# Challenges and Solutions for Model Driven Web Service Composition

Konrad Pfadenhauer[1], Schahram Dustdar[2] , Burkhard Kittl[1]

[1] *Vienna University of Technology, Institute for Production Engineering, Karlsplatz 13, 1040 Vienna, Austria*
*{pfadenhauer, kittl}@mail.ift.tuwien.ac.at*

[2] *Vienna University of Technology, Information Systems Institute, Distributed Systems Group, Argentinierstraße 8/184-1, 1040 Vienna, Austria*
*dustdar@infosys.tuwien.ac.at*

## Abstract

*System theory propagates the use of models which raise the level of abstraction to cope with complexity, evolving out of variety and connectivity. Different modeling techniques have been invented for the abstract description of certain aspects of dynamic system behaviour. Nowadays process models are also an accepted part of service oriented process design. Although service oriented architecture (SOA) is the aim, rather isolated processes are the starting point for executable code generation and not the changing functionality distribution the system offers. Thus we claim the system view, necessary for process lifecycle management, is missing. Moreover, in the context of Web Service Composition, a vertical, bi- directional model driven approach allows for manifold allocation of e.g. testing, validation or binding tasks within the layer stack of MDAs (model driven architecture). In this paper we discuss two distinctive proposals for model driven service composition, followed by a comparison of representative mapping scenarios offered by IBM and Microsoft.*

## 1. Introduction

In our current project "Modeling the shop floor for a service oriented architecture" [18] we use the shop floor domain to implement a methodology for platform independent, service oriented model generation. To overcome a situation of vertical, interrupted processes and partly unavailable, partly static accessible functionalities we introduce our concept of a MDSA (Model Driven Service Architecture) for the shop floor (Figure 1), together with a top-down methodology and a tool for user friendly model creation, which in addition is the basis for automated flow execution. Therefore, we introduced three levels of model granularity and completeness which allow for static and dynamic modeling, starting with generic constructs and ending with a model of a given shop floor domain.

Fast and easy initial, computation independent modeling of a given shop floor system has to be supported, focusing on functionality and connectivity of the system as a hole. We achieve this by a generic high level model construct collection called "Shop Floor Tool-Box". By means of scenario specific selection of generic artefacts from the toolbox a sophisticated platform independent model (PIM) evolves. Residing at a strategic level this model does not include detailed information enabling automated flow execution, but takes into account the socio-technical structure of the domain.

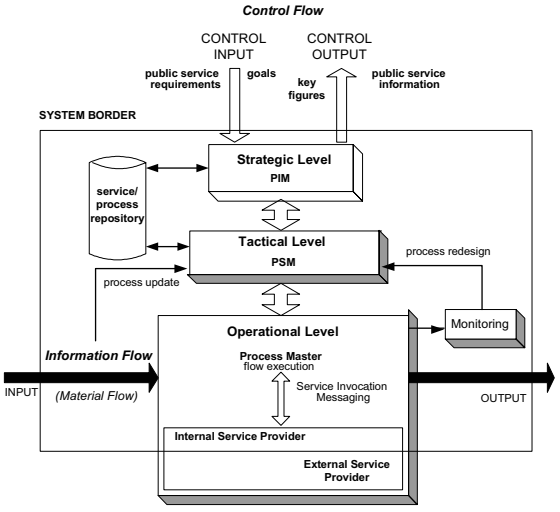


**Figure 1.** Process life cycle oriented domain modeling

Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05)
0-7695-2362-5/05 $20.00 © 2005 IEEE

It has to support long term platform, infrastructure and service provider decisions through as-is and to-be comparisons. This high level model has to interact with the PSM concerning process definition. The latter serves at a tactical level for the (re)design of service flow definitions which are semantically rich enough for executable code generation. Both levels interact with the service repository. It provides patterns specifying typical system processes, each on the one hand in a more generic (computation independent) model form for the use within the PIM and on the other hand particular process models for the interaction with the PSM. Alternative service flows, different versions or diverse technological specifications of one and the same process are, therefore, separated from the actual valid PSM. Templates allow for fast process creation, either directly within the PSM or within the repository. This proceeding takes into consideration the necessity of business- and IT-view alignment, allowing e.g. architectural views or process definition at distinct levels of abstraction.

