Extending Choreography Spheres to Improve Simulations

Oliver Kopp, Katharina Görlach, Frank Leymann

Institute of Architecture of Application Systems,
University of Stuttgart, Germany
{kopp, goerlach, leymann}@iaas.uni-stuttgart.de

Oliver Kopp, Katharina Görlach, and Frank Leymann. 2010. Extending choreography spheres to improve simulations. In *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services* (iiWAS '10). ACM, New York, NY, USA, 696-699. DOI=10.1145/1967486.1967598 http://doi.acm.org/10.1145/1967486.1967598

BIBTEX

```
@inproceedings{ChorSphere4Sim,
  author    = {Oliver Kopp and Katharina G\"orlach and Frank Leymann},
  title     = {Extending Choreography Spheres to Improve Simulations},
  booktitle = {Proceedings of the 12\textsuperscript{th} International
               Conference on Information Integration and Web-based
               Applications \& Services (iiWAS '10)},
  year      = {2010},
  pages     = {694-697},
  publisher = {ACM},
  doi       = {10.1145/1967486.1967598},
  acmid     = {1967598}
}
```

Extending Choreography Spheres to Improve Simulations

Oliver Kopp, Katharina Görlach, Frank Leymann
Institute of Architecture of Application Systems
University of Stuttgart
70569 Stuttgart, Germany
lastname@iaas.uni-stuttgart.de

ABSTRACT

In simulations scientific workflows are used to coordinate complex implementations incorporating different kinds of simulations. Typically, the amount of data to be analyzed is huge and it is impossible to store all intermediate or alternative results. Thus, the access to data services has to be coordinated such that applications read the right data and do not overwrite one another. In this paper, we present a possibility to coordinate different scientific simulations accessing and updating the same data using existing Web service technologies: We extend the concept of choreography spheres by allowing control-links between them and the property "permeability" stating whether a cross-boundary link may be traversed before the start or completion of a choreography sphere. This paper is the first presenting a state model for choreography spheres.

Categories and Subject Descriptors

H.4.1 [**Information Systems Applications**]: Office Automation—*Workflow management*

1. OVERVIEW

Scientific workflows typically deal with a huge amount of data [2] which results in an extensive use of data storages such as databases. Using workflow technology for scientific simulations arises particular challenges such as the automation of scientists work during a simulation. Naturally scientists successively improve their simulation models with the help of multiple simulation runs varying the simulation model or simulation method. In between the scientist compares simulation results with the reality and decides about the possibly needed improvement.

For instance, a linguist studies names and wishes to understand their origins. He has some hypotheses about the evolution of names, e. g. randomly arising nicknames, compound names by marriage, and the lengthening with place names. For evaluation of this hypotheses the scientist models them in a simulation and compares the simulation run with the reality. In our example, the scientist utilizes the Monte Carlo method [6] as simulation method. It is a stochastic simulation method for iteratively evaluating a deterministic model using sets of random numbers as inputs. A simulation can typically involve over 10,000 iterations. This simulation method exists in diverse types and can be implemented by Web services. In our example the Monte Carlo method is parameterized with the simulation model in order to compute the evolution of names based on the name universe. The simulation model is built by a name universe in conjunction with a mathematical representation of the user hypotheses. Since the linguist "only" has hypotheses, there exist some parameters correlated to the hypotheses, which cannot be known by the scientist; e. g. probability distributions of arising nicknames. In the simulation model such values are provided by adjustable parameters that the scientist wants to optimize with the help of different simulation workflows.

In case the user wants to use multiple types of the simulation method we can improve the performance of the simulation by parallel execution of simulation workflows each implementing one simulation method. Since the simulation methods work on the same simulation model, the model is context information for the parallel executed workflows and should be hold outside the workflow models. The linguist in our example, however, defines three different simulation workflows, which vary in the utilized Monte Carlo method and manipulations on adjustable parameter values (see Figure 1). First of all, the simulation workflows fetch values for the simulation model from an external database. Afterward the values of adjustable parameters are optionally manipulated and passed together with the name universe to the particular Monte Carlo method. After execution a resulting name universe and some interesting values about name evolution are returned. The writing service manages the assumption of improved values in the simulation model. In this service the user has to specify a condition that decides whether a simulation result is nearer to the real world than another. If the condition is fulfilled manipulated values of adjustable parameters (and possibly the name universe) are written into the database. As the data is stored at the same place in the database, it has to be ensured that each simulation workflow reads the data before the data is updated by another workflow. In order to automate such work of scientists external coordination techniques have to be applied if the simulation and the simulation database do *not* implement a locking concept allowing only the best result to be stored. This paper presents a possible coordination technique based on *choreography spheres* [9]. Choreography

