Building Orchestrations in B2Bi – The Case of BPEL 2.0 and BPMN 2.0

Jörg Lenhard and Guido Wirtz

Distributed and Mobile Systems Group, University of Bamberg, Germany
{joerg.lenhard,guido.wirtz}@uni-bamberg.de

Abstract. Various approaches for service-oriented business-to-business integration (B2Bi) rely on a top-down development methodology. The starting point is a choreography model which is subsequently partitioned into multiple orchestrations. Most current approaches use the Web Services Business Process Execution Language (BPEL) for implementing the latter. At the same time, a plethora of other languages, such as Business Process Model and Notation (BPMN) 2.0 process diagrams, is available. As integration partners are free to select the orchestration language of their choice, it should be easy to integrate different orchestration languages with current choreography technology. Language transformation, starting from a suitable format, is a means to achieve this. In this paper, we assess BPEL 2.0 and BPMN 2.0 process diagrams for their suitability for this transformation in a services-based B2Bi setting using a requirements framework identified through a literature study.

Keywords: Orchestration, BPEL 2.0, BPMN 2.0, B2Bi

1 Introduction

Over the last years, the influence of service-orientation in the implementation of interorganizational processes has grown rapidly. Many approaches[1] for implementing such processes employ a combination of choreography and orchestration models [6, 19] to capture different viewpoints on an enterprise-crossing business process. Top-down approaches refine the first into a set of the latter which thereafter is implemented at each partner's side. A variety of languages has emerged, and is continuing to do so, on both levels of abstraction. Integration among the two types of languages and an easy translation from a choreography to a set of orchestrations is seen as a core issue [4].

Although BPEL [16] is widely used in these approaches for implementing orchestrations, it is facing rising competion by other languages, such as BPMN 2.0 process diagrams [17] or the Windows Workflow Foundation (WF) [3]. Consequently, an integration partner may choose to implement her orchestration not in BPEL, but in any other orchestration language, using the derived orchestration, which with current approaches mostly is a BPEL process, only as blueprint.

[1] Some examples, without claiming to be complete, are: [4, 8, 14, 18, 23, 30]

In this case, there is still a considerable gap between a BPEL process derived from a choreography and the final running orchestration. For instance, target languages may rely on a different (i.e., graph-based or block-structured [11]) execution model with a differing level of expressiveness. Bridging this gap manually, which is the current option, it is easy to introduce problems that hinder later execution. Behavioural incompatibilities that were eliminated using model checking techniques may be reintroduced and adjustments to the original interfaces and message definitions that arise from the limitations of the final execution platforms may become necessary. Here, an automatic transformation from an orchestration derived from a choreography model to orchestrations implemented in other languages can help. The idea is to not to compile choreography models to an executable artifact tailored to a specific execution platform in a single step, but to provide an artifact that can be automatically transformed to a variety of different types of languages and platforms, and also be executed directly. By automating this transformation step, it would be possible to:

1. Use model checking and related techniques [27] to ensure that no behavioural incompatibilites are introduced during the transformation.
2. Leverage the information of what execution platforms are ultimately used and will be communicating with one another to avoid pitfalls and communication problems known for these platforms[2] and perform optimizations such as the configuration of the most efficient communication binding known to exist among the platforms.

The starting point of this functionality is a suitable format to which choreography models are compiled and from which such a transformation is possible in the first place. The aim of this paper is to deliminate criteria (requirements) that are essential for such a format and which can be used to evaluate the suitability of a language, as well as to use these criteria to assess two languages that are natural candidates for this kind of task. BPEL could be seen as such a format and most researchers use it because of its status as de-facto standard; A more recent option is BPMN 2.0 process diagrams [13].

The requirements are identified through a literature study. Since the aformentioned format is to be used for model transformation and optimization in service-oriented top-down B2Bi, the requirements are derived from relevant sources in the B2Bi domain, service composition languages and transformation of process models that reside on the same level of abstraction[3].

Furthermore, we assess the support of BPEL 2.0 and BPMN 2.0 process diagrams for the identified requirements and discuss the evaluation. The outcome suggests that BPMN 2.0 process diagrams are more suitable for this type of application.

[2] For example known problems among Java-based and .NET-based Web Services [25].
[3] This is called horizontal transformation [15].

