TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Faculty of Computer Science, Institute of Systems Architecture, Chair of Computer Networks

Diploma Thesis

ADAPTIVE MODELING OF INTER-ORGANIZATIONAL BUSINESS PROCESSES

Submitted by:

Silvio Tschapke

born 21. May 1984 in Lutherstadt Wittenberg
Matr.-Nr.: 3139523

Professor

Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

Supervisor (TU-Dresden)

Dr.-Ing. Iris Braun

Supervisor (SAP Research Dresden)

Dr. rer. nat. Michael Stollberg

Submitted on 30. September 2012

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik, Institut für Systemarchitektur, Lehrstuhl Rechnernetze

AUFGABENSTELLUNG FÜR DIE DIPLOMARBEIT

Name, Vorname: Tschapke, Silvio
Studiengang: Informatik
Matr.-Nr.: 3139523

Thema: `Adaptive Modeling of Inter-Organizational Business Processes`

ZIELSTELLUNG

The tight collaboration of different business partners is one big challenge in many domains. This applies for instance to the Transport and Logistics sector (T&L). The delivery of transport goods involves a variety of stakeholders, which all have to work hand in hand to successfully implement the business process. Despite strong inter-organizational collaboration in such scenarios, the application of modern information and communication technologies is rather rare. The primary causes are different realizations of business processes and heterogeneity within IT infrastructures among the involved partners, although the overall organizational structures are often similar and only the concrete process realization differs.

Future Internet technologies, such as Software as a service (SaaS) solutions, cloud-based storage, Internet of Services (IoS), Internet of Things (IoT), may tremendously facilitate the coordination between partners, since they can be used on-demand and therewith, in a very cost-effective manner. As a result, there is no need to install uniform software in all IT infrastructures of the involved business partners. One major impediment to better collaboration tool support can thus be eliminated. However, the provision of suitable solutions also requires a flexible and adaptive modeling approach that is able to support the business process realizations among the involved partners. This requires an appropriate representation and a set of suitable abstract types, which can be instantiated for concrete processes. Although the involved stakeholders are physically distributed, they all require certain information of the overall process at different points in time. To provide a global view and to encapsulate the concrete data from the corresponding stakeholders, the management of the process should be centralized. Furthermore, business processes are usually executed in dynamic environments. It is impossible to predict all events that may occur during the execution of a process. Hence, the chosen approach must be able to integrate changes at runtime in order to reflect changed conditions and plans. Additionally, the application of the desired approach for enterprise scenarios requires the availability of industrial strength execution environments for defined business models.

The aim of the thesis is the flexible and adaptive modeling, representation and execution of inter-organizational business processes in order to enable the application of Future Internet tools for distributed business processes. A suitable modeling concept shall be leveraged and prototypically implemented to demonstrate the practicability of the envisioned approach. Finally, an evaluation based on real-world use cases of the T&L domain will validate the elaborated results with regard to qualitative and quantitative aspects.

SCHWERPUNKTE

- State-of-the-Art analysis and technology selection
- Conceptual solution design
- Prototypical implementation
- Qualitative and quantitative evaluation by means of real-world examples of the T&L domain

Betreuer:	Dr. Iris Braun
Verantwortlicher Hochschullehrer:	Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill
Institut:	Institut für Systemarchitektur

Lehrstuhl:	Rechnernetze
Externer Betreuer:	Dr. Michael Stollberg (SAP Research CEC Dresden)
Beginn am:	01.01.2012
Einzureichen am:	30.06.2012

i. A. Braun

Unterschrift des verantwortlichen Hochschullehrers

Verteiler: 1 x Prüfungsamt, 1 x HSL, 1 x Betreuer, 1 x Student

SOLEMN DECLARATION

With my signature I truthfully and solemnly declare that I have prepared the Diploma Thesis with the topic:

Adaptive Modeling of Inter-Organizational Business Processes

entirely myself and without the help of any third party and that all literal quotations and other authors have completely been accounted for.

Silvio Tschapke

Dresden, 30. September 2012

ABSTRACT

Inter-organizational business processes are becoming increasingly prevalent. Numerous partners collaborate across organizational borders in a highly dynamic environment. Seamless integration and efficient management of data and information is indispensable for such processes characterized by permanent re-planning. In short, the right information has to be available at the right time, in the right place. Although appropriate information management is necessary, the technological support is still limited. As a consequence, the exchange of data and process information between companies often requires considerable manual effort. Research approaches related to business process management are aiming to solve these problems, however existing solutions lack of adequate agility and transparency of the overall process. Therefore, additional management capabilities must be provided by a solution that builds on top of existing technologies and infrastructures.

In this thesis, we developed an adaptive modeling approach based on *Configurable Collaboration Artifacts* as key elements for representing collaborative business processes. The solution approach extends the concept of Business Artifacts by mechanisms of runtime variation. This was motivated by the need to support changing business scenarios without depending on a predefined set of Business Artifacts. The starting point of the conception was the decomposition of an artifact system into a collection of artifact templates, and into a set of collaboration features, which can be dynamically assigned to an artifact. Both types have declared extension and variation points, which allowed for a distinction between fixed and variable parts of Collaboration Artifacts, to finally enable adaption and customization of business processes.

A prototypical implementation introduced a technical architecture for execution, and demonstrated technical feasibility of the developed concepts. An evaluation verified the central concerns of transparency, flexibility, and scalability, which were considered as most relevant in order to solve the problems regarding collaborative business process management.

The developed solution approach allows efficient management of constantly changing business processes, as well as provides real-time notification support for process deviations. The central scientific contribution is in extending the artifact-modeling approach, by introducing concepts for process configuration. This provides more flexibility by varying and extending single artifacts. Furthermore, such modularization keeps the design slim and the set of artifact types manageable. This solution approach can be used as an adaptive design-, and execution model for business collaboration management systems.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ACSI	Artifact-Centric Service Interoperation.
BPEL	Business Process Execution Language.
BPMN	Business Process Modeling Notation.
CA	Collaboration Artifact.
CHS	Case Handling System.
ECA	Event Condition Action.
EDI	Electronic Data Interchange.
GSM	Guard-Stage-Milestone.
OASIS	Organization for the Advancement of Structured Information Standards.
OMG	Object Management Group.
RDF	Resource Description Framework.
SOA	Service-Oriented Architecture.
TAM	Technical Architecture Modeling.
TCP	Transport Chain Plan.
TEP	Transport Execution Plan.
W3C	World Wide Web Consortium.
WfMC	Workflow Management Coalition.
WS-CDL	Web Services Choreography Language.
XML	Extensible Markup Language.

1 INTRODUCTION

This chapter gives a comprehensive introduction into the topic of adaptive modeling of inter-organizational business processes. The first section describes central characteristics of collaborative business processes to provide the context and background of research. A motivating, real-world example of a transport process thereby is used for illustration. In the second section, we discuss general problems in this context and identify general solution requirements. The third section outlines relevant research areas and developments corresponding to the topic. Subsequent to this, we define the overall problem statement, and formulate the central research question and its corresponding working hypothesis. Finally, the structure of the thesis is outlined.

1.1 RESEARCH CONTEXT AND BACKGROUND

This section provides a context overview of this work to identify main characteristics of inter-organizational business processes. This is done by means of a real-world example of a transport and logistics process. The domain of transport and logistics is characterized by highly distributed processes. Hence, it is taken as a demonstrative example throughout the thesis. In the following, we describe a concrete transport process scenario to illustrate characteristics of collaborative processes.

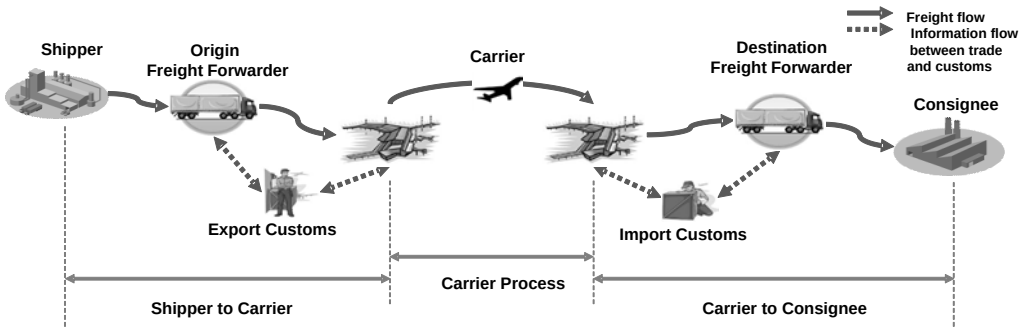

Figure 1.1: Air Freight Transport [Ria12]

Inter-organizational business processes are characterized by the involvement of numerous, collaborating stakeholders. In such a process, each stakeholder has an essential contribution to make with respect to its role. Collaboration thereby is based on information exchange, while activities are planned, re-planned, and executed. The execution of activities however depends on the access to information, and on secure exchange of data in the business network. Events are processed, and notifications are delegated to involved stakeholders. In addition to that, the overall process flow is often affected by deviations. Those deviations are common in complex scenarios and require re-planning of the running process. In the following, we illustrate such characteristics by means of a transport business process.

The stakeholders, which are involved into logistics processes collaborate with each other by moving and handling goods and materials with the goal to transport these goods to a designated destination. Two basic roles of stakeholders are the transport service provider on the one hand, and at the other hand a transport service client. Transport providers, such as a trucker or carrier are responsible for the physical transport. A freight forwarder has also the role of transport service provider, however he/she is responsible for transport coordination. In contrast to that, a transport client requests a transport service. Transport clients can be customers, shippers, or transport coordinators who request a logistics service. Customs control the import and export of cargo to and from a country, and governmental authorities document the transport for statistics. The single parts of a transportation process are denoted as transportation legs.
The air freight transport process, which is shown in Figure 1.1, consists of three transportation legs. The goods thereby must be transported from a shipper to a consignee, using road- and air transport means. The transport is initiated by the consignee, who in this case is the customer. He/she requests a supplier in Kiev to send machinery parts to Amsterdam. A freight forwarder, who is acting on behalf of the customer, plans and coordinates the first part of transport from the supplier to the airport in Kiev. The forwarder therefore requires a shipment order including time periods, good details, and destination. After a verification and confirmation step, an invoice for the customer and for sub-contracted service providers is created. One of those contracted service providers is the trucker company, which is responsible for the road transport. The freight forwarder finally sends pick-up information to the trucker, while the shipper prepares a load list with all relevant data for transporting the goods. As soon as the customs documents are forwarded and are finally approved, the shipment can start, if all goods are ready for pick-up. Once the delivery is completed, the freight forwarder contacts a carrier to send information about the goods and the estimated time of arrival. Once the carrier agrees the transport of goods, the trucker starts its route to the airport, where the hand-over of goods and documents is established.
The subsequent air-, as well as the final road transport are realized in a similar way. The carrier requests an air way bill by the forwarder, security checks and container registrations are performed, and the destination airport is informed about the estimated time of arrival after take-off. By continuing the transport in Amsterdam, customs checks are required before the goods can be finally transported to the consignee and the invoice payment is finished.

With respect to the main characteristics, this example illustrates the importance of data and information exchange of inter-organizational processes. The availability of invoice-, load list-, or customs documents, determines whether the transport process continues or not. The knowledge about where the goods are, has influence on decision making and thus improves the process. The air carrier for instance is very interested in the exact time of arrival and in details about the transported goods in order to plan the internal process. In addition to that, deviations in each part of the process may occur. Freight might not be accepted by customs for export, delays in production might happen, or traffic jam requires to change the initial plan. Further, the customer in Amsterdam might change the need for capacity, or even cancels the transport order on short notice.

This section illustrates the main characteristics of inter-organizational business processes by means of a transport process scenario. These processes are based on information exchange, notifications, and deviations. Thus, it is a complex collaborative business process. All that must be further achieved in a highly dynamic environment, characterized by ad-hoc changing collaboration partners or schedules. From these properties we outline problems and challenges in the next chapter.

1.2 PROBLEMS, REQUIREMENTS, CHALLENGES

In this section, we identify problems, which result from the outlined characteristics of inter-organizational business processes. For each set of related problems we formulate requirements for a possible solution.

The importance of data and knowledge in order to realize specific tasks, together with the likelihood of deviations are general characteristics of inter-organizational business processes. This states a problem, since information still is exchanged via e-mail or telephone, whereas automated, standardized data exchange is used by big companies only. As a consequence, data is transferred along the chain in different formats using different data exchange technologies. Missing consistency checks reduce the quality of data even more. These media breaks are not only inefficient, but error-prone. There is no global accepted standard for data exchange in inter-organizational business, which is applicable for all involved stakeholders. Besides the lack of information quality, the main problem is to effectively cooperate with each other in the first place. Each service provider needs information of business partners in order to plan, and to coordinate its process. Due to missing IT capabilities not all partners are able to provide the required data in real-time. Although the big players in the field have appropriate system technologies and infrastructures for collaboration, this does not apply to small-sized, or ad-hoc contracted companies. The consequence of such insufficient information technology support is a time consuming, manual data exchange via telephone, e-mail or hard copy.

In order to provide an on-demand integration platform for business networks, a solution that builds on top of existing information technology is required to facilitate inter-organizational collaboration. To benefit from collaboration in terms of increasing quality, speed, and business agility, an efficient management and coordination for both, data and processes, is required. This includes data integration, event processing, re-planning, and real-time notification to pro-actively control the process. The realization of such management solution requires the following.

First, it needs more *transparency*. Usually, each peer knows its direct business partners in a graph-based collaboration network. Information however must be accessible across different layers of sub-contracted clients. Stakeholders are not able to get any information without requesting intermediates, because they are not directly involved into a certain part of the process. Such information is required for planning and effective decision making. A business partner at the end of the transport chain could tremendously accelerate the process, if all information about the goods and delays, happening at the beginning of the chain, would be directly accessible. This problem is described as a lack of an end-to-end view on the process.
The challenge is to close these information gaps. A *global view* on the whole process would increase predictability and improve decision making. The requirement is to provide a common information base for the multitude of actors involved. This is a central requirement for efficient process management across company borders, however it requires an appropriate information representation and data management. Corresponding information includes general data, the pro-

cess status, conditions, and critical events. This must be enriched by customization support and secured by access restrictions.

Second, the process model must be *adaptable*. The business environment is highly dynamic and unpredictable. Cooperations between partners do change, traffic or delivery problems cause delays in transport, and all kinds of deviations might require the re-planning of the process. Single stakeholders need to adapt and easily customize a process to their internal needs. Automated process management however is often inflexible, which leads to closed supply chains, and hinders agile inter-organizational information exchange. The challenge hereby is to provide more agile business collaboration. In order to face this challenge, a basic requirement is the effective management of events and deviations. At first, real-time notification support is required to immediately inform affected stakeholders about unforeseen events. Those notifications are the basic requirement to even start with the processing of a deviation. Deviation handling includes the need for re-planning, which results in process model modifications. This however requires an adaptable process model with reusable parts and declared points of variability.

Third, the execution of the process model must guarantee high *performance*. The increase of data volume for complex scenarios states a problem for the execution of process models. This includes the number of documents, events, and the complexity of transport plans. The performance thereby should not primarily depend on concrete technology stacks or infrastructures. As a consequence, the model must be scalable to ensure performance even for complex scenarios.

Finally, the process model must be *universally applicable* for different scenarios within specific domains. Inter-organizational business processes do not rely on fixed workflow patterns. Although, single sequences might be standardized for a domain, the concrete process varies across different scenarios. As a consequence, a scenario-specific application, which covers transparency and flexibility for a certain use case only, is not appropriate. The challenge is to support transparency and flexibility by design, instead by means of a concrete implementation. A generic, and abstract modeling approach, which is capable to represent various scenarios within a domain, therefore is a requirement.

Summarizing, the problems for inter-organizational business process modeling are manifold. This leads to various challenges. In short: *the right information has to be available at the right time, in the right place*. We envision a generic modeling approach which provides transparency and flexibility by design. Besides that, it must be able to manage even complex scenarios within a highly dynamic environment.

1.3 RELATED RESEARCH AND DEVELOPMENT FIELDS

This section gives a short overview of relevant research areas. Regarding to the identified problems in the previous section, research areas are of interest, which allow to satisfy the requirements. This includes especially developments related to adaptive business process modeling.

Efficient inter-organizational collaboration hugely depends on the ability of electronic data exchange between heterogeneous systems. Hence, relevant research for collaboration solutions would include communication technologies, cloud-based computing, security and privacy mechanisms, or service-based integration systems. Solutions and technologies to bridge organizational borders do already exist. In order to support efficient business process management, technology

however is not the main problem. Business process collaboration and management is expressed at a more conceptual level. Therefore, relevant research areas are related to business process modeling for representing data, activities, stakeholders, and all other types of relevant information. The following research areas are directly related to this work:

- *Business process modeling.* Various methodologies, standards and tools for business process modeling already exist. Key organizations for standardization in this field are the World Wide Web Consortium (W3C)[1], Organization for the Advancement of Structured Information Standards (OASIS)[2], and the Object Management Group (OMG)[3]. These organizations contribute in standardization of languages, notations, or architectural models. We are interested in developments for the support of human-driven, data extensive business process modeling. Furthermore, modeling approaches are classified regarding on what elements are considered as most important for representing the domain in an intuitive way. These are mainly process-centric, data-centric, and artifact-centric modeling approaches. Besides investigating a certain modeling approach, we are further interested in how appropriate models can be efficiently executed.
- *Flexible, inter-operable, and scalable systems.* These properties are addressed (amongst others) by Service-Oriented Architecture (SOA), which allows the cooperation of distributed companies via services. We are not primarily interested in architectures, but in the related modeling concepts and coordination strategies. Besides SOA, there exist other fundamental software concepts for loose coupling of functionality to support flexibility and scalability.
- *Adaptive modeling.* This research field is of interest with respect to extend and modify process models at design- and at runtime. General concepts for modularization, variation, and verification are related to this field.

Summarizing, we are not interested in concrete technologies or architectures to realize a suitable solution. All relevant research areas are focused on general techniques or strategies for adaptive, and scalable business process modeling.

1.4 PROBLEM STATEMENT

In this section, we emphasize the basic problem and the scope, which is finally addressed by this thesis. The problem results from the findings of the preceding sections.

The basic motivation of this work is to reduce manual effort for ad-hoc inter-organizational collaboration, and to increase business agility and efficiency by eliminating media breaks among involved stakeholders. To achieve this goal, concrete technology matters, however we envision a solution which builds on top of existing infrastructures to allow effective data and process management. The addressed problem in this thesis, is to develop a modeling approach, which satisfies the following:

- It must be able to hold, and to represent all relevant information. This includes information about process activities, related data, involved stakeholders, events, and constraints.

[1] http://www.w3.org/
[2] http://www.oasis-open.org/
[3] http://www.omg.org/

- It must be capable of providing a global view on all of those information and events, a certain stakeholder might be interested in. This includes data- and status information for monitoring, consistency checks, efficient planning and issue handling. This data view, must be restricted by access control mechanisms, and it must allow for customizing views.
- The solution must support real-time notification for critical events.
- The business process model must be adaptable, extendable, and reusable. This is required to handle process deviations and re-planning at execution time.
- The model must be intuitive. An intuitive representation of the business process allows business people as well as implementation teams to analyze, manage, and to control business operations, and to simplify stakeholder communication [CH09].
- The modeling approach must be scalable. This is required to guarantee performance, even for complex business processes.
- It must be possible to validate the solution approach by applying real-world scenarios of transport and logistics. Therefore, the instantiated model must be generic enough to allow the representation and execution of different scenarios.

This section provides a vision of how the solution has to be. In the following, we formulate the research question, which is supposed to be answered in this thesis.

1.5 RESEARCH QUESTION AND HYPOTHESIS

The previous section describes the general background, related problems, and relevant research areas. This section formulates the central research question and the corresponding hypothesis. The overall research question targeted by the thesis is:

> *How can the modeling of inter-organizational business processes be improved regarding flexibility and scalability, and how can a global process view for all involved stakeholders be provided?*

The working hypothesis consists of several parts and is constructed as a statement of expectations. It constitutes the starting point for this thesis and reflects the main steps of investigation:

H1 Inefficiency of collaborative business processes is mainly caused by inadequate underlying modeling approaches.

