Guiding the Creation of Choreographed Processes with Multiple Instances Based on Data Models

María Teresa Gómez-López, José Miguel Pérez-Álvarez, Angel Jesús Varela-Vaca, and Rafael M. Gasca

Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, Seville, Spain
{maytegomez,josemi,ajvarela,gasca}@us.es
http://www.idea.us.es/

Abstract. Choreography in business processes is used as a mechanism to communicate various organizations, by providing a method to isolate the behaviour of each part and keeping the privacy of their data. Nevertheless, choreography diagrams can also be necessary inside an organization when a single instance of a process needs to interact and be synchronized with multiple instances of another process simultaneously. The description, by business experts, and the implementation, by developers, of these choreographed models are highly complex, especially when the activities involved in the processes exchange various data objects and with different cardinalities. We propose the automatic detection of the synchronization points, when a choreographed process model is needed. The choreography will be derived from the analysis of the process model, data objects consumed and generated through the process, and the data conceptual model that relates the data objects. A graphical tool has been developed to support where the synchronization points must be included, helping to decide about the patterns that describe how a single model can be transformed into a choreographed model.

Keywords: Business process choreography · Multiple instances · Conceptual model · Data model relation

1 Introduction

Process choreography is frequently related to a business contract between two or more organizations, used as a mechanism to communicate various entities, by providing a method to isolate the behaviour of each part and keeping the privacy of their data. Nevertheless, choreography diagrams can also be necessary within an organization when the relation of single and multiple instances need to be synchronized simultaneously. Unfortunately, one aspect that remains a challenge is the description and development of these choreographed business processes. The challenges that choreography establishes, such as correlation and

the inclusion of heterogeneous data, once the process is choreographed, have been studied in [1,2]. However, the creation of the choreographed model remains highly complex, specially when the data objects involved have an N:M cardinality relation. Also, the choreography diagrams are typically too low-level, and the terms of the interactions between the two parties are not always clear.

Regarding our proposal, in an activity-centric paradigm, such as BPMN [3], where the process is described by an explicit order between the activities, the choreography between processes becomes difficult to model and implement in a Business Process Management System (BPMS). One example, where difficulty in choreographing a model can be observed is given by the process that manages the *Calls for Residence* to help University students that want to apply for an assignment of a room in a residence. Figure 1 depicts a reduced example that is enlarged in the next sections: (1) the call for residences is created; (2) students apply by filling out a form; (3) administrative staff of the University evaluate the proposals; (4) the administrative staff notify each student; and, finally, (5) the accepted students can formalize documents and make payments. Although the process is easy to model and understand, it is totally incorrect and incomplete, since the activities cannot be executed in the same and single instance. For example, the activity *Fill out the form* can be executed several times for each *Publish Residence Call*, and not only once per call as the model describes. Something similar occurs between *Evaluate the Proposals* and *Publish a list with the resolution*, since they are not executed the same number of times. The problem of the different number of executions of the activities cannot be solved with a loop activity, since there exists a correlation between each execution of *Fill out the form* and *Formalize documents and make payments* depending on the student. This problem can be solved by using event handlers, however how to create the pools, the activities for the synchronization and the exchanged data is a hard problem that we want to reduce in this paper.

Sometimes inconsistencies in business processes are derived from the activities that consume and produce different data objects. For the example, *Fill out the form* generates a data object *Residence Call*, but *Fill our the form* generates only one *Student Form* per instance, and since it is a competitive process *Evaluate the proposal* uses a set of *Student Forms*. Therefore, in order to discover the inconsistencies related to the data object evolutions and to help in the creation of a correct choreography, it is necessary to include the data objects in the process model. In order to ascertain the points where choreography is required, both an analysis of the expected relation between the objects described in the Data Model, and the data evolution during the process model are needed. The importance of the data perspective and its verification in business process models has been studied in previous work [4]; the challenge now is to ascertain how it can affect the choreography.

In this paper, we propose the combination of various modelling paradigms: business process models described by using BPMN, the data object evolution also included in the BPMN model, and the Data Model that supports the managed data. With these three models combined, it is possible to analyse the process

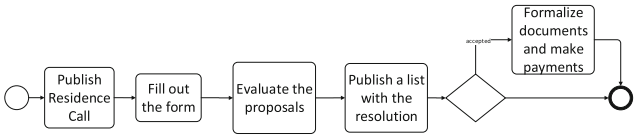

Fig. 1. Example of business process which needs to be choreographed.

