

A Conceptual Framework for Composition in Business Process Management

Ingo Weber, Ivan Markovic, and Christian Drumm

SAP Research, Karlsruhe, Germany
{firstname.lastname}@sap.com

Abstract. In this work, we present a conceptual framework for deriving executable business process models from high-level, graphical business process models based on the paradigm of Service-Oriented Architectures and Semantic Web technology. We hereby envision a direct, but implicit link from a business analyst's view on a process model to its execution driven by an IT system. This linkage enables the derivation of an execution-level model for newly created business process models as well as adaptation of the execution model after re-engineering processes, possibly under certain re-design goals (such as quality, cost, execution time, flexibility, or others).

The framework includes a component architecture and an algorithm that describes how to combine executable artifacts, such as (Semantic) Web services, in order to find an implementation that matches a given business process model. An extensible set of criteria can be used for validating the composition.¹

1 Introduction

One of the promises of the Service-Oriented Architecture (SOA) paradigm is increased flexibility by coupling components loosely. In the area of enterprise applications, Web services can accordingly be used to encapsulate business functionality. The loose coupling of business functions aims to increase the flexibility in executable business processes: the process flow can then be separated to a large degree from the implementation of business functions. This increased flexibility in process modeling and enactment is highly desired: in today's business world the business models² are changed at an ever increasing frequency in order to react to changing market situations. Allowing for a swift adaptation of operational business processes is a key requirement for modern enterprise application systems.

Among other challenges, the question of how to leverage process modeling at the execution level is a key question for the uptake of SOA in enterprise software solutions [15]. This paper addresses the question how changes can be propagated from the process modeling level to the execution level. That is, if a new process model is created or an existing model is changed, then the respective implementation

¹ This work has in part been funded through the European Union's 6th Framework Programme, within Information Society Technologies (IST) priority under the SUPER project (FP6-026850, <http://www.ip-super.org>).

² Under business model we refer to the concept of how an enterprise makes money.

has to be created or modified to reflect the changes in the process model³ at the execution level (i.e. the information technology (IT) infrastructure).

A solution to this problem would provide an enterprise with the opportunity to react faster to changes in their respective environments, e.g., changes in regulations or business models, and would allow for leveraging small windows of opportunities. Such a solution would basically serve for re-arranging available capabilities in an enterprise in order to meet a changed business goal, while missing capabilities or services can be identified. An identified lack of capabilities could potentially be compensated by extending the application infrastructure or outsourcing concerned parts of a process.

We envision a solution that is based on the novel combination of a number of known techniques, namely Web service composition, discovery, mediation, and orchestration, structural process translation, model checking and process validation, as well as machine-accessible semantics. Ontologies⁴ aim at making the semantics, i.e. the meaning of terms in a domain of discourse, machine-accessible, which enables reasoners to infer logical conclusions from the presented facts. This functionality can for example be applied to the discovery of Web services. Also, by modeling the relevant domain as an ontology that captures the potential states of the world, state-based composition approaches [3][13] can be employed. The output of the composition of Web services is then expressed as an orchestration of Web service calls, e.g., in Business Process Execution Language (BPEL) [1]. Ultimately, model checking and process validation add to the approach by checking the output under various criteria. The advantage over performing those functions on a high-level process model lies in the increased degree of formalism of the executable process. This paper presents a coarse-grained algorithm for applying these techniques to the given problem and describes the particularities of each point where they are used. The modular approach of the composition component enables different usage scenarios of the components and flexible inclusion of additional ways to automatically validate and provide feedback for the derived executable process model.

The presented work can be used as follows: A business expert models a business process on a high level⁵ of abstraction, which should be made executable on the available technical infrastructure. For this purpose, the composition component is called, which attempts to combine available executable artifacts and returns an executable process model or a failure note. The suggested executable process can subsequently be validated or directly get deployed for enactment. Benefits of this approach are increased reuse of artifacts, easier accessibility of the process space in an organization, increased flexibility through simpler change management, the potential for a more fine-grained evaluation of the validity, correctness, and compliance of a process model under various viewpoints, lower costs in maintaining the enterprise application infrastructure, and more.