The visualized aim made UML the first choice, because UML is the de facto standard for IT professionals and, as we demonstrate with our "Shop Floor Tool-Box" methodology, is well suited for abstract domain analysis too. One of the main platforms of interest is WS/SOAP/XML, a promising set of technologies for service oriented architecture implementation. Business process management within this environment has to cover the complete range from high-level, graphical process design down to low-level, coded process execution [14], [12].

The actual project state is this. On the one hand we created platform independent UML models of the shop floor and on the other a repository of Web Service interfaces realized by typical shop floor service providers is available. The next step is to implement the MDSA control loop, thus we are forced to emphasise how to develop and implement executable service flows. This is the aim of this paper, to figure out which distinctive proposals and tools for model driven WS Composition are available.

The structure of this paper is as follows. We first introduce in Section 2 common service composition challenges regarding model support. We continue with a more detailed discussion of model driven Web Service Composition and emphasize two distinctive paradigms. For the realization of Web Service oriented architectures, we need to make our platform independent process models executable. Therefore, we choose one representative from each group mentioned above, fulfilling the requirement of real-world proceedings for model driven WS Composition with tool support. For the "top-down group" notably an IBM-BPWS4J approach, which is at a rather early testing stage, and the Microsoft-BizTalk approach, representing the "bottom-up group". A theoretical comparison and evaluation can be found in Section 3. With a conclusion and an outlook at the work to come we will finish this paper.

## 2. MDA, SOA and Composition challenges

Figure 2 illustrates the three main areas which have to be combined for model driven service composition. Alonso et al. [2] mention three main elements for WS composition middleware: the modeling environment, the development environment (IDE) and the run-time environment. The latter one has to execute the coded composition specification. How this specification is achieved, i.e. how and within which environment the necessary specifications and tasks from high level system model down to code execution are fulfilled, is a matter of ongoing discussion. Tasks like syntactical and semantic verifications have to be considered, just as service discovery and binding. Which type of composition specification, the composition model, the model representation language or the executable composition language, is best suited for which kind of task, can not be answered definitely yet. The dependencies between these composition specifications on the one hand and their integration within the three environments on the other determine the demand for mapping and testing functionality. The challenging factors depicted in Figure 2 establish our collection of criteria for evaluation of existing approaches.

The number of enclosing real-world approaches especially for "top-down" model driven WS Composition with an appropriate tool support is limited.

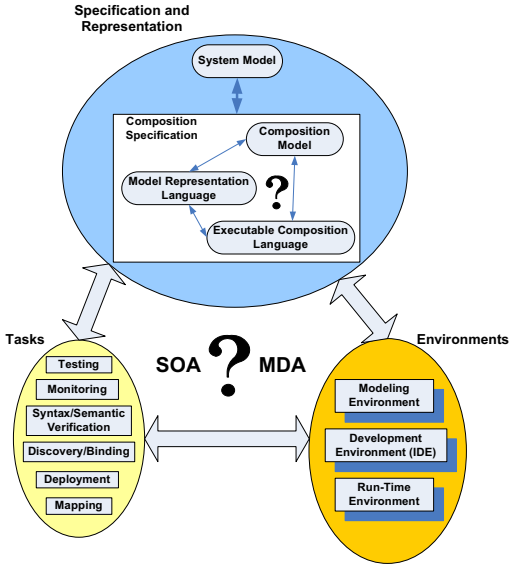


**Figure 2.** Composition challenges

The reasons, therefore, are amongst others:

- Still a number of process mark-up languages, the "missing link" between model and execution, like BPEL4WS or WSCI/BPML exist. None of them can be called a standard yet. Worse, the execution engines availability is limited (one is IBM's BPWS4J for BPEL4WS).
- Model and code have to support equally certain control patterns, the evaluation of this is still in progress (see [5], [1]). By means of the emergence of second generation WS Technologies in conjunction with the service oriented architecture [6], [7] it is likely these patterns have to be further expanded.
- Model mapping and transformation is still limited to static constructs like classes and components, the complex task of dynamic information transcription is just at the beginning.
- The transformations have to be completely bi-directional, a requirement which is hard to achieve.
- Intelligent validation and testing mechanism between the mapping steps hardly exist yet.

