# Towards The Essential Flow Model

Oliver Kopp, Frank Leymann, Tobias Unger, and Sebastian Wagner*

Institute of Architecture of Application Systems, University of Stuttgart, Germany
lastname@iaas.uni-stuttgart.de

**Abstract** Many of today's manufacturing projects are so complex that they cannot be conducted only by one company anymore. Current approaches for modeling inter-enterprise processes require an early decision on the way activities are connected. The modeler has to decide between control flow and message flow. This implies an early decision on the used IT-technology. We present a modeling approach where this decision is postponed to a later modeling phase. This enables modelers to concentrate on the essentials of the model.

## 1 Introduction

Many of today's manufacturing projects are so complex that they cannot be conducted only by one company anymore. These collaborations are mostly modeled and executed using business processes. State of the art in modeling collaborations is to model (1) a centralized process model, where the involved partners are represented by swimlanes and connected by sequence flows or to model (2) a choreography, where each partner is represented in a separate pool and connected via message flows. Thus, a modeler has to decide early on how the connection between two activities is established. This has also implications on the used IT infrastructure: In case a single process is used, a single workflow engine coordinates the activities. In case multiple processes are used, a workflow is executed at each participant in the choreography. These workflows have to exchange the agreed messages in order to coordinate. We argue that the decision whether to use a single workflow engine or multiple workflow engines should be taken after capturing the business process. We call such a model "Essential Flow Model".

The idea of an Essential Flow Model is illustrated in Fig. 1. First, an Essential Flow Model is modeled. In principle, this model may be implemented by one of the following infrastructure types: (1) As orchestration, where the workflow is executed by a single workflow engine and the activities are implemented by services or a human task manager [17], (2) as choreography, where a group of activities (typically one lane) becomes a participant in the choreography and thus is executed within a separate workflows engines, (3) using a distributed workflow engine, where the workflow engine is distributed across the different participants. A detailed explanation of each option is provided in the remainder of the paper.

We use BPMN2.0 [20] as illustration for our concept. The concept, however, is independent of the language used. For instance, it is possible to model an Essential Flow Model using PM-Graphs [17] and use BPEL as implementation language [5].

* The authors contributed equally to this paper.

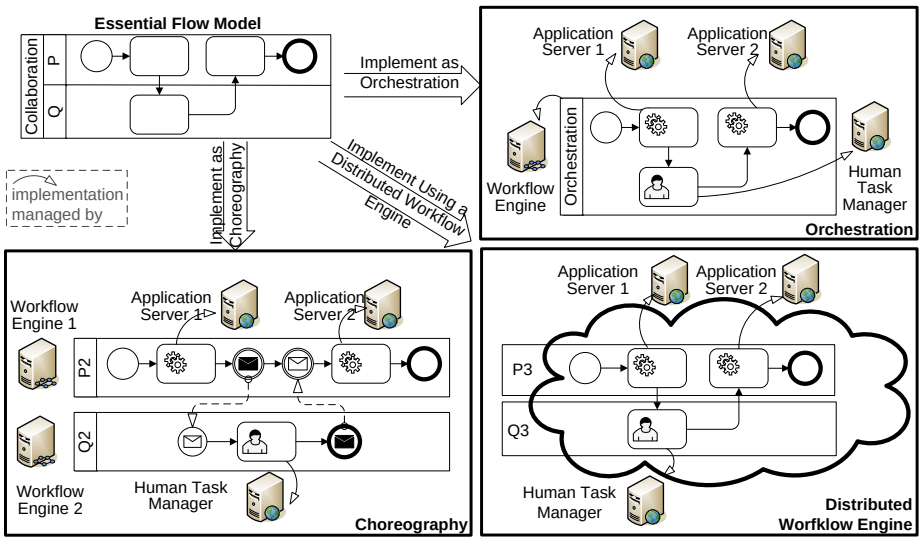


**Figure 1.** Idea of the Essential Flow Modeling Approach

The remainder of the paper starts with an overview on the Essential Flow Model in Sect. 2. There, an example model is provided. The different implementation possibilities are illustrated using this example in Sect. 3. Related work is presented in Sect. 4. Finally, we conclude and present an outlook on future work in Sect. 5.