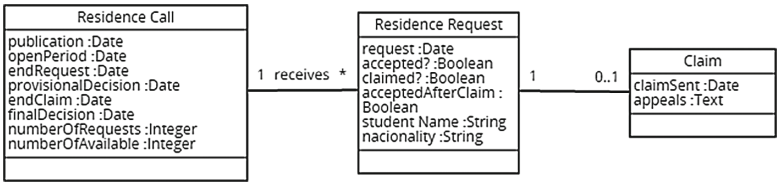

Fig. 2. A piece of the conceptual model.

correctness according to the exchanged data. If any inconsistency is found, the necessary modifications are proposed. The main advantages of our proposal are: (1) we define a model that combines various and different paradigms that are easy to describe independently, but difficult to choreograph; (2) the relations between the objects to generate a correct choreographed model are analysed providing the mechanisms to create a correct model; and, (3) the warnings and the transformation patterns have been implemented on a graphical tool as an extension of Activiti^{TM}.

The remainder of this paper is organized as follows. Section 2 presents the formalization of the concepts and the defined model, including a motivating example. Section 3 presents the suggested approach for the automatic detection of synchronization points and how they can be implemented. Section 4 discusses related work. Finally, Sect. 5 concludes and presents lines for further research.

2 Formalization of Concepts

The modelling paradigms that are combined and included in the formalization are:

- **Conceptual Model for Data Modelling.** The data management used in a process is crucial in achieving the established objectives. Various models can be used to describe the data relation, but we propose the use of a Conceptual Model (cf. Fig. 2) for a higher level of abstraction managed by using Object-Relational Mapping [5].
- **BPMN for Business Process Modelling.** The main goal of BPMN [3] is to provide a standard notation which is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology, and finally, to the business people who will manage and monitor those processes.

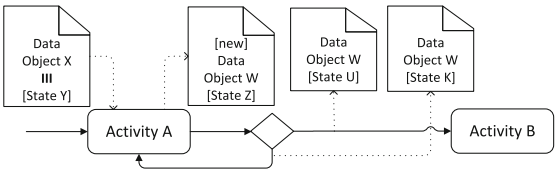

Fig. 3. Example of data object relations.

Thus, BPMN creates a standardized bridge for the gap between the business process design and process implementation.

– **Data Object Life-cycle Modelling.** Although BPMN is not primarily designed for data modelling, there exists a set of notations that enables the designer to model the data involved in a business process. The primary construct of BPMN for modelling data within the process flow is the *Data Object* element. The *Data Objects* can include the State of Data Objects at various points in a process [3]. An activity in a process can *Consume* a Data Object, which implies that the Data Object is in a particular state. When the activity is executed, the object may transit to a new state (an object in a new state is *Produced*). Figure 3 represents an example of accessing the data objects related to *Activity A*. A data flow edge from a data object to an activity describes a read access to an instance of the data object (cf. Data object X), which has to be in this state to execute the activity. Likewise, a data flow edge from an activity to a data object describes a write access (cf. Data object W), which either creates a data object instance if it does not already exist (labelled with [new] as proposed in [6]), or updates the instance if it already exists. A data flow edge connecting a data object with a sequence flow indicates the data object is flowing through that connection, and the state of the flowing data object (cf. in Fig. 3, Data Object W in State U). Data objects can be modelled as a single object or as a collection of objects (marked by three parallel bars, |||). Only the execution of an activity can imply the creation of a new object, although it is possible to represent different states of an object depending on the executed branch, for example, after the execution of activity A, an XOR-split is executed, where the data object W in state U flows through the upper branch, or in state K for the lower branch.

2.1 Model Formalization

The model applied in the formalization and automatic choreography of our problem is composed of two sub-models: (1) one model to describe the Conceptual Model (CM_{Graph}); and (2) one model to describe the components related to the business process model (BP_{Graph}).