To sum it up, the proceeding from WS Composition graphs to executable service flows is in the state of lively theoretical discussion, although first implementations exist. At this time two rather distinctive approaches can be observed, one with model environment focus, the other with IDE focus. They also stand for completely different ways how the model driven concept can be interpreted. Both paradigms have knowledge alignment through modeling [11] in mind, but the difference in the realization shall be described by means of two examples for model driven Web Service Composition. Due to the need for publicly available tools we focused on an IBM and a Microsoft scenario.

## 2.1. "Top-down" model driven WS Composition

Generally speaking, the first approach illustrated in Figure 3 postulates the concept of model enrichment through extensions combined with elaborate transformation and mapping mechanisms, most prominent represented by means of OMG's MDA (Model Driven Architecture) [16]. This ambitious concept demands for a strict separation of concerns, expressed through a layer stack consisting of different levels of abstraction. To achieve this, the models have to cover a broad set of requirements.

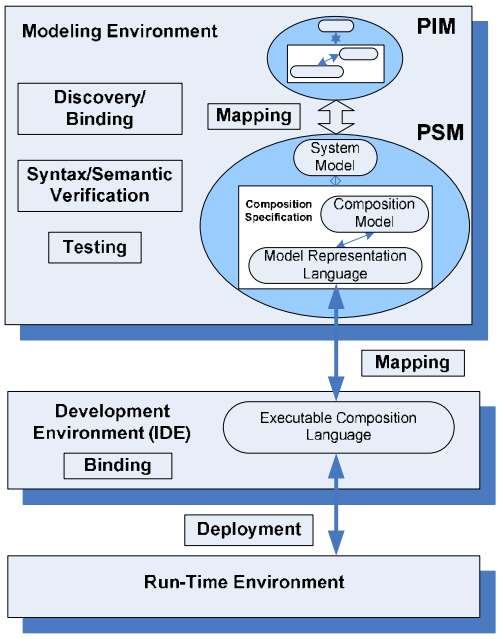


**Figure 3.** Top-down model driven WS Composition

We mention as an example the complex task of PIM (Platform Independent Model) to PSM (Platform Specific Model) mapping, extended by an additional executable UML layer, introducing mark-ups and languages like ASL (Action Specific Language. This intermediate step makes the model executable, especially valuable for testing before mapping. Skogan, Gronmo and Solheim ([19] and [10]) present a very interesting approach. They introduce not only a technique how to import WSDL service descriptions into UML, but also how to use UML activity graphs enriched by means of certain UML extensions to define WS compositions. In addition they use XMI in combination with corresponding XSLTs to achieve a transformation in several execution languages, at the moment BPEL4WS and WorkSCo are supported. The use of templates and patterns can be applied, either for activity graph design [8], or for consistency analysis. Voigt [20] has implemented the latter by means of UML-CSP (Communicating Sequential Processing) language mapping. The benefit of this high level "top-down" approach is a stringent process from abstract domain models down to executable code.

The IBM approach [see also [15]) begins with an UML model, which is specific for the BPEL4WS platform through extensions following the "UML 1.4 Profile for Automated Business Processes with a mapping to BPEL 1.0" [3]. Modeling according to the profile makes the WS descriptions (porttypes,

operations, messages) of the involved services necessary, but no WSDL import support is available. Hence, this information has to be extracted and included into the class diagrams (static view of data types/messages, protocols and roles) and activity diagram (dynamic view) by hand. After finishing the model gets exported into XMI format, which then is imported into an Eclipse 2.1.3 IDE java project. There an add-in performs the mapping to BPEL4WS 1.0 or 1.1, generating the WSDL, XSD and BEPL files. Bindings, location paths and service links are added by means of a WSDL modify tool. Deployment is separated from the IDE, afterwards the BPEL process can be executed via the BPWS4J 2.0 runtime engine.

### 2.2. "Bottom-up" model driven WS Composition

The second approach (Figure 4) is best described as "bottom-up", because not the portable architecture model is the starting point, but the existing process development and execution possibilities within a certain vendor environment. The vendor independent visualization is just an extension of the central graphical drag & drop interactions offered by the IDE. In this group we classify Microsoft Biz Talk/Visual Studio, IBM WebSphere [13] or the SAP NetWeaver Platform [21]. In addition to platform specific (PSM), the IDE models have to be called vendor specific (VSM). Only very recently within this group portability is not only achieved through the mapping to process mark-up languages like BPEL4WS, but also at model level through e.g. UML-XMI mapping [4].

