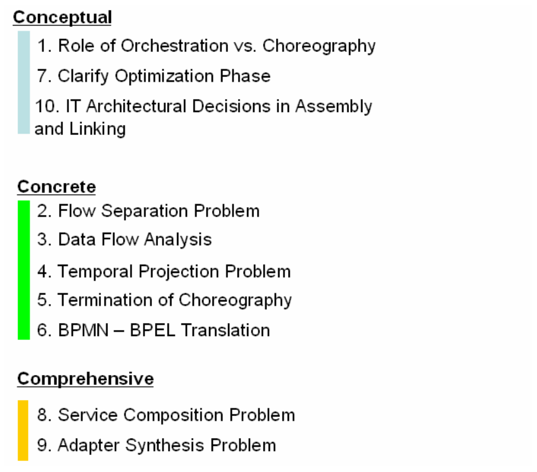
With this list of 10 specific research problems, the vision of a compiler for Business-IT systems is broken down into a specific set of key problems. We believe that a solution of these problems constitutes the essential cornerstones for such a compiler. The positioning of the problems within the various compilation phases makes it possible to tackle them systematically, study their interrelationships, and 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 version merging for process models in horizontal (distributed modeling) and vertical scenarios (refinement), versions of the process models can be tagged, compared, and merged. Alternative views on the processes for different purposes can 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.

3. Summary

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, because 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 of research results.

The 10 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. Furthermore, we introduced two comprehensive areas of research around the two synthesis problems where we believe that they not only require a sophisticated combination of various techniques developed in different fields of computer science, but where we have the impression that these problems cover a large class of related subproblems. Solutions to the concrete problems that we defined are a prerequisite for making progress on the synthesis problems. We believe that this mix makes the proposed problems particularly interesting and would enable researchers to drive progress in complementary strands of work.



Conceptual
1. Role of Orchestration vs. Choreography
7. Clarify Optimization Phase
10. IT Architectural Decisions in Assembly and Linking

Concrete
2. Flow Separation Problem
3. Data Flow Analysis
4. Temporal Projection Problem
5. Termination of Choreography
6. BPMN – BPEL Translation

Comprehensive
8. Service Composition Problem
9. Adapter Synthesis Problem

We presented our initial thoughts on the Business-IT systems compiler to various researcher groups and the feedback was very encouraging. It would be nice to make compiler experts excited in the BPM/SOA compilation problem, which has not been widely recognized so far. The main challenge, however, does not lie in understanding the state of the art in compiler theory, but in applying the known techniques to the new problems and combining them with novel solutions. Our current results have been built on combining results from compiler theory, graph theory, the theory of programs, formal languages, Petri nets and software engineering.

Bibliography
[1] J. Vanhatalo, H. Völzer, F. Leymann: Faster and more focused control-flow analysis for business process models though SESE decomposition, ICSOC-07. LNCS 4749, pages 43-55, 2007.
[2], J. Küster, C. Gerth, A. Foerster, G. Engels: Process Merging in Business-Driven Development. ZRL Research Report RZ 3703, February 2008.
[3] H.M.W. Verbeek , T. Basten, W.M.P. van der Aalst: Diagnosing Workflow Processes using Woflan, The Computer Journal, 44(4):246-279, British Computer Society, 2001.
[4] L. Baresi, E. Di Nitto (Eds.): Test and Analysis of Web Services, Springer, 2007.
[5] R. Johnson, D. Pearson, K. Pingali. The program structure tree: Computing control regions in linear time. In Proceedings of the ACM SIGPLAN’94 Conference on Programming Language Design and Implementation (PLDI), pages 171–185, 1994.
[6] J. Koehler, J. Vanhatalo: Process Anti-Patterns: How to Avoid the Common Traps of Business Process Modeling. IBM Websphere Developer Technical Journal, Issues 10.2 and 10.4, February and April 2007.