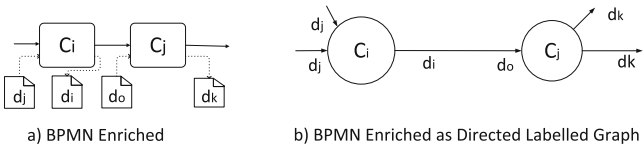

Fig. 4. Labelled directed graph to formalize the BP_{Graph}.

Formalization of the Conceptual Model. A conceptual model is described as a connected labelled graph (CM_{Graph}) composed of nodes (EN) and edges (AS), $CM_{Graph} = \langle \text{EN, AS} \rangle$, where:

- EN is the disjoint union of the entities of the conceptual model ($e_1 \ldots e_n$),
- AS is the set of edges that represent the association between two entities (e_i, e_j). Each edge can be labelled with two values that represent the cardinality ($card_i$, $card_j$), and each one can represent simple (1 or 0..1) or multiple (0..N or 1..N) cardinality.

Formalization of the Enriched Process Model. The enriched business process is modelled as a directed labelled connected graph (BP_{Graph}), based on the annotated graph presented in [7,8], composed of nodes (C) and edges (F), $BP_{Graph} = \langle \text{C, F} \rangle$ whose equivalences are shown in Fig. 4, where:

- C is the disjoint sets of components formed of activities A, events E, and gateways G:
  - A is the set of activities.
  - E is the set of events, that can be partitioned into disjoint sets of one start event E^s, intermediate events E^i and end events E^e. Just one start event is allowed in the model, and therefore there is one and only one node whose input degree is equal to 0.
  - G can be partitioned into disjoint sets of parallel-fork gateways G^F, parallel-join gateways G^J, data-based *XOR*-decision gateways G^D and *XOR*-merge gateways G^M.
- F is the set of edges that represent the association between the components, such as activities. Each edge is labelled with two values (origin and destination of the arc) $\langle d_i, d_o \rangle$. From the point of view of the nodes (C components), $d_{Consumed}$ and $d_{Produced}$ are the incoming and outgoing data objects respectively. Each Data Object (DO), associated to the origin (d_i) or to the end (d_o) of each directed edge, is described by the tuple $\langle$*Entity*, *Cardinality*, *new*$\rangle$, where:
  - *Entity* is one of the entities EN described in the CM_{Graph} ($\in e_1 \ldots e_n$),
  - *Cardinality*: single or multiples (collection represented by |||) objects,
  - *new:* a Boolean that represents whether the data object is *[new]*.

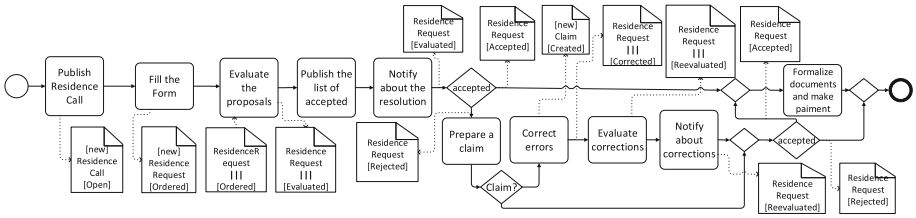

Fig. 5. No choreographed process for the management of University Residences.

2.2 Application Case Study

The business process shown in Fig. 5 extends the previous example for the assignment of rooms in the University Halls of Residences of Seville. Firstly, the call for positions is published, and students can apply by filling out a request form. Once the requests have been sent to the administration staff, a competitive analysis is performed. According to the resolution, the students are notified. Each student analyses whether the assigned residence is suitable and decides whether to claim. If a student decides to claim, then the documentation is revised again. The final decision is sent to each candidate, and the final list is published.

3 Guiding the Transformation to a Consistent Choreographed Model

As explained in the introduction, a correct business process workflow can be incorrect for the data object consumed and produced in the model. The necessity or not to transform a BPMN model into another choreographed model can be derived from the necessity to synchronize processes with multiple and different number of instances.