H2 Existing modeling approaches lack in at least one of the following requirements: flexibility, visibility, or scalability.

H3 A modular design allows for flexible, and adaptable process models.

H4 Encapsulation of data and process information into single domain entities enables a global view on the process.

H5 A prototypical implementation verifies the practicability of the solution approach.

H6 The solution model allows the representation of various concrete transport scenarios.

H7 The modeling approach improves the quality of collaborative business processes and satisfies requirements for performance and scalability.

1.6 OUTLINE

This section describes the structure, and the scientific methodology of the thesis. The remainder of the thesis is as follows:

In Chapter 2, a detailed requirement analysis is conducted. A motivating scenario is used to identify and to illustrate central requirements. These requirements constitute the basis for a subsequent analysis of the state-of-the-art. In Chapter 3, the conceptual solution approach is developed. The conceptual design is mainly influenced by the final assessment in Chapter 2. Technical feasibility is demonstrated by a prototypical implementation of the conceptual architecture (Chapter 4). In Chapter 5, the qualitative benefit of the solution approach is evaluated. Besides that, quantitative aspects such as performance and scalability are evaluated as well. This chapter thus proves if, and to what extend the initial identified requirements are satisfied. Finally, a summary of the results and an outlook on future work are given in Chapter 6.

2 REQUIREMENT ANALYSIS AND RELATED WORK

This chapter illustrates the need for an adaptive modeling approach in order to improve the representation and the management of inter-organizational business processes. First, a detailed requirement analysis is conducted by means of a real-world example. The outlined challenges thus motivate the solution approach. Subsequent to this, the main requirements for the envisioned solution are discussed. In the second section, an analysis of existing research developments is made. As a starting point, a short overview of relevant research areas is provided. The following state-of-the-art analysis then examines each research area in detail, and discusses the latest developments. At the end of this chapter, we summarize the examined research approaches and its assessments. We emphasize the shortcomings of current research and finally come to a conclusion with respect to a conceptual design.

2.1 REQUIREMENT ANALYSIS

The management of inter-organizational business processes is a challenging task. Numerous stakeholders interact with each other across organizations by exchanging information. In a dynamic environment, frequent deviations require real-time notification support and an efficient deviation management. The success of a business process depends on the availability of information. The main challenge is to provide each stakeholder with all the information, which is required for the execution of a specific task. Furthermore, the process management system must deal with process changes due to frequent re-planning. In order to provide an on-demand platform with information management and notification support, a solution that builds on top of existing technology and infrastructure is required. The aim of this section is to identify the challenges and requirements for such a solution. First, we describe a real-world use-case scenario for air-freight transport by considering the normal process flow, as well as exceptional behavior. Based on this scenario, we identify the main requirements by analyzing characteristics that can be derived from this scenario.

2.1.1 Motivating Example

In the introduction a typical door-to-door air freight transport scenario is described to introduce the big picture of logistics processes. Now, the first part of this process is examined, while taking unforeseen changes and their impacts during process execution into account. These deviations or exceptions are the basis for identifying requirements of the envisioned modeling approach. As already described, the scenario is about the transport of machinery parts by plane from a shipper in Kiev to a consignee in Amsterdam. The transport from shipper to airport involves the export freight forwarder, export customs, and a trucker. The roles involved into the process at the other end of the chain respectively are import customs, the import freight forwarder, and a trucker. While truckers and carriers are solely transport service providers, the freight forwarders coordinate the process in terms of a global manager. The forwarder is legally responsible for a satisfactory overall performance of all its employed agents involved into the process. Depending on the specific case, forwarders can be responsible for the global process, for specific transport legs only, or a freight forwarder is not required at all. The trucker needs to know where and when to pick-up what kinds of goods. The internal process of the air carrier mainly depends on its predecessor in the transport chain, which is the trucker. In the following the transport from the shipper to the carrier is described, which is the first of three parts of the door-to-door transport scenario. In order simplify the scenario, negotiation interactions, and invoicing processes are excluded. Both are important in logistics, however they are not required for the illustration of the main problems and challenges.

At first, the shipper sends a transport order to the export freight forwarder. This document includes origin and the destination of transport, its timing constraints and good details. The forwarder then books a trucker and transmits the pick-up information. Next, the freight forwarder sends customs documents to the export customs and searches for carriers, which are able to execute the transport. The carrier is interested in the trucker identification, its expected arrival time, and in the transported goods. If the carrier verifies transport capacity and availability, a confirmation is sent back to the freight forwarder. He/she can now instruct the trucker with all required information about the destination, and the trucker can start to pick-up the goods. Meanwhile, the forwarder builds an air way-bill document in coordination with the carrier. This bill covers the transport of cargo from airport to airport and serves as an evidence of the contract of carriage. When the trucker arrives at the terminal, the carrier checks the goods documents and builds its unit load devices. The hand-over thus is finished and the plane is ready for take-off. Such process flow, which exactly reflects what is expected, is referred to as a "sunny day" scenario.

Process variants are derived from "rainy day" scenarios, which include possible errors and exceptions. By taking an exemplary deviation into account, it is illustrated what impact an unforeseen event has on the involved stakeholders. The management of deviations is referred to as exception handling and issue tracking.
Supposing, the order changes on short notice. While the trucker is on its way to the shipper, the customer order changes, and the amount of goods to be transported increases. Since the export freight forwarder is the direct partner of the shipper, he/she is notified about the changed order and must inform its agents about the changed amount of goods. For instance, the truck might be too small for the changed order and an additional or bigger truck is needed. It is the job of the forwarder or of the trucker itself to organize additional resources for transport. The air carrier needs to recalculate aviation fuel and load volume and decides whether he/she can execute the transport or if interim storage capacities are required. Further, transport documents are updated and export customs must be notified. Such an order change also affects transport providers at the end of the logistics chain. The export forwarder requests the import forwarder to re-plan the shipment from the destination airport in Amsterdam to the consignee.

This example shows that changes at the one end of the logistic chain might influence the whole process and requires notification and communication between all the involved partners. Deviations often require subsequent re-planning of the internal processes of the business partners; however in some cases a single event influences the structure of the overall process. If an airport closes because of weather conditions or strikes, the transportation mode changes to road or rail, sub-contracts are made, and the sequence of transportation legs is changed. In international logistics, the challenges become even more complex, which means more stakeholders, more documents, and more possible deviations. Due to the application of different communication media and standards the integration of data is error-prone, which again might lead to new delays. Summarizing, agile business processes are the leading factors for efficient process operation.

2.1.2 Requirement Identification

Based on the previously described example and on its characteristics, in the following, we conduct a requirement analysis. First, each requirement is explained in detail. At the end, we summarize the requirements, which are considered as indispensable for an appropriate solution approach.

Global View

Typically, a business process is divided into different sub-processes with various stakeholders involved. These sub-processes have activities for marketing and sales, for planning, for warehousing, or transport execution. The accessible information on data and activities mainly depends on the stakeholders role. This is described as the need for customized views to provide a stakeholder only the information he/she is interested in, or is allowed to see.
In inter-organizational collaboration sub-processes might be distributed over different countries. Although stakeholders usually have full information about their local activities, there is a lack of information about the relevant process context. Stakeholders thus depend on the information they receive from their environment, for example from direct partners in the process. If this information is incomplete or is received too late, the recipient becomes a bottleneck of the overall process. With respect to the motivating example this is the case for different stakeholders after an unforeseen event. The air carrier is responsible for the transport of machinery parts from airport to airport. He knows its direct partners in the logistics chain and has the focus on its tasks. If the shipper changes its order, he/she at first has to inform its freight forwarder. The freight forwarder then notifies trucker, customs, and carrier about the issue and about possible delays. The fact that the amount of shipped machinery parts has changed however also affects the import customs and the import trucker. This means that the forwarder at the export side informs the forwarder at the import side, which again delegates the information to its peers.
These mechanism are inefficient due to the limited knowledge about the global process. Each stakeholder should have centralized access to all information relevant for the execution of its tasks. Even if the process is in a phase the stakeholder is not directly involved in, it might depend indirectly on the business operations. If a stakeholder not just has information about the steps he/she is actually working on, but about the context, it increases confidence and simplifies planning [CL04]. Sub-processes are transparently integrated and encapsulate all end-to-end process information the involved stakeholders might be interested in. This is called a global process view or end-to-end process visibility and thus is a central requirement for the envisioned solution approach.

Flexibility

Today's enterprises need to keep their processes flexible to cope with increasing change, uncertainty and unpredictability of business. The term flexibility covers a group of requirements, since it is a property which can be realized on different aspects and abstraction levels. [Sch+08] and [Aal+09] introduce a taxonomy for this issue of flexibility. Subsequently, we discuss different facets of flexibility and identify requirements for the modeling of business processes.

Support of process variants. The exemplified air freight transport scenario is just one possible logistic scenario. Railway transportation of natural resources or over-sea transportation of frozen fish might be other case scenarios. The air freight industry requires processes including airports and contracts, whereas oversea logistics providers need to represent ports and revocable bookings of containers. An air carrier might be interested in weight and volume specifications of cargo, whereas a container line is interested in ensuring temperature conditions for frozen fish. Although there are different concepts and terms for different scenarios, there are also commonalities in logistics. For instance, each transport scenario has an origin, goods to transport, and a destination. The envisioned model must be able to represent different variants of logistic scenarios.

Flexibility at design time and runtime. Business processes must be flexible to keep pace with a constantly changing business world. Eicker et al. address this topic from different points of view [ENS10]. From various interpretations of several authors they define flexibility as "the capability to cope with new, different, or changing requirements". Requirements on costs, time and quality as well as environmental circumstances in business do change frequently. This matter is described as business evolution. For instance, an air carrier could need concepts for warehousing or subcontracting, which have not been required before. Business processes thus must be flexible and extensible to adapt to changing needs by modifying parameters or re-ordering of activities. In this context it is important to mention, that flexibility is not equal to adaptability, since a system might be highly flexible due to the fact that it does not need to adapt any of its processes [ENS10].

In contrast to adaption at design time, the "rainy day" use case scenario illustrates the need to efficiently handle uncertainty and unpredictability at runtime. A single unforeseen event for instance can require modifications in the sequence of legs. The global process thereafter must be updated to the new conditions. A business process, which is flexible during execution, is able to react to a new, unforeseen situation. This includes the ability to handle errors and exceptions. Such a system, which can quickly adapt to changes at runtime, is referred to as being agile. Since knowledge produces agility, it is the responsibility of the business process model to provide data and process information for efficient decision making and re-planning. For the modeling of business processes this implies increasing needs for flexibility both at design and at runtime.

Scalability

Besides supporting an end-to-end view and process flexibility, the solution must be scalable. With increasing complexity, more aspects arise which must be included into the business process. Process configuration must be changed and additional roles such as customs or insurances might be added. In addition to that, the information load increases due to the involvement of more and more stakeholders, and the workflow must be scaled up to deliver higher throughput. An appropriate design and technologies for scalability thus are requirements for the envisioned solution.

Summary of identified Requirements

The subsequent list summarizes the requirements, which must be met for a solution, which aims at efficient data and process management of inter-organizational business processes. The solution approach must provide the following:

REQ1.1 **Support a global process view.**
Stakeholders must be provided with all the relevant information for their tasks. They should have also insight into processes they are only indirectly involved in. The information should be logically centralized and available without depending on explicit notification chains of partners.

REQ2.1 **Support process variants**
The model must be generic, but at the same time complete, to serve as a basis for the representation of various logistics scenarios. This generalization increases flexibility in creating models for concrete use cases. Flexibility at design time is supported by generic models, which can be adapted for a specific context.

REQ2.2 **Provide process agility.**
The business process model must be able to react to unforeseen events, exceptions and errors. This means, the system must be flexible to allow replanning at runtime, which affects the underlying process model.

REQ2.3 **Provide process adaption.**
The business process model must be configurable for specific scenarios and circumstances at design time.

REQ3.1 **Ensure scalability.**
Since real world logistic scenarios are complex, the solution must be scalable.

These requirements are all non-functional requirements. Information transparency, flexibility, and scalability are not directly measurable, however they constitute criteria which can be used for a judgment. The next section discusses research areas, which are of interest in order to satisfy the requirements.

2.2 RELEVANT RESEARCH AREAS

This section gives an overview of relevant research areas. These fields state the basis for further examination of candidate approaches in order to realizing the envisioned solution. First of all, we clarify the basic terminology and the scope of the thesis. Subsequently, the most relevant developments are mentioned. It is further argued, why certain investigations are considered for a state-of-the-art analysis, and at the same time, why others seem less important and therefore are excluded for further examination.

The topic of this thesis aims at the modeling and at the runtime management of dynamic, and collaborative business processes. A business process consists of a network of activities, related data, and stakeholder interaction. The field of business process modeling has a broad spectrum of modeling concepts, languages, and technologies. We are mainly interested in general concepts,

which allow flexible business processes management, and efficient knowledge representation. Besides that, a process model must deal with deviations at execution time. Modeling aspects such as flexibility, adaptability, or reuse are addressed in general software design and system architectures. The basic idea is based on the decomposition of an application design into logical, loosely-coupled components with well-defined communication interfaces. Since this paradigm allows flexibility and reuse, research developments are of interest as well. Summarizing, we are interested in the analysis of appropriate techniques to represent control- and data flow, process states, events, and involved stakeholders.

Adaptive-, Process-centric Modeling. Various standards, and tools for business process modeling already exist. Organizations such as the W3C, OASIS, the OMG, or the Workflow Management Coalition (WfMC)[1] make contributions in standardization of languages and architectural models for business process modeling and process management. Most technologies which aim at the modeling of business processes, are based on a procedural and/or graph-based paradigm for specifying how a business process is supposed to operate [NC03]. Conventional business process modeling however is often inflexible, in terms that it requires much effort to consistently change the model. One research area of interest thus are process-centric modeling approaches, that aim at adapting process models and thus provide more flexibility.

Service-oriented Modeling. Aspects like flexibility, interoperability, and scalability, are addressed by SOA. It can be considered as an evolution of centralized process management as it is based on the idea of modeling a business process as a loosely coupled network of services, instead of static predefined tasks. This architectural style provides standardized languages for this purpose; however, we are rather interested in related modeling concepts and coordination strategies.

Data-centric Modeling. Besides process-centric modeling, and the service-oriented modeling paradigms, process models can also be designed from a more data-centric perspective. Research approaches in this direction are of interest, because inter-organizational business process are based on extensive data exchange, and thus are highly data-driven processes. The execution of human tasks for instance, mainly depends on the availability of certain data documents. Therefore, the state-of-the-art analysis includes the latest developments of data-centric modeling.

Artifact-centric Modeling. Following the different perspectives of process modeling, the paradigm of artifact-centric modeling is another way to think of business processes. The idea thereby is to consider both, data and processes, to the same extend. Since this paradigm seems to be a compromise between purely process-centric and purely data-centric approaches, we examine latest research in the next section. The subsequent state-of-the-art analysis covers this spectrum of approaches for business process modeling.

To complete the discussion on relevant research areas, this paragraph summarizes research developments, which are not considered for further examination. As already mentioned, general modeling concepts are of central interest, since the solution approach should not depend on specific language or standards.
Petri Nets provide a formal process modeling language to describe and to implement control flow, including rules for synchronization and branching. The Pi-Calculus is an algebraic system for building processes and states the mathematical foundation for certain process modeling standards. Despite its academic influence on Business Process Modeling, neither Petri Nets nor Pi-Calculus are appropriate languages for business analysts or software developers and therefore are not considered for further examination. Similarly, approaches like state machines or activity diagrams have little direct influence on major standards for Business Process Modeling

[1] http://www.wfmc.org/

[Hav09]. Ontology-based process modeling uses semantic descriptions to model business processes [KO05],[Hep+05]. Such knowledge representation allow for more flexibility by bridging the gap between the business world, and the technical world [FKM09]. However, an ontology is based at a very abstract model level. Furthermore, ontologies are grounded on a hard logical knowledge base and therefore it requires high knowledge of the concrete business domain to be ontologically modeled. Although knowledge and data representation is of central interest, concrete technologies and standards are not considered. For instance, the Extensible Markup Language (XML), Electronic Data Interchange (EDI) standards, or the Resource Description Framework (RDF) support data exchange and integration. The envisioned solution however has the focus on the conceptual design, independent of concrete data representation languages or standards.

2.3 STATE-OF-THE-ART ANALYSIS

This section gives an overview of the state-of-the-art for those research areas considered to be relevant for the envisioned solution. The most important approaches are discussed in detail, and their underlying concepts are set into relation to each other. Each candidate approach is assessed in order to argue whether it the identified requirements or not. In the following, each section discusses a different paradigm for business process modeling. For each paradigm we outline research approaches, which aim at more adaptive process modeling.

2.3.1 Adaptive Process-centric Modeling

All research approaches discussed in this section envision more flexible, process-centric modeling. This is achieved by using modularization principles to encapsulate common things like activities and then select between pre-defined variants in order to construct the process from a standard repertoire rather than reinventing the wheel. No matter of what is encapsulated within a component (a business process or a data structure), the overall goal is to build software units which can be reused in different scenarios. This motivation, because of a paradigm-shift from time and cost optimization towards flexibility of processes and customization [Sch08]. A good theoretical foundation on the concepts for componentization is provided in [LR10]. These concepts are applied for business process modeling. We differentiate in the following between configurative and compositional approaches.

Configurative Mechanisms

The motivation of configurative process modeling is based on the idea that a customer is interested in deploying a subset of available features to support specific needs, without designing the process model from scratch. The concept of configuration relies on declared variable parts in the process, to enable the construction of process variants. Flexibility thereby is achieved by under-specification.

[Dre+05] propose the concept of *Configurable Event-Driven Process Chains*. They use generic patterns of configuration alternatives, in order to understand what situations can occur during business process configuration. These patterns include optionality, multiple choice, exclusive choice, and propose a notation. Configurable node and configuration attributes.

Another project targeting at adaptable workflows is *ADEPT* [Hen+00]. The aim of this project was the development of an adaptive workflow management system. ADEPT offers concepts for the modeling, analysis, and verification of workflow templates and thereby guarantees static and dynamic correctness properties [RRD03]. It is based on process schemas as graph-based representations of workflows providing formal semantics. The resulting process then can be adapted at instance level; this includes adding and deleting activities, or re-ordering their sequences. It also provides adaption at workflow type level as well, by propagating workflow schema changes to running instances. Since 2008 there exist an industrial product version called *AristaFlow BPM Suite* [Dad+10].
Another approach to gain dynamic flexibility in workflows are *Worklets* [Ada+06], [Ada07]. A worklet is a small, self-contained, complete workflow process which handles one specific task in a larger, composite process. Flexibility in this approach is achieved by late binding. Thereby, the concrete workflow implementation is chosen at runtime from a set of pre-defined actions. At built time placeholders are specified, which are linked to concrete implementations during process execution. These worklets build an extensible repertoire of actions, which are dynamically substituted to the placeholders of incomplete tasks.

Compositional Mechanisms

Process modeling approaches, which use compositional mechanisms are based on the idea of composing pre-defined, generic activities to complete processes. This concept offers high flexibility, however it requires additional effort for consistency and verification checking procedures.

[BMR09] propose a component-based modeling approach based on predefined *Generic Activities*. Those activities establish a common terminology and understanding of what a process is. In their paper, the authors propose a procedure for an identification of the activities. Furthermore they introduce the term of a *Processed Object*, which represents organizational units, documents, or events. Both, the standardized generic activities and a set of processed objects are finally combined to complete processes. The applicability of their concept is evaluated in a case study in the health care domain.
[GD11] introduce a set of different *Process-Meta-models* to describe a process at different abstraction levels. This component paradigm helps in developing process components with specified interfaces. The activities are modeled as black-box components, triggered by events. This allows dynamic sequencing of activities and results in reactive control for process models. Further, needed and produced resources are defined which can be either optional or mandatory for an activity. Contracts are used to connect activities and to define control flow between them. This results in the capability of restructuring the control flow at runtime. In their paper they define a general process meta-model, however application specific meta-models are under development.
[Den+04] differentiate between *general- and flexible activities*. Flexible activities thereby encapsulate the uncertain sub-process at design time. These activities are then replaced at runtime with a concrete sub-process composed by general-, or newly added activities. They present a workflow model enabling those activities to be composed automatically or manually at runtime. They designed an algorithm to efficiently compose selected activities into a process. Furthermore, they apply this model to a multi-agent based workflow management system. Finally, they make a case study in the medical sector.