## 2 Designing an Essential Flow Model

An Essential Flow Model (EFM) captures the essential flows in a business process. In this paper, an EFM is a restricted BPMN process. For modeling control flow, we allow tasks, sequence flow, data-based gateways, one none start event, and one none end event only. The process may only be started by one none start event. We allow one pool only, which may contain multiple lanes. Each lane in the pool is interpreted as distinct entity having the *responsibility* for the activities contained in its lane. In future work, we intend to extend EFMs with other constructs, such as IT system assignments or data access rules. These extensions are out of scope of this paper.

An EFM is designed by humans to agree on a collaboration. In this paper, we assume that such a model has been created by business experts. In other scenarios, an EFM might be created out of existing interacting processes. For instance, a BPMN collaboration could be transformed to a BPMN process and thus forming an EFM. Such a merging procedure is part of our future work.

Figure 2 depicts an example scenario of an EFM. In this scenario a train manufacturer wants to develop a new rail car prototype. In the first activity the requirements of the rail car prototype are determined (e. g., the number of seats and toilets in the car). Based on these requirements the chassis of the rail car has to be designed by an engineer of the train manufacturer. After the design is completed, the wheels and the axes for the

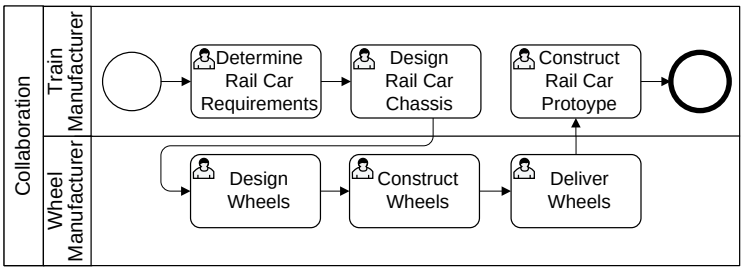


**Figure 2.** Example Scenario Modeled as Essential Flow Model

rail car are ordered from another organizational unit. This wheel manufacturing unit has to design the wheels in a way such that they fit into the chassis design. Thus, they have to use the chassis design as foundation. After the wheels are designed, prototypes are constructed and delivered to the train manufacturer, which is then able to construct a prototype of the rail car.

In the scenario, each activity has been configured to be a user task. It is also possible to configure each task as other type or postpone this decision to the implementation phase.

## 3 EFM Implementation Approaches

After the EFM was modeled, it has to be transformed to an executable process model. The first step is to create disjoint sets of the activities. Each set defines the activities of one workflow. In the following, we use swimlanes to define these sets. In case all activities should be executed in one workflow, one has to decide on using a standard workflow engine or a distributed workflow engine. In case the activities are distributed in more than one set, the choreography approach has to be taken. In each approach, the generated process models are abstract process models. That means, the process models are not executable by themselves, but have to be enriched with information needed for execution. This includes typing of variables, adding tasks for data transformation, and binding to concrete services.

**Implementation as Orchestration** The EFM is implemented on a single workflow engine, i.ė., the activities of each participant are executed on the same engine. The activity implementations of human activities can be performed by one or several human task managers (this depends on the binding information). Tasks which have not been typed, have to be typed. That means, one has to decide whether to use a user task, a service task, or another specific task type for implementing a task. In case a task has already assigned a task type, this assignment may not be changed. Figure 3 presents the orchestration model for our scenario. The final orchestration is executed on a workflow engine of the Train Manufacturer. The human tasks of the Train Manufacturer are executed by its human task manager and the human tasks of the Wheel Manufacturer are

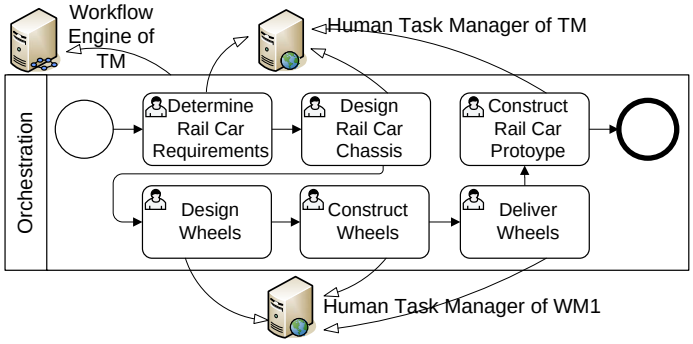


**Figure 3.** Train Prototype Scenario Implemented as Orchestration

executed by its own task manager, which is coordinated by the workflow engine of the Train Manufacturer.