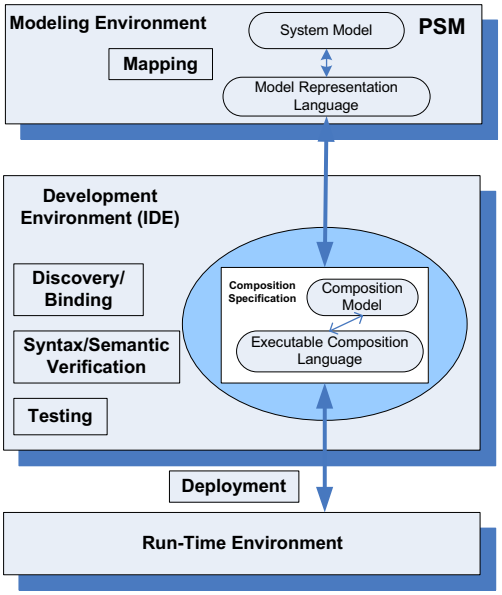


**Figure 4.** Bottom-up model driven WS Composition

The Microsoft proposal can be described as follows. In the Visio Orchestration Designer a proprietary modeling semantic is used to design the basic flow pattern, allowing fork, join, group, decide and loop constructions. An add-in exports this activity graph into a BizTalk Orchestration XLANG/s .odx format, which can be imported into the BizTalk Orchestration Designer, which again runs within the Visual Studio IDE. The rudimentary flow is now supplemented with ports and messages, which are created by means of wizards utilizing the port types and message types retrieved from the added web references. All WSDL, XSD and the .odx files are generated and updated with strong graphical support. The process can be deployed within the IDE using an administration console.

## 3. Comparison and Evaluation

As we mentioned above, both approaches are model driven Web Service Orchestration implementations, but with very diverse emphases. In chapter two we emphasized current challenges on WS Composition. This discussion led to a collection of criteria which in the following shall be used for comparison purposes (Table 1).

Generally speaking, IBM tries to design the WS Composition at a higher level of abstraction, using an UML profile for nearly complete process definition. Therefore, the mapping afford is rather high, unfortunately no syntax validation takes place before XMI export, which makes the failure risk during the subsequent BPEL4WS mapping very high. The IBM approach offers at no level WS discovery or reference mechanisms.

**Table 1**. Evaluation results

| | IBM "top-down" | MS "bottom-up" |
|---|---|---|
| **Environment** | | |
| Modeling | substantial | rudimentary |
| Development | code oriented | graphical/wizards |
| Run-time | BPWS4J | BizTalk Server |
| **Specification** | | |
| System model | no | no |
| Composition model | enriched UML | poor + proprietary |
| Model representation language | standardized XMI | proprietary |
| Executable composition language | standardized BPEL4WS v1.0/v1.1 | proprietary XLANG/s |
| **Tasks** | | |
| Testing WS links | no | within IDE |
| Testing orchestration | run-time | run-time |
| Monitoring | poor | strong |
| Syntax Verification | limited | strong |
| Semantic Verification | no | no |
| Discovery | not supported | search functionality |
| Binding | static | static |
| Deployment | isolated | integrated |
| Mapping | isolated | integrated |

The manual WS description import, i.e. the WSDL file transformation into different types of stereotyped classes within UML is tedious. On the other hand the XMI format containing the whole business logic and platform specific information offers theoretically a certain flexibility concerning model reuse and transformation. In practice the reimport of created XMI files into Rational Rose, our UML tool, was not complete, manual adjustments were required. The necessary support within the IDE is reduced to WSDL modifications. The XMI mapping to BPEL4WS itself takes place automatically, but again without XMI validation. If an error occurs, the mapping stops without error handling. A BPWS4J Editor plug-in for the Eclipse 2.1.3 IDE offers enhanced verification functionality and more important rudimentary graphical support for process editing. Nevertheless the hierarchical visualization is much less expressive and functional than the one within the MS Orchestration Designer.