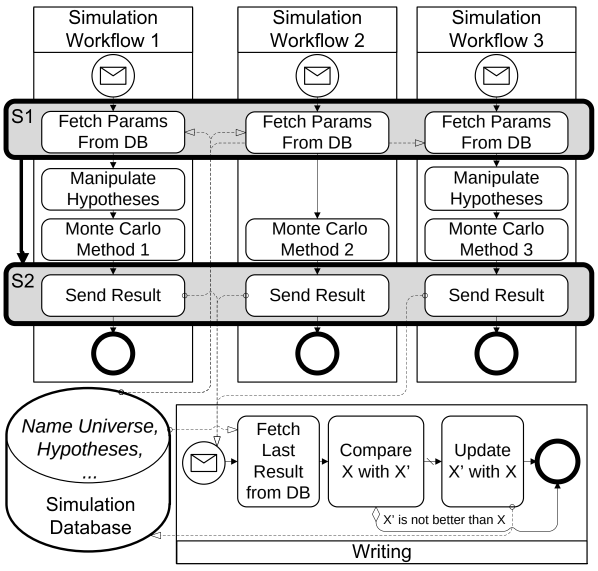

Figure 1: Scenario using choreography spheres

spheres are a technique for placing arbitrary activities of different participants in a choreography in one transaction.

In the scenario, two *choreography spheres* are defined and connected via an additional *control link* (see Figure 1). Sphere S1 is used to combine the three read operations into a single unit of work. Sphere S2 combines the sending operations. In each simulation process, the existing control link from "Fetch Params From DB" to "Manipulate Hypotheses" ("Monte Carlo Method" in the case of simulation workflow 2) and the existing control link from "Monte Carlo Method" to "Send Result" are crossing the boundary of a choreography sphere. S1 is configured to let the control flow pass the cross-boundary control link ("permable"="both"). S2 is configured to defer the execution of the activities inside until S2 is activated by the incoming control link ("permeable"="out"). As a result the cross-boundary control link from "Fetch Params From DB" is processed as usual and the control link to "Send Result" is deferred until all data is fetched. In that way it is ensured that simulation results are written after all workflows have read the initial simulation data.

The choreography spheres used in the scenario do not add custom fault or compensation handlers. Compensation handlers are not necessary as simulation workflows scientists simply rerun simulations and place less value on compensation. A fault can happen at all activities of the workflow. Regardless where the fault happens, the choreography spheres propagate the fault to the other workflows, which finally leads to a termination of each process.

The remainder of this paper is structured as follows: Section 2 provides information on the background and related work. Section 3 presents the overall concept of choreography spheres, the property "permeability", and a state model. Section 4 sketches alternative approaches solving the coordination problem. Finally, Section 5 concludes and provides an outlook on future work.

2. BACKGROUND AND RELATED WORK

In this paper, we use BPEL4Chor [5] as choreography modeling technique. BPEL4Chor builds on the Web Services Business Process Execution Language (BPEL for short) [11]: Each participant is modeled by an abstract BPEL process, which are interconnected using message links listed in the participant topology. As BPEL (and BPEL4Chor) do not specify a graphical rendering method, we use BPMN 1.2 as rendering as presented in [13].

A general evaluation of the Web Services Business Process Execution Language in the context of scientific workflows has been done in [1]. There, exception handling, user interaction, recovery and rollback mechanisms were identified as advantages of BPEL in the scientific domain.

Karastoyanova et al. [7] present an event model for BPEL4WS 1.1, which allows monitoring of BPEL processes and modifying BPEL behavior. A general usage is described in [8]: the workflow engine publishes the events of each activity on a JMS topic. Custom controllers are subscribed to the topic. A dedicated custom controller may sent events to the workflow engine in order to steer the behavior of the engine. In our paper, a custom controller is implementing the behavior of the choreography sphere.