**Implementation as Choreography** This approach splits the implementation of the EFM into one process model per participant. A workflow engine is assigned to one or more swimlanes. In our case, the Wheel Manufacturer activities are executed on a different workflow engine than the Train Manufacturer activities. Each EFM control flow dependency between activities assigned to different workflows is replaced by an intermediate message throw event, a message link, and a message catch event. The intermediate message throw event is connected to the source of the sequence flow to be replaced and the intermediate message catch event is connected to the target of the sequence flow. The first intermediate message catch event has to be replaced with a message start event. The other intermediate message events have to be connected to the preceding intermediate message throw events. The precedence relation is given by the precedence relation of the original EFM. It may be the case that this approach is too straight-forward for workflows where the control flow is split using gateways

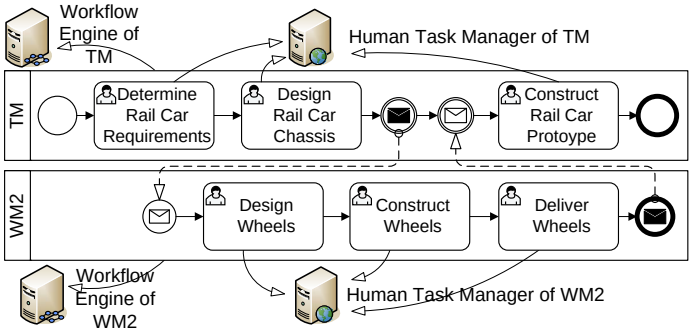


**Figure 4.** Train Prototype Scenario Implemented as Choreography

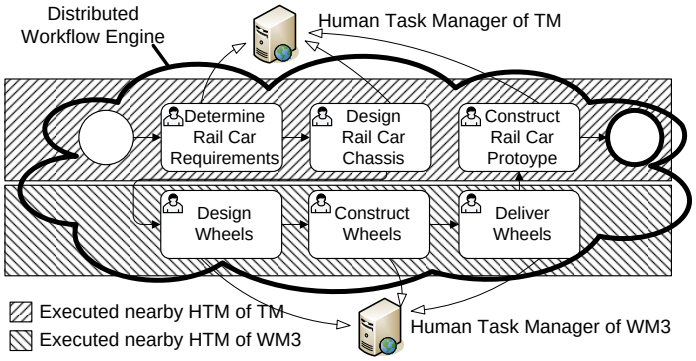


**Figure 5.** Train Prototype Scenario Implemented by a Distributed Workflow Engine

and where multiple intermediate message catch events without a local predecessor are generated. These aspects have been discussed by Khalaf and Leymann [9, 11] in the case of BPEL (see Sect. 4). A discussion of these aspects regarding BPMN should be tackled in future work. Figure 4 presents the final transformation result. The result forms a BPMN collaboration diagram, which is an interconnection choreography model [4]. The difference to the orchestration is that the activities of the Train Manufacturer are executed by its workflow engine and the activities of the Wheel Manufacturer are executed by its own workflow engine. The task manager of the Wheel Manufacturer is coordinated by its own the workflow engine.

**Implementation using a Distributed Workflow Engine** In a distributed workflow engine activities of the same workflow are executed on different nodes (e. g., a physical or virtual machine instance). For example, the distribution of the activities may be defined based on certain technical constraints or based on service level agreements. A technical constraint may be the overall decrease of the amount of data exchanged remotely between two activities by letting data-intensive activities on the same node. In our example the activities “Design Wheels” and “Design Train Chassis” might run on the same node as they exchange data as illustrated by Fig. 5. This means, that the EFM implementation may not be split by swimlanes (although this is still possible in this approach) but also by other criteria. Consequently, an appropriate fragmentation that meets those criteria has to be defined, this can be either done by a workflow designer or the fragmentation is automatically derived by the distributed workflow engine (e. g. based on the workload of their nodes). Based on the fragmentation the activities are distributed. Wutke et al. [18, 24] describe concrete runtime behavior and further concepts of distributed workflow engines.

## 4 Related Work

Khalaf et al. [9–12] discuss issues when doing a role-based decomposition of a single BPEL process into multiple BPEL processes. The main difficulties are (1) splitting

the control flow such that the split processes resemble the same order of activities, (2) correlating exchanged messages to the right the instances of each partner process, (3) keeping data consistent across partners, and (4) coordinating split scopes and split loops. The distribution of control flow among the split processes is solved by propagating the status of each link using messages [11]. Khalaf solves the correlation problem by using a globally unique correlation set for each process instance [9]. In case tasks read from and write to the same variable in the unsplit model and are placed in different partners in the split model, the data has to be kept consistent. A solution is to separate data flow from control flow [10]. In case of split scopes, the fault handling has to be coordinated: A fault occuring on one partner has to lead to a proper handling in the respective parts of the other part [12]. A transfer of these concepts to BPMN is part of our future work.