Also concerning deployment, monitoring and testing the MS approach is superior, but that is what we expected. It should be mentioned here the pointlessness of a comprehensive IDE comparison. In that case IBM WebSphere Studio rather than Eclipse would have to be the challenger, resulting in a comparison between two IDE centric approaches.

The second proposal places the Web Service Orchestration at a lower level of abstraction, within the IDE and, therefore, establishing a tight coupling between visualization and coding. The initial Visio modeling in the MS proposal offers very limited additional value, since beside the control flow no additional information can be included. Moreover, for control flow modeling the IDE functionality is as easy to use. The diagram is not integrated within other stencils, therefore, the integration in an architecture model is missing. After the import into the IDE the model is platform specific twofold (WS/XML and .NET). Once the basic service flow is set up, the service binding is very comfortable by means of drag&drop and wizard functionality. At this level code and graph are updated real time, a benefit which is only possible by means of a tight integration into the IDE. Although the underlying XLANG/s language is proprietary, restricted BPEL4WS 1.1 import and export functionality is offered which allows for a certain degree of portability.

Both approaches are concerned with single process definition and do not take into account a system model from which the process may be derived. That is to say, the UML allows for further extensions referring to this issue, whereas the Visio model doesn't offer this flexibility. As one can see in Figure 1, the dependencies between a single composition specification and an overall architectural model are in our opinion a vital part of a methodology which supports life cycle oriented service composition. In our opinion process lifecycle management without a complete model covering all relevant interaction aspects is not possible. To see the process isolated from its environment has many disadvantages:

- Inter process dependencies like synchronization or process hierarchies (nested sub-processes) are missing. Hence, appropriate business rules and constraints have to be coded. Monitoring by model is no longer possible.
- Within architectures different control flow paths exist and new evolve to achieve a certain process aim. Without an architecture model and the knowledge about the given connections, one has to design for every path a single process model, not knowing whether the needed interoperability exists or not.
- The consequences of a change in the architecture or the introduction of new service providers for the process designs can not be monitored accordingly.
- Service repositories are missing at initial modeling level. The design of the activities takes place without knowledge about their availability according to given QoSs (Quality of Service).

Reactive semantic verification mechanisms should be implemented twofold, firstly concerning the control logic (e.g. the detection of deadlocks) and secondly concerning inconsistencies regarding the semantic at different levels of abstraction after manual intervention. Both ignore the first requirement, and only the MS approach inhibits inconsistencies between code and graphical representation due to the tight integration within the IDE (but not between the Visio model and the IDE graph). After our discussion it should be evident that for single process definition one can choose between a top-down and a bottom-up approach, but for service oriented architectures the top-down approach is crucial.

## 4. Conclusion and Outlook

In this paper we compared two general approaches of process design and execution, one with strong focus on the abstract, vendor independent model, which is semantically rich enough so that an IDE is exclusively needed for (complex) mapping tasks. The other with the focus on "applied" modeling within the IDE, supported by a rudimentary graph describing the control flow.

The latter "bottom-up" approach, represented by MS BizTalk Server, has its main advantage concerning easy integration of existing Web Service definitions, which is not possible within the "top-down" approach. On the other hand the integration of the Visio or the Biz Talk Orchestration models in an overall, syntactical homogeneous architecture model is not possible, thus these directed graphs have to be built from the scratch. Here the UML-BPEL4WS approach offers much more possibilities of integration in an enclosing MDA concept. In the context of a "top-down" approach, mappings from XML representations of widespread used and well-defined modeling techniques (EPC Markup Language in the case of EPC, or PNML Petri Net Markup Language in the case of Petri Nets) to platform independent MDA-UML models seem promising. OMG's Business Process Definition Metamodel [17], an UML 2.0 profile, aims at this goal. This proposal supports the mapping to a common metamodel and thus facilitates the communication among a variety of process models [9]. Moreover, it makes business process specifications part of complete system specification to assure consistency and completeness.

Next steps to come are investigations, whether these approaches are suited for shop floor domain usage. Actually we implement some of our processes in a use case scenario, allowing for tests regarding applicable mechanism for process model and process specification synchronization, with BPEL4WS as the main specification of interest. Another open question is still how the process models have to be embedded in the overall architecture model e.g. how mechanisms, which synchronize between architecture structure ("the sum of all possible processes") and single process lifecycle management, can be established.

