A scenario is a behavioral view – Orchestrating services by scenario integration

Dirk Fahland

Humboldt-Universität zu Berlin, Institut für Informatik,
Unter den Linden 6, 10099 Berlin, Germany
fahland@informatik.hu-berlin.de

Abstract. The construction of a complex service orchestration is a tedious and error-prone tasks as multiple service interactions with a single orchestrating service must be specified and combined. We suggest to specify a service orchestration in terms of behavioral scenarios that capture a specific aspect of service interaction, a *behavioral view* in isolation. By synchronizing the different scenarios, the views get integrated and define the behavior of a complex service orchestration. Our formal model for scenarios and their integration is a class of Petri nets called oclets.

Keywords: service choreography, view, scenario, Petri nets

1 Behavioral views for service orchestration

In service-oriented computing [1], services serve as building blocks for complex, distributed systems. A service orchestration is a means to coordinate n services by a $(n+1)$st service, called *orchestrator* that communicates with each of the n given services directly and coordinates the overall exchange by its internal logic [2]. Process modeling languages like BPEL combine workflow modeling with the SoC paradigm for specifying and executing orchestrator services.

While specifying the orchestrator's interaction with a specific, individual service is usually straight forward, the coordination of all interactions with all partner services remains a challenge. If service interactions are sufficiently complex and depend on each other, the resulting orchestrator logic will be complex, and so will be the orchestrator model. Constructing a comprehensible orchestrator model with appropriate sub-processes etc. is a tedious task. The sub-process hierarchy usually needs to be remodeled if another service interaction, that cuts across the present hierarchy, is to be integrated into the model.

The problem itself is not new and relates to the problem of *process integration* for classical (non-communicating) process models [3]. A recurring solution approach for behavioral integration takes inspiration from databases where a complex relational database is constructed by integrating the *views* of the various users on the database [4]. A database view is, in principle, a projection of a complete database onto a few specific somehow connected objects of interest. Such a view has a valid interpretation in isolation. Conversely, a "complete"

set of views describes the entire database; hence it can be constructed from its views. If the set of views is not complete or do not fit, they have to be adjusted, viz. *integrated*.

In this paper, we propose the concept of a *behavioral view* for the construction of behavioral models. In analogy to databases, a behavioral view is, in principle, a projection of a complete behavioral model onto a specific behavior, i.e. a partial process execution also called *scenario*. Conversely, a "complete" set of behavioral views describes the entire behavior. A behavioral view can have arbitrary structure (as longs as it is a connected partial execution); it may therefore cut hierarchies and hence is a means to express cross-cutting concerns in process models. While the definition of a behavioral view is straight forward, the converse, their integration to form a complete behavioral model is non-trivial.

We argue that a well-founded approach for behavior modeling by view integration requires an appropriate formal model. In [5] we proposed the Petri net class of *oclets* as a formal model for scenarios (or behavioral views) on the basis of a formal, operational semantics [6]. Intuitively, this formal model constructs complex behavior by concatenating and merging "fitting" scenarios. This reduces the problem of behavioral integration to make a given set of scenarios "fitting" to each other. In general, the solution is to unify the tasks and resources occurring in the scenarios appropriately.

In the remainder of this paper, we first substantiate the concept of a behavioral view in Sect. 2 as we introduce oclets, and their semantics at an intuitive level. We subsequently explain the problem of behavioral integration and suggest a procedure for process integration by the help of an example. We conclude the paper with a discussion of related work and a presentation of open research problems in Section 4.

2 Scenario-based service modeling with oclets

For formally modeling services, the Petri net class of *open nets* has been established. Open nets allow for a rigorous analysis of behavioral properties of services while industrial service modeling languages can be translated to open nets [7], and vice versa [8]. The behavior of an open net is defined by standard Petri net semantics; this operational semantics defines the (partially-ordered) runs of the modeled service. As every service model is finite, these runs are not arbitrary, but exhibit a certain structure, e.g. transitions always fire in a specific order.

Scenario-based models exploit this regularity of the runs: One can identify various repeating patterns (*scenarios*) that are partial executions of the service. The entire service behavior is composed of these scenarios. A scenario-based model makes a scenario a modeling artifact. The entire service behavior is expressed as a set of scenario; a corresponding formal semantics describes how scenarios compose to runs.

In [5], we propose the Petri net class of oclets for formally describing scenarios with an operational formal semantics. An *oclet* is an acyclic Petri net with a local precondition describing which requirements must be satisfied in order

to execute the subsequent scenario. We denote service communication by annotating transitions by incoming arrows (receive a message) and outgoing arrows (send a message).