Koschmider et al. [16] discuss perspective-compliant business process design. One perspective is the view-based perspective. The EFM model may be regarded as one view on the collaboration, whereas the choreography model may be regarded as another view. When going from abstract process models to executable process models, the modeling perspective changes (analysis vs. execution). Koschmider et al. propose a tool supporting different perspectives by using process fragments. In our work, we do not focus on different views, but argue that the decision on the concrete connection between activities should be delayed to the implementation phase. The concrete mapping between the different views is left as future work.

Werth [23] presents a method, a metamodel, and a notation for "collaborative business processes". In his method, the designer has to distinguish between material flow, information flow, energy flow, and control flow between organizations. Werth leaves the implementation as future work. In our work, control flow is the only connection between tasks and we sketch a way from the model to an implementation.

Van der Aalst et al. [1] propose a method to describe a business process spanning multiple partners. They use open worfklow nets (oWFNs [19]) to model a "process-oriented contract which can be seen as the composition of the public views of all participating parties." [19]. The decision whether a place is an interface place is an integral part of the method. In our approach, this decision is deferred to the implementation phase. Decker et al. [5] describe a method for going from a choreography to executable processes. The starting point is a choreography, where the realization choice has been made in the starting model. The same applies for the methods proposed by Barros et al. [2], Dijkman and Dumas [6], Greiner et al. [8], and Werth [23].

Barros et al. [3] reason about possible interaction types between participants. In the current version of the EFMs, request-for-bid scenarios cannot be directly expressed. Thus, our current research is to investigate whether and how those scenarios have to be supported by EFMs.

Kiepuszewski et al. [13] reason about fundamental control flow structures in workflows. In contrast, our paper focused on the overall fundamental ingredients of a model where a workflow (possibly involving multiple parties) is modeled. Patig and Casanova-Brito [21] conducted a survey on general requirements of process modeling languages. Users mostly started modeling by capturing the interactions between departments. The survey did not distinguish between control flow and data flow.

Zimmermann et al. [25] present a concrete reusable architectural decisions framework for enterprise application development. Our decision on whether to use one process engine or a set of engines falls in the class RADM-C, where conceptional decisions have to be tackled.

We assumed that the modeling starts by defining an EFM model. Another start might be existing processes, which are merged into an EFM. Regarding merging, Steinmetz [22] shows how an interconnection choreography model can be generated out of multiple interacting processes. Kopp et al. [15] show how an interconnection choreography model can be converted to an interaction choreography model. This approach might also be used to merge multiple pools into a single pool, which is out of scope of this paper.

When generating process models, which need to be modified by a process modeler, the consistent and clear layout becomes important [14]. In this paper, we do not investigate on appropriate process visualization techniques and process layouting techniques [7], but leave their application as future work.

## 5 Conclusion and Outlook

We have presented the idea of an "Essential Flow Model". This model captures the most basic flows between activities required for communicating the model between the involved modelers and with the persons responsible for the implementation of the model. We have presented three different approaches to implement an EFM. Each mapping from an EFM to a skeleton for each implementation approach has been explained using an example. Thus, our next step is to provide a formal presentation of the transformation.

The presented EFM is very basic. In industrial settings, the participants have to agree to use certain IT systems or agree on data access rules. We are going to integrate these issues in a refined version of the EFM. The result is a concrete description of the ingredients of an EFM. In addition, we assumed that an EFM is created from scratch by humans. An EFM might be created out of an existing BPMN collaboration. This conversion is part of our future work.

**Acknowledgments** This work is partially funded by the projects ALLOW (`http://www.allow-project.eu/`), COMPAS (`http://www.compas-ict.eu`), and MASTER (`http://www.master-fp7.eu/`). They all are part of the EU 7th Framework Programme (contract no. FP7-213339, FP7-215175, and FP7-216917).