³ Note that the opposite direction of the problem is also of high interest: How to adapt a process model to changes on the execution level.

⁴ We here refer to ontologies in notations such as OWL [25] and WSMO [20].

⁵ Commonly used graphical notations for business processes on this level are for instance the Business Process Modeling Notation (BPMN), the Event-driven Process Chains (EPC), and UML Activity Diagrams.

This paper describes a conceptual framework for composition in the described context. It builds upon the requirements analysis formulated in [18] and contains an outline of the most important components to instantiate the framework as well as an algorithm that describes the interplay between those components. The most relevant functions span from the discovery of artifacts, their actual composition, the compatibility of the data exchange between the artifacts (handled through mediators), to the validation of the composed process. As the paper addresses our current work, the granularity of the description of each individual building block is rather high-level. The goal is to formulate on an abstract level how the different techniques can be brought together in order to realize the larger vision. However, the current status of the work is purely conceptual.

The remainder of the paper is organized as follows: The following section describes the addressed problem in more detail. The conceptual framework in Section 3 explains the presented solution in terms of a component overview, a composition algorithm, and a further investigation of the usage of discovery, mediation, and validation techniques. Section 4 examines related work and Section 5 concludes.

2 Problem Description

When modeling a business process today, the modeler usually creates the process manually in a graphical tool. The outcome is a process model that reflects the business expert's view on real-world activities or a to-be process. Subsequently, this process model is implemented by IT experts - or, rather, its control and data flow are mapped to implemented artifacts in information systems. The relationship between the business-level model and its IT-level implementation is oftentimes weak. Consequently, the process implementation can deviate substantially from the model, and changes on either one of the levels cannot be easily propagated to the respective other level.

The approach of automated composition which we pursue attempts to bridge the gap between the modeled processes and their implementation by finding program parts which can be used for the implementation of a process model and defining their usage in an executable process model. For this vision to become reality, several needs must be met: Application program fragments with business functionality must be available for remote invocation as encapsulated and executable IT artifacts that can be enacted through an IT infrastructure. Examples for such executable IT artifacts include: Web services with simple and complex interfaces, partial processes and sub-processes.

In order to allow the automation of tasks such as composition, these artifacts are annotated with formal, machine-accessible semantics. In particular, we assume that the executable artifacts are annotated by semantic descriptions of their functionality and non-functional properties, e.g., by linking the meaning of the used terms to the content of ontologies which model the domain of interest. In addition, we assume that process tasks are annotated with goals as formalizations of desired task functionalities and, optionally, quality requirements. This allows a composition component to find suitable artifacts in a repository and evaluate their applicability and compatibility in the context of other artifacts.

While other recent work addressed similar issues, there is a notable difference to this paper: we are looking at composition from the viewpoint of an enterprise, not an end user. While business processes in business-to-consumer (B2C) scenarios can be a point of contact with end users, service composition on the consumer's end is not the focus of our work. Our work is rather placed in the context of enterprise application software and enterprise application integration. In the scope of this work, the final results must be correct and compliant with both internal policies and external regulations such as Basel II or Sarbanes-Oxley (SOX). Thus, the focus is on design-time composition in a known domain, which simplifies the problem in two ways: Firstly, design-time composition can always be checked and approved manually before deploying and enacting it. Thus, unanticipated side effects⁶ can be avoided because in the manual control step the decisions from the automated composition can be overruled. And secondly, in our known domain - the enterprise - we can assume to own the artifacts and can thus enforce a uniform way and formalism of their description.

On the above basis, the problem addressed by this paper can be restated in the following way: Given a high-level process model and a set of previously modeled artifacts in a repository, all semantically annotated, a composition approach should come up with the required set of artifacts from the repository, orchestrating the artifacts in a way that reflects the business process model's structure and business semantics.

In preliminary work [18] we examined the requirements on a solution of this problem in more depth, providing a list of 14 such requirements. In this paper, we describe a conceptual framework as a general solution strategy, which does not yet address all of the listed requirements completely. The framework rather serves as an extensible base structure on which solutions to the individual requirements from [18] can be combined. The following section describes the framework.