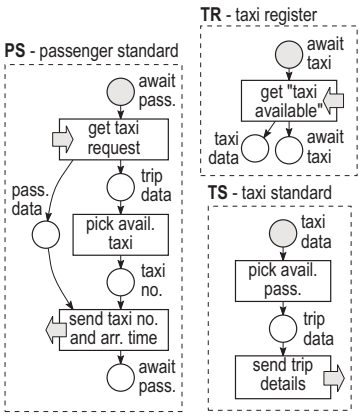

Fig. 1. Standard service interaction with passengers and taxis.

Figure 1 depicts some oclets; the minimal (no predecessor) grey-shaded places denote the precondition of the oclet. Our running example is a passenger-taxi coordination service which is specified in two views:

(1) The first view (oclet PS) specifies that a passenger can send a pickup request to the service, which it processes by picking an available taxi and returning taxi number and arrival time.

(2) The second view (oclets TR and TS) specifies that an available taxi can register at the service at any time. If taxi data is registered at the service, the service will pick a matching passenger and send the corresponding trip details to the taxi.

Figure 3 depicts a partially ordered run of the oclets PS, TR, TS as an acylic Petri net. The run is constructed by merging (copies of) oclets at equally labeled nodes in the obvious and intuitive manner. The run of Fig. 3 shows explicitly that the passenger view and the taxi view are not related to each other, while each makes sense on its own.

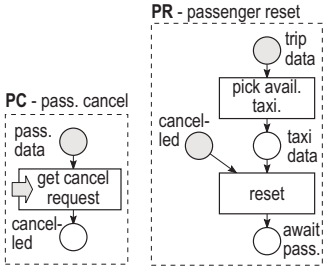

Fig. 2. Service cancelation by passenger and service reset.

We can easily extend our view based model with another view, see Fig. 2: A passenger may cancel a request at any time (oclet PC); the service resets its processing subsequently (PR). This allows to construct the run depicted in Fig. 4. Thereby oclet PS of Fig. 1 is not executed completely (transition send taxi no is not enabled because get cancel request of PC occurred. Instead, oclet PR is appended by merging transitions pick avail. taxi of PS and PR.

The run of Fig. 4 shows again that the given behavioral views do not fit to each other; the taxi still gets notified about the passenger although the request has been canceled. The views must be integrated.

3 View integration with scenarios

We just have introduced oclets as a modeling language for behavioral views and show that if behavioral views do not fit to each other, they cannot be composed to a run. In this section, we sketch how behavioral views of services can be integrated to define a consistent orchestrator service.

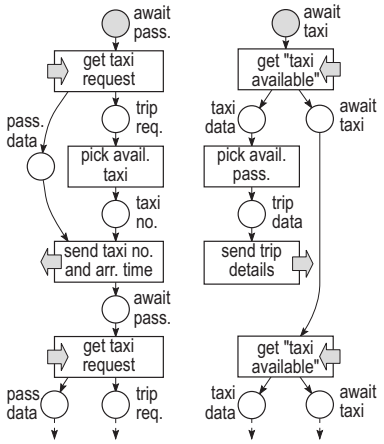

Fig. 3. A standard run constructed from scenarios ps, tr, ts.

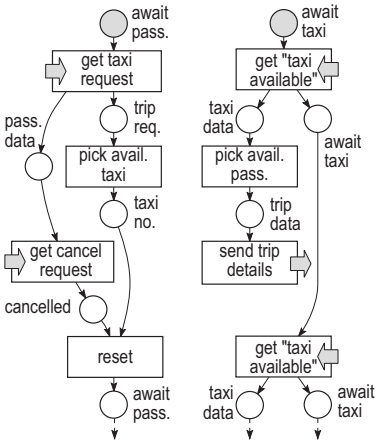

Fig. 4. A run with service cancellation constructed from scenarios ps, tr, ts, pc, tc.

We suggest the following *integration procedure* depicted in Fig. 5. The orchestrator's interaction with each of its partner services is specified in a behavioral view. To integrate the different views, (1) all views are refined to the same granularity of resources and tasks; (2) a modeler identifies points of synchronization (specific resources or tasks) and defines integration options. Finally, (3) synchronization is made explicit in each scenario by applying the integration options. This adjusts the different scenarios to each other s.t. the formal semantics of oclets constructs the orchestrator's behavior by concatenating and merging the now fitting scenarios. Possibly, some integration issues are not visible after the first integration step, so steps (2) and (3) are iterated until satisfaction. The grey tasks of Fig. 5 require interaction with a human modeler.