2 Related Work

Approaches for service composition using choreography and orchestration technology have attracted considerable interest. Most of these approaches (e.g. [4, 8, 14, 18, 23, 30]) rely on BPEL as the target language to which choreography models are compiled. Although being widely supported, BPEL is more and more rivaled by other languages [13], both based on standardization initiatives, such as BPMN 2.0 [17], or proprietary environments, such as WF [3]. Just as for BPEL, engines for executing orchestrations built in these languages are available.

There are many studies that explicitly specify and assess requirements for service composition languages, B2Bi, or model transformation with varying design goals. In the area of service composition, focus lies on choreography languages [4, 23]. In this paper, we center on orchestration languages instead, but take into account these studies, where requirements for choreographies and orchestrations intersect. The requirements defined in [4] concentrate on language expressiveness. [22, 23] on the other hand, take B2Bi-related requirements into account. General requirements for process languages and requirements related to horizontal transformation of these languages can be found in [7, 15, 29]. In this paper, we extract and unify the requirements from the preceding studies that are relevant to horizontal transformation of orchestration languages in B2Bi.

An assessment of BPEL 2.0 and BPMN 1.1 for parts of these requirements can be found in [4]. In [10], the requirements from [4] are used to assess BPMN 2.0 *collaboration* and *choreography diagrams*. Here instead, we concentrate on BPMN 2.0 *process diagrams* with the addition of `participants`, use an extended set of requirements and a different design goal; that is the assessment of orchestration instead of choreography capabilities.

3 Requirements for Orchestration Languages in B2Bi

Myriads of requirements could be taken into account when considering either B2Bi, service composition or language transformation and a vast amount of literature on these topics is available with varying design goals. We do not intent to start from scratch and therefore extract common requirements from several influential studies of recent years that did explicitly post such requirements and which match well our domain and design focus. Like in any literature study, this selection of sources is biased to some extent by our knowledge and we do not claim the completeness of the requirements listed here.

The requirements are sorted in four groups: (i) General requirements, (ii) B2Bi-related requirements (iii) interaction-related requirements and (iv) derived requirements. The final group does not originate from the requirement sources, but is derived in the context of this study.

I. General requirements:

R1 *Support for common control-flow structures*: An orchestration language must include a suitable amount of control-flow structures to allow for a direct implementation of domain relevant scenarios. This requirement is explicitly stated in [7,15,22,23]. Assessing languages for their support for control-flow patterns which describe such common structures can be used as benchmark for this requirement [28].

R2 *Mechanism for hierarchical decomposition*: A key feature for dealing with the complexity of realistic orchestrations is a mechanism for hierarchical decomposition. The necessity of this feature is stated in [7,15,22,23].

R3 *Data handling mechanisms*: Just as for control-flow structures, appropriate mechanisms for defining, transfering, and manipulating data structures must be in place [4,7,15,22,23]. This requirement can be evaluated by assessing pattern support as well [20].

R4 *Exception handling mechanisms*: Being executable, orchestrations must not only deal with best-case scenarios, but take into account erroneous circumstances that may arise during execution. This requirement is backed up by [4,7,22,23]. It can also be assessed using pattern-based analysis [21].

R5 *Extensibility*: An orchestration language should be extensible to allow for an easy adaptation and the introduction of new or modified constructs to support use cases with specific needs [7,22,23].

II. B2Bi-related requirements:

R6 *Transactions*: An important primitive in enterprise computing is transactional integrity of interactions. For instance, the reliable transmission of business documents is crucial and a common means to this end are transactions. An orchestration language should provide mechanisms to denote transactional contexts during process execution [7,22,23].

R7 *Quality of service (QoS)*: Several nonfunctional properties, esp. QoS parameters, are vital in B2Bi. These are authentication, message encryption and signatures, non-repudation of message exchanges and time constraints. An orchestration language should provide explicit mechanisms to express these properties [4,22,23].

R8 *Standards*: In the B2Bi setting, it is not possible to enforce technologies on different independent partners. A higher degree of interoperability is likely by relying on essential standards [22,23].

III. Interaction-related requirements:

R9 *Message correlation*: During execution, multiple orchestration instances run in parallel. To support a correct routing of messages by an engine, an orchestration language must provide mechanisms for message correlation [4,22,23]. As before, this aspect can be evaluated using patterns [1].

R10 *Service selection and reference passing*: In realistic interaction scenarios, not all communicating parties may be known at design time. Instead, partner references are transfered in messages and are bound at run-time [4].

R11 *Multi-lateral interaction*: Choreographies may consist of more than two partners. Consequently, orchestrations must be able to represent and communicate with multiple different parties [4].