3 Conceptual Framework for BPM Composition

The framework described in the following addresses the problem laid out in the preceding section. It describes an architecture in terms of the required components (Fig. 1), an algorithm explaining the interplay between the components, and the most important functions.

3.1 Component Overview and Explanation

The Process Modeling Environment is used for designing process models and serves as the Graphical User Interface (GUI) to the rest of the architecture. In this modeling environment, the user must be provided with a convenient way to attach semantic annotations to his process models, which are part of the necessary input for the composition approach. At any point during the modeling phase or after finishing the design of the process, the user may request the composition of an executable

⁶ Cf. [7] for such a side effect: a composite process that satisfies the pre-condition of having a credit card prior to the execution of a service by directly applying for a credit card.

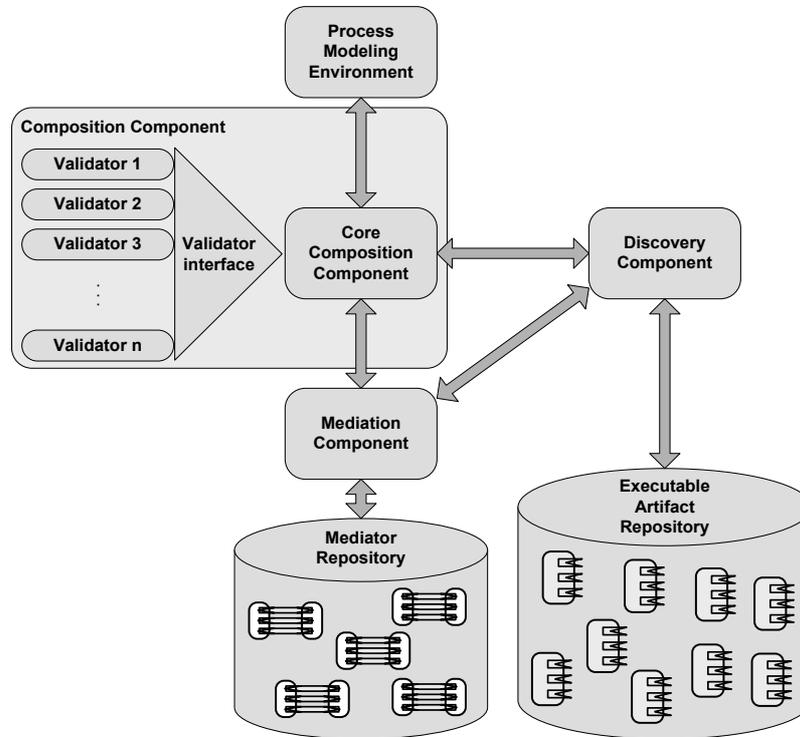


Fig. 1. Component architecture and interactions

process for his process model or parts thereof, which triggers an interaction between the Modeling Environment and the Composition Component.

The Executable Artifact Repository stores descriptions of the available artifacts. For each artifact this repository contains the description of its functionality, its non-functional properties and in particular how it can be accessed.

The Discovery Component serves as a clever interface to the Executable Artifact Repository, in that it answers simple and complex requests for artifacts by matching requests to semantic descriptions of artifacts in the repository.

The Mediation Component provides a simple querying interface for the **Mediator Repository**. It enables the other components to retrieve the available mediators for a given mediation problem. The Mediator Repository contains descriptions of the available mediators.

The Composition Component interacts with the Process Modeling Environment and the Discovery Component in order to provide the composition of executable artifacts. It holds the Core Composition Component, which implements the composition algorithm described below and handles the interactions with the other components, along with a Validator Interface and the required Validator Plug-Ins. Validation is

desired in order to provide an achievable degree of automatic assertion. The suggested structure allows for flexibly plugging in Validators with respect to the current context of the application.