The concept of spheres with transactional behavior has been first introduced in [10]. The spheres were allowed to overlap, but the semantics for choreography spheres was put as future work. Currently, there exist no work on cross-organizational transactions, where arbitrary activities may be chosen to be coordinated. The only work adding cross-process coordination capabilities to choreography models by employing spheres is the concept of choreography spheres [9]. The work presented in this paper builds on that work.

3. CHOREOGRAPHY SPHERES

A choreography sphere is used to group activities of participants in a BPEL4Chor choreography together and to assign properties to the group. In [9], the Boolean properties "2PC" and "compensatable" are used. In case a sphere has the property "2PC", the activities in a sphere are coordinated to ensure an all-or-nothing behavior. The sphere has been called "2PC sphere". Control flow is allowed to leave the sphere as soon as the sphere is completed (and not earlier). Control flow is always allowed to enter the sphere. The property of allowing control flow crossing the boundary of a sphere regardless of its state is called "permeability": A sphere can be permeable for incoming control links as well as permeable for outgoing control links. In the case of 2PC spheres, the sphere is permeable for incoming links, but not permeable for outgoing links: That means, the activities in the sphere may start regardless of a start of the sphere. Control flow crossing the boundary of the sphere may only be followed as soon as all activities in the sphere completed.

Compensatable spheres are called "BPEL spheres" [9]. Similar to a BPEL scope, a compensation handler is attached to the sphere and is installed as soon as the sphere completes. "BPEL spheres" are permeable for both incoming as well as outgoing control links.

This paper introduces the configurable property "permeable" to a choreography sphere. Possible values are "none", "in", "out" and "both"; with "both" being the default. This enables control flow links crossing the boundary of a sphere to be followed even if the sphere itself is not active. BPEL offers the concept of isolated scopes, where outgoing control flow is not allowed to leave the scope until the scope completed and thus are comparable to choreography spheres with in-permeability. The difference to choreography spheres is

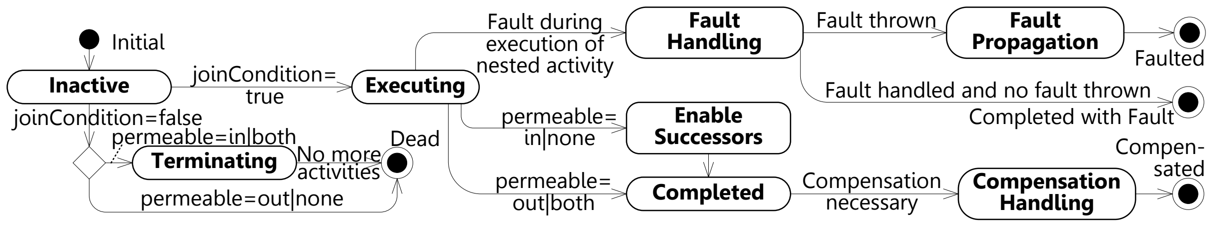

Figure 2: Choreography sphere state model

that only one isolated scope may be active at the same time, whereas choreography spheres may be active in parallel.

Regarding control flow, this paper introduces the connection of spheres with control links. For this paper, we disallow the control flow forming cycles and spheres to overlap. Spheres may only contain activities in the positive control flow of the workflow (and not in the exception handling flow). We do not allow spheres to be partially or fully nested in a loop. This requires more tracking effort in the custom controller and is out of scope of this paper. Aspects of repeatable constructs will be addressed in future work. The semantics of the control links follows the semantics of control links of BPEL. We disallow block-oriented control-flow modeling and force dead-path elimination to be disabled.

The *state model* of a choreography sphere is presented in Figure 2. A sphere is first in the state "Initial". After one participant of the choreography has been instantiated, the sphere is in the State "Inactive". As soon as all incoming links are visited, the join condition is evaluated.

In case the join condition evaluates to false, the sphere is dead. In case "permeable" is set to "in" or "both", activities in the sphere may currently execute or may have been executed. Executing activities have to be terminated and completed activities must be compensated. After termination and compensation of all nested activities the sphere transitions to the state "Dead". In this state, no activity in the sphere may execute any more. In case control flow reaches an activity in the sphere, the activity is handled as dead activity.