IV. Derived requirements:

R12 *Contract-first development and integration with choreography approaches*: It is a general engineering best practice to define interfaces or contracts before implementing them. This is inevitable in a top-down development approach. Orchestration languages therefore must support contract-first development[4]. [4,23] specify that choreography languages must easily integrate with orchestration languages. Also the reverse is important: The applicability of an orchestration language in top-down approaches should be demonstrated.

R13 *Web Services and XML*: Choreographies should be technology-independent [4, 23]. This does not apply to orchestrations, which need to be executable. Consequently, they should work with contemporary communication and integration technologies, most notably Web Services and SOAP. To allow for easy processing and transformation of orchestration models, languages should provide a XML serialization format [7, 15, 22, 23].

4 Assessment of BPEL 2.0 and BPMN 2.0

In the following, BPEL 2.0 and BPMN 2.0 process diagrams are assessed for their support for the requirements. We state whether a requirement is supported *directly* (+), *partially* (+/-) or *not in a direct fashion* (-). This trivalent measure is relatively simplistic and subjective to a certain extent. Although enhanced alternatives do exist [12], it is used extensively [4, 5, 10, 20, 21, 28, 31]. For that reason and the space constraints of this paper, we use the above measure.

Assessment of BPEL 2.0: A detailed analysis of the control-flow capacity of BPEL can be found in [12]. BPEL 2.0 supports a range of control-flow structures and block-structured and graph-oriented control-flow definition. Given typical B2Bi use cases[5] generally only require simple control-flow constructs [24], we consider this as evidence for the support of R1. BPEL 2.0 provides no explicit construct for hierarchical decomposition; that is, no direct notion of a subprocess[6]. It is possible to work around this requirement using nested scopes or Web Service invocation of another BPEL 2.0 process, which qualifies as partial support for R2. Compensation and try-catch constructs are present for exception handling and XML Schema and XPath 1.0 for data definition and manipulation. Although closer evaluations are only present for BPEL 1.1, we consider this as support for R3 and R4. BPEL 2.0 allows to extend the language with new engine-specific activities using `extensionActivity` and `extensionAssignOperation`, thereby fullfilling R5. As for the B2Bi requirements, BPEL 2.0 has no built-in mechanisms for scoping transactions, which must be implemented using additional standards such as WS-Coordination and WS-AtomicTransaction. Policies using these standards can be attached to operations at the WSDL-level. This policy-based approach

[4] This may seem obvious. Nevertheless, there are languages, such as Windows Workflow in its current revision 4 [3], that do not support contract-first development.

[5] Examples of such use cases are the *RosettaNet Implementation Guides*: `http://www.rosettanet.org/Support/ImplementingRosettaNetStandards/RosettaNetImplementationGuides/tabid/2985/Default.aspx`

[6] Such a structure is introduced by the BPEL-SPE specification [9], a BPEL extension for subprocesses. However, this specification is not widely adopted and thus we limit ourselves to the BPEL specification [16] and related WS-standards in this evaluation.

enhances the flexibility and composability of the Web Services stack. However, in the case of B2Bi, it would be reasonable to insert the notion of transactions into the process itself[7]. BPEL 2.0 directly provides only *quasi-atomic* transactions through compensation [2]. Altogether, we consider it to provide only partial support for R6. The same applies to QoS requirements which cannot be represented directly, but with the help of additional standards, such as WS-ReliableMessaging and WS-Security. Moreover, BPEL's support for time constraints is fairly limited [12]. This results in partial support for R7. As BPEL is an OASIS standard, it fullfills R8. BPEL 2.0 uses key-based correlation with `correlationSets` that can be initialized and used by messaging activities, thus fullfilling R9. References can be passed and set via WS-Addressing `endpointReferences` which are first-class citizens of the specification. As this is only an implict way of service selection, it is considered as partial support for R10 [4]. Multi-lateral interaction is possible using multiple `partnerLinks`, fullfilling R11. The applicability of BPEL in top-down approaches has been proven in multiple settings [8, 14, 18, 23, 30], fullfilling R12. Finally, the language is built on Web Services and provides a XML format (R13).