## 5. References


[1] van der Aalst, W.M.P.: ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P., Workflow Patterns, BETA Working Paper Series, WP 47, Eindhoven University of Technology, Eindhoven, 2000

[2] Alonso, G. et al.: Web Services – Concepts, Architectures and Applications, Springer Verlag, Berlin Heidelberg, 2004

[3] Amsden, J., Gardner, T. et al.: Draft UML 1.4 Profile for Automated Business Processes with a mapping to BPEL 1.0, Version 1.1, IBM, June 9th 2003

[4] Beck, K., Joseph, J., Goldszmidt, G.: Learn business process modeling basics for the analyst, IBM developerworks, Februar 22nd 2005, http://www-128.ibm.com/developerworks/ webservices/library/ ws-bpm4analyst/index.html

[5] Dumas, M., ter Hofstede, A.H.M.: UML Activity Diagrams as a Workflow Specification Language, In: M. Gogolla, C. Kobryn, editors, UML 2001 – The Unified Modeling Language, Proc. Of the Int. Conference in Toronto, Canada, October, Springer Verlag, Berlin Heidelberg, 2001, pp76-90.

[6] Dustdar, S.: Web Services Workflows - Composition, Coordination, and Transactions in Service-Oriented Computing, Concurrent Engineering: Research and Applications, Sage Publications, September 2004, p. 237-246

[7] Erl, T.: Service-Oriented Architecture, Prentice Hall PTR, New Jersey, 2004

[8] Förster, A.: Pattern-basierte Modellierung von Geschäftsprozessen, Thesis at the Institute for Database- and Informationsystems, University of Paderborn, November 2002

[9] Giaglis, G.M.: A Taxonomy of Business Process Modeling and Information Systems Modeling Techniques, International Journal of Flexible Manufacturing Systems, Volume 13, Issue 2, April 2001, 209 – 228

[10] Grønmo, R., Solheim, I.: Towards Modeling Web Service Composition in UML, presented at The 2nd International Workshop on Web Services: Modeling, Architecture and Infrastructure (WSMAI-2004), Porto, Portugal, 2004

[11] Haeckel, S.H.: Leading on demand business-Executives as architects, IBM Systems Journal, Volume 42, Issue 3, 2003, pp405-413.

[12] Kalogeras, A.P. et al.: Vertical Integration of Enterprise Industrial Systems Utilizing Web Services, in Sauter, T., Vasques, F.: Proceedings of the 2004 IEEE International Workshop on Factory Communication Systems, Vienna, September 2004

[13] Kloppmann, M., König, D. et al.: Business process choreography in Websphere: combining the power of BPEL and J2EE, IBM Systems Journal, Volume 43, Issue 2, 2004, pp. 270 - 296

[14] Leymann, F., Roller, D., Schmidt, M.-T.: Web services and business process management, IBM Systems Journal, Volume 41, Issue 2, 2002, pp198 – 211

[15] Mantell, K.: From UML to BPEL, IBM developerworks, Sept 9th 2003, http://www-106.ibm.com/developerworks/webservices/library/ws-uml2bpel/

[16] Miller, J., Mukerji, J. (Editors): MDA Guide Version 1.0.1, OMG, June 12th 2003

[17] OMG: Business Process Definition Metamodel, Request for Proposal, OMG document, 2003-01-06, http://www.omg.org/docs/bei/03-01-06.pdf

[18] Pfadenhauer, K., Kittl, B.: With a system approach towards a model driven service architecture for the shop floor, in: Proceedings of the 2004 International Research Conference on Innovations in Information Technology (IIT2004), College of Information Technology, UAE University, Dubai, 2004

[19] Skogan, D., Grønmo, R., Solheim, I.: Web Service Composition in UML, The 8th International IEEE Enterprise Distributed Object Computing Conference (EDOC), Monterey, California, 2004

[20] Voigt, H.: Modell-basierte Analyse von ausführbaren Geschäftsprozessen für Web Services, Thesis at the Institute for Database- and Informationsystems, University of Paderborn, September 2003

[21] Woods, D. and Word, J.: SAP NetWeaver for dummies, Wiley Publishing, Indianapolis, Indiana, USA, 2004