3.2 Composition Algorithm

After describing the high-level functionality of the components involved in our framework we will in this section focus on the details of the composition component. The composition component performs the following steps in order to provide the modeling environment with the desired functionality of composing an executable process for a given business process model. The flowchart in Figure 2 corresponds to this algorithm:

1. Identification of existing target-process parts (e.g., in re-engineering, parts of a process may stay unchanged, which implies that the respective executable process does not necessarily have to be changed in those parts, either.)
2. Translating the process structure, which may be necessary, e.g., if there is a change in the underlying process description paradigm⁷.
3. For each task / step in the source process: derivation of executable artifacts
 - 3.1. Discovery of single artifacts that implements a task. Pre-existing research results for service discovery such as [8][9][11][16][17] can be adapted for this purpose.
 - 3.2. If no single artifact can achieve the required functionality:
 - 3.2.1. Composition of artifacts for implementing a single task. Here, known approaches to Web service composition, such as [2][3][10][12][13] can be adapted.
 - 3.2.2. Consistency checking of the composed artifacts on the level of this single task implementation (data flow, detection of inconsistencies such as goal invalidation, adherence to policies and regulations...). Note that this step can potentially be integrated to a degree with the previous step.
4. Combination of the task implementation to form the complete executable process. If an inconsistency is detected, another solution for affected task implementations is searched by calling step 3 again.
 - 4.1. Global control flow vs. local (artifact-level) constraint checking. Potentially there are inconsistencies between the global control flow as defined in the process model and constraints over the artifacts implementing the individual tasks. They may be resolved by adding additional ordering constraints over certain activities. If the problem is not solvable in another way, the task implementations are changed.
 - 4.2. Validation through the validator plug-ins, e.g., sequentially based on a prioritization.

⁷ Many high-level process modeling notations are graph-based, while execution-level process descriptions as BPEL are often block-based. In this step, the graph-based structure, i.e., the control flow “skeleton” of the process, would be translated to an according block-based process structure. [22] and [21] deal with such translations.

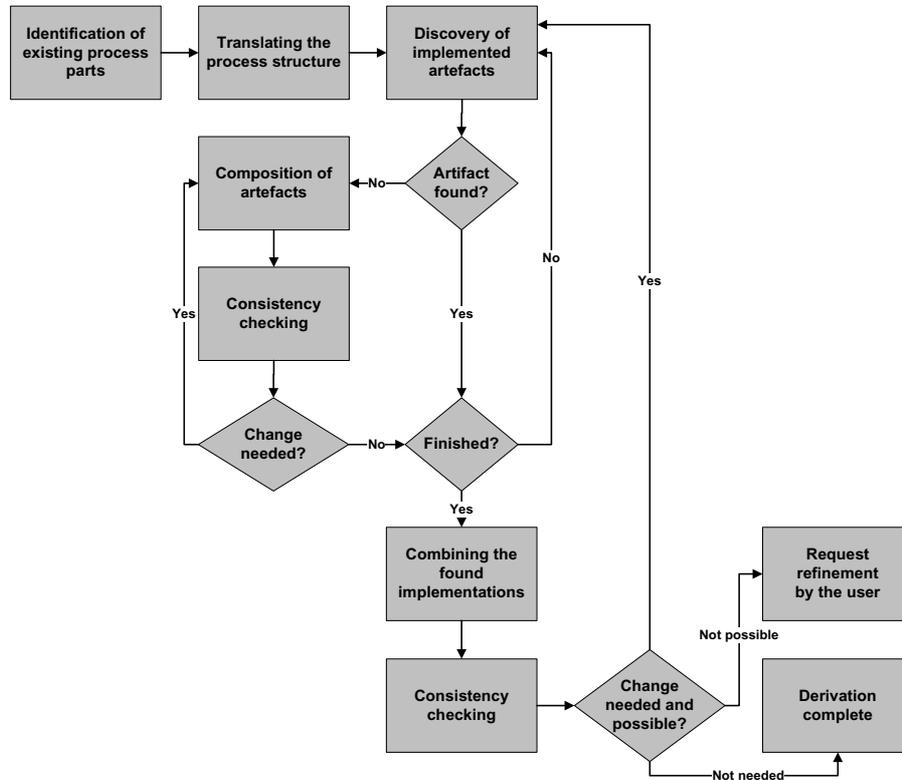


Fig. 2. Flowchart depicting the Composition Algorithm

In order to explain the steps of the algorithm in a more tangible way, Figures 3 (a) – 3 (d) depict how the resulting composed process evolves with the steps of the algorithm. Fig. 3 (a) shows a BPMN process model with tasks, connectors, and events. Fig. 3 (b) presents the same process after discovering single services that can implement a full task in the process (step 3.1 in the algorithm). Fig. 3 (c) shows additionally groups of services that jointly implement tasks (as composed together by step 3.2.1 in the algorithm). Finally, Fig. 3 (d) depicts the process in terms of only the services after the combination (step 4 in the algorithm).