Assessment of BPMN 2.0: A discussion on control-flow pattern support is part of the BPMN 2.0 specification [17], granting support for R1. This is not the case for data or exception handling patterns, but here results from an older revision are applicable [21, 31], fullfilling R3 and R4. R2 is directly supported using `subProcesses` and `callActivities` which are considerably more powerful than BPEL `scopes` as they allow for a reuse of process definitions without having to resort to external Web Service invocation. BPMN 2.0 comes with an extension mechanism, based on `extension` and `extensionDefinition`, that can be used to define additional elements at the level of existing elements. For instance, by extending a `task` with additional attributes, it is possible to provide new functionality. This mechanism fullfills R5. A special type of `subProcess`, `transaction`, can be used to demark a transactional context in a process to provide a consistent outcome to the execution of a set of activities and allows for the configuration of the coordination protocol applied (typically WS-AtomicTransaction or WS-BusinessActivity). Additionally, `hazards` mark events in a `transaction` that enforce its immediate termination without compensation, but without terminating the complete process. Such *scope-termination* is not possible in BPEL. Altogether, this fullfills R6. Concerning QoS, BPMN processes are limited to simple time constraints in the same fashion as BPEL. Although it would be possible to annotate such configurations using `properties`, this only qualifies for partial support with respect to R7. BPMN is an OMG standard, satisfying R8. BPMN 2.0 supports key-based and, in contrast to BPEL 2.0, context-based correlation. This is sufficient for R9. `Participants` can be used to represent interaction partners of a process, thereby fullfilling R11. Service selection is no first-class member of the BPMN 2.0 specification. However, BPMN 2.0 allows to define `endPoints`, which may be comprised of WS-Addressing `endpointReferences`. It is possible to reference these `endPoints` in a `participant` and reassign them

[7] For instance, this is also the strategy followed by [26]. There, policies are introduced at the level of `scopes` or `partnerLinks`, resulting in *coordinated scopes / partnerLinks*.

during process execution. This resembles the solution of BPEL. Therefore, we conclude that R10 is partially supported. The use of BPMN executable processes as orchestrations is just in its start. Nevertheless, in the BPMN environment, they are integrated into diagrams for modeling choreographies, so their applicability in a top-down development approach seems given (R12). Finally, BPMN 2.0 comes with a XML serialization format and in the context of messaging activities and tasks, Web Services are considered the default technology (R13).

5 Conclusion and Future Work

The results are summarized in Table 1. Both languages provide a strong degree of support for the requirements at hand, which is not surprising and the reason they were selected in the first place. Here, nuances in the support are of interest. As there is a more powerful mechanism for hierarchical decomposition and a concept for demarking transactions in BPMN 2.0 processes, they are considered as more suitable for the output of B2B-choreographies. To levitate existing deficiencies, it would be helpful to extend process elements with explicit notions for QoS. Also, the definition of mappings to further languages, such as WF, is promising.

Table 1. Assessment of Languages

Requirement	BPEL 2.0	BPMN 2.0
R1 Control-flow structures	+	+
R2 hierarchical decomposition	+/-	+
R3 Data handling	+	+
R4 Exception handling	+	+
R5 Extensibility	+	+
R6 Transactions	+/-	+
R7 QoS	+/-	+/-
R8 Standards	+	+
R9 Correlation	+	+
R10 Reference passing	+/-	+/-
R11 Multi-lateral interaction	+	+
R12 Choreography integration	+	+
R13 XML, Web Services	+	+

References

1. A. P. Barros, G. Decker, M. Dumas, and F. Weber. Correlation Patterns in Service-Oriented Architectures. In *FASE*, pages 245–259, Braga, Portugal, 2007.
2. A. P. Barros, M. Dumas, and A. H. M. ter Hofstede. Service Interaction Patterns. In *BPM*, pages 302–318, Nancy, France, September 2005.
3. B. Bukovics. *Pro WF: Windows Workflow in .NET 4*. Apress, June 2010. ISBN-13: 978-1-4302-2721-2.
4. G. Decker, O. Kopp, F. Leymann, and M. Weske. Interacting services: From specification to execution. *Data & Knowledge Engineering, Elsevier*, 68(10):946–972, 2009.
5. G. Decker, H. Overdick, and J. Zaha. On the Suitability of WS-CDL for Choreography Modeling. In *EMISA*, pages 21–33, Hamburg, Germany, October 2006.
6. R. Dijkman and M. Dumas. Service-oriented Design: A Multi-viewpoint Approach. *International Journal of Cooperative Information Systems*, 13:337–368, 2004.
7. S. I. Fernando, D. Creager, and A. Simpson. Towards Build-Time Interoperability of Workflow Definition Languages. In *SYNASC*, Washington DC, USA, 2007.
8. B. Hofreiter and C. Huemer. A model-driven top-down approach to inter-organizational systems: From global choreography models to executable BPEL. In *Join Conf CEC, EEE*, 2008.