Assessment

First of all, both concepts, provide flexibility for business processes. It is possible to react on deviations at design- and even at runtime. However, there are challenges that must be considered, and there are also limitations.

The correctness of configurable or compositional processes must be enforced by domain and process constraints. The process must be guided by a reference model. However these concepts still suffer from notations and methods to document and understand variations for both process analysts and providers of reference models [RD08]. Second, these process constraints depend on the domain, and they did not focus on Transport and Logistics. Third, the level of flexibility is restricted by, or depends on the underlying model. This is discussed at the example of ADEPT and Worklets.
In the case of ADEPT, the manual adaption of a process schema, triggers formal analysis and reduction rules to guarantee certain pre- and post-conditions. Flexibility thus is provided by concepts that build on top of conventional workflow models. However, we envision a solution which provides flexibility by design and not by semantic consistency checks built on top of usually rather static workflows.
For Worklets, the general process structure of the model is pre-defined. As a consequence you are restricted to a specific order of tasks. Although the behavior of tasks can be adapted at runtime, the overall process structure is not adaptable. We require more flexibility without being restricted to a determined sequence. Furthermore, the substitution of tasks is done dynamically by rules. A central role in the processes we want to realize play humans, who decide about changes, sub-contracts, or the re-ordering of transport steps. For the envisioned solution we are searching for a adaption concept which allows more human interaction without intelligent substitution rules.

Apart from the declared shortcomings respecting our settings and requirements, all of the before mentioned approaches have one common disadvantage. Workflow modeling approaches which provide runtime adaptability such as ADEPT and Worklets are process-centric. By focusing on a pre-described control flow, the context which is the related data, is moved to the background. This phenomena is called context tunneling and results in errors and inefficiencies [AWG05]. Data objects, including their structure and life cycles, are rather modeled implicitly, although they are the main driver for decision making in our case.

2.3.2 Service-Oriented Modeling

The modeling paradigm discussed in this section is based on the SOA design principle, which is based on design principles including reuse, modularity, interoperability, loose coupling, and standard compliance. The OASIS group[2] defined a reference model for this architectural style. We are not interested in technical architectures to build a service-based system, but on the general design principles. In a SOA, functionality and behavior is encapsulated into *services*, that are combined to realize business processes driven by message interaction and information exchange. This section is not supposed to provide an overview about specific research in the field of SOA. It rather aims at describing the two fundamental concepts for wiring processes together. In context of the included assessment we describe certain properties in detail, which affect our identified requirements.

[2] http://www.oasis-open.org

Service Orchestration defines a process as the composition of existing services. Thereby, a central process always has the role of a controller to the involved services and the interaction between the services follows an explicitly defined control flow. The Business Process Execution Language (BPEL) [OAS07] is the de-facto standard for defining and executing business processes based on flow-oriented orchestration. A BPEL process describes activities, partner links, and events which are declared in a XML file and is finally executed on a BPEL engine.
Service Choreography is another approach for service coordination. A choreography does not require a central coordinator, which is responsible for the overall process. The process instead emerges on the interaction of the services, which is defined in form of peer-to-peer interaction rules. The individual services thus behave autonomously after message exchange patterns are defined. This makes a choreography dynamic and due to a decentralized control more scalable than an orchestration. The Web Services Choreography Language (WS-CDL) [W3C05] is an XML-based language that was developed to formally describe service interactions. Choreography is more collaborative in nature [BWH08]. Describes a collaboration between services in order to achieve a common goal / shared goal. All participants are treated equally in a peer-to-peer fashion.
The Business Process Modeling Notation (BPMN) [OMG11] finally provides an intuitive, visual representation of a process sequence defined within a BPEL process. Similar to UML activity diagrams, BPMN is a graphical flowchart-like language, specifying different diagram types for representing a choreography or general collaborative behavior.

Assessment

SOA as well as service-oriented modeling in general provide flexibility to deal with environmental changes by reusing or exchanging services or adapting their interaction protocol. This ability as a central motivation of SOA is based on loosely coupled services communicating with each other via standardized interfaces and message exchange protocols. Scalability is supported by the architectural design with small chunks of functionality implemented as separate services, which can be run on multiple servers. However, scalability mainly depends on the concrete realization of this service-oriented architecture.
In order to assess global view properties, we analyze orchestration and choreography as techniques for system structure and coordination. An orchestration describes and composes the process from the viewpoint of one central element. This perspective however is appropriate for relatively static process definitions only, and therefore it does not scale well for complex scenarios [Ros08] Besides that, we envision a solution design, which does not necessarily depend on one central controller steering the whole process. In a service choreography all services are considered as equal, intercommunicating peers. This allows the adaption of process parts, without the needs of redefining the overall process structure, and thus makes the model more dynamic and scalable. In a choreography, the services however are considered as black boxes, without having insight into internal behavior. This restriction sometimes is desired to hide implementation details, or it is indispensable because the concrete implementation is not known at design time. We are searching for an approach to describe and to adapt both, internal details of a functional component and their interaction. A choreography therefore is often combined with an orchestration to complement each other. The language for choreography WS-CDL however is a very abstract language, lacking a graphical notation, and without a comprehensive formal grounding [BDO05]. [Fre05] indicates this as the reason why WS-CDL so far has little use in industry. Independent of the advantages or disadvantages, which are implied in the service paradigm, our envisioned solution requires an abstract model structure and management properties similar to orchestration and choreography.
Service aggregation techniques like orchestration and choreography consider processes and col-

laborations as first-class citizens [OR03], [Hil+04]. This again leads to the phenomena of context-tunneling, which we want to avoid. We think, that a service-oriented modeling perspective is not appropriate for our purpose, because we considered the representation of data as the driving force.

2.3.3 Data-centric Process Modeling

This section describes the data-centric modeling paradigm to represent business process. Classical process-centric modeling approaches focus on *how* a business process is supposed to operate. The state of a process is strictly determined by the control-flow, which means by the number of already executed activities. This however does not represent the reality of how work is carried out in most non-manufacturing environments [AB01]. By prescribing what *must* be done instead of formulating what *can* be done, decision paths are limited. Data-centric process modeling aims to improve and to simplify process management, by letting the user decide what to do next, depending on the availability of data.

Case Handling [AWG05] is one concrete data-centric modeling approach. The whole business process thereby is modeled as *cases*, which are presented to the user. A case represents the product whereas the business process is the recipe for handling the case. The user can view and modify the data represented within a case. You specify on data level what information is required in order to fulfill a task, and when a task is considered as completed. The availability of data then determines the control flow, without statically prescribing the order of tasks and all possible deviations. This is called implicit modeling. The author states that this strategy leads to an increased flexibility of knowledge intensive business processes. Summarizing both, modeling and the execution of a so called Case Handling System (CHS) are data-driven.

Assessment

Case handling is not universally superior to traditional process-centric modeling approaches, however it provides a better view on the overall process by avoiding context-tunneling [RRA03].
Respecting our settings and requirements, we belief that data-centric process modeling is an appropriate strategy. In transport and logistics, the decision about the execution of a specific task depends on the availability of certain information and documents. Involved stakeholders generate and consume data, which is then shared among each other. While case handling solves the problem of context-tunneling, it grants the user a lot of liberties on how a task should be completed. In a collaboration environment however there exist certain standard procedures a stakeholder must stick to. Hence, we are looking for an approach that is slightly more restrictive but still offers enough flexibility and information transparency.

2.3.4 Artifact-centric Modeling

In this section we give an executive introduction into artifact-centric process modeling, which emerged from concepts of the previously discussed paradigms. In the first part, we describe the general idea and a modeling framework for this concept. In the second part, we given an overview of latest research in this field.

Modeling Framework for Business Artifacts

Artifact-centric modeling, is a unified business process and information modeling approach based on key domain *artifacts*. It is based on the idea to combine data and processes in a fundamental way. It provides a unifying basis for representing business rules and user interactions. A concrete artifact-centric research approach are *Business Artifacts* [NC03], [Hul08],[CH09],[BHS09]. A Business Artifact is described as a *business-relevant, conceptual object that evolves as it passes through the operations of an enterprise* [Nan+10]. One example of a Business Artifact is a purchase order. A order request document is described by an document identifier, a deadline, requested services, and related service costs. The process model (including the business tasks) of this artifact includes stages like request-creation, verification, order-execution, and order-payment. The order artifact then passes these stages while being manipulated by human interactions. Regarding the structure of a single Business Artifact, the authors in [Hul08] distinguish between four "interrelated but separable dimensions".

Information model. An artifact consists of an information model to capture all data and information which corresponds to this artifact during its lifetime. This includes business data, as well as meta-data including identifiers, relations, or state information. This model thus represents the knowledge base of this artifact. This knowledge base however evolves at runtime. A Business Artifact receives data from its environment, or produces data on its own. The artifact modeling approach does not prescribe any language, how the information model must be represented, as soon as data conflicts are prevented, and artifact life-cycle history can be stored. Alternatives for a technical implementation are XML-based information models [NC03], or information represented by database schema [BHS09].

Life-cycle Model. In addition to the information model for representing data, each artifact has a life-cycle. The life-cycle represents one or more relevant steps the artifact passes during business operation, and it therefore defines the control flow of the artifact. The life-cycle of a business artifact can be either relatively short-lived, or long-lived in case the artifact performs monitoring tasks or other time intensive operations. One possibility to define artifact life-cycles is to use variants of imperative, finite state machines [NC03].

Tasks/Services. A business task encapsulates a unit of work meaningful to the whole business process [Hul08]. Business tasks are either automated, or they require human interaction. In the context of Business Artifacts, tasks sometimes are referred to as services to emphasize the close correspondence to services in SOA. Technically, an executed service makes transactional changes to one or more business artifacts. These changes typically reflect measurable steps of progress of this artifact towards its operational business goal. At execution time, the interaction of possibly distributed artifact tasks drives the overall business process.

Associations. While tasks divide a business process into units of work, associations put them into context. An association defines under which conditions a task can be executed, and links it to point in the artifact life-cycle. This association of tasks can be defined in an procedural, or in a declarative manner. A finite state machine for instance, associates tasks to state transitions [Nan+10].

Artifact-centric modeling provides a methodology to effectively, and intuitively design business processes. A graphical notation for business artifacts however is required to make this concept usable. The GSM meta-model is one solution, which at the same time represents the latest development in this field [Hul+11c], [Hul+11a], [Hul+11b]. GSM however is more than a graphi-

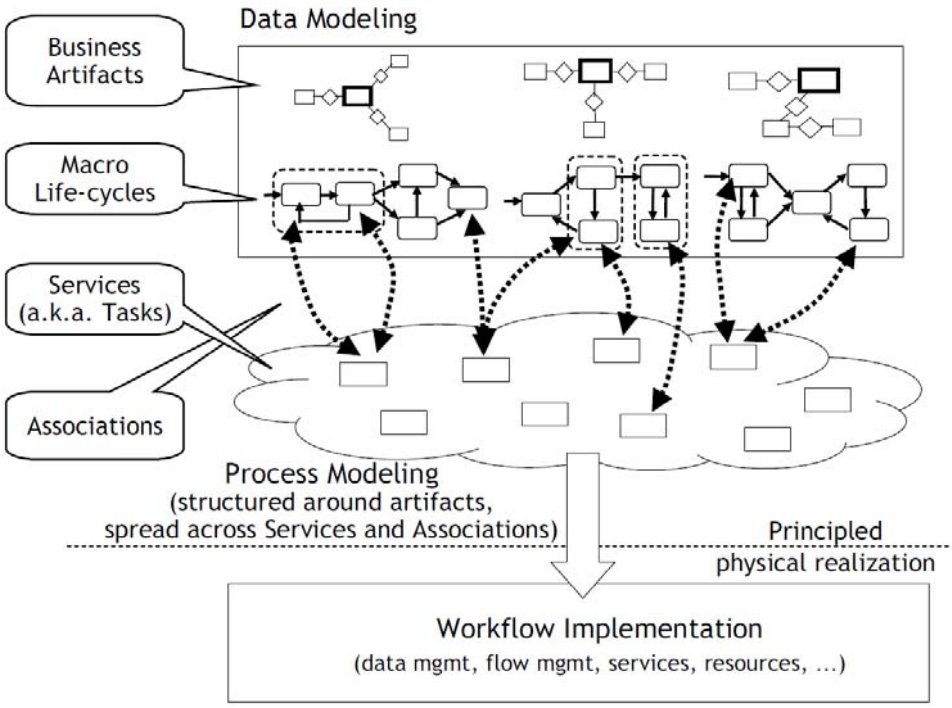

Figure 2.1: Business Artifact Dimensions [Hul08]

cal representation of Business Artifacts. In the first place, it provides a modeling framework for specifying artifact life-cycles. The central motivation thereby is to create an intuitive meta-model that corresponds to how business people think [Hul+11b]. GSM supports the management of business-related activities, however without supporting details of their execution [Hul+11b]. The formal semantic is rather declarative with an emphasis on *what* should be performed to accomplish a specific goal, instead of describing *how* it should be performed. Logic programming or Event Condition Action (ECA) rules are examples of specifying what task can be executed under what circumstances. In the following, we describe the three life-cycle elements of GSM (see Figure 2.2). To explain the semantics we describe the elements in the opposite directions, first milestones, then stages, and finally guards.

Milestone. The focus in business artifact modeling is on identifying business goals. A business artifact always has a declared operational goal which is supposed to be achieve by executing business operations. Such goals are designated as artifact *milestones*. Each milestone has a name and a boolean value, which indicates its achievement or its invalidation. An approved booking request, a completed invoice, or a successful order cancellation might be valid milestones of an artifact type. If a booking request milestone is achieved or not, depends on the actual data values and on triggering events. The milestone condition is formulated as a logical expression on data and events.

Stages. In order to achieve a milestone, an artifact passes various *stages* in its life-cycle. A stage is a cluster of activities. Just as milestones can be achieved or invalidated, stages can be activated or de-activated. An currently achieved milestone de-activates (or closes) a stage. Alternatives in completing a stage are defined by numerous milestones. Stages are depicted by rounded-corner rectangles.

Guards. Achieved milestones de-active the corresponding stage, since the goal of the stage is already achieved. The *guards* are responsible for activating stages and thus represent entering

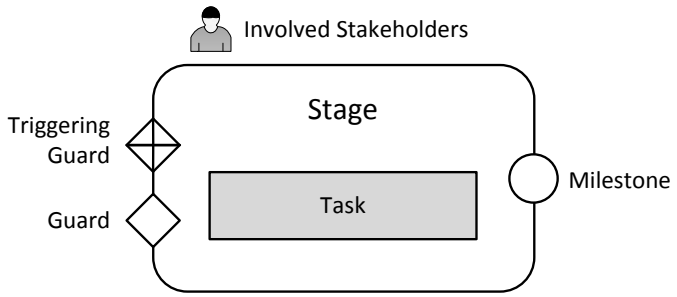

Figure 2.2: GSM Stage Structure

conditions of stages. A guard can be true or false, depending on the computed logical expression. Just like a stage might have more than one milestones, a stage can also have one or more guards. Guards are depicted as diamond nodes. A cross inside the diamond indicates the initial stage of an artifact which is that these guards trigger the creation of an artifact.

Figure 2.3 provides an example of how the life-cycle of a Business Artifact is described. In this case, a transport order is modeled as a Business Artifact. The central phases of its life-cycle are illustrated as three different stages. These stages named Order Creation, Order Execution, and Invoicing must be passed by the artifact in order to achieve its operational business goal. Invoicing thereby is an example of a nested stage consisting of two sub-stages for invoice creation and payment. Each stage is defined by one or more milestones, which indicate the operational goal. Each atomar stage has a business task associated, which is depicted as a grey rectangle. The transition from one stage to another is triggered by incoming environmental events. The environment of a Business Artifact consists of the involved stakeholders and all Business Artifacts.

The illustrated example describes, that the initial event for the creation of a transport order is an incoming order request. If this request is inserted into the system, the Order Creation stage is activated and the associated task to check the transport feasibility is executed. This stage closes as soon as the order is accepted, which can be signaled either by the automated task, or by the corresponding stakeholder. The same event immediately triggers the next stage of invoicing, whereby the first task is the invoice creation by making a proposal to the client. The operational goal of this stage is either to accept the proposal, or to reject it. Stages can further be executed concurrently. This is the case for the Invoicing stage and the Order Execution stage. As soon as the invoice proposal is accepted, the Order Execution stage is opened. At the same time, the Invoicing stage stays active until the invoice finally is paid.

Related Research

This section summarizes research areas which are directly related to Business Artifacts. There are investigations in formal analysis [Bha+07b], in automatic construction of artifact-based business processes [FHS09], [Yon+10], or in view-based modeling of inter-organizational processes [YLZ11]. A theoretical foundation for artifact verification and constraint checking provides [Zha+09] and [Deu+09]. The execution, data access, and message-based interaction of business artifacts is coordinated and controlled by a concept architecture named "artifact-centric inter-operation hubs", introduced in [HNN09]. In [Nan+10] the Business Entity Definition Language (BEDL) is proposed as a standard language for modeling Business Artifacts. These concepts were already successfully applied by in various real customer projects and engagements [Bha+07a],[LBW07].

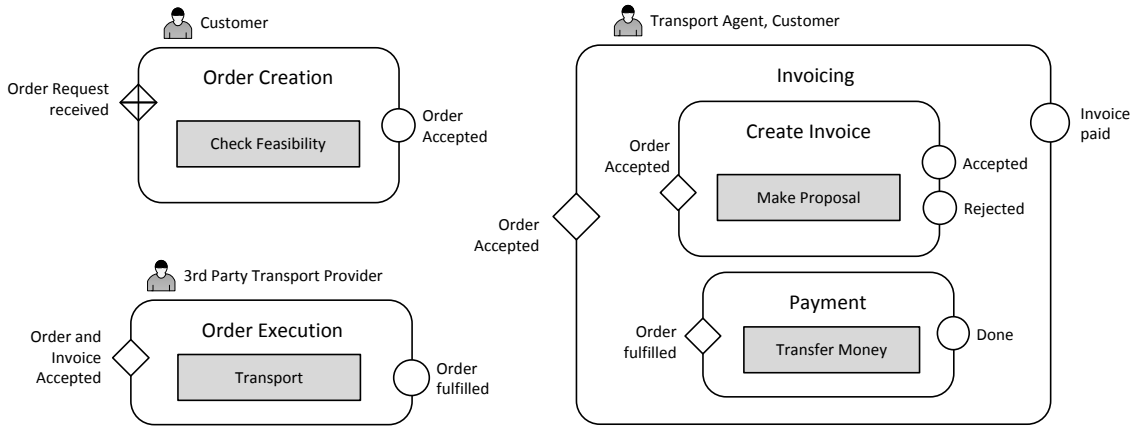

Figure 2.3: GSM life-cycle of a transport order Business Artifact

In [Mei+11] the authors present an artifact-based *Collaboration Hub* partners can use on-demand to coordinate their business collaboration. Process variability thereby is supported by allowing the business users to specify process variants. These process variants are provided by varying sets of Business Artifacts that are used to realize a process.

The *Artifact-Centric Service Interoperation (ACSI)* [ACS10] project, is an ongoing, european project. The objective is to develop a framework for service collaborations based on *Interoperation Hubs* and Business Artifacts. The research activities in this project focus on extending the current concepts of Business Artifacts with verification and synthesis techniques. In addition to that, the GSM meta-model is a central aspect for further investigation within this project. Therefore, an execution environment is under development, which is able to process GSM-based business artifacts.