In case the join condition evaluates to true, the sphere is in the state "Executing". In case "permeable" is set to "out" or "none", the sphere enables all activities in the state "Ready" to execute. Otherwise, the activities already started execution—independent of the state of the sphere. In all cases, the sphere waits for all nested activities to be completed or being dead. For instance, the latter is the case for activities in `if` branches not chosen. After the execution is finished, the subsequent state depends again on the setting of the "permeable" property: in the case of "in" or "none", activities following the sphere and outgoing links of activities, where the links are crossing the boundary of the sphere, may only followed if the sphere has executed completely. These links are followed in the state "Enabled Successors". In the case of "out" or "both", these links (and activities following the sphere) are executed as soon as the workflow model itself visits them—independent of the state of the sphere.

Activities in a choreography sphere may fault. A fault is raised by an activity either due to a BPEL internal fault such as `uninitializedVariable` or a fault raised by an invoke activity, which propagated a fault of a called Web service. Both kinds of faults are treated the same way in BPEL: A fault is propagated to the next enclosing scope. This scope first terminates all running activities and subsequently looks up a fault handler applicable for the current fault. In case a specific fault handler is found, this fault handler is executed. If no specific fault handler is found, the default fault handler (`catchAll`) is called. In case this fault handler is not explicitly modeled, it compensates all completed nested activities and rethrows the catched fault. A choreography sphere also has to handle faults of nested activities. Thus, a choreography sphere extends local fault handling to cross-partner fault handling: A choreography sphere interrupts the hierarchy relation of scopes and nested activities. That means, if an activity is nested in a scope and assigned to a choreography sphere, this sphere handles the fault (state "Fault Handling"). The behavior of a choreography sphere with respect to fault handling is aligned to the behavior of a BPEL scope: As soon as the fault reaches the choreography sphere, the sphere terminates all running nested activities. This also affects activities of activities residing at processes not being the process where the fault is raised. In case multiple faults are raised, the first raised fault is handled and the others are ignored. An explicit fault handler can include arbitrary activities including compensation of nested activities and throwing a new fault (state "Fault Propagation"). Choreography spheres also offer a default fault handler. If this fault handler is not explicitly modeled, the default behavior is to compensate all completed nested activities and to rethrow the fault. A fault thrown from a choreography sphere is thrown to *all* processes where activities belong to the choreography sphere. In case the property "permeable" of the choreography sphere is set to "out" or "both", the control flow may have continued and the direct parent scope of the activity of the sphere may have completed, too. To ensure a consistent execution of the process, the fault is thrown in the closest parent scope, which is running and where the activity is nested in.

A choreography sphere is completed as soon as all enclosed activities completed or are dead due to dead-path elimination (state "Completed"). An activity in the sphere may be compensated due to a fault or compensation handler in its parent scope. A compensation of an activity in a sphere leads to an inconsistent state of the sphere, since part of the work of the sphere is undone, while other work still has effects. Thus, the sphere also compensates the other activities of the sphere as soon as one activity gets compensated (state "Compensation Handling"). This compensation happens also in other processes. This leads to inconsistent states in the other processes, too. Thus, the sphere raises a fault `unableToComplete` to all running parent scopes of the activities nested in the choreography sphere. The scope which has triggered the compensation is excluded, as this handling ensures the consistency in its process.

4. DISCUSSION

Employing choreography spheres with control links is not the only solution to the coordination problem. Examples for other solutions are: (i) explicitly modeled coordination of the simulation workflows, (ii) customized WS-Coordination protocol and (iii) locks in a customized database system.

Instead of using an additional modeling construct, the coordination between the workflows can be *modeled explicitly*: After fetching the parameters from the simulation database, each workflow may send a "fetched" message to all other workflows. The other workflows wait for the finished message before writing the results back into the simulation database. The drawback of this approach is that each simulation workflow has to be aware of the other workflows used. In case a new parallel workflow is added, all other workflows have to be adapted.