3.1 Choreography Derived from Data Object Relation

As explained previously, incorrectness in business processes can be detected through the data used in each activity, and therefore the choreography becomes necessary. This analysis is based on allocating the activities involved with objects with the same cardinality in the same pool, where a pool is the representation of a participant in a collaboration [3]. The cardinality and relation of the objects are described in the conceptual model. For the fragment of the Conceptual Model shown in Fig. 2, two activities that consume *Residence Call* and *Residence Request* respectively, cannot be in the same pool, since both data objects have a 1:N relation in the conceptual model. This incompatibility arises due to the fact that both activities cannot be executed the same number of times in the same instance, and for each evaluation of a *Residence Call*, several managements of *Residence Request* can be performed. Otherwise, two activities that consume *Residence Call* and a set of *Residence Requests*, respectively, can be

in the same pool, since both data objects have a 1:N relation in the conceptual model. In an equivalent way, the activities that manage a *Residence Request* and a *Claim*, can be in the same pool since there exists a 1:1 relation between them in the conceptual model. The relations in the conceptual model are determined by a direct association between two entities, such as between *Residence Call* and *Residence Request*, or transitively, such as between *Residence Call* and *Claim*. Derived from this idea, we include the following definitions:

Definition 1. Data Context. A Data Object *D* is **in the same context** as a Data Object *D'* iff:

- D and D' are single objects and they have a 1:1 relation in the conceptual model,
- D is a single object and D' is a collection of objects, and they have a 1:N relation in the conceptual model,
- D is a collection of objects and D' is a single object, and they have an N:1 relation in the conceptual model,
- D and D' are collections of objects, and they have a 1:1 or N:N relation in the conceptual model.

Definition 2. Points of Choreography. Being f_i an edge that connects two C components (c_i, c_j) with the data object d_i and d_j, if d_i and d_j do not belong to the same Data Context, then f_i is a point of choreography. There are two types of Points of Choreography:

- **Many-to-one point:** In BP_{Graph}, it implies that $card_i > card_j$ in CM_{Graph}, and the cardinality of d_i is $\leq$ to the cardinality of d_j. This is, for example, the relation between *Residence Call* and *Residence Request*.
- **One-to-many point:** It implies that $card_i < card_j$ in CM_{Graph}, and $d_i \geq d_j$ in BP_{Graph}. This is, for example, the relation between *Residence Request* and *Collection of Residence Request*.

In order to model the Point of Choreography following BPMN 2.0 [3], it is necessary to take into account that each interaction has an initiator or sender (the party sending the message), and a recipient or receiver (the party receiving the message, who may reply with a return message). In order to choreograph the instances of two pools, multi-instance activities ($\equiv$) to send and receive messages are used. They are divided into:

- **Loop Activity (↻)** is a type of activity that acts as a wrapper for an inner activity that can be executed multiple times in sequence. The Loop Activity executes the inner Activity as long as the *loop Condition* evaluates to true.
- **Parallel Activity (|||)** is a type of activity that acts as a wrapper for an activity that has multiple instances spawned in parallel, and can be used to receive the messages of another process.

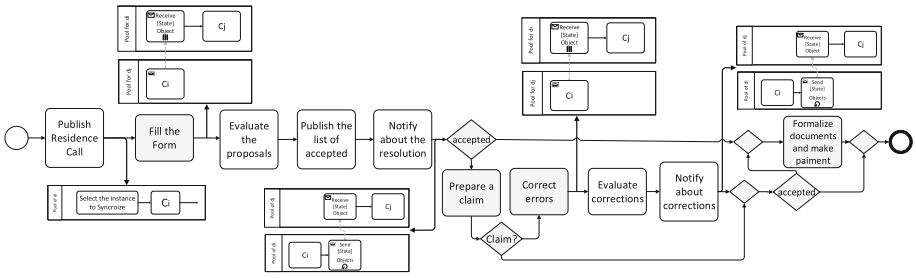

Fig. 6. Example of annotated process about University Residences.

3.2 Patterns for Synchronization Point Identification

Assuming that BP_{Graph} must be correct according to the workflow [9–11], our analysis about the data aspect correctness is centred in the existence of various pools to support the components that manage different Data Contexts than pools. In order to ascertain that a business process model formed by one or more than one pool is correct related to exchanged data, two properties must be satisfied:

1. Every $d_{Consumed}$ and $d_{Produced}$ by the components c_i of a pool P, must belong to the same *Data Context* of P (Definition 1).
2. For every point of synchronization (Definition 2) where (c_i, c_j) are involved, c_i and c_j must belong to different pools.