Assessment

The approach of Business Artifacts is closely related to the the Case Management paradigm. Both concepts use key data elements which evolve as they move through the process. In artifact-centric modeling however both, data and processes are modeled within artifacts. This allows the modeling of standard procedures of a business process. Furthermore, this is more flexible, since data and processes are combined in single reusable, exchangeable artifacts. This combination further solves the problem of context tunneling by providing a global view. By accessing a business artifact, a stakeholder gets all information which is relevant for a specic task fulfillment. The global view thus improves decision making, especially in cases of deviations. The choice between imperative or declarative languages for business process models often is a matter of choice regarding personal experiences and preferences. Declarative approaches have its strength in reflecting circumstantial information, therefore, we consider GSM as appropriate. We think it is intuitive to think in operational goals of an artifact, and in stages, this artifact must pass to achieve this goal. Artifacts modeled with GSM and integrated into an artifact system, communicate via messages. The guards and milestones. The current publications and projects such as ACSI are indicators for usability and erhoffen further impact.

The artifact-centric approach however has some limitations. Since artifact-centric modeling is a relatively new approach, as far as we know, there is no engine available which supports the execution of Business Artifacts. Further, artifact cannot be configured. The processes we consider however have changes, for what you would need single artifact types to represent each variation.

2.4 SUMMARY

The basic requirements are to provide a global view on the process, to be flexibly, and scalable. Relevant research areas are the modeling paradigms, that address the representation and management of business process, in a flexible way. The state-of-the-art describes the latest research for each paradigm and assesses it, with respect to the declared requirements. This section summarizes the findings of the state of the art analysis to come to a final conclusion of what approach is suitable or helpful for development of a solution approach.

Adaptive-, Process-centric Modeling Approaches related to this kind of modeling make use of configurative, or compositional mechanisms to enrich conventional process modeling with more flexibility. Both approaches are based on the concepts of modularization. Configurative approaches thereby provide predefined variants to allow flexible process modeling. Compositional approaches on the other hand are based on the idea to construct a business process by connecting single fragments.
Although modularization has positive impact to flexibility, all related research approaches have one common drawback. The focus on a pre-described control flow, by considering process activities as first-class citizens, results in the phenomena called context tunneling. We consider this design as not appropriate, since the business processes we address are highly human-, and data driven. A global view on the process therefore cannot be established without additional effort to get knowledge about affected data, or involved events. Aspects of scalability are not directly addressed. This mainly depends on an underlying component architecture.

Orchestration and Choreography This modeling paradigm relates to service-oriented approaches to allow for more flexible and reusable business processes. Flexibility thereby is achieved by well-defined service interfaces and message-based interaction patterns. Service orientation theoretically satisfies the requirement for scalability, however this mainly depends on the underlying architecture and implementation.
Orchestration is considered as not suitable, since it relies on a central process, which acts as the main controller to all involved services. This provides a global view on the process, however only from the perspective of this central component. Choreography on the other hand is more collaborative. However, in a choreography the services are considered as black-boxes, which interact by message exchange patterns. We envision a solution, where the internals of single component are of central interest, and therefore should not be seen as a black-box. In addition to that, both concepts suffer from context tunneling, which we want to avoid in order to provide a global view on all important parts of a process.
Both techniques are important for coordinating business processes, however with the application to service orientation, they do not fulfill our needs.

Data-Centric Modeling The idea of data-centric modeling is to consider data as the main driver in a business process. Users are therefore free to decide what to do next, depending on the available data set at a certain point in time. This modeling perspective is considered as suitable for our human-driven, and data-intense collaborative processes. However, we envision a solution approach, where it is possible to represent data and certain standard procedures in the process flow. Purely data-centric approaches however do not support the representation of such activity sequences or constraints. Besides that, we are searching for an approach which is slightly more restrictive in prescribing certain behavior.

Requirement (ID)	(Adaptive-), Process-Centric	Service-Centric	Data-Centric	Artifact-Centric
Global Process View (1.1)	○	○	◐	●
Process Variants (2.1)	●	◐	○	◐
Process Agility (2.2)	◐	◐	–	●
Process Adaption (2.3)	●	◐	◐	○
Scalability (3.1)	–	●	–	●

Table 2.1: Summarized Assessment of the State-of-the-Art

Artifact-Centric Modeling Artifact-centric modeling is based on the idea to combine data and processes in a fundamental way. This encapsulation of business relevant information in single data entities provides a global view on the business process. Furthermore, autonomous, and loosely-coupled artifacts can easily be exchanged at runtime, without the need to adapt the whole artifact system. However, flexibility is limited. Although artifacts can easily be added or removed, the artifact model itself, is rather static. Variants, or configuration of single artifacts is not supported.

Summarizing, no research approach or modeling paradigm fulfills all declared requirements. Process modularization is considered as necessary to provide the required flexibility. Mechanisms for configuration or composition therefore are considered as useful to allow for process variants or process extensions. The artifact-centric modeling approach seems to be an appropriate way for intuitive, and data-oriented process modeling. The GSM meta-model further provides a declarative, visual description of the life-cycle model including process-, and service constraints.
We conclude, that a combination of adaptable process-modeling, and artifact-centric modeling is a promising way to combine the advantages of both concepts. Our contribution thus lies in extending the artifact-concept with mechanisms for variability and extensibility. A summary of all discussed paradigms is shown in Table 2.1. A filled circle indicates that the requirement is fully satisfied. A semi filled circle indicates partial requirement fulfillment, whereas a white circle means that the requirement is not met by the approach. A minus sign indicates that the concept either does not explicitly address the requirement, or that it simply depends on the concrete implementation or underlying infrastructure.

3 CONCEPTUAL DESIGN

This chapter presents the conceptual design of the solution approach for more adaptive, and flexible business process modeling. The previous chapter concludes that no sophisticated solution exists. However, general concepts are identified, which are considered as a appropriate basis for a solution approach. In the following, we discuss all required aspects and theoretical concepts. The first section gives a general overview of the solution approach. The second section motivates mechanisms for more configurative business processes in artifact-centric modeling and introduces the terminology, which is used for the subsequent sections. The third section provides a formal introduction and defines the central elements of the solution approach, as well as the configuration process. The last section describes the conceptual architecture, which is required for a conceptual realization of the previously discussed concepts.

3.1 SOLUTION APPROACH

This section introduces the overall conceptual solution design. It provides an overview of the involved components, and describes the interaction with the business environment. More details, especially on capabilities for process model configuration are discussed in subsequent sections.

The overall goal is to provide a solution to efficiently manage dynamic, inter-organizational business processes. Such a system however requires an appropriate model, which is adaptable, and which further provides a global knowledge base for all involved stakeholders. The general approach to this challenge is shown in Figure 3.1.
The front-end represents involved stakeholders, which require customized views on consistent business data. The solution approach however addresses the underlying management model, which provides schemas for a predefined class of problems. The business model is designed from an artifact-centric modeling perspective. The execution environment finally is responsible for the runtime management of the model instances.
The basic element to represent a business process is the *Collaboration Artifact (CA)*, which is based on concepts of conventional Business Artifacts. By representing both, data and internal process models, a CA provides a global view on the represented process, and moreover is easily exchangeable. The solution model consists of a set of collaboration artifact types. These core types are derived by an executive domain analysis to represent general behavior and data types.

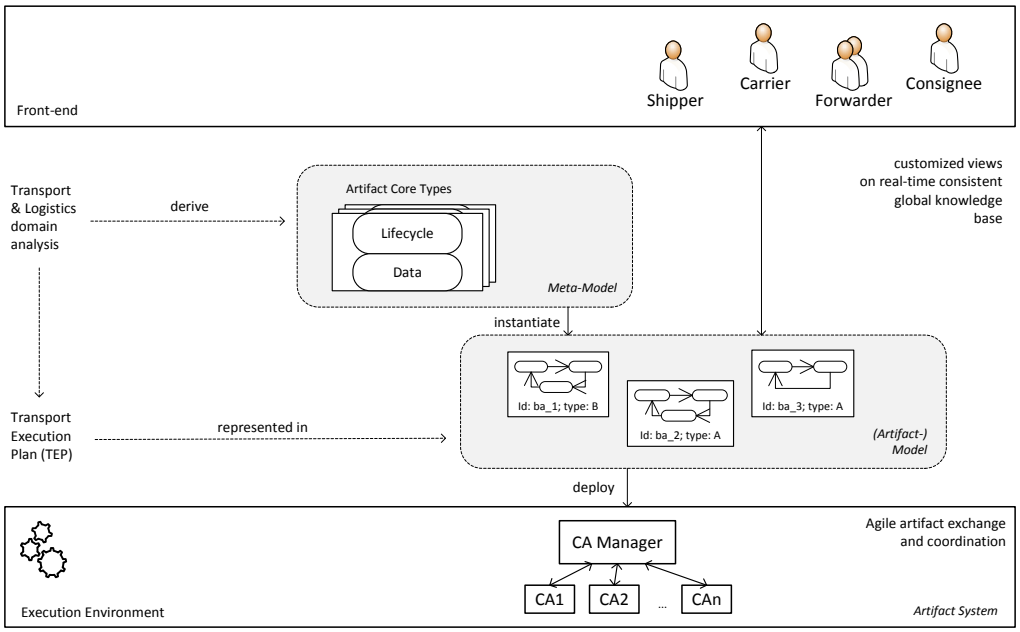

Figure 3.1: Solution Approach (Overview)

A concrete scenario, for example a transport process, is then represented by an instance of this meta-model. Finally, this model instance is deployed to the execution environment. A corresponding CA Manager is responsible for the runtime management. Since Collaboration Artifacts are encapsulated, autonomous elements with well-defined communication interfaces, they can be easily exchanged at execution time. The overall business process behavior is represented by the interaction of Collaboration Artifacts.

The (meta-)model supports flexibility by exchanging Collaboration Artifacts at runtime, and thus allows dealing with process re-planning. This artifact design however has limitations. The initial domain analysis to set up a collection of artifact core types is cost-intensive. For frequently changing business process scenarios this leads to problems, if the set of core types is insufficient to represent the changed requirements. This is considered as a bottleneck for efficiency and flexibility.

3.2 CONFIGURABLE COLLABORATION ARTIFACTS

The loosely-coupled nature of an artifact-centric design provides a certain degree of flexibility. This section introduces the concept of *Configurable Collaboration Artifacts* in order to provide a higher level of adaptability at design and at runtime. Starting with a brief discussion about general characteristics of artifact-centric modeling, we identify aspects, which must be adapted to meet our identified solution requirements. The second part introduces a theoretical background of design variability and describes how such variability can be provided by Collaboration Artifacts.

3.2.1 Concept Motivation

In this section we motivate the need for artifact configuration. In a first part, we describe those characteristics of business processes, which emphasize the limitations of Business Artifacts. After discussing possible solutions, we explain how our solution approach addresses to this problem. Finally, we describe different variability aspects that must be considered in order to realize the proposed solution.

Configurable Collaboration Artifacts are based on the approach of Business Artifacts [Hul08] and on the related GSM meta-model [Hul+11c], [Hul+11a], [Hul+11b]. The main objective thus is to intuitively represent business processes by core artifacts including both data and life-cycle information [CH09]. A Collaboration Artifact, as well as a conventional Business Artifact is a central place for all kind of information needed in completing a specific business goal. This concept allows for efficient monitoring and management of business operations. One example of such an artifact type is a customer order. After being created, different stakeholders execute tasks on this entity, depending on certain conditions. The purpose of this data-centric approach is to provide all information within single key domain entities, and to let these entities evolve by executing associated services.
This is fundamentally different to conventional workflows with pre-defined activity sequences. The modeling paradigm provides flexibility in form of adding, deleting, or replacing artifacts without the need to stop and adapt the overall business process. This is mainly achieved by encapsulating data and process information in single artifacts. Beyond that, message-based process execution reduces the coupling between running instances. The domain analysis showed that there are likely changing aspects among transport scenarios on the one hand, as well as common, and more universal aspects on the other hand. This holds for the overall process structure, and even more important for the sub-processes implemented by domain artifacts. The conventional artifact approach however is based on fixed artifact types, which cannot be varied, neither at deploy nor at runtime. That means, that the deployed artifact system is limited to predefined data and life-cycle models. Even for slightly changing requirements regarding data types, tasks, or life-cycle constraints, considerable effort is required to adapt to this new scenario. The following items are examples for variables that influence the required process model:

1. Transport means
   (a) Air / Road / Sea / Rail
2. Cargo type
   (a) Dangerous goods
   (b) Perishable goods
3. Differences among stakeholders and their "best practices"
4. Country regulations
   (a) Prohibited truck transport on Sunday in Germany
   (b) Import / Export procedures
5. Climate differences and weather conditions

These examples illustrate, that there are reasons not to model Business Artifacts with pre-defined data and process models since it is very likely that later adaptations are indispensable. Variability

thus is an essential characteristic of logistics processes, which must be taken into account. The conventional artifact-centric approach offers the following ways to deal with the described aspects of uncertainty:

a) Predefine as many artifact type variants as possible.

b) Interrupt process execution to (re-)implement and (re-)deploy the required artifact types.

c) Define complex artifact types, which by default contain all possible data and life-cycle manifestations. At deploy time, the required parts are instantiated, whereas the optional parts are ignored.

On top of that, there is always the alternative to restrict an artifact model to certain scenarios only. Such limitation however is non-compliant with the aim of supporting as many scenarios as possible with one generic business process model. In our opinion, a) is no alternative, since the variety of possible scenarios is unpredictable high according to domain experts. Furthermore, such approach is neither elegant nor efficient, because it requires the definition of an enormous amount of very similar, but yet slightly different artifact types. Option b) requires considerable effort for customization whereby c) results in huge and for this reason hardly manageable and non-intuitive artifact definitions. Summarizing, none of the above is an acceptable alternative to meet the identified requirements.

We want to support variability at a fundamental level for the development and reuse of adaptable artifacts. Instead of relying on an artifact system with pre-defined type definitions, the envisioned model should efficiently deal with changing requirements. The basic idea consists of adapting *Artifact Templates* with *Fragments*. In a first step, the common parts among different scenarios must be separated from the variable parts. An in-depth domain analysis is the basis to make these decisions. Assuming that a business process is already represented by Business Artifacts, the idea of decomposing artifact types into adequate pieces is illustrated in Figure 3.2. The common parts among the artifact types *Art1 - Art4* are represented as generic Artifact Template types. Parts that are likely to vary from case to case, are encapsulated into Artifact Fragments. Depending on this design decision, variability can be induced at various abstraction layers. Artifact variants are finally constructed by combining a Templates with one or more Fragments, regarding certain rules. By implication, the reconstruction of the artifacts *Art1 - Art4* must be possible.
As a consequence, an artifact type does not depend on pre-defined information models, which results in more flexibility and in process variants. By splitting up a monolithic model into reusable parts with well-defined interfaces, the design comprehensibility increases, and as a consequence the software quality increases as well. Following the "divide and conquer" principle to break down a complex problem into manageable parts, such modular design simplifies testing, error handling, or security issues. The reuse of functionality across different scenarios saves time and money. Beyond that, modularization allows the assembling of process variants and therefore covers a broader spectrum of use-cases, which is exactly the goal of our work.

Although a modular design has several advantages, it also has challenges. First, it must be decided at which abstraction level modularization should be employed. For the meta-model of Business Artifacts, modularization could be applied to single data attributes, life-cycle stages, or pre- and post-conditions. The goal thereby is to achieve the desired flexibility while keeping the complexity of the variation space manageable. If the resulting model is too fine-grained, this increases flexibility, however with the disadvantage of an unmanageable variation space. On the other hand, a very coarse-grained model, which does not provide the required flexibility is the other extreme.

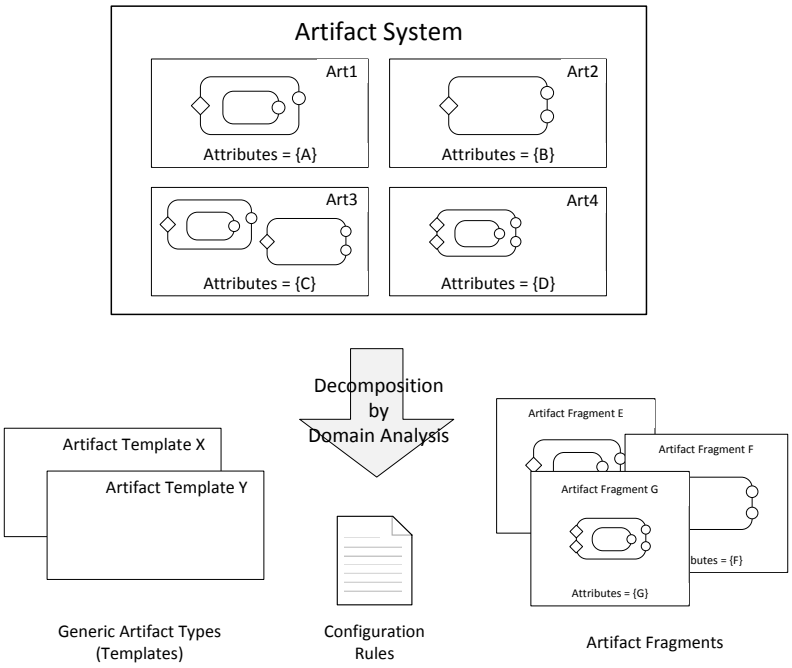

Figure 3.2: Decomposed Business Artifacts

A second challenge is in identifying both the common and varying aspects of the domain model. To get maximum benefit, this step requires domain knowledge and therefore must be supported by domain experts.
Third, a modular design might suffer from inconsistencies, which requires additional effort for consistency checking at design time, as well as for process verification at runtime. The complexity of this problem depends on the level of modularization, as well as on the expressive power of the specific verification languages. This topic of verification is heavily discussed for the approach of Business Artifacts [BLB12], [Deu+09], [FHS09]. In [Bha+07b], the authors state that the complexity ranges from undecidable for the general case, over PSPACE-complete when no new artifacts are created during execution, up to linear complexity however with more restrictions. As a consequence, the degree of modularization introduced into business processes must be considered carefully. In Section 3.3.2 we discuss verification with regard to the configurable design of Collaboration Artifacts.

Summarizing, the decomposition of Business Artifacts into Templates and Fragments, allows to flexible combine process parts to new artifact variants. Such modularization however has its own challenges. Based on these findings, the following section describes how variability can be approached, and what aspects determine variability in general.

3.2.2 Approaching Variability

In order to provide a framework for an adaptable design, this section introduces the required terminology. First, we describe a paradigm which allows to define artifact variation points. Subsequent to this, we discuss aspects which must be considered in Collaboration Artifact design.

In common language use, the term of variability refers to the ability or the tendency to change [PBL05]. We are interested in the kind of variability, which does not occur by chance but is brought about on purpose. Extensibility has also the ability to vary, however by adding an a-priori not

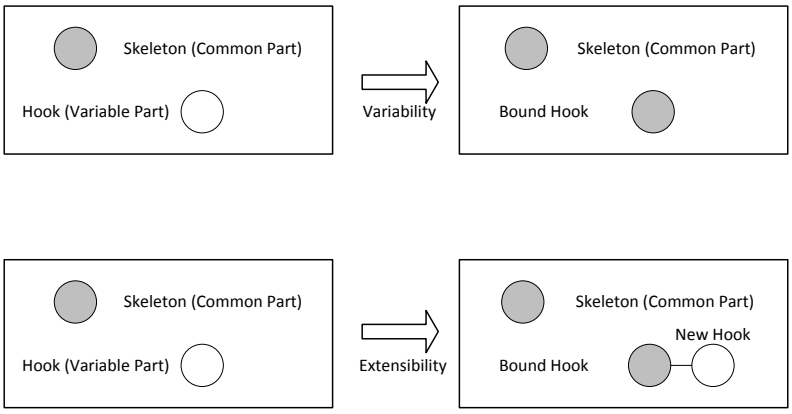

Figure 3.3: Variability and Extensibility

predetermined set of functionality to an already existing core. The author of [Pre95] introduces a paradigm for describing common and variable parts. The "template-and-hook" (T&H) concept allows to describe the "hot-spots" of variation and extension. Although this concept is introduced at the level of design patterns with classes and operations, we use this terminology to explain the adaptation of Collaboration Artifacts and Artifact Fragments. The distinction between fixed and variable parts are represented by role types named *templates* and *hooks*. A template provides a skeleton (frozen spots) of a component, whereby a hook (hot spot) declares a point of variation. Figure 3.3 illustrates this idea. Variation is finally achieved by binding a hook with a functional fragment. Extension on the other hand is accomplished by extending a hook of a component. In the following, binding/extending a hook with a fragment, is called a configuration step.
A system, which provides variability as well as extensibility at a fine-grained level, both at design and at runtime, would be the most desirable solution. However to find the best solution is not a trivial task, since there is a trade-off between flexibility and complexity. In the following, we describe different facets of extension and variation. This states the basis for subsequent design decisions in order to provide the kind of flexibility, which is characteristic in the transport and logistics domain.