3.3 Underlying Composition Approaches

In step 3.2.1 of the Algorithm, we plan to build on existing approaches to Web service composition, such as AI Planning⁸. The applicability of this technology for service composition has been examined, e.g. in [3][10][13], and is rather well understood. Certain approaches can be adapted for the usage in the algorithm above and extended towards directly handling a subset of the requirements from [18].

⁸ An overview over Artificial Intelligence (AI) planning can be found in [14], recent publications in the field at [3].

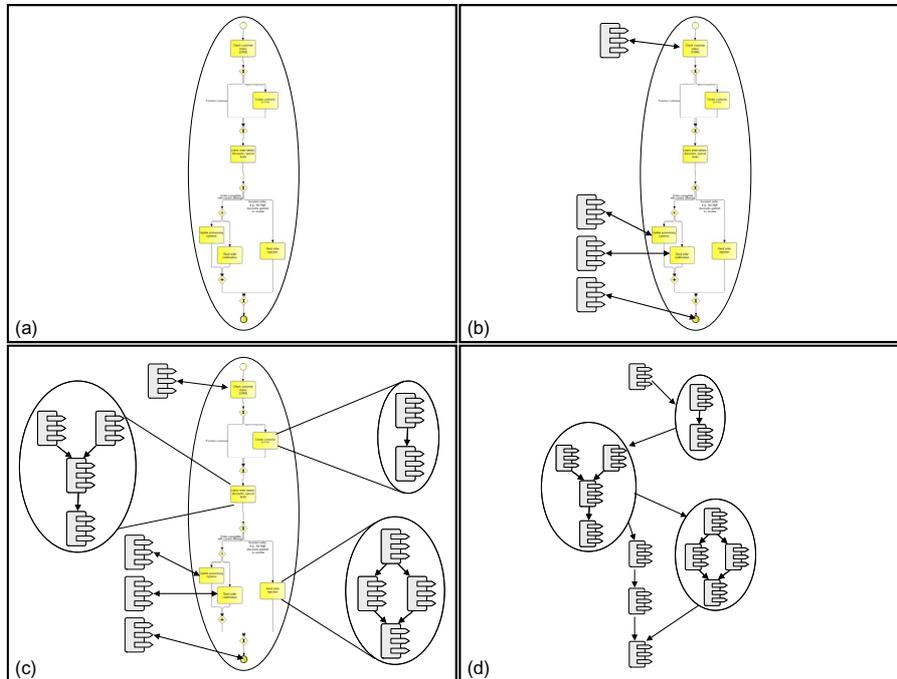


Fig. 3. Results from the steps of the Composition Algorithm

Our ongoing work in the SUPER project¹ includes specifying and implementing forward search AI planning guided by heuristics. We research using domain ontologies as background theories in the planning. The approach builds on the Fast-Forward (FF) planning algorithm [24].

3.4 Discovery

Two steps of the composition algorithm make use of the discovery functionality: in step 3.1 an attempt is made to discover an individual artifact that can implement a task, while in step 3.2.1 a set of artifacts is composed together and must be discovered beforehand. For this task we plan to adapt and extend the Semantic Web service discovery techniques proposed in [11][16][17] to find single artifacts that implement process tasks.

The annotations of executable artifacts and process tasks serve as an input to the discovery component that decides whether an executable artifact fits to the goal of a process task. Discovery of executable artifacts from the repository is realized by utilizing matchmaking techniques for comparing their semantic descriptions against the semantic descriptions of process tasks. The matching artifact descriptions are then ranked according to their relevance w.r.t. the goal.