[7] J. Koehler, T. Gschwind, J. Küster, C. Pautasso, K. Ryndina, J. Vanhatalo, H. Völzer: Combining Quality Assurance and Model Transformations in Business-Driven Development. International Workshop and Symposium on Applications of Graph Transformation with Industrial Relevance, LNCS, forthcoming 2008.
[8] B. Nebel, C. Bäckström: On the Computational Complexity of Temporal Projection, Planning, and Plan Validation. Artif. Intell. 66(1): 125-160,1994.
[9] P. Massuthe, W. Reisig, and K. Schmidt: An Operating Guideline Approach to the SOA. AMCT, 1(3):35-43, 2005.
[10] C. Ouyang, M. Dumas, A. H.M. ter Hofstede, W. M.P. van der Aalst: From BPMN Process Models to BPEL Web Services, ICWS-06), pages 285-292, 2006.
[11] N. Lohmann, P. Massuthe, K. Wolf: Operating Guidelines for Finite-State Services. ICATPN-07, LNCS 4546, pages 321-341 2007.
[12] P. Massuthe and K. Wolf: An Algorithm for Matching Nondeterministic Services with Operating Guidelines. IJBPIM, 2(2):81-90, 2007.
[13] C. S. Ananian: The static single information form, Master's thesis, MIT 1999.
[14] R. P. Sriganesh, G. Brose, and M. Silvermanohn: Mastering Enterprise JavaBeans 3.0, John Wiley, 2006.
[15] J. Rao, X. Su: A Survey of Automated Web Service Composition Methods, First International Workshop on Semantic Web Services and Web Process Composition, LNCS 3387, pages 43-54, 2004.
[16]. J. Hoffmann, P. Bertoli, M. Pistore: Web Service Composition as Planning, Revisited: In Between Background Theories and Initial State Uncertainty, AAAI-07, pages 1013-1018, 2007.
[17] M. Reichert, P. Dadam. ADEPT_Flex - Supporting Dynamic Changes of Workflows Without Losing Control. J. Intell. Inf. Syst., 10(2):93-129, 1998.
[18] S. Rinderle, M. Reichert, P. Dadam. Disjoint and Overlapping Process Changes: Challenges, Solutions, Applications, CoopIS-04, LNCS 3290, pages 101-120, 2004.
[19] P. Bertoli, J. Hoffmann, F. Lécué, M. Pistore: Integrating Discovery and Automated Composition: from Semantic Requirements to Executable Code. ICWS-07, pages 815-822, 2007.
[20], A. Brogi, R. Popescu: Automated Generation of BPEL Adapters. ICSOC-06, LNCS 4294, pages 27-39, 2006.
[21] Kruchten P., Lago P., van Vliet H, Building up and reasoning about architectural knowledge. QoSA-06, LNCS 4214, pages 43-58, 2006.
[22] Jansen, A. and Bosch, J. 2005. Software Architecture as a Set of Architectural Design Decisions, Wicsa-05, IEEE Computer Society, 2005.
[23] Tyree, J., Akerman, A., Architecture Decisions: Demystifying Architecture, IEEE Software 22(2): 19-27, 2005.
[24] O. Zimmermann, J. Koehler, F. Leymann: Architectural Decision Models as Micro-Methodology for Service-Oriented Analysis and Design, SEMSOA CEUR-WS.org/Vol. 244, 2007.
[25] O. Zimmermann, U. Zdun, T. Gschwind, F. Leymann: Combining Pattern Languages and Architectural Decision Models into a Comprehensive and Comprehensible Design Method, WICSA-08. IEEE Computer Society, 2008.
[26] Business Process Modeling Notation Specification1.1, OMG Final Adopted Specification, 2007.
[27] E. Gamma, R. Helm, R. Johnson, and J. Vlissides: Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley, 1995.
[28] X. Fu, T. Bultan, J. Su: Analysis of interacting BPEL Web services. WWW-04, pages 621-630, ACM, 2004.
[29] M. Trainotti, M. Pistore, G. Calabrese, G. Zacco, G. Lucchese, F. Barbon, P. Bertoli, P. Traverso: ASTRO: Supporting Composition and Execution of Web Services. ICSOC-05, LNCS 3826, pages 495-501, 2005.
[30] J. Vanhatalo, H. Völzer, J. Koehler: The Refined Process Structure Tree, forthcoming 2008.
[31] J. Lee, S. P. Midkiff, D. A. Padua: Concurrent Static Single Assignment Form and Constant Propagation for Explicitly Parallel Programs. LCPC-97, LNCS 1366, pages 114-130, 1997.
[32] M. P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann, B. J. Krämer: Service-Oriented Computing: A Research Roadmap. Dagstuhl Seminar Proceedings on Service-Oriented Computing, 2006.
[33] BPMN 2.0 Submission by IBM, Oracle, SAP for the BPMN 2.0 Request for Proposals by the OMG, 2008.
[34] H. Ehrig, G. Engels, H.-J. Kreowski, G. Rozenberg: Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 2, 1999.
[35] F. Leymann, D. Roller: Production Workflow, Prentice Hall, 2000.