Granularity The first facet expresses the level of modularization granularity. At design time, it must be decided, which element is subject for variation, and at which level of abstraction variation or even extension is required. The proposed design provides extension at the level of artifacts and thus allows to efficiently deal with changing requirements by extending an artifact with the desired functionality. This approach of a core artifact type, which is extendable with a theoretically infinite but a-priori unknown number of fragments is represented in Figure 3.4 and 3.5, before and after configuration. One configuration step consists of binding an Artifact Fragment to a Artifact Template and results in a modified template with transparently included fragment details.
According to the Guard-Stage-Milestone meta-model, the stages with its guards and milestones are identified as the variable parts representing sub-segments of the Business Artifact life-cycle. Such sub-processes are representative steps in logistics processes, and are therefore encapsulated into Artifact Fragments. At this level of abstraction, the Artifact Fragment itself must be adaptable in order to allow for fragment variants. This is required since those sub-processes and their related data models might differ between use-case scenarios. A loading fragment for instance might be slightly adapted to changing requirements to support a changed loading-check strategy, or a changed document format. In order to support variation, the flexible spots, also referred to as variation points, must be determined a-priori. A configuration file describes the configuration process by using pairs of variation points and a corresponding value. Figure 3.6

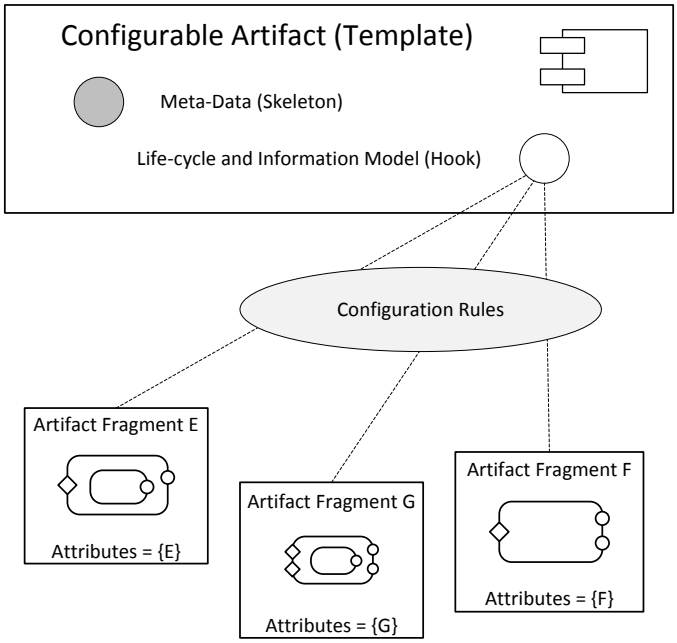

Figure 3.4: Configurable Artifact (before configuration)

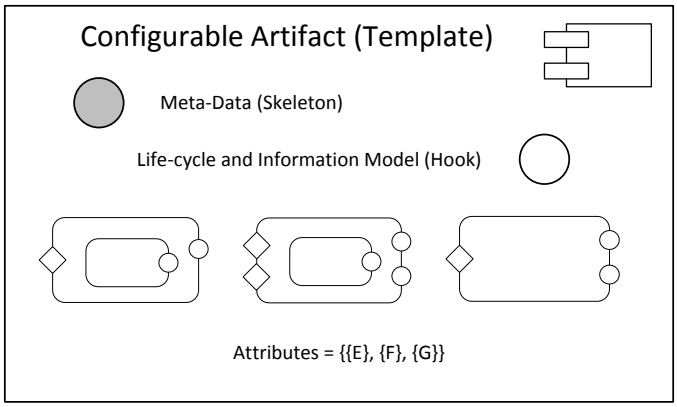

Figure 3.5: Configurable Artifact (after configuration)

illustrates the concept by means of an Artifact Fragment for aircraft loading. This design decision does not allow fragment extensibility. If certain functionality is required, which is not represented by any fragment variant, a new fragment must be implemented. This decision thus limits the level of possible variability, however it keeps the resulting complexity manageable.

Contract between Components This facet of variability is about the relation of a template and possible fragments. In order to define those relations, it is required to identify mandatory, alternative, and depending functionality. Applied to Artifact Fragments, this could mean; a fragment *frA* requires fragment *frB*, but excludes fragment *frC*. One example of such dependency in transport and logistics is: a loading fragment requires a fragment for unloading. This kind of relations can be represented by Feature Trees or Variability Diagrams as used in [PBL05]. Another aspect which relates to this facet is the declaration of variation points. The proposed design assumes an explicitly defined interface for artifact extension. This composition interface referred to as the extension point, is supposed to bind an a-priori unknown number of fragments at deploy or at

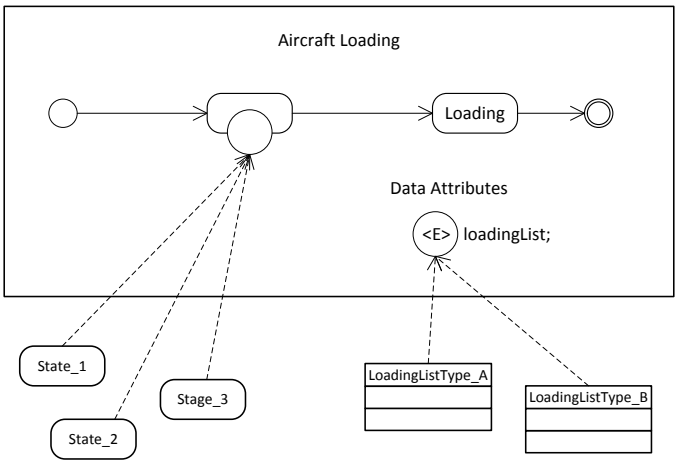

Figure 3.6: Artifact Fragment for Aircraft Loading

runtime. In contrast to the artifact template, an artifact fragment requires the declaration of variation points at explicitly defined points of the implemented process and data model, which might include the labeling of certain states in the life-cycle model, or a specific exchangeable attribute.

Binding Time Besides considering the level of granularity and describing the interfaces for variation, it must be decided whether the binding of a hook is performed at design time, or at runtime. In general, the goal is to postpone configurations as late as possible. If a transport plan, which includes all required knowledge for configuration is analyzed at runtime to dynamically configure the artifacts, then the binding is accomplished at runtime as well. This holds for artifact and for fragment configurations, since the system is already running and is not stopped to reconfigure the process.

This section provides an abstract introduction into the theoretical concepts for artifact variation. In the following section, we define the detailed structure of Collaboration Artifacts.

3.3 ARTIFACT CONFIGURATION AND EXECUTION

This section aims at providing a detailed description of Collaboration Artifacts. The structure is oriented on the different phases in process modeling, including configuration, deployment, and execution. This life-cycle of configurable process modeling is described in [RD08]. Configurable process models are realized by splitting up the traditional design phase into two phases: one where the reference model is designed, and another where this model is configured to a particular context. After a deployment phase including model verification, the artifact model finally is executed. Figure 3.7 shows the involved cyclic modeling phases. On the left, a traditional process is illustrated, whereas on the right the configuration process is described. In the following, we discuss these phases, each one in separate section.

In order to add a semantic to an Artifact Fragment, we introduce the term of a *Collaboration Feature*, because a technical fragment represents an artifact- or collaboration feature, which can be added to the business process. In the following we use both terms interchangeable. The first section defines Collaboration Artifacts and Collaboration Features, which states the basis for the description of a Configuration Step. Since a configuration must guarantee correctness of the

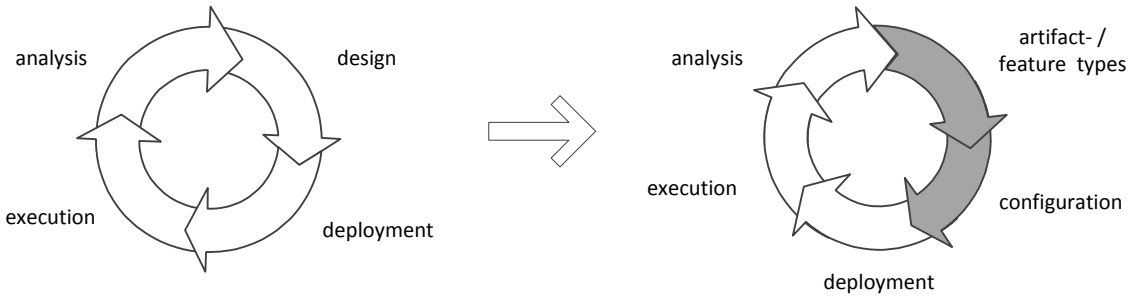

Figure 3.7: Artifact Configuration Process

resulting artifact, in the second section we describe concerns for verification and model checking. Finally, we describe the runtime behavior of Collaboration Artifacts

3.3.1 Configuration of Collaboration Artifacts

This section provides the theoretical framework for the modeling with Collaboration Artifacts. The aim is to describe the detailed structure and the basic elements and concepts. Therefore, we define the structure of Collaboration Artifacts, of Collaboration Features, and of a Configuration Step. The important aspects thereby are the declaration of extension- and variation points. The way of defining artifacts and features, is oriented on the notation, which is introduced in [Hul+11a].

Definition 3.3.1 Collaboration Artifact:

A Collaboration Artifact c *has the form* $c = (R, D, Att, S, M, \mathcal{L}, F_c)$*:*

- R *is the name of the collaboration artifact.*
- D *is the meta-data belonging to the collaboration artifact.*
- Att *is the set of data-, event-, and status attributes.*
- S *is a set of stages.*
- M *is a set of milestones.*
- $\mathcal{L}$ *is the life-cycle model.*
- F_c *is a set of bound features to* c *with* $F_c = \emptyset$ *at time of initialization.*

A Collaboration Artifact is an essential component for representing business processes. Each artifact has a name R, as a unique classifier of this type. Furthermore, each Collaboration Artifact contains some meta-data D required for management or monitoring purposes. Such meta-data includes information on the artifact version and timestamps for instantiation or storage. The set of attributes Att is subdivided into a set of data attributes that define the data model of an artifact, a set of event attributes, which represent registered event types, and a set of status attributes. The latter type of attributes are booleans to indicate whether a stage $s \in S$ is currently active, and to indicate if a milestone $m \in M$ is currently achieved or not. Besides that, the point of time is tracked when a status attribute has changed. The life-cycle model $\mathcal{L}$ of a Collaboration

Artifact c is further defined according to [Hul+11a] by describing the nesting of stages and their corresponding guards and milestones including their pre-and post conditions, and by describing the associated tasks. The most important item in the list to realize our approach is the set F_c which indicates the bound Features. This set, represents the point of variability, and is usually empty after initialization, however it can also be initialized with Features. The hook represented by the set F_c must have a unique name and a type to represent valid fragment types. This declared hook represents the configuration interface of the Collaboration Artifact. A Collaboration Artifact c thus represents the fixed part or the template which is configured. In the following we describe the variable part, named Collaboration Feature.

Definition 3.3.2 Collaboration Feature:

A collaboration feature f has the form $f = (R, D, Att, S, M, \mathcal{L}, V_f)$:

- *R is the name of the collaboration feature.*
- *D is a description of the behavior the feature is meant to provide.*
- *Att is the set of data- and status attributes.*
- *S is a set of stages.*
- *M is a set of milestones.*
- *$\mathcal{L}$ is the life-cycle model.*
- *X_f is a set of variation points $\{x_1, x_2, \ldots, x_n\}$.*

The name R of a collaboration feature is a unique classifier of this type. A description D contains information about the purpose of the hook type. The sets of attributes Att, stages S, milestones M, as well as the life-cycle model $\mathcal{L}$ are defined in exactly the same way as they are formulated in Definition 3.3.1. X_f is a set of variation points supposed to realize the parametrization of variable parts with concrete values. Each variation point $x \in X_f$ refers to one element in the life-cycle, or to one element in the sets Att, S, M that is subject to change. To ensure a valid parametrization of predefined variation points, each variation point has a unique name and a type to represent valid fragment types that can be bound. In contrast to extension points the the variation points $x_i \in X_f$ are allowed for variation only. The collection of variation points makes the configuration interface of an Artifact Feature.

Definition 3.3.3 Artifact Configuration Step:

A configuration step of an artifact is the concatenation of a feature parametrization and a subsequent artifact extension. In the following let:

- *$C = \{c_1, c_2, \ldots, c_l\}$ be the set of available artifacts,*
- *$F = \{f_1, f_2, \ldots, f_m\}$ be the set of available features,*
- *$X_{f_i} = \{x_1, x_2, \ldots, x_n\}$ be the set of declared variation points of a feature $f_i \in F$,*
- *$V = \{v_1, v_2, \ldots, v_o\}$ be the set of valid assignments of the variation point $x_i \in X_f$.*

Parametrization Step: The parametrization of a feature $f \in F$ with n variation points, is described by an n-dimensional vector consisting of pairs of an variation point $x_i \in X_f$ and the chosen values $v_i \in V$. This tuple declares the assignment of value $v_j \in V$ to the variation point x_i. The result of a feature parametrization is a feature f' with closed variation points.

Extension Step: The second step is the assignment of a feature $f' \in F$ to an artifact $c \in C$: This means that a collaboration artifact $c_{old} \in C$ with $F_{c_{old}} = \{f_1, f_2, \ldots, f_n\}$ results in a collaboration artifact c_{new} with $F'_{c_{new}} = \{f_1, f_2, \ldots, f_n, f'\}$. This extension can be expressed as a function $ext_{C,F}$: $(c, f') \rightarrow c'$.

The configuration typically is accomplished using configuration files. This is a kind of configuration recipe, which defines the artifact manifestation. The first step is to identify the artifact types. A configuration of a Collaboration Artifact consists of one or more configuration steps. Depending on how many Features are bound to the artifact. Such configuration of an artifact results in an executable process.

This section shows, that the concept of Configurable Collaboration Artifacts and variable Features can be formally defined. The next section describes the verification of artifact configurations at deploy time.

3.3.2 Verification of Collaboration Artifacts

The process of designing and configuring Collaboration Artifacts requires well structured validation and verification, before the artifacts are deployed. The reliability of a system can be increased by checking, in advance, if a deployed program behaves as supposed. This section addresses challenges of verifying an artifact configuration. We give an overview of how the verification problem could be defined for our case, however without providing a concrete solution. We illustrate the topic by means of a simple example. Finally, we provide an overview of related work for future investigations.

Model Checking

Feature composition and subsequent artifact instantiation result in an artifact system which represents a process model. The main challenge of process modeling is in determining whether the model exhibits a certain behavior. For instance, the model is checked for invalid states, since those might lead to deadlocks or live-locks. Another desirable property especially for systems with human interaction is that the system cannot run into a state where information sensitivity is not guaranteed. Properties are also generalized to characteristics that are desirable for each kind of business process. An example is the *soundness* property, which is defined of three subproperties. A started process can always complete; an ended process should not have any tasks that are still running for that process; and the process should not contain tasks that will never be executed. Verification is considered as an important step before deploying executable business process models since it can greatly improve the reliability of such systems [Wyn+09].
With regard to our solution approach, we are interested in guaranteeing certain behavior after artifact configuration, which primarily covers the reachability of its operational goals. This is required before verifying the whole artifact system, with the goal to verify whether all runs of an artifact system satisfy desirable correctness properties. Potential errors do not arise from adding

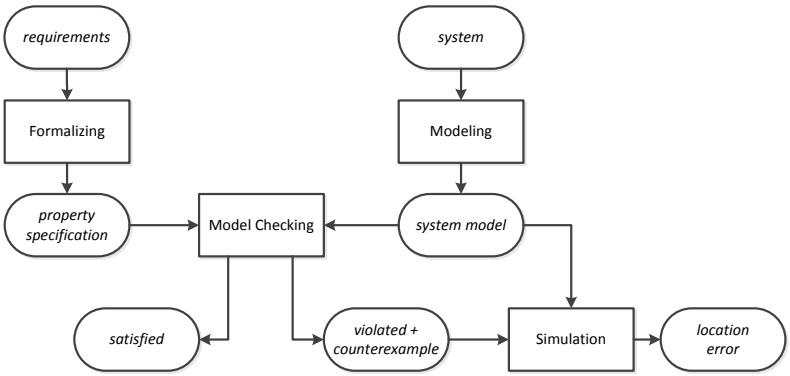

Figure 3.8: Schematic view of Model-Checking [BK08]

or removing artifacts to or from a system only, but also might be consequences of feature binding operations.
Let a transport scenario serve as an example, whereby a service provider is responsible for the loading at the port terminal. The representing artifact must guarantee specific behavior to achieve an operational goal. Such guarantee for example concerns the availability of certain data and the enforcement of specific task sequences in order to finish the loading. A test might be formulated as follows:

> "Some artifact must provide the stuffing of goods in advance into containers, and a load list must be available at some point in time, in order to load the ship properly."

In order to satisfy the correctness of artifact systems, the authors in the domain agree in the major problem for artifact verification, which is the presence of data and the infiniteness of possible states [Deu+09], [GS07]. The verification of such setting is typically undecidable, which means it is impossible to construct an algorithm that always leads to a correct yes-or-no answer. The problem of decidability can be handled by model abstraction, whereby details are removed which are not relevant for the property under verification. Another possibility are rather severe restrictions for the domain or for accepted values. In this section we will not discuss those approaches. Instead, we introduce model checking as a general technique to solve the problem of verification. Although model-checking has limitations for problems beyond control-flow systems, it is important for the basic understanding of the problem. Subsequently, we define properties, that are desirable to be checked for an artifact configuration.

Model checking is defined as an *"automated technique that, given a finite-state model of a system and a formal property, systematically checks whether this property holds for (a given state in) that model"* [BK08]. The model checking process is shown in Figure 3.8 and is divided into three phases with corresponding tasks [BK08]:

- *Modeling*: Model the system using a model description language and formalize the property to be checked using a suitable specification language.
- *Running*: Run the model checker to check the validity of the property in the system model, this is to check whether the given model satisfies a certain property or not.
- *Analysis*: Handle property satisfaction or property violation. A counterexample which leads to the result serves as a basis for further model adaption.

In order to verify a configuration of an artifact these steps of model-checking must be passed. An artifact system represents both, control-flow and data-flow and further constitutes an infinite-state system. Therefore, model-checking in general is not effectively computable [BK08]. Instead of describing a workflow by activities, the life-cycle of an artifact evolves by executing associated tasks, which are constrained by declarative, event-condition-action rules. Thus, advanced mechanisms are required to support parallelism, communication, and data-dependencies. Despite those limitations, the concept of model checking provides effective techniques to expose potential design errors of a configuration.
The definition of a concrete decision problem requires a description of the underlying model and the formalization of the desired properties that are subject for verification. Even if we assume a consistent set of single feature definitions, the correctness of feature combinations is not guaranteed. We define four properties that should be verified before deploying a configured artifact.

- Precondition conflicts: There are no conflicting guard definitions among different features.
- Effect conflicts: There are no conflicting milestone definitions among different features.
- Reachability: All milestones of a configured artifact can be reached.
- Executability: The life-cycle of the artifact cannot run into any deadlock.

The guards and milestones of a stage can be represented by terms like input parameters, output parameter, pre-conditions, and post-conditions. Post-conditions permit non-determinism in the outcome of a service, as a typical case in human-driven business processes [Deu+09]. Finally, the composition of various features leads to a set of conditions, which should not be in a conflicted state. Based on conflict prevention, the property of reachability and executability can be derived. It is desired that all stages within the features are theoretically reachable during runtime, and that the system never runs into a deadlock. If stages are not reachable, they are superfluous. In order to test desirable properties we need a formal notation for verification.