In the field of Web services, *WS-Coordination* is used to coordinate participants [12]. Regarding the example scenario, each workflow is a participant in the coordination. The standardized coordination protocols WS-BA and WS-AT cannot be used to coordinate the simulation workflows as they are targeted to transactions and not to synchronize read and write operations the way required by the scenario. A possible customized coordination protocol is as follows: First, each coordination participant sends a "read" message to the coordinator. After the coordinator collected "read" from all participants, it sends "write" message to each participant. After receiving the "write" message, the participant is allowed to write its result. This customized coordination protocol may be realized in two ways: (a) putting the activities used for WS-Coordination in the simulation workflows or (b) using a BPEL engine, which is aware of the customized coordination protocol. Option (a) is similar to the general option (i), where the coordination between the workflows is modeled explicitly. The advantage of using WS-Coordination is that existing workflows do not need to be adapted in case an additional parallel workflow is added to the simulation or an existing workflow is not used any more.

A *customized database system* may provide locks to ensure that results are only written if the previous results are read [4]. The drawback here is that the database system has to be adapted. A similar option is to implement a check-in/check-out mechanism with branching and merging support. In case the simulation results are too large, the database might not be able to store all versions.

We do not claim that the choreography sphere solution is the best solution in all cases. In case the simulation is executed using BPEL workflows, choreography spheres seem to be the most efficient option: A modeler can define a choreography sphere where it is required. The sphere is then either transformed to activities in the workflow or is directly executed by a workflow engine. The modeler is not required to model the coordination activities explicitly. A proof of concept implementation of choreography spheres is based on WS-Coordination and customized coordination protocols. It is described in [3].

5. CONCLUSION AND OUTLOOK

This paper presented a state model for choreography spheres and introduced control links connecting choreography spheres. We have shown how choreography spheres can be used to automate improvements of simulation models. Especially, we have presented how a simulation can be enrolled by different workflows concurrently and how it can be ensured that the input data is not overwritten by workflows running in parallel. We position the proposed choreography sphere extension as modeling artifact in scenarios where concurrency control is not offered by the used services. One may argue that the presented application scenario for choreography spheres is a solution for a technical detail of the whole simulation process and that the choreography sphere is not used in the first step of choreography modeling, but in the step toward execution. Nevertheless, the construct offers an additional choice on the way from a choreography to executable processes.

The concept has been presented as a solution to coordination issues in parallel scientific workflows. Our future work includes an evaluation of the concept regarding general choreography modeling. Questions driving our work are: What is the performance of the approach compared with other approaches? Which control flow constructs are needed to connect choreography spheres? Does the concept of choreography spheres ease choreography modeling? Is there a higher understandability of the choreography models when using choreography spheres instead of using existing constructs for coordination?

6. ACKNOWLEDGMENTS

This work is partially funded by the DFG Cluster of Excellence "Simulation Technology" (EXC310).

7. REFERENCES

[1] A. Akram, D. Meredith, and R. Allan. Evaluation of BPEL to Scientific Workflows. In *Cluster Computing and the Grid (CCGRID)*. IEEE, 2006.

[2] S. Bharathi et al. Characterization of Scientific Workflows. In *3rd Workshop on Workflows in Support of Large-Scale Science (WORKS08)*, 2008.

[3] S. Bors. A Runtime for BPEL4Chor Cross-Partner-Scopes. Diploma thesis 2990, IAAS, 2010. (in German).

[4] G. Coulouris et al. *Distributed Systems: Concepts and Design*. Addison-Wesley, 2005.

[5] G. Decker, O. Kopp, F. Leymann, and M. Weske. Interacting Services: From Specification to Execution. *Data & Knowledge Engineering*, April 2009.

[6] G. Fishman. *Monte Carlo*. Springer, Feb. 2003.

[7] D. Karastoyanova et al. BPEL Event Model. Technical Report Computer Science 2006/10, IAAS, 2006.

[8] R. Khalaf, D. Karastoyanova, and F. Leymann. Pluggable Framework for Enabling the Execution of Extended BPEL Behavior. In *WESOA*. Springer, 2007.

[9] O. Kopp, M. Wieland, and F. Leymann. Towards Choreography Transactions. In *ZEUS 2009*.

[10] F. Leymann. Supporting Business Transactions via Partial Backward Recovery in Workflow Management Systems. In *BTW'95*. Springer, 1995.

[11] OASIS. *Web Services Business Process Execution Language Version 2.0 – OASIS Standard*, 2007.

[12] OASIS. *Web Services Coordination (WS-Coordination) Version 1.2*, February 2009.

[13] D. Schumm et al. On Visualizing and Modelling BPEL with BPMN. In *4th International Workshop on Workflow Management (ICWM2009)*. IEEE, 2009.