As all oclets of our example process already have equal granularity, the first step changes nothing. We now have to define integration options for our scenarios in order to unify the given oclets accordingly. An *integration option* maps a set of transitions of the oclets to be integrated to a (possibly new) transition. For some kinds of integrations like *parallel synchronization* the resulting integrated transition is a function of the integration inputs; see [3].

In our example, the standard behavior in Fig. 3 suggests to integrate the views via transitions pick avail. taxi and pick avail. pass by the parallel composition depicted in Fig 6. Further, the reset oclet pr of Fig. 2 must be extended to properly handle cancelation also in the service's interaction with the taxi. Applying this integration on all oclets in the subsequent unification step re-

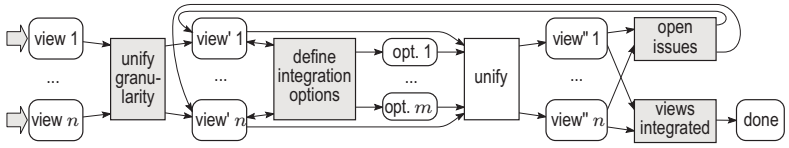

Fig. 5. Procedure for integrating behavioral views.

places transitions pick avail. taxi and pick avail. pass each by the integration result match request; see oclets PS2, TS2, and TCS in Fig. 7.

One can quickly see that this integration option alone is insufficient: The reset transition of TCS only resets the passenger thread of the process while the taxi thread is unmodified. A human modeler who inspects the integration result TCS can detect this problem. Hence, another (obvious) integration option that extends transition reset of Fig. 7 by the dashed dependencies must be specified. After this integration step, the passenger view and the taxi view are integrated, but still exist in isolation.

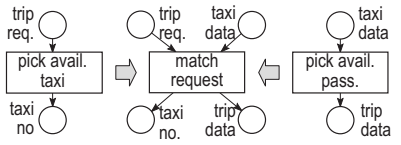

Fig. 6. View integration option

The integrated service model is now given by oclets PS2, TR, TS2, PC, and TCS. Figure 8 depicts a standard run of the integrated service while Fig. 9 depicts a run with cancelation. Thereby, the match request transitions of the various oclets are merged upon construction of the runs as they are now pairwise compatible in term of enabling condition and effect. The integration of the different scenarios on common transitions is a consequence of our formal model [6].

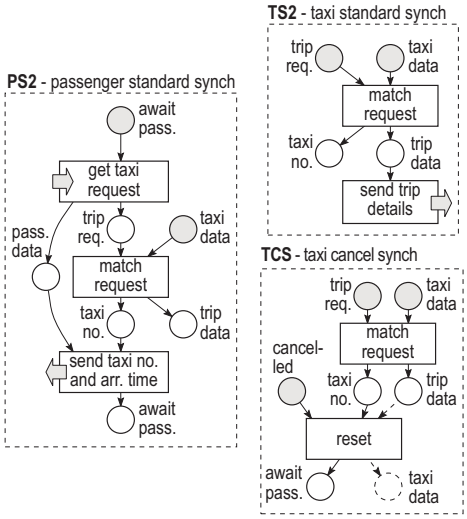

Fig. 7. Integrated oclets with unified transitions and enabling conditions.

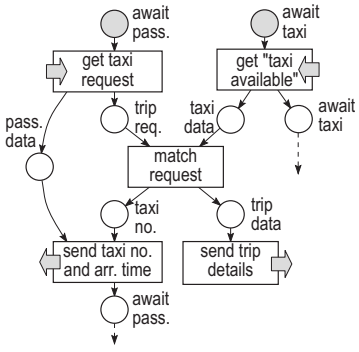

Fig. 8. A standard run of the integrated scenarios ps2, tr, ts2.

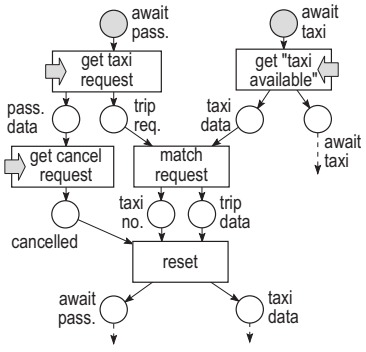

Fig. 9. A run with service cancelation of the integrated scenarios ps2, tr, ts2, pc, tcs.

4 Conclusion

We proposed scenarios as behavioral views for modeling complex service orchestrations. The interaction of an orchestrating service with each of its partner services is modeled separately in terms of their interaction scenarios. In a subsequent integration step, the different views are first unified in granularity; then integration options for synchronizing different views are defined by the modeler; finally, the views are unified according to the integration options. We proposed the Petri net class of oclets as a formal model for scenarios; their operational semantics allow to construct the behavior of the orchestrator directly from the unified scenarios.