Towards a Formalization of the Decision Problem

Model checking can be defined as a decision problem: given a desired property in form of a logical formula p, and a model M with initial state s, then decide if $M, s \models p$. One prerequisite for model-checking thus is a model of the system. This means, the behavior of the system must be described. Transition systems are models to describe the behavior of concurrent systems. The system thereby is basically modeled as a directed graph with nodes representing states, and edges modeling transitions indicate state changes. In order to address the configuration of one Collaboration Artifact, we define the behavior of an artifact as a transition system. This notation is the basis for the definition of certain properties that must be checked. Finally, the overall problem statement of Collaboration Artifact verification can be formulated.

Definition 3.3.4 Artifact Behavior as a Finite Transition System:

A finite transition system is a tuple $(S, I, G, Act, \longrightarrow)$ *where:*

- S *is a finite set of states,*
- $I \subseteq S$ *is a finite set of initial states,*

- $G \subseteq S$ *is a finite set of goal states,*
- Act *is a set of actions,*
- $\longrightarrow \subseteq S \times Act \times S$ *is a transition relation.*

An artifact passes different stages before reaching its operational goal. We consider a state $s \in S$ as an artifact snapshot at a certain moment of behavior. This state includes the status of all guards, stages, and milestones, as well as the values of the data attributes. The initial state $s_0 \in I$ is the snapshot of the artifact just after its configuration. A goal state $s_n \in G$ is a milestone, which represent the operational goal of the artifact. An action $\alpha \in Act$ represents any activity, that changes the data/status attributes of the artifact. The guard/milestone sentries are considered as possible pre-and post-conditions $pre(\alpha)$ of an activity, whereby events are the effects $eff(\alpha)$. The transition relation finally indicates the state change from s to s' triggered by an action α.

In the following we write $s \xrightarrow{\alpha} s'$ instead of $(s, \alpha, s') \in \longrightarrow$. According to [BK08] the behavior of a transition system can be described as follows: The system starts in some initial state $s_0 \in I$ and evolves according to the transition relation $\longrightarrow$. If s is the current state, then $s \xrightarrow{\alpha} s'$ states that the action α is performed and the transition system evolves from state s into the state s'. If $E = \{(s, s') \in S \times S \mid s' \in successor(s)\}$ is the set of possible transitions, then the selection of $s \xrightarrow{\alpha} s'$ originating from s is done non-deterministically. This selection procedure is repeated in state s' and finishes if the system reaches a state s where $successor(s) = \emptyset$. A state transition is accomplished under consideration of the GSM model of the artifact, and under consideration of possible events and messages from within the environment or the artifact itself.

The following definitions comply with the model of transition systems. In a first step the properties that must be checked during verification are defined, whereas the second definition summarizes the overall problem statement for verification of an artifact configuration.

Definition 3.3.5 Desired Properties (P1-P4):

The desired verification properties are defined as follows:

- *Reachability (P1).*
  A state s_j is reachable if(f) either $s_j \in I$ or there exist a reachable state s_i and an executable action α with $s_i \xrightarrow{\alpha} s_j$.
- *Executability (P2).*
  An action α is executable if $pre(\alpha)$ is satisfiable.
- *Precondition Conflicts (P3).*
  α_i has a precondition conflict with α_j if $eff(\alpha_i) \wedge pre(\alpha_j)$ is unsatisfiable.
- *Effect Conflicts (P4).*
  α_i and α_j have an effect conflict if $eff(\alpha_i) \wedge eff(\alpha_j)$ is unsatisfiable.

By definition of these properties, the verification task is to guarantee reachability as well as executability for all states, and to prevent from precondition conflicts and effect conflicts, whereby it is debatable to some extend whether precondition/effect conflicts represent flaws, or whether they are a natural phenomenon of the modeled process [WHM08].

Definition 3.3.6 Problem Statement for Verifying an Artifact Configuration:

Let I be the initial state after configuration with a potential number of empty data attributes. Let G be the set of those milestones, which constitute an operational goal of an artifact. Let D represent all information, which is possibly available for the artifact, including data from the environment and from within the artifact system. Let C be the set of the pre-conditions and post-conditions defined within the guards and milestones of the artifact. Let p define a sequence of activities that lead to a state transition of the artifact.

Under usage of the data D and under consideration of all constraints C, the decision problem is answering the question if there exists a finite path $p = s_0 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} s_n$ with $s_0 \in I$, that leads to a goal state $s_n \in G$. This requires that all states in the GSM model are reachable, with regard to the reachability property P1. The checking for precondition/effect conflicts (P3 + P4), as well as if the process is always executable should be guaranteed first. In some cases P1 then follows from the P2 [WHM08].

Illustrative Example

In the following, we describe two artifact configurations, which serve as an illustrative example. One configuration, which allows to accomplish the business goal, and another one, which represents an invalid configuration.

The scenario is about loading a ship with goods, whereby the loading stage has two sub-stages; the first stage represents cargo checking, whereby the second stage represents the ramp transport. This behavior is modeled by a single Collaboration Artifact type called a Transport Execution Plan (TEP). This term is also used for illustration within the implementation and evaluation chapter. A TEP is a domain type, which represents all data and process information, required to provide a certain transport service. The required loading behavior is assigned as a Feature to this TEP, and thus determines the operational goal *Container loaded*. Figure 3.9 illustrates both configurations. The upper part represents the messages, which can be expected from the environment. Those messages are part of the available data of an artifact. The lower part represents the TEP Collaboration Artifact.
The invalid configuration thereby contains the loading feature only, with all the guards, stages, and milestones. The model checker would respond with an error in this case, since the first guard will never be true, which leads to the matter that the milestone *Container loaded* cannot be achieved. More precisely, there are three violations that hinder the artifact to accomplish its goal. First, the artifact does not contain any milestone which produces the *cargo stuffed* event. Second, there is no party registered which is expected to send a load list to the system. Third, the *Check Cargo* stage requires a data attribute *containerSize* in order to check if the cargo fits into a container. This attribute however is not represented in the loading feature itself. On the right side, an additional stuffing feature is bound to the TEP, and a message type containing a load list is expected. The overall milestone *Container loaded* thus can be achieved theoretically, since all four desired properties are satisfied.

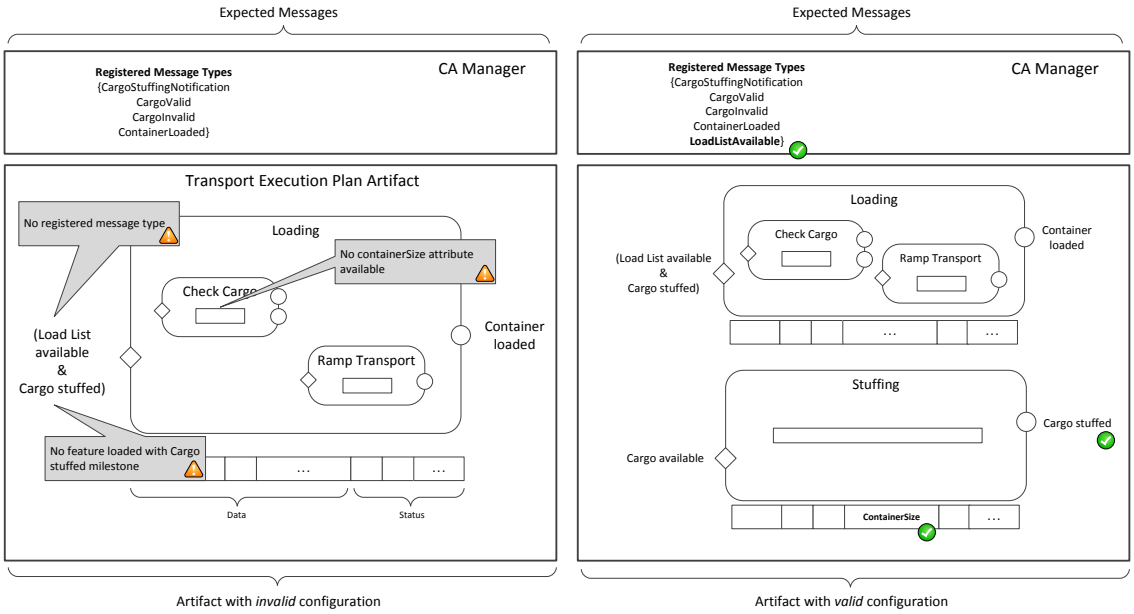

Figure 3.9: Invalid and Valid Artifact Configuration

Conclusion and Related Work

The general idea of transition systems fits to our needs from an abstract point of view. However, there are limitations of transition systems and limitation of model checking in general. Transition systems can effectively model control-flow or even data-dependent systems, however it is less suited for data-intensive applications as data typically ranges over infinite domains [BK08]. In order to handle the undecidability problem, the problem domain must be simplified to reduce the search space.
The authors of [WHM08] discuss related work of process verification beyond control-flow verification. This includes verification of access control, as well as verification techniques to assure structural and temporal correctness of inter-organizational workflows and event-driven systems. Exemplary approaches rely on semantic checks using pre-conditions, effects, and ontological axioms (related to service composition); or they rely on data-flow analysis using petri-nets with data-associated tokens.
There are several investigations which address the verification of Business Artifact systems. Temporal constraints are discussed [GS07], whereas the existence of deadlocks is addressed in [Bha+07b]. Although [BLB12] and [Deu+09] propose abstraction techniques to handle the complexity of verification, the authors conclude, that there are no current results which provide a solution to this problem. Nevertheless, these developments give an overview of related work, and provide a theoretical foundation for artifact verification.

Summarizing, automated verification of the configured process model is desirable, however it requires a formal specification of both, the model and the desired properties to check. Depending on what properties must be checked, and how complex the abstract artifact model is, this influences the decidability and complexity of verification techniques. There is a trade-off between the variability of artifact construction and the effort to assure its correctness. In order to prevent the system from errors, verification and validation however is indispensable.

3.3.3 Execution of Collaboration Artifacts

This section describes the runtime behavior of Collaboration Artifacts. The aim is to explain the relation between internal behavior and artifact interaction with the environment. The internal behavior is in setting attribute values and updating the internal information model. The behavior with the outside world is in making requests to external services and to send real-time notification on state changes to the environment. Both are triggered by incoming events and by associated tasks. The runtime behavior of Business Artifacts states the basis for describing the runtime behavior of Collaboration Artifacts.

The runtime behavior of a Collaboration Artifact representing data- and process information, is influenced by the interaction with its environment. The environment represents stakeholders involved into the process, as well as other Collaboration Artifacts. The interaction with the environment is based on message exchange. The life-cycle model consisting of stages is controlled by guards and milestones and by their corresponding conditions. These conditions contol if a stage gets opened or closed. When a stage becomes active, the associated tasks are executed. Atomic stages may contain one or more tasks of different types [Hul+11c], including: (a) assignment of attribute values; (b) invocation of (one-way or two-way) external services; (c) request to send response to an incoming two-way service call; (d) request to send an event (message) to one or more identified artifact instance(s); and (e) request to create a new artifact instance. The actual business process however is driven by the involved stakeholders. By inserting data into artifacts, the associated tasks are executed, and the artifact life-cycle evolves.
Figure 3.10 shows an example of an external service call, which is used to describe the interaction between guards, stages, milestones, and associated tasks / services. A loading feature, implemented by a *Loading Stage* is embedded into a Collaboration Artifact. Thus, the artifact consists of a life-cycle model and of a data model including stages and data attributes related to cargo loading. The data attributes for instance contain information about a load list, cargo details, destinations, or involved parties. The status attributes store the status history of the guards and milestones. Since this figure represents a snapshot of the artifact, extension and variation points are omitted.
The top-level stage with the operational goal to load the cargo consists of two sub-stages named *Cargo Check* and *Ramp Transport*. At the point of time represented by this snapshot, the *Loading* stage is already opened and the sub-stage *Check Cargo* is active. This means the corresponding guard is true, whereas both milestones are false. This atomic stage has a task associated, which is supposed to invoke an external service for checking the cargo. Once this stage is opened the service request is sent immediately and the artifact waits for a service response in order to reach the milestone and to close the stage. In this case, the response triggers the *Cargo Valid* milestone. At the same time, the *Cargo Valid* guard of the *Ramp Transport* stage is set true and the stage is opened. This procedure of sending and receiving messages, and executing tasks is continued until the global milestone *Cargo loaded* is achieved.

3.4 CONCEPTUAL ARCHITECTURE

In this section, we describe the conceptual design of the solution approach. This design represents the abstract architecture for a Collaboration Artifact management system.
Different standards for the visualization of conceptual architectures exist. In this section we choose the Technical Architecture Modeling (TAM) [SAP07] specification. One advantage of TAM is that it allows for describing a system with a distinction of the conceptual level and the design

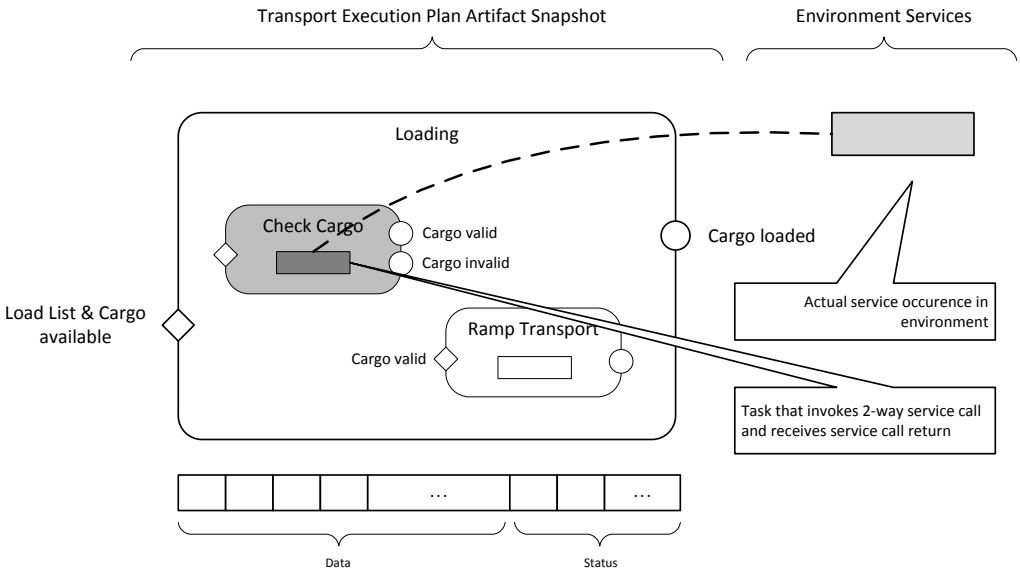

Figure 3.10: Two-way Service Call [Hul+11a]

level. It defines a set of diagram types and elements for visualizing the structures of an overall systems. The TAM block diagram is used subsequently to illustrate the design. It basically consists of three graphical elements: *Agents* are the only active elements, depicted by rectangular shapes, which process information. *Storages* on the other hand are passive elements, depicted by rounded boxes, which contain information, that can be processed by an agent. Access arcs indicate interactions between agents and storages, as well as the direction of information flow. *Channels*, visualized as a small circle on an connecting arc, represent connections between agents.
In Figure 3.11, the conceptual architecture is illustrated using the previously described design elements. The upper part thereby represents environment and security agents. The focus however is on the Collaboration Manager agent, which realizes the solution approach.

3.4.1 Environment and Security Agent

Since the Collaboration Artifact system is supposed to manage human-driven business processes, the environment agent represents all the involved stakeholders. The concern of security is crucial for the management of inter-organizational business processes. The security agent therefore controls the access to sensitive data, which is managed by the artifact system. This layer however is not subject for further investigation. Security mechanisms like data access control can be incorporated into a Collaboration Artifact itself to represent stakeholders of different roles and permissions. One example of those concepts are discussed in [HNN09]. Summarizing, it is a design decision in which layers or agents the security or user management concerns are realized.

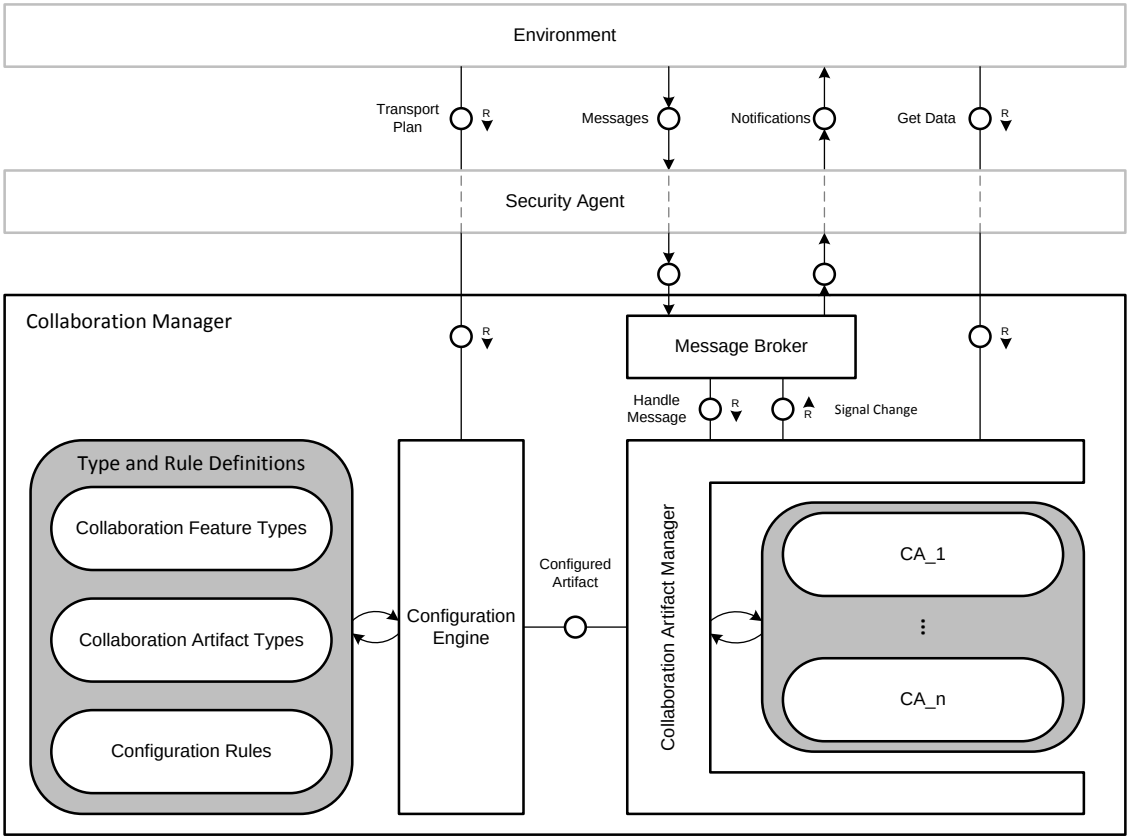

Figure 3.11: Conceptual Architecture

3.4.2 Collaboration Manager

The Collaboration Manager provides the entire infrastructure to manage inter-organizational business processes realized by Collaboration Artifacts. This includes message interfaces in order to send or receive messages and notifications, or interfaces for direct data access. The Collaboration Manager further provides agents for configuration and runtime management of Collaboration Artifacts. The responsibilities of included sub-components are outlined in the following.

Collaboration Artifact Manager

The Collaboration Artifact Manager is the a central component in the architecture. It fulfills the role of a runtime container for a set of CAs, which are intended to represent business related data and activities. In particular, the CA Manager is responsible for the internal management of Collaboration Artifacts. In addition to that, incoming messages are processed and forwarded to the corresponding artifact, and outgoing notifications are delegated to the Message Broker. Besides messaging and management functionalities, the CA Manager provides an interface for direct data access, which is required for create-read-update-delete (CRUD) operations. Another relation exists between the CA Manager and a Configuration Engine. The CA Manager interacts with this engine in two directions. On the one hand, the Configuration Engine deploys configured artifacts into the runtime container, and on the other hand, the CA Manager itself can request the Configuration Engine for dynamic artifact (re-)configuration.