Functionality-based discovery of a single artifact for process task implementation consists of two phases. In the first phase, we utilize semantic descriptions of artifacts to filter only invocable artifacts. Here, by invocable we mean artifacts whose preconditions

for execution in the current state are satisfied. In the second phase, the resulting set of executable artifacts is queried against the desired goal of a process task. The final set of discovered artifacts for each process task is then ordered by using the widely used degrees of match introduced in [9] and [8]. If a single artifact with required functionality can not be found, the discovery component will provide a set of invocable artifacts as an input for artifact composition.

In the business domain, besides discovering the artifacts meeting the functional requirements, it is important to discover artifacts which best meet the non-functional (quality) requirements of a process task [17]. We envision that the before-mentioned functionality-based discovery will take place at design time. As a result of this process, the semantic description of each task will contain a set of links that associate a goal with the discovered artifacts. During process execution, ranking of the artifacts using the current values of selected non-functional requirements (e.g., price, execution-time, availability, etc.) can be performed.

In case that a single artifact fully matches the desired goal, the top ranked artifact is selected for process task implementation. Otherwise the discovery component returns a set of artifacts for composition that both provide the desired functionality and comply with the non-functional requirements.

Note that the described discovery process can be performed only if the goals and artifacts are annotated using the same ontology. If this is not the case, the discovery component first has to find a mediator for translating between different ontologies, as described in the following section.

3.5 Mediators

Mediators play an important role in two areas of our framework. Firstly they are required in order to compose different independently developed Web service into one executable process. In this context it is necessary to mediate between the different message formats used by these services. Secondly the discovery component needs mediators to cope with artifacts annotated using different ontologies as described in the previous section. In the context of this composition framework we do not want to focus on how necessary mediators are developed. Therefore we assume that the mediators have already been defined beforehand and furthermore have been deployed to a mediator repository [19]. In addition to that we also assume that a mediator is available through a standard Web service interface.

In our framework two types of mediators are necessary: i) mediators capable of translating between syntactic messages formats of Web service and ontology instances and ii) mediators capable of mediating between ontologies. Depending on the type of the mediator invoking the associated Web service interface requires different input data. Invoking a mediator of the first type will for example require an instance of an XML message as an input whereas invoking a mediator of the second type will require instances of concepts according to some ontology. Note that our notion of mediators is broader than the one used by WSMO[20]. WSMO mediators are only concerned with the mediation on the ontology level whereas our notion of mediators also takes the mediation between syntactic message formats and ontologies into account.

Based on these assumptions we are now able to differentiate between two possible usage scenarios of our framework. In the first scenario we assume that all artifacts are annotated using the same ontology whereas we assume in the second one that the artifacts are annotated using different, not integrated ontologies. Each of these scenarios results in certain requirements on the integration of mediation and is detailed in the following subsections.

Annotation using a single ontology: If all artifacts are annotated using a single ontology no mediators need to be executed during the composition of the process. In addition the composition step doesn't need to insert any ontology mediation steps into the process. However, in order to execute the composed process, mediators of the first type might need to be inserted before and after calls to the Web services implementing tasks or parts of them. The necessary calls to the mediators therefore need to be inserted into the process before it is deployed onto the runtime.

Annotation using different ontologies: If all artifacts are annotated using different ontologies the situation becomes much more complex. In order to discover artifacts capable of implementing a given task the discovery components need to execute mediators of the second type as the tasks and each of the artifacts might be annotated using different ontologies. Therefore the discovery needs to interact with the mediator repository in order to locate the required mediators – given they exist. However, this approach might result in the need for the execution of a very large number of mediators. Therefore a pre-selection might be necessary in order to identify promising candidates before executing the mediator. How such a pre-selection could be performed efficiently is currently an open research question and will not be further discussed in this paper.

After the discovery component has found suitable artifacts for a given task it will return this artifact as well as the mediator for mediating between the ontology in which the task is described and the ontology in which the artifact is described to the composition component. The composition component will then use this mediator to create the composed process by inserting it before and after the discovered artifact. Note that this approach results in the usage of the ontology in which the tasks are described as a central hub format. If this is not desired it would also be possible to query the mediator repository for suitable mediators during the composition step again.

3.6 A Sample of Validators

The following two examples show possible validators for the composition component, as depicted in Figure 1. Note that these validators correspond to common requirements from [18].