If it is found an incorrect synchronization point that does not satisfy the first property, or a $d_{Produced}$ that does not satisfy the second property, the following warnings (in Fig. 6) will be annotated in the original model (Fig. 5):

1. **Data Objects Consumed and Produced by an Activity** (Fig. 7.a): If $d_{Consumed}$ (d_i) and $d_{Produced}$ (d_j) by a component C_i do not belong to the same Data Context, two situations and labelled patterns can be found:
   - If d_i is a *new* data object, such as the activity *Fill out the form*, then synchronization is necessary. To synchronize the instance of d_i and d_j, a new activity to correlate the instances of different tasks must be located immediately before the component C_i (Fig. 7.a.I), such as the activity *Select the Residence Call* for the example of Fig. 6. This is necessary because for each d_i, several d_j can be created, and therefore when a d_j is created, it is necessary to determine the corresponding d_i.
   - Else, two types of synchronization can be carried out:
     - **Many-to-One Relation** (Fig. 7.a.II)) advises the creation of synchronization by means of a *Loop Activity* that forms a wrapper around for the inner activity C_i, which can be executed multiple times to send messages, and a single activity that receives the messages. In Fig. 6, an example is the activity *Notify the resolution*.

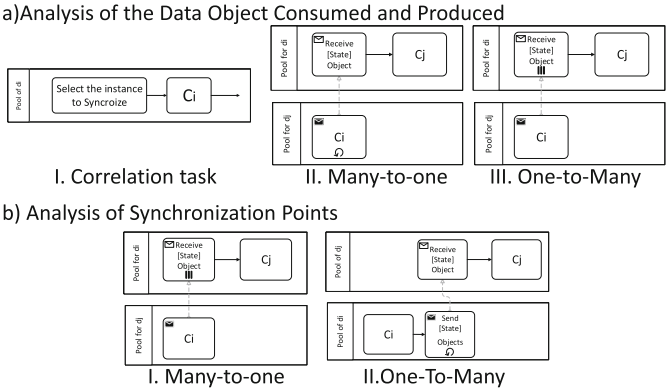

Fig. 7. Patterns of synchronization

- **One-to-Many Relation** (Fig. 7.a.III)) advises the creation of synchronization by means of the single activity C_i that also sends messages to synchronize with a *Parallel Activity* that receives these messages.

2. **Points of Synchronization between two Activities** (Fig. 7.b): The synchronization points involve components that must be in different pools. It implies that a correct choreographed process must have, at least, one pool for each Data Context. For the example, at least, two pools must be necessary to cover the activities that manage the data objects {*Residence Call*, *Collection of Residence Request*} and {*Residence Request*, *Claim*}, respectively. In order to highlight the components that can belong to the same pool, we have coloured some activities of Fig. 6 to distinguish the tasks of both pools. There exist two types of synchronization to satisfy the properties expressed before when d_i and d_o are not in the same Data Context:
   - **Many-to-One Relation** (Fig. 7.b.I) which synchronizes by means of two extra activities: a *Loop Activity*, to send messages; and an activity to receive the messages in the another pool, such as the activity *Notify the resolution*.
   - **One-to-Many Relation** (Fig. 7.b.II), which synchronizes by means of two extra activities: a *Parallel Activity*, to receive messages; and a single activity to send the messages, such as *Send Residence Request*.

3.3 Architecture of the Implementation

In order to evaluate our proposal, we have implemented a graphical tool that enables the inclusion of a connection to a database to create the Conceptual Model automatically using ORM (Hibernate in our case). It has been developed as an extension of Activiti^{TM} [12] with additional components (cf. Fig. 8), since this is an open-source distribution that offers the business expert a friendly