Configuration Engine

The Configuration Engine is responsible for the creation, and for the configuration of Collaboration Artifacts. For this purpose, it has access to artifact type definitions, and feature definitions. These types in addition with a set of configuration rules are the requirements for correct artifact configuration and are managed by a storage agent. The configuration typically is triggered by an incoming process description, which for instance can represent a transport plan. At first, this transport plan is analyzed to identify the required artifact types and features. In a second step, the combination of artifacts and features is established under consideration of the given configuration and verification rules. Finally the configured Collaboration Artifact is deployed to the Collaboration Artifact Manager.

Message Broker

The Message Broker is an intermediate agent, which is responsible for message-based interaction between the environment and the Collaboration Artifact Manager. Incoming messages are directly delegated to the CA Manager, whereby outgoing notifications are sent to the corresponding stakeholders. The Message Broker further is responsible for message transformations in case the incoming message format cannot be processed by the Artifact Manager.

3.5 SUMMARY

This chapter introduces the concepts that are required to enable efficient, and flexible management of dynamic business processes. Starting with an abstract solution approach, the concept of a modular artifact design is described, followed by a detailed introduction of Configurable Collaboration Artifacts. The concept is based on the idea of explicitly defined extension- and variation points for configuration. The cyclic configuration process is illustrated and related concerns of verification and constraint checking are discussed. This chapter finally introduces a conceptual architecture, which is capable to control the configuration and the execution of a Collaboration Artifact system.

4 PROTOTYPICAL IMPLEMENTATION

This chapter presents the prototypical implementation of the solution approach, and thus describes the realization of the conceptual design introduced in Chapter 3. The prototype thereby serves as a proof-of-concept implementation to demonstrate the technical feasibility. While describing the single parts of the implementation, we reason why certain implementation decisions are made.
At first, the overall technical architecture of the Collaboration Manager is presented. Subsequent to this, we discuss the realization of sub-components, and describe how the concepts introduced in the conceptual design are implemented. The focus thereby is on the realization of Collaboration Artifacts, and on related aspects for configuration and runtime interaction. Finally, we summarize the results of this section.

4.1 TECHNICAL ARCHITECTURE

Figure 4.1 presents the technical architecture of the implementation. As already introduced in Section 3.4 we use the TAM standard to represent technical design elements. In order to describe a technical design, the *component* is the central functional element. Following [SAP07], a component is an implementation-close, active part of the software system, which is embedded into an environment using well-defined interfaces. A component is either indicated by the same-named stereotype, or/and by a decent symbol in the upper right corner. Components can be nested and do interact with its environment using well-defined interfaces. Provided interfaces represent certain functionality which is offered to other components. This type of interface is depicted using a lollipop notation. Required interfaces on the other hand are represented as sockets and describe certain functionality or services that are required in order to realize the components' functionality.
The focus thus is on the explicit representation of functional interfaces and on the description of component relations. In many cases, a technical architecture diagram also includes more technical details about interfaces or about data formats. Those details however are not included in our diagram for the following reasons. On the one hand, all interfaces corresponding to the Collaboration Management component are realized by Java interfaces and method declarations. The

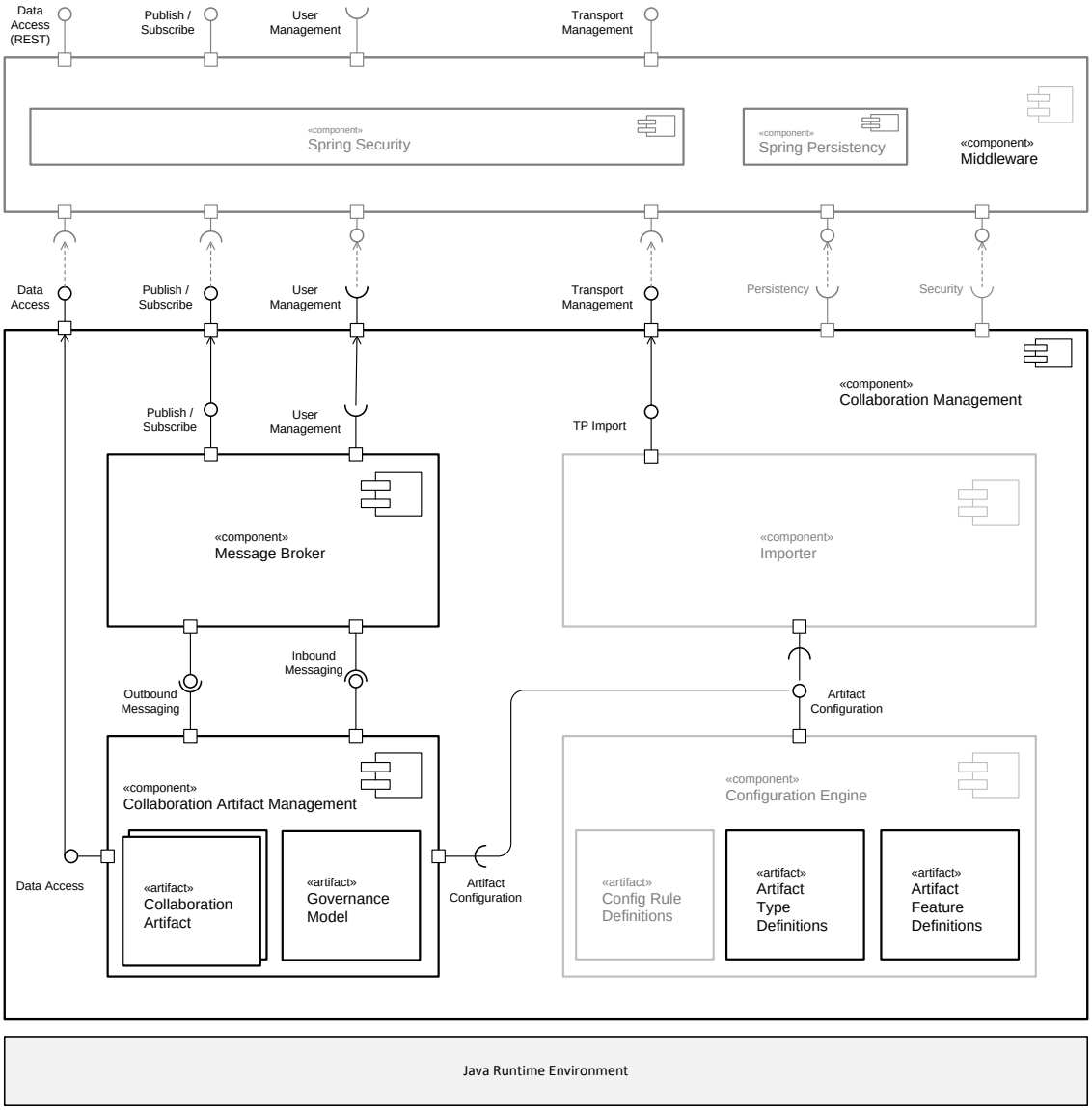

Figure 4.1: Technical Architecture of the Collaboration Manager

same applies to the format of exchanged data, which is represented by plain Java objects. On the other hand, the technical realization of interfaces corresponding to the middle-ware component are not of interest for the prototypical implementation. Therefore, we abstract from more specific technologies that might be subject for a possible system integration.

The Collaboration Management component is the main component of the technical architecture, since it offers all functionality to support the management of collaborative business processes by using Collaboration Artifacts. One provided interface is called Transport Management. Transport Management means the general ability to import a transport plan, and to manage the corresponding business process. Another provided interface is called Publish/Subscribe, which describes the ability of subscribing external components or involved stakeholders for certain notifications. The third interface provided by the Collaboration Management offers direct data access to Collaboration Artifacts. Typically, this includes create-read-update-delete (CRUD) operations. All these interfaces describe functionality, which is realized by the collaboration of sub-components. The Importer for instance is responsible to analyze an incoming transport plan and to trigger artifact

configuration. The Configuration Engine is in charge to build appropriate Collaboration Artifacts. This is achieved by artifact and feature instantiation, and by the subsequent configuration with regard to given configuration rules. The Artifact Management component is responsible for the runtime management of Collaboration Artifact instances. This includes the following tasks: First, the Artifact Manager tracks the state of all artifacts, and triggers creation, removal, and configuration of them. Second, it is responsible for the coordination of inter-artifact communication using a governance model of artifact relations. Third, it must handle incoming as well as outgoing messages or notifications and therefore is directly connected to the Message Broker component. The Message Broker provides messaging in both directions; incoming messages from middle-ware components to artifacts, and outgoing message from artifacts to external components.

The core component for collaboration management is supposed to integrate into a complex component architecture, realized with *Spring*[1]. Spring is an application development framework for Enterprise Java, based on dependency injection. Dependency injection is a software design paradigm, which allows a selection among multiple implementations of a given dependency interface at runtime instead of at compile time. Besides that, Spring provides a large set of features and projects such as support for persistence or security, which simplifies Java development [Wal11]. This technology stack including Java and Spring, finally is the reason for using Java as a programming language within the prototypical implementation. The relation to external components is illustrated by a middle-ware. The implementation of this middle-ware component however is not part of the prototype. Nevertheless, it indicates support for security and persistence, which is required by the Collaboration Management component. In the following we describe those components which are directly involved into the runtime management of Collaboration Artifacts. The light-grey parts are out of scope of the prototype implementation.

4.2 IMPLEMENTATION OF COLLABORATION ARTIFACTS

This section describes the implementation of Configurable Collaboration Artifacts. This includes those concepts presented in Chapter 3, which are in the scope of the prototype implementation. The first part describes the implementation of general artifact aspects, including data-model, life-cycle, and tasks. In a second step we discuss the implemented support for variability of artifacts and features. The third section explains the implementation for artifact management and interaction.

4.2.1 Realization of GSM Concepts

The realization of Collaboration Artifacts is based on the GSM meta-model. We decided to implement a Collaboration Artifact as a Java class, since it is the blueprint from which individual artifacts are created. The instantiated object then holds all information for this domain entity with possible *null* values after instantiation. It evolves at runtime until it reaches a specified operational business goal. Following the GSM meta-model, each artifact has a data-model, a life-cycle, and associated tasks.

The life-cycle model based on stages is described by the abstract *Stage* class, containing guards and milestones. The relation between stages, guards and milestones is shown in Figure 4.2 Each stage refers to the containing Collaboration Artifact and holds a set of data attributes for its

[1] `http://www.springsource.org/`

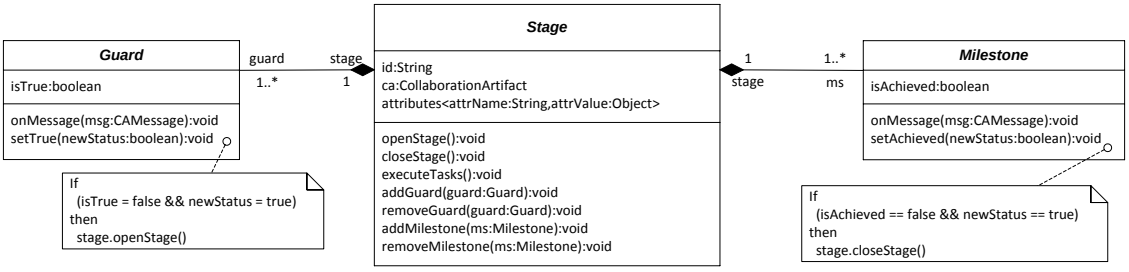

Figure 4.2: Guard-Stage-Milestone Class Diagram

information model. Furthermore, each stage has a method called *executeTasks()*. Sub-classes are free to overwrite this method with specific behavior. This method represents the actual behavior of this stage. Guards and milestones act as the pre- and post-conditions of a stage. They represent a condition in form of boolean attributes that are combined by logical operators to a logical expression. If a message arrives at a guard or milestone, the logical expression is calculated. The corresponding stage is opened if the guard switches from false to true. As soon as the stage is opened, the implemented tasks are executed. After executing the tasks, the artifact waits for subsequent messages to close the stage. Another way would be to include pre- and post-conditions directly within the stage, surrounding executed tasks. However, the implementation of guards and milestones as separate classes has the advantage, that the objects can be reused and be flexible assigned to different stages. Furthermore, an encapsulation of the conditions into single classes provide an increased control over their state. The data model of an Collaboration Artifact is represented by a set of data attributes.

4.2.2 Runtime Extension and Variation

This section explains how the mechanisms for the configuration of Collaboration Artifacts are realized. The first section describes the implementation of extension capabilities for artifacts. Subsequent to this, we discuss possible implementation approaches to realize feature variation. In the last part, we finally describe how the configuration process is accomplished by the prototype implementation.

Extensibility

Extensibility refers to the ability to extend a Collaboration Artifact with functionality. The artifact itself serves as a template declaring an explicit extension point to which one or more Features are bound at runtime. In object-oriented systems, classes can be extended with new features, attributes and methods, which is known as inheritance. According to that, a Collaboration Artifact *A'*, which results from an extension step of an artifact *A* and a Feature *F* includes properties of both of them. This sub-classing however has some drawbacks, which are explained in the following.
First, Java does not support multiple inheritance in terms of *Artifact extends FeatureA, FeatureB*. However, this is an essential requirement for the realization of the concept. The programming concept of *mixins* provides the ability of collecting functionality rather than traditional sub-classing by inheritance. [Am03] defines a mixin as a class (i.e., a set of features) by which a superclass can be extended to derive a subclass. During the extension step, one or more mixins are then integrated into the superclass to form the subclass. However, such mixin-based inheritance is

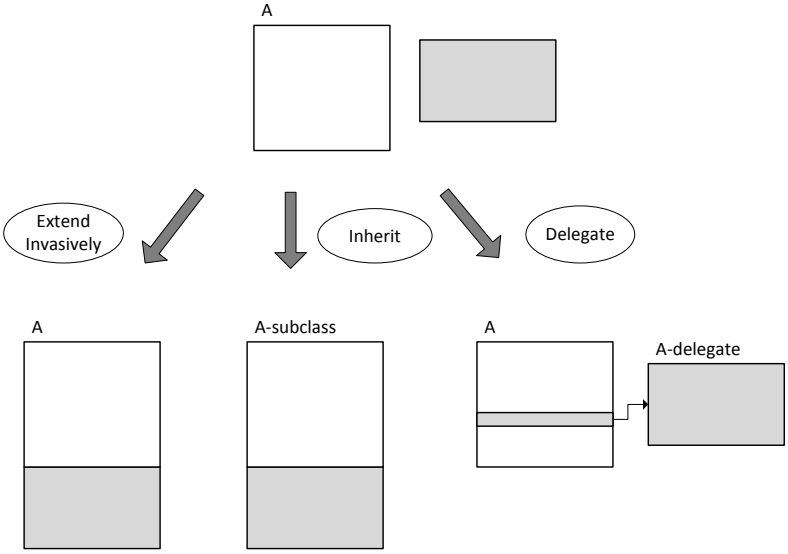

Figure 4.3: Comparing invasive extension, inheritance, and delegation. [Am03]

not supported in Java neither.
Another limitation of any kind of inheritance is that it leads to a new class type. However, we do not want to change the type while extending the capabilities. Ideally, we want to extend a base class invasively at runtime. This means, the class should be adapted or extended by a transformation while keeping its class type. For realizing those concepts, the book named "Invasive Software Composition" [Am03] provides more details. This distinction between invasive extension and sub-classing is visualized in Figure 4.3.
The most crucial drawback of using inheritance for our implementation is the fact that inheritance happens at design time. First, for the kind of flexibility required in our setting this is sufficient, and second, in order to support an artifact hierarchy which is capable of reflecting all possible combinations, this would end in a class explosion.

If a technique is not supported by a programming language, it must be simulated by a workaround. We decided to use delegation as a workaround to extend the functionality of the artifact type, although it introduces indirections, which might deteriorate runtime efficiency [Am03]. We consider this as adequate for the prototype, because it prevents from class explosion and allows dynamic extension. The extension point of a Collaboration Artifact is a typed set which stores the bound Features. A Feature is implemented by a class and represents a code fragment with certain functionality. Events that arrive at an artifact are then delegated to the referenced features. This delegation however has the extra effort of bypassing. One possibility is to use the *Decorator* design pattern [Gam+95]. The pattern allows to add functionality to an object at runtime, without extending its type. One restriction is that you have to use the same interface for the decorated object and for the decorator. However, such restriction sometimes is not useful, since we want to distinguish between a Collaboration Artifact and its Features. Both should contain different methods and capabilities. Therefore, we decided to implement the relation as shown in Figure 4.4. This relations reminds on the *Bridge* design pattern described in [Gam+95]. It allows a flexible binding between an abstraction and its implementations. Flexibility thereby is achieved via indirection, which results in two class hierarchies that can be varied independently. The client only sees the abstract Collaboration Artifact, whereby the concrete Features can be added or removed transparently at runtime. The concrete behavior of a Collaboration Artifact is then determined by the bound Features.

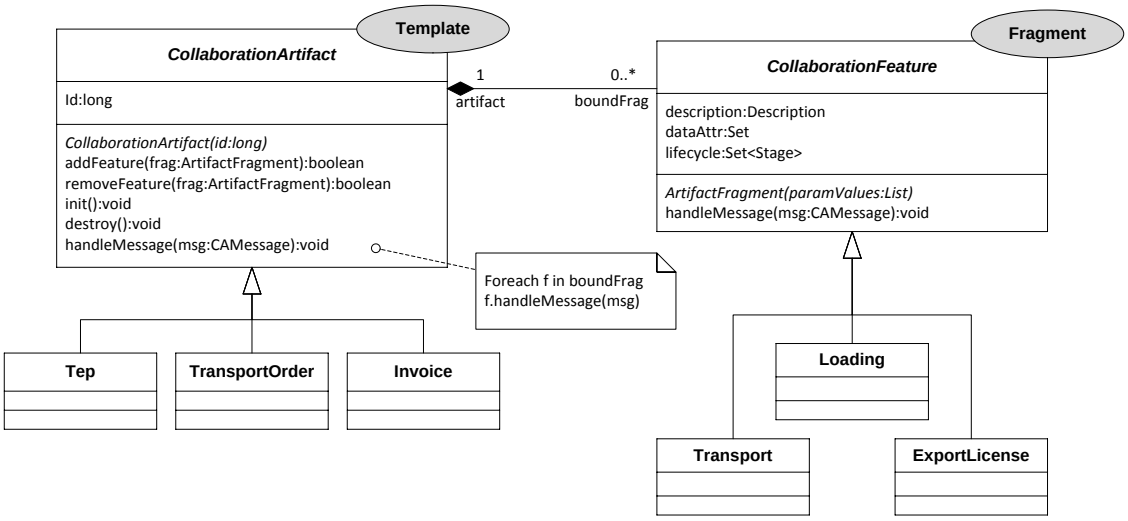

Figure 4.4: Artifact - Feature Relation Class Diagram

Variability

This section outlines possible realization strategies for supporting feature variation. In general, all concepts are based on the idea of under-specification to provide flexibility. Under-specification means that a software fragment is only partially specified. The decision of how exactly these undefined parts are realized is postponed until deployment or execution. These variation points are then bound with a concrete value or algorithms. In the following we discuss candidate solutions approaches.

One possible solution is to use*generic programming*. A component with generic types and identifiers is variable in binding concrete typed values to these generic elements. This allows the parametrization of a component in a fine-grained way. The Java programming language supports generics. However, in case of Java they are only checked at compile time for type correctness. The use of generics thus are a possibility to insert a certain degree of adaptation, where dynamic variability is not required.
Another option to support variability is to make use of appropriate design patterns. The *Strategy* design pattern is one example of variation patterns described in [Pre95]. This pattern is based on the idea of defining a family of interchangeable algorithms, which can be assigned at runtime to a fragment. Applied to the concept of variable Collaboration Features this means a definition of interchangeable feature processes. This pattern thus allows to change the sequence of loading activities within a loading feature, or to change the strategy by which a certain guard activates a stage. For the client object, these changes are transparent, because the initial method declaration remains the same and the method call is delegated to the chosen strategy implementation.

We consider the Strategy pattern as most suitable for realizing the concept. The pattern is based on delegation and therefore allows variability at runtime. The prototype allows to vary Artifact Features by exchanging specific guards or milestones of a stage. The decision about what kind of guard or milestone implementation is used influences the pre- or post-conditions of a stage and thus changes the strategy of when a task is executed.