- Validation w.r.t. policies and regulations: Organizations typically have internal policies and are subject to laws or other regulations. If these policies and regulations are expressible in a machine-accessible way, e.g., in Business Rule Engines, they can potentially be used to evaluate in how far the composed executable process is compliant with them.

- Taking into account transactional requirements: A certain set of artifacts might have an interrelationship with respect to a joint outcome, e.g., either all artifacts achieve a positive outcome or the effects of all artifacts' executions have to be undone [10]. Such a setting would require i) an understanding for the joint outcome, i.e., which outcomes denote success and which ones represent failures; ii) using services that provide cancellation actions; and iii) case-based cancellation policies, depending on which actions actually have to be undone in which cases. Detecting such situations, evaluating and correcting the composition could be achieved by a validator. Also, besides atomic behavior (yes-or-no outcomes only), more complex transactional properties can be presented and handled.

Together, the components and the algorithm form an extensible framework for the problem described in Section 2. The modeling environment serves as the user entry point to the system. The composition component executes the composition algorithm, which explains how and when discovery is used for finding suitable available artifacts; the mediation component bridges heterogeneities; and the validators are executed for evaluating the results of the composition.

4 Related Work

Recently service composition has been an active area of research [2][3][10][12][13]. However, this paper addresses a different problem than most of the current work around Web service composition, as argued in Section 2.

More related issues address mixed-initiative approaches to composition, such as [13] or [23], where composition of services is performed in an interleaved fashion with human modeling steps. Still, the two mentioned works operate on end-to-end composition scenarios, i.e., the automated composition fills in the missing services between the anticipated start and end states. In comparison to our approach, [13] and [23] would perform steps 3.1 and 3.2.1 of the algorithm in Section 3.2 automatically and leave the rest of the work to the process engineer.

Business-Driven Development (BDD) [5][6] is also related, but assumes many manual steps and is more directed to traditional software engineering, in contrast to the service-assumption made here. Probably it is not feasible to automate all the tasks in BDD, but the use of explicit, formal semantics could provide many desired features. The framework in this paper might serve the advancement of automation in BDD.

So far, we have not encountered other approaches that address composition of services over a complete business process model, in particular not if the granularity of the items that have to be composed is not necessarily that of a Web service.

5 Conclusion and Outlook

In this paper we have presented a conceptual framework for composition in business process management. We have introduced the necessary components and described their interactions. Furthermore we have presented a composition algorithm that details how and when these components are used. The described components and the

validator plug-ins in combination with the composition algorithm should allow the implementation of business process models by composing an executable process out of executable artifacts.

The main contribution of this paper is the composition component including the validation technique and the algorithm. Additional contributions are the novel combination of existing technology and the introduction of business requirements into composition present additional contributions.

In future work, we plan to implement the framework and develop more details of how the underlying composition technology can directly be extended towards further business requirements. Also, the seamless integration of the presented approach with a modeling environment is going to be addressed.

References

- [1] Alexandre Alves et al. Web Services Business Process Execution Language Version 2.0, OASIS Public Review Draft, 23rd August, 2006, <http://docs.oasis-open.org/wsbpel/2.0/>
- [2] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Richard Hull, and Massimo Mecella. Automatic Composition of Transition-based Semantic Web Services With Messaging. In VLDB '05: Proceedings of the 31st international conference on Very Large Data Bases, pages 613–624. VLDB Endowment, 2005.
- [3] International Conference on Automated Planning and Scheduling Systems, ICAPS 2003 – 2006, proceedings at <http://www.aaai.org/Library/ICAPS/icaps-library.php>
- [4] Sheila McIlraith and Tran Cao Son. Adapting Golog for Composition of Semantic Web Services. In Proc. of the 8th Int. Conf. on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France, 2002.
- [5] Jana Koehler, Rainer Hauser, Jochen Küster, Ksenia Ryndina, Jussi Vanhatalo, and Michael Wahler. The Role of Visual Modeling and Model Transformations in Business-driven Development. In Fifth International Workshop on Graph Transformation and Visual Modeling Techniques, April 2006.
- [6] Jochen Küster, Jana Koehler, and Ksenia Ryndina. Improving Business Process Models with Reference Models in Business-Driven Development. In Proc. 2nd Workshop on Business Processes Design (BPD'06), LNCS, Springer-Verlag, 2006.
- [7] Ulrich Küster, Mirco Stern, and Birgitta König-Ries. A Classification of Issues and Approaches in Automatic Service Composition. In Proc. of the First International Workshop on Engineering Service Compositions (WESC'05), pages 25–33, December 2005.
- [8] L. Li and I. Horrocks. A Software Framework For Matchmaking Based on Semantic Web Technology. In Proceedings of the 12th World Wide Web Conference, 2003.
- [9] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic Matching of Web Service Capabilities. In Proceedings of the 1st International Semantic Web Conference (ISWC), 2002.
- [10] Marco Pistore, Paolo Traverso, and Piergiorgio Bertoli. Automated Composition of Web Services by Planning in Asynchronous Domains. In Proceedings of the International Conference on Automated Planning and Scheduling, 2005.
- [11] C. Preist. A Conceptual Architecture for Semantic Web Services. In Proceedings of the 3rd International Semantic Web Conference (ISWC), 2004.
- [12] Jinghai Rao. Semantic Web Service Composition via Logic-based Program Synthesis. PhD thesis, Norwegian University of Science and Technology, 2004.