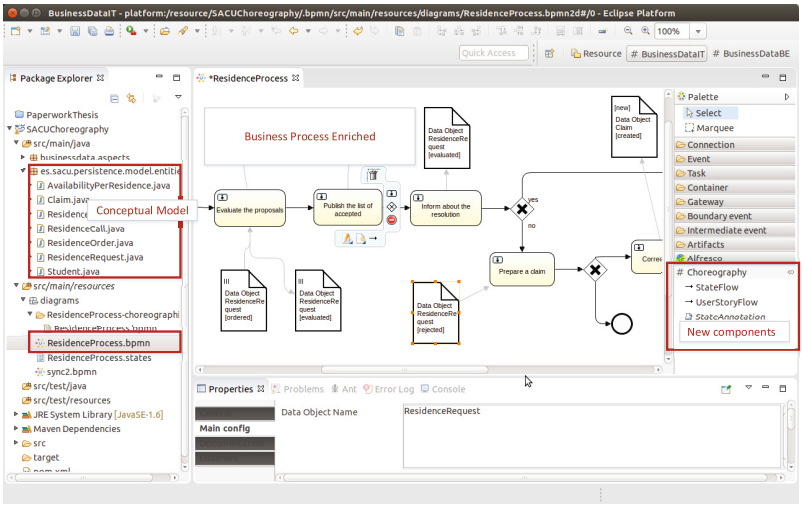

Fig. 8. Screenshot of the Activiti^{TM} process modeller plugin.

interface to model the process using BPMN elements and easy to extend with new components and functionality.

Activiti^{TM} is a light-weight workflow and Business Process Management Platform targeted at business people, developers and system administration. Since neither the data state description nor warning patterns are included in Activiti^{TM}, an extension of Eclipse-based Activiti^{TM} Designer has been developed to support the new graphical elements, and included in the graphical tool (with a video available in: http://www.idea.us.es/condchoreography/).

4 Related Work

The complexity of managing the choreography of business processes is well-known. Certain studies have been published describing the implementation of business process choreographies [13,14]. However, these only take into account the control flow, although the data flow across messages is also an important aspect. In [15], the authors introduce a first proposal about data-aware collaboration.

According to the data inclusion the process models, it is crucial to analyse the evolution of the objects [16], and to determine whether there are data dependencies and which these dependencies are [17] in order to define whose model can be aligned in BPMN model [6,18].

The automating data exchange in a process choreography has been solved in [1], where the choreography model is known. Nevertheless, we consider the difficulty encountered when faced with how to create a correct choreography model specially derived from the limitations of BPMN studied in [19], or how to perform the choreography using technologies such as BPEL [20].

[21,22] propose extensions to choreography modelling by means of interaction patterns between the involved participants. But these works do not study the choreography problem derived from the synchronization of multiple instances necessary for the data dependencies. They only face the choreography problem when the activities must be executed by different organizations, but with a one-to-one relation between the instances executed in the different pools that tend to be produced in conversation environment. Khalaf et al. have introduced an approach for the partitioning of single process models into multiple participants [2,23] including the data aspect. However this partitioning cannot detect the necessity to synchronize N instances of a process with another derived from the data that manage.

5 Conclusions and Future Work

In this paper, we propose an automatic detection of the necessary and type of synchronization points in a process models derived from the exchanged objects. The detection of the synchronization points is based on the comparison between the type and cardinality of the data objects consumed and produced by the components of a business process, and on the relation in the conceptual model. We have developed a graphical tool as an extension of Activiti, where the business process, conceptual model and data object description can be included in the same model, and the warnings related to synchronization correctness are located in the process automatically. The result has been applied to a real example of the University of Seville.

As future work, we propose an extension of the proposal to create a new synchronized model automatically. Also we consider relevant the application of user stories combined with business processes to facilitate the recovery of requirements, and the automatic creation of lanes and pools during the process of choreography.

Acknowledgement. This work has been partially funded by the Ministry of Science and Technology of Spain (TIN2015-63502-C3-2-R) and the European Regional Development Fund (ERDF/FEDER). We would like to thank SACU of the University of Seville for the valuable information that has contributed towards the development of the ideas in this paper.