## References

1. Aalst, W.M.P.v.d., Lohmann, N., Massuthe, P., Stahl, C., Wolf, K.: Multiparty Contracts: Agreeing and Implementing Interorganizational Processes. Comput. J. 53(1), 90–106 (2008)
2. Barros, A., Decker, G., Dumas, M.: Multi-staged and Multi-viewpoint Service Choreography Modelling. In: SEMSOA (2007)
3. Barros, A., Dumas, M., ter Hofstede, A.: Service Interaction Patterns. In: BPM. Springer (2005)
4. Decker, G., Kopp, O., Barros, A.: An Introduction to Service Choreographies. Information Technology 50(2), 122–127 (Feb 2008)
5. Decker, G., Kopp, O., Leymann, F., Weske, M.: Interacting services: From specification to execution. Data & Knowledge Engineering 68(10), 946–972 (Apr 2009)

6. Dijkman, R., Dumas, M.: Service-oriented Design: A Multi-viewpoint Approach. International Journal of Cooperative Information Systems 13(4), 337–368 (2004)
7. Effinger, P., Jogsch, N., Seiz, S.: On a study of layout aesthetics for business process models using BPMN. In: Second International Workshop on Business Process Modeling Notation. Springer (2010)
8. Greiner, U., Lippe, S., Kahl, T., Ziemann, J., Jkel, F.W.: Designing and Implementing Cross-Organizational Business Processes - Description and Application of a Modelling Framework, pp. 137–147. Springer (2007)
9. Khalaf, R.: Supporting business process fragmentation while maintaining operational semantics: a BPEL perspective. Doctoral thesis, University of Stuttgart, Faculty of Computer Science, Electrical Engineering, and Information Technology, Germany (2008)
10. Khalaf, R., Kopp, O., Leymann, F.: Maintaining Data Dependencies Across BPEL Process Fragments. International Journal of Cooperative Information Systems (IJCIS) 17(3), 259–282 (September 2008)
11. Khalaf, R., Leymann, F.: Role-based Decomposition of Business Processes using BPEL. In: ICWS 2006. IEEE Computer Society (2006)
12. Khalaf, R., Leymann, F.: Coordination for Fragmented Loops and Scopes in a Distributed Business Process. In: BPM. Springer (2010)
13. Kiepuszewski, B., ter Hofstede, A., van der Aalst, W.: Fundamentals of control flow in workflows. Acta Informatica 39(3), 143–209 (2003)
14. Kitzmann, I., Konig, C., Lubke, D., Singer, L.: A Simple Algorithm for Automatic Layout of BPMN Processes. E-Commerce Technology 0, 391–398 (2009)
15. Kopp, O., Leymann, F., Wu, F.: Mapping interconnection choreography models to interaction choreography models. In: ZEUS (2010)
16. Koschmider, A., Habryn, F., Gottschalk, F.: Real Support for Perspective-Compliant Business Process Design. In: Aalst, W., Mylopoulos, J., Sadeh, N.M., Shaw, M.J., Szyperski, C., Ardagna, D., Mecella, M., Yang, J. (eds.) Business Process Management Workshops. Springer (2009)
17. Leymann, F., Roller, D.: Production Workflow – Concepts and Techniques. Prentice Hall PTR (2000)
18. Martin, D., Wutke, D., Leymann, F.: A Novel Approach to Decentralized Workflow Enactment. In: EDOC. IEEE Computer Society (2008)
19. Massuthe, P., Reisig, W., Schmidt, K.: An Operating Guideline Approach to the SOA. Annals of Mathematics, Computing & Teleinformatics 1(3), 35–43 (2005)
20. Object Management Group (OMG): Business Process Model and Notation (BPMN) Version 2.0 (2011), OMG Document Number: formal/2011-01-03
21. Patig, S., Casanova-Brito, V.: Requirements of Process Modeling Languages – Results from an Empirical Investigation. In: Wirtschaftsinformatik (2011)
22. Steinmetz, T.: Generierung einer BPEL4Chor-Beschreibung aus BPEL-Prozessen. Student Thesis: University of Stuttgart, Institute of Architecture of Application Systems (2007)
23. Werth, D.: Modellierung unternehmensübergreifender Geschäftsprozesse – Modelle, Notation und Vorgehen fr Geschäftsprozesse. Salzwasser Verlag (2007)
24. Wutke, D., Martin, D., Leymann, F.: Tuplespace-based Infrastructure for Decentralized Enactment of BPEL Processes. In: Wirtschaftsinformatik. OCG, Vienna, Austria (2009)
25. Zimmermann, O., Zdun, U., Gschwind, T., Leymann, F.: Combining Pattern Languages and Reusable Architectural Decision Models into a Comprehensive and Comprehensible Design Method. In: WICSA (2008)