9. IBM, SAP. *WS-BPEL Extension for Sub-Processes – BPEL-SPE*, September 2005.
10. O. Kopp, F. Leymann, and S. Wagner. Modeling Choreographies: BPMN 2.0 versus BPEL-based Approaches. In *EMISA*, 2011.
11. O. Kopp, D. Martin, D. Wutke, and F. Leymann. The Difference Between Graph-Based and Block-Structured Business Process Modelling Languages. *Enterprise Modelling and Information Systems Architecture, GI e.V.*, 4(1):3–13, 2009.
12. J. Lenhard, A. Schönberger, and G. Wirtz. Edit Distance-Based Pattern Support Assessment of Orchestration Languages. In *OTM Conferences*, Hersonissos, 2011.
13. F. Leymann. BPEL vs. BPMN 2.0: Should You Care? In *2nd International Workshop on BPMN*, 2010.
14. J. Mendling and M. Hafner. From WS-CDL choreography to BPEL process orchestration. *JEIM*, 21(5):525 – 542, 2008.
15. M. Murzek and G. Kramler. The Model Morphing Approach - Horizontal Transformations between Business Process Models. In *BIR*, pages 88 – 103, 2007.
16. OASIS. *Web Services Business Process Execution Language*, April 2007. v2.0.
17. OMG. *Business Process Model and Notation (BPMN) Version 2.0*, January 2011.
18. C. Ouyang, M. Dumas, A. H. M. ter Hofstede, and W. M. P. van der Aalst. From BPMN Process Models to BPEL Web Services. In *ICWS*, pages 285–292, 2006.
19. C. Peltz. Web Services Orchestration and Choreography. *IEEE Computer*, 36(10):46–52, October 2003.
20. N. Russell, A. H. M. ter Hofstede, D. Edmond, and W. M. P. van der Aalst. Workflow Data Patterns: Identification, Representation and Tool Support. In *ER*, LNCS, pages 353–368, Klagenfurt, Austria, October 2005. Springer, Heidelberg.
21. N. Russell, W. M. P. van der Aalst, and A. H. M. ter Hofstede. Workflow Exception Patterns. In *CAiSE*, pages 288–302, Luxembourg, Luxembourg, June 2006. Springer.
22. A. Schönberger, C. Wilms, and G. Wirtz. A Requirements Analysis of Business-to-Business Integration. Technical Report 83, Otto-Friedrich-Universität Bamberg, December 2009. ISSN 0937-3349.
23. A. Schönberger. The CHORCH B2Bi approach: Performing ebBP choreographies as distributed BPEL orchestrations. In *SC4B2B*, Miami, Florida, USA, July 2010.
24. A. Schönberger. Visualizing B2Bi choreographies. In *4th IEEE International Conference on Service-Oriented Computing and Applications*. IEEE, 2011.
25. S. Shetty and S. Vadivel. Interoperability issues seen in Web Services. *International Journal of Computer Science and Network Security*, 9(8):160 – 168, 2009.
26. S. Tai, R. Khalaf, and T. Mikalsen. Composition of Coordinated Web Services. In *ACM/IFIP/USENIX International Middleware Conference*, 2004.
27. W. M. P. van der Aalst, N. Lohmann, P. Massuthe, C. Stahl, and K. Wolf. From Public Views to Private Views - Correctness-by-Design for Services. In *Web Services and Formal Methods, Fourth International Workshop (WS-FM)*, Brisbane, 2007.
28. W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow Patterns. *Distributed and Parallel Databases, Springer*, 14(1):5–51, 2003.
29. D. Vanderhaeghen, S. Zang, A. Hofer, and O. Adam. XMLbased Transformation of Business Process Models - Enabler for Collaborative Business Process Management. In *XML4BPM*, pages 81 – 94, 2005.
30. I. Weber, J. Haller, and J. Mulle. Automated Derivation of Executable Business Processes from Choreographies in Virtual Organisations. *IJBPIM*, 3:85–95, 2008.
31. P. Wohed, W. M. P. van der Aalst, M. Dumas, A. H. M. ter Hofstede, and N. Russell. On the Suitability of BPMN for Business Process Modelling. In *Business Process Management*, pages 161–176, Vienna, Austria, September 2006.