References

1. Meyer, A., Pufahl, L., Batoulis, K., Fahland, D., Weske, M.: Automating data exchange in process choreographies. Inf. Syst. **53**, 296–329 (2015)
2. Khalaf, R., Kopp, O., Leymann, F.: Maintaining data dependencies across BPEL process fragments. Int. J. Coop. Inf. Syst. **17**(3), 259–282 (2008)
3. OMG: Object Management Group, Business Process Model and Notation (BPMN) Version 2.0. OMG Standard (2011)
4. Borrego, D., Gasca, R.M., Gómez-López, M.: Automating correctness verification of artifact-centric business process models. Inf. Softw. Technol. **62**, 187–197 (2015)

5. Vennam, S., Dezhgosha, K.: Application development with object relational mapping framework - hibernate. In: Proceedings of the 2009 International Conference on Internet Computing, ICOMP 2009, 13–16 July 2009, Las Vegas, Nevada, USA, pp. 166–169 (2009)
6. Meyer, A., Pufahl, L., Fahland, D., Weske, M.: Modeling and enacting complex data dependencies in business processes. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 171–186. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40176-3_14
7. Weber, I., Hoffmann, J., Mendling, J.: Semantic business process validation. In: 3rd International Workshop on Semantic Business Process Management at ESWC 2008, SBPM 2008, June 2008
8. Gómez-López, M.T., Gasca, R.M., Pérez-Álvarez, J.M.: Decision-making support for the correctness of input data at runtime in business processes. Int. J. Coop. Inf. Syst. **23**(4), 1–29 (2014)
9. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Inf. Softw. Technol. **50**(12), 1281–1294 (2008)
10. Borrego, D., Eshuis, R., López, M.T.G., Gasca, R.M.: Diagnosing correctness of semantic workflow models. Data Knowl. Eng. **87**, 167–184 (2013)
11. Kheldoun, A., Barkaoui, K., Ioualalen, M.: Specification and verification of complex business processes - A high-level petri net-based approach. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 55–71. Springer, Cham (2015). doi:10.1007/978-3-319-23063-4_4
12. Rademakers, T.: Activiti Documentation (2015)
13. Decker, G., Weske, M.: Interaction-centric modeling of process choreographies. Inf. Syst. **36**(2), 292–312 (2011)
14. van der Aalst, W.M.P., Lohmann, N., Massuthe, P., Stahl, C., Wolf, K.: Multiparty contracts: Agreeing and implementing interorganizational processes. Comput. J. **53**(1), 90–106 (2010)
15. Knuplesch, D., Pryss, R., Reichert, M.: Data-aware interaction in distributed and collaborative workflows: Modeling, semantics, correctness. In: 2012 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2012, Pittsburgh, PA, USA, October 14–17, pp. 223–232 (2012)
16. Herzberg, N., Meyer, A., Weske, M.: Improving business process intelligence by observing object state transitions. Data Knowl. Eng. **98**, 144–164 (2015)
17. Senderovich, A., Rogge-Solti, A., Gal, A., Mendling, J., Mandelbaum, A., Kadish, S., Bunnell, C.A.: Data-driven performance analysis of scheduled processes. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 35–52. Springer, Cham (2015). doi:10.1007/978-3-319-23063-4_3
18. Gómez-López, M.T., Borrego, D., Gasca, R.M.: Data state description for the migration to activity-centric business process model maintaining legacy databases. In: Abramowicz, W., Kokkinaki, A. (eds.) BIS 2014. LNBIP, vol. 176, pp. 86–97. Springer, Cham (2014). doi:10.1007/978-3-319-06695-0_8
19. Cornax, M.C., Dupuy-Chessa, S., Rieu, D., Mandran, N.: Evaluating the appropriateness of the BPMN 2.0 standard for modeling service choreographies: Using an extended quality framework. Softw. Syst. Model. **15**(1), 219–255 (2016)
20. Decker, G., Kopp, O., Leymann, F., Pfitzner, K., Weske, M.: Modeling service choreographies using BPMN and BPEL4Chor. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 79–93. Springer, Heidelberg (2008). doi:10.1007/978-3-540-69534-9_6

21. Barros, A., Dumas, M., Hofstede, A.H.M.: Service interaction patterns. In: Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 302–318. Springer, Heidelberg (2005). doi:10.1007/11538394_20
22. Barros, A., Hettel, T., Flender, C.: Process choreography modeling. In: vom Brocke, J., Rosemann, M. (eds.) Handbook on Business Process Management. International Handbooks on Information Systems, pp. 257–277. Springer, Heidelberg (2010)
23. Khalaf, R., Leymann, F.: E role-based decomposition of business processes using BPEL. In: 2006 IEEE International Conference on Web Services (ICWS 2006), 18–22 September 2006, Chicago, Illinois, USA, pp. 770–780 (2006)