Our proposition is a contribution to the relatively novel field of service (or process construction) by view integration [9]. Currently, there exists no systematic solution for behavioral process integration. Initial works like [4, 10] consider the problem from an information system perspective where database schema integration is extended by considering behavior of data processing as well. The problem of behavioral integration is now also being research in isolation. The general line of research aims on constructing complex, integrated process models by merging smaller process models (the views) on synchronization points. In [11, 12] different behavioral process integration options are considered for constructing such merged process models. These integration options can be applied when merging UML activity diagrams for constructing complex process models [3]. Similar results are available for Petri net based models [10] and EPCs [13].

One markable observation in all these approaches is that prior to integration, the process models have to be flattened in order to define and apply the integration options. While some sort of task grouping can still be applied [10], the modeler essentially works on an ever growing complex model. Because process integration involves frequent human interaction (see Sect. 3), this complexity becomes a problem. One of the main problems appears to be that in classical models, behavioral integration is achieved only by model composition. The notion of behavioral view must essentially be given up in order to integrate. Because, at the same time, the model must be flattened, there are no abstraction techniques available to support the modeler in reducing the complexity.

We argue that a scenario-based service model, as the one suggested on this paper, preserves the advantages of view-based service definitions. We have that different behavioral views can be unified in a way that they yields behavioral integration of the views while *preserving* each view. Only local changes must be made to achieve integration. Thus original view, and integrated views can still be related to each other. This provides smaller modeling artifacts, and hence an abstraction technique that suits the problem of process integration. The formal semantics of oclets yields a precise behavioral definition.

While the formalization of a scenario as an oclet and its formal semantics are available, the following questions remain to be answered for actually applying oclets for service integration: If two oclets specify behavior at different levels of granularity, how can a common granularity be achieved? What are the available options to integrate two given oclets of same granularity? Can these options

be computed automatically? How does the integration of two oclets affect the integration of other oclets? Given a set of integration options, are the unified oclets that realize the integration canonical? Are the integrated oclets consistent? Can inconsistencies be computed automatically?

We do not claim this list to be complete, but we consider answers to these questions to be fundamental for a systematic solution for process integration, whether using oclets or any other approach.

Acknowledgements We would like to thank Jan Mendling for bringing this topic to our attention and for the reviewers' comments and suggestions that helped improving this paper. D. Fahland is funded by the DFG-Graduiertenkolleg 1324 "METRIK".

References

1. Papazoglou, M.P.: Agent-oriented technology in support of e-business. Commun. ACM **44**(4) (2001) 71–77
2. Dijkman, R.M., Dumas, M.: Service-oriented design: A multi-viewpoint approach. Int. J. Cooperative Inf. Syst. **13**(4) (2004) 337–368
3. Grossmann, G., Ren, Y., Schrefl, M., Stumptner, M.: Behavior based integration of composite business processes. In: BPM'05. Volume 3649 of LNCS. (2005) 186–204
4. Schmitt, I., Saake, G.: A comprehensive database schema integration method based on the theory of formal concepts. Acta Inf. **41**(7-8) (2005) 475–524
5. Fahland, D., Woith, H.: Towards process models for disaster response. In: Workshops of the BPM'08, Milan, Italy (September 2008) LNBIP to appear.
6. Fahland, D.: Oclets - a formal approach to adaptive systems using scenario-based concepts. Informatik-Berichte 223, Humboldt-Universität zu Berlin (2008)
7. Lohmann, N., Massuthe, P., Stahl, C., Weinberg, D.: Analyzing interacting WS-BPEL processes using flexible model generation. Data Knowl. Eng. **64**(1) (January 2008) 38–54
8. Lohmann, N., Kleine, J.: Fully-automatic Translation of Open Workflow Net Models into Human-readable Abstract BPEL Processes. In: Modellierung 2008. Volume P-127 of LNI., GI (March 2008) 57–72
9. ACM, C.: Special issue: Developing and integrating enterprise components and services. Commun. ACM **45**(10) (Oct. 2002)
10. Preuner, G., Conrad, S., Schrefl, M.: View integration of behavior in object-oriented design. Data Knowl. Eng. **36** (2001) 153–183
11. Shen, J., Grossmann, G., Yang, Y., Stumptner, M., Schrefl, M., Reiter, T.: Analysis of business process integration in web service context. Future Generation Comp. Syst. **23**(3) (2007) 283–294
12. Grossmann, G., Schrefl, M., Stumptner, M.: Classification of business process correspondences and associated integration operators. In: ER (Workshops). (2004) 653–666
13. Mendling, J., Simon, C.: Business process design by view integration. In: Business Process Management Workshops. (2006) 55–64