- [13] Jinghai Rao, Dimitar Dimitrov, Paul Hofmann, and Norman Sadeh. A Mixed Initiative Approach to Semantic Web Service Discovery and Composition: SAP's Guided Procedures Framework. In Proceedings of the 2006 IEEE International Conference on Web Services (ICWS 2006), Chicago, USA, September 18 - 22, 2006.
- [14] Stuart J. Russell and Peter Norvig. Artificial Intelligence: A Modern Approach. Pearson Education, 2003.
- [15] Jim Sinur and Janelle B. Hill. Align BPM and SOA Initiatives Now to Increase Chances of Becoming a Leader by 2010. Gartner Predicts 2007. 10 November 2006.
- [16] I. Toma, K. Iqbal, M. Moran, D. Roman, T. Strang, and D. Fensel: An Evaluation of Discovery approaches in Grid and Web services Environments. NODe/GSEM 2005: 233-247
- [17] Vu, L.-H., Hauswirth, M., Porto, F., and Aberer, K. A Search Engine for QoS-enabled Discovery of Semantic Web Services. Special Issue of the International Journal on Business Process Integration and Management (IJBPM), 2006, to be published.
- [18] Ingo Weber, Requirements for the Implementation of Business Process Models through Composition of Semantic Web Services. Proceedings of the 3rd International Conference on Interoperability for Enterprise Software and Applications (I-ESA) March 2007, Funchal, Portugal
- [19] L. Cabral, C. Drumm, J. Domingue, C. Pedrinaci, A. Goyal: D5.6 Mediator Library. Deliverable of the DIP project, July 2006
- [20] Digital Enterprise Research Institute (DERI): Web Service Modelling Ontology, <http://www.wsmo.org>, 2004.
- [21] C. Ouyang, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. Translating BPMN to BPEL. BPM Center Report BPM-06-02, BPMcenter.org, 2006.
- [22] Jan Mendling, Kristian Bisgaard Lassen, Uwe Zdun: Transformation Strategies between Block-Oriented and Graph-Oriented Process Modelling Languages. In: F. Lehner, H. Nösekabel, P. Kleinschmidt, eds.: Multikonferenz Wirtschaftsinformatik 2006, Band 2, XML4BPM Track, GITO-Verlag Berlin, 2006, ISBN 3-936771-62-6, pages 297-312.
- [23] J. Schaffner, H. Meyer, C. Tosun: A Semi-automated Orchestration Tool for Service-based Business Processes. In: Proceedings of the 2nd International Workshop on Engineering Service-Oriented Applications: Design and Composition, Chicago, USA (to appear)
- [24] J. Hoffmann, FF: The Fast-Forward Planning System. In: AI Magazine, Volume 22, Number 3, 2001, Pages 57 - 62.
- [25] World Wide Web Consortium (W3C), Web Ontology Language (OWL), W3C Recommendation 10 February 2004, <http://www.w3.org/2004/OWL/>