Artifact Configuration

In this section, we outline how the configuration of Artifacts and Features is realized by the prototypical implementation.
As described in Section 3.3.1, the configuration process consists of numerous configuration steps, which represents the binding or removal of one parametrized Feature to, or from a Collaboration Artifact. The prototype does not provide a configuration engine. This has two reasons. First, a configuration engine requires a configuration program or configuration rules formulated in a formal language. These formal descriptions of valid and invalid configurations are required to automatically realize the requested composition. Second, an automated artifact configuration is followed by a verification process. Verification concerns are discussed in Chapter 3, however these aspects are out of scope of the thesis.
In the prototype implementation, instantiation and parametrization tasks are delegated to a feature builder, which is aware of the pre-defined feature types. If the corresponding method is called, the feature is constructed by initializing and referencing stages, guards, and milestones. The result is a *CollaborationFeature* object which can be assigned to a Collaboration Artifact. The initial set of artifacts for a given transport plan is constructed by the Artifact Management component in the first place. In a next step, the feature builder is called to extend these artifact templates with a set of feature objects.

4.2.3 Artifact Management and Interaction

This section describes the technical realization of Collaboration Artifact runtime management. First, we explain how the messaging functionality is realized by involved management agents. In the second part, we describe the implementation of the Collaboration Artifact Manager, and provide an example of a governance model implementation for transport scenarios.

Messaging

In this section, we describe the realization of Collaboration Artifact messaging. Therefore, we explain the structure of artifact messages, and describe the components that are involved into the messaging process. Finally, a short scenario illustrates how the involved components interact with each other.

Collaboration Artifacts communicate by exchanging messages, which are coordinated by a central component, the Collaboration Artifact Manager. Such design decreases the coupling of components and allows the integration into a message-oriented middle-ware. A Collaboration Artifact Message (CAMessage) is the object which is transferred between artifacts and is structured as follows. A header contains sender and target information as well as a timestamp, whereas the actual content is included in the payload. The payload provides information about the message type, which defines the kind of the transmitted event or notification. The actual content is implemented by a set of key-value pairs.
Figure 4.5 shows an example of a message-based interaction cycle, which starts with the arrival of an external message, and finishes with the generated notification event for the subscribed stakeholders. The involved agents are: the Message Broker, the Artifact Manager, the Message Handler, and finally a Collaboration Artifact.
The Message Broker is responsible for forwarding incoming messages to the Artifact Manager, and outgoing messages to subscribed stakeholders or components. It therefore takes the role of

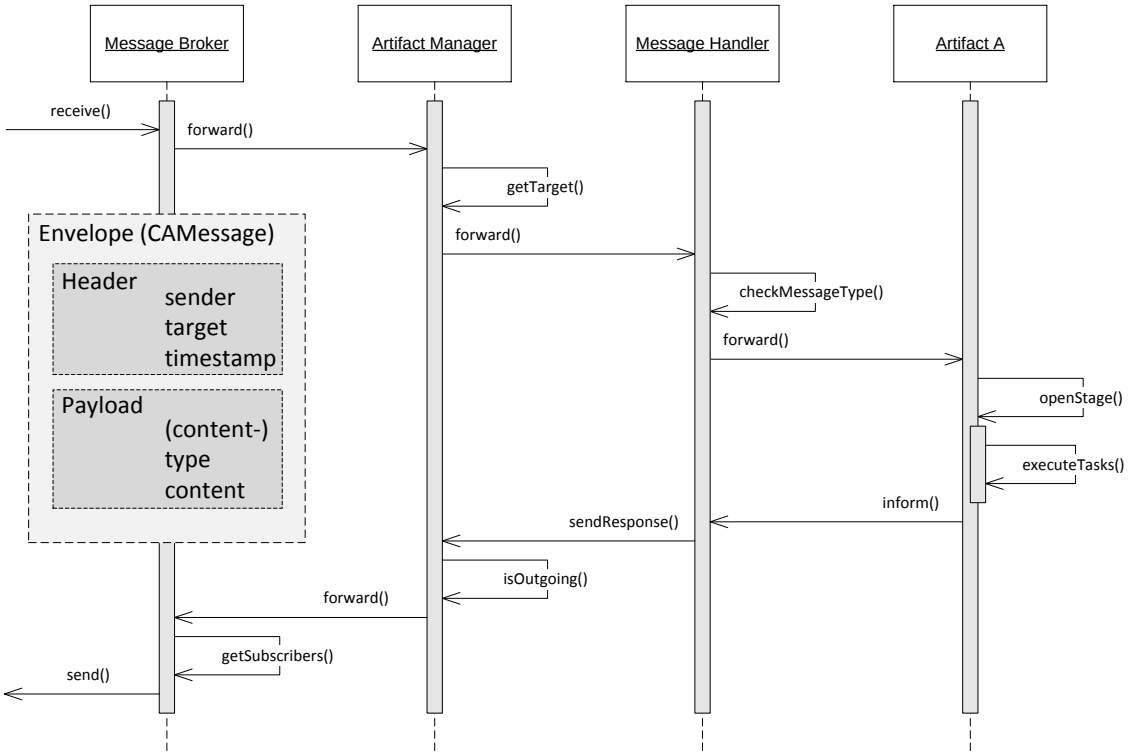

Figure 4.5: Message Interaction

a message gateway of the Collaboration Management component. Message subscriptions are stored as key-value pairs inside a Java Map. The Artifact Manager is responsible for inter-artifact coordination. It stores the identifiers of registered artifacts and thus is able to delegate an incoming message to the requested target artifact. For outgoing messages, the Artifact Manager checks whether the registry contains the target identifier, or if the message is supposed to be forwarded to the Message Broker. If an internal, inter-artifact message does not specify a target-id, the Message Broker uses its governance model to find possibly affected artifacts.
The Message Handler is a proxy of a Collaboration Artifact in order to encapsulate messaging functionality from the actual artifact logic. The main task of the Message Handler is to delegate artifact messages to the corresponding guards or milestones. Therefore, a Java Map stores for each possible message type the number of recipients, which are able to process this kind of message. Finally, the message is processed by the guards and milestones of the artifact and triggers the (de-)activation of its stages.

Artifact Management

The runtime management of Collaboration Artifacts is the task of the Collaboration Artifact Management component. As already described in Section 4.1, this basically includes the following tasks: first, creation and removal of Collaboration Artifacts, as well as the awareness of their actual state; second, the cooperation with the Message Broker to process incoming and outgoing messages; and third, the support of inter-artifact communication. This section describes how these tasks are realized in the implementation.

The management of artifacts is established by an artifact registry, consisting of key-value pairs, whereby the key indicates the artifact-id, and the value stores the reference to the corresponding

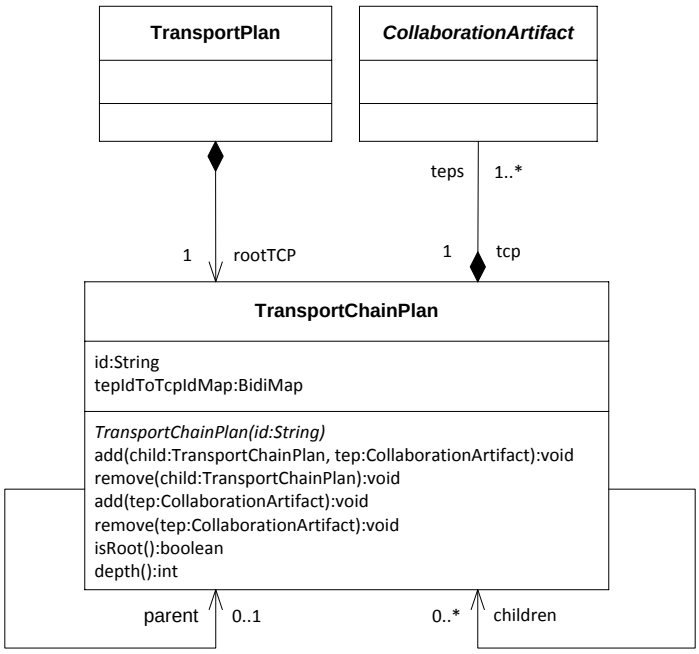

Figure 4.6: Transport Chain Plan Class Diagram

object. Registration or unregistration of artifacts to/from the artifact system thus is accomplished by adding/removing entries to/from the registry map. The artifact object then allows to get an insight into the current state of the artifact, combined by the state of all included milestones.
Inter-artifact communication is mediated by the Artifact Manager using a governance model of the business process. This model serves as a management structure to ensure efficient message re-direction among artifacts. This necessity is described in the following by example of a hypothetical transport plan. The transport plan is represented by a hierarchical structure of plan segments, whereby each segment is called a *Transport Execution Plan* (TEP). Certain TEPs are assigned to one common *Transport Chain Plan* (TCP) to describe the shipment of goods along these transport plan segments. The possibility of sub-contracting for transport realization, requires a parent-child relationship among those TCPs. Finally, the transport plan is described as a tree-structure, with TCPs representing the nodes. For the moment, it is not required to know how exactly a single TEP is modeled. This issue is taken up again within the evaluation chapter.
Lets assume, a TEP is implemented as a Collaboration Artifact. By importing a transport plan, TEP artifacts are instantiated, configured, and finally deployed. Since Collaboration Artifacts are supposed to be autonomous, loosely-coupled elements, they are neither aware of other artifacts, nor do they have any reference to them. This does not hold for each case, however such design decreases the coupling and thus increases flexibility.
The tree-structure of TEP artifacts is preserved in the governance model. This structure can be used to determine the artifact instance, which is affected by a particular event. The governance model of a TEP-based transport plan is shown in Figure 4.6. Each TCP contains one or more TEP Collaboration Artifacts, whereas an artifact belongs to exactly one TCP. The tree-structure is implemented by references on the parent or child TCPs, because it must be explicitly defined which TEP "sub-contracts" a child-TCP. To enable the Artifact Manager to walk through the tree structure in both directions, these connection points where a sub-TCP is hooked into is stored by a bidirectional map. The overall transport plan is then represented in the governance model by only the root TCP.

In order to process incoming or outgoing messages, the CA Manager must collaborate with the Message Broker. An incoming message received from the middle-ware is processed by the CA Manager. In case this message requests the creation of an artifact, the CA Manager instantiates the corresponding artifact type and calls the feature builder to extend the template with the desired functionality, which is encoded in the message. Finally, the resulting object is added to the artifact registry. The same holds for the removal of a running artifact. Messages which are sent by a Collaboration Artifact, are either re-directed to the target artifact (internal messaging), or they are forwarded to the Message Broker, if it is a notification without an artifact target-id. In the latter case, the Message Broker makes a look-up for subscribed components or stakeholders, which are interested in this notification (external messaging).

4.3 SUMMARY

This chapter describes the prototypical implementation of the concepts, which are introduced in Chapter 3. The aim of the prototype is to proof the technical feasibility of the conception of Configurable Collaboration Artifacts. The chapter is separated into two sections. In the first part, the technical architecture is described, which is required for Collaboration Artifact management. This includes the artifact creation, configuration, and inter-artifact communication. A Collaboration Management component as the main solution component describes all provided, and required interfaces to the middle-ware. The realization of the sub-components including the Message Broker, and the Collaboration Artifact Manager, are described in separate sections. Further, this chapter explains how extension and variation concerns are realized. At the end, the implementation concepts for artifact interaction management are described. This helps the reader to understand the runtime behavior of the system.

5 EVALUATION

This section investigates the solution approach, which is elaborated within the preceding sections. The evaluation consists of two parts; first, a qualitative, scenario-based evaluation, and second, a quantitative evaluation. As a first step, the identified requirements are recapitulated, and the evaluation strategy is described. Subsequent to this, the evaluation is used to discuss if and to what extend the solution approach satisfies the required properties. A quantitative evaluation briefly addresses performance and scalability aspects, before the main findings and limitations are summarized.

5.1 AIM AND METHODOLOGY

The aim of the evaluation is to verify if the identified requirements are satisfied by the proposed solution. The requirements named global view, scalability, and flexibility are defined in Chapter 2. First, we outline the central idea of each of those requirements. Second, we describe the modeling process, which is applied for artifact and feature discovery. This process is applied during the use-case realization within the evaluation. Finally, we describe the strategy, which is chosen for the evaluation.

The global view requirement characterizes the ability of a model to support an end-to-end process view on collaborative tasks. Thereby, it highly depends on the surrounding system whether the it is able to imitate a global view by gathering all information from different sources. The validation however is interested in the question if the artifact-centric modeling approach provides a global process view *by design*. The requirement for scalability is about the ability of a system to handle a growing amount of work in a capable manner. So called functional scalability allows system evolution by adding new functionality with moderate effort. Load scalability on the other hand is about the ability to enable concurrent, distributed computing in order to handle increasing complexity. Finally, the requirement of flexibility addresses the capability of adapting a design or a running system to external changes. In the qualitative evaluation we discuss how the solution approach provides different types of flexibility.

The modeling process of Collaboration Artifacts requires an in-depth domain analysis in the first place. Domain expert feedback is helpful to define the overall business goal, and to choose an

1. *Artifact Type Discovery*: Discover critical Collaboration Artifact types and their operational goal(s).
   (a) Staging: Formulate Artifact sub-goals to identify top-level Stages. Define data attributes related to each Stage.
   (b) Service Discovery: Identify internal/external tasks for atomic Stages.
   (c) Refinement: Define Guards and Milestones with event-condition-action rules.
2. *Feature Type Discovery:* Identify behavior and operational goals, which are supposed to be reused in different Artifacts.
   (a) Staging: Formulate Stages that are part of the Feature life-cycle.
   (b) Service Discovery: Identify internal/external tasks, which are related to the Feature.
   (c) Refinement: Define Guards and Milestones with event-condition-action rules. Define points of variation.
3. *Dependency Definition:* Set up configuration rules and dependencies between Feature types.

Figure 5.1: Applied Process for Collaboration Artifact Modeling

appropriate granularity of artifacts and features. Based on a data-centric perspective, the identified types of artifacts are stepwise refined and extended with life-cycle information and runtime constraints. In Figure 5.1 we outline the modeling process, which is used in the evaluation to define model Collaboration Artifacts and its Features. This process is considered as one possible strategy to come to a Collaboration Artifact model for a concrete scenario. However, this does not state a prescribed methodology for Collaboration Artifact modeling. In fact, it is based on an abstract methodology for Business Artifact modeling presented in [BHS09]. The first step aims at a high-level specification of business operations by defining key artifacts and their most important life-cycle stages. Typically, this is done in the context of a top-down analysis by considering user stories and exceptional use-cases. Key domain entities and their top-level stages are identified to represent the entity life-cycle and the operational business goal. Subsequent to this, tasks, which are supposed to represent the artifact behavior, are associated to atomic stages. Finally, dependencies and their resulting constraints are designed at the level of guards and milestones. This life-cycle model and the corresponding data model build the skeleton for the artifact type, which is common for all instances of this type. The goal of step 2 is to identify artifact features as the reusable fragments that extend the capabilities of artifacts. Each feature thus has its own data and life-cycle model, and encapsulates functionality reused among different artifact instances. Since features are rather generic and adaptable fragments, the second step finally requires the explicit selection of feature variation points. The third step of the modeling process aims at defining the relationships among features and artifacts, which mainly consists of a dependency analysis for later consistency checking and verification procedures.

In order to verify the conceptual solution, a *scenario-based evaluation* [Loo+08] is used in the following section. Thereby, non-functional requirements representing business goals are formulated as *scenarios* to evaluate non-functional properties of the model. The identified requirements of visibility, flexibility, and on scalability are considered as non-functional requirements [GS05]. These requirements are architectural drivers and indicate how the system is supposed to be. Scalability for instance is a rather technical term; nevertheless, it is a non-functional requirement.

The evaluation then relies on a discussion about the ways in which the system may be expected to scale up and in which ways the solution approach supports such scaling.
In our case, two real-world scenarios provide the general setting for the evaluation. More concrete, but hypothetical use-cases describe particular situations, which are commonly encountered among stakeholders in this scenario. These cases serve as test-cases to demonstrate how the solution is applied to the scenario, and to assess if the system satisfies the corresponding requirement. The evaluation process consists of three steps: first, a real world scenario and a related set of use-cases are defined; in a second step, the solution approach is applied to the previously defined use-case scenario; finally, for each use-case it is evaluated if and to what extend the solution requirements are satisfied by the applied approach.
Apart from describing how the use-cases are realized within the second evaluation step, it is emphasized in which way expert feedback influenced the applied solution design. This is regarded as part of the evaluation since best practices and years of experience of domain experts constitute a certain degree of usability. In order to validate the realization of concrete scenarios, expert reviews are an appropriate way, since an interactive prototype is not required. Beyond this, it presents an alternative to empirical usability tests, which is helpful to make assumptions if the solution fits the needs of the domain.

The requirements which are subject for the subsequent evaluation are declared as a global process view, flexibility, and scalability. The strategy which is chosen for evaluation is called scenario-based evaluation, which is considered as a appropriate technique for the evaluation of non-functional requirements. The process of applying the solution approach to concrete scenarios, consists of different steps of identifying and refining Collaboration Artifacts. These steps are applied to each use-case scenario in the following.

5.2 SCENARIO-BASED QUALITATIVE EVALUATION

In this section, the previously described evaluation strategy is applied to two simplified, but concrete, real-world scenarios; both with a different purpose. The first scenario constitutes the proof-of-concept. The aim is to evaluate the requirements of a global process view and scalability. In contrast to that, the second scenario evaluates the requirements of flexibility and re-planning. In a first step, we describe the scenario and define concrete use-cases that are subject for evaluation. Subsequent to this, we demonstrate how the solution approach is applied to this scenario and thus demonstrate Collaboration Artifact modeling. In the third section, we describe how the realization satisfies the requirements corresponding to the use-cases. Finally, we summarize the results by referring to the initially defined requirements.

5.2.1 Scenario I - Fish transport

The first scenario describes the export of fish from Norway to Europe with three roles involved: a shipping line, the port control, and the container terminal. This scenario is defined in the FInest Use-Case Description Deliverable [Ria12]. Before we describe the transport process more in detail, we briefly explain the idea of a TEP and a Transport Chain Plan (TCP). A TEP is an information model that holds all data related to the execution of a transport service, whereby at least a Logistics Service Client (LSC) and a Logistics Service Provider (LSP) are involved. A TCP represents a logistics chain, which is represented by a collection of TEPs. Although there exists an implicit sequence of steps, the contained TEPs can be executed concurrently. The

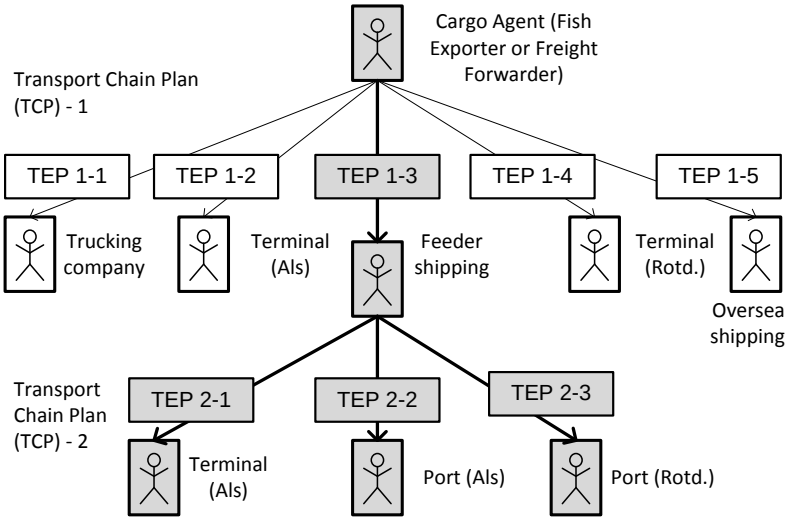

Figure 5.2: Fish Export Transport Plan

TEP information model is specified in context of the e-Freight research project [BMT10], which supports the development of a standardized framework for real-time freight information exchange covering various transport means. In the following, we describe the scenario in detail and we define use-cases for the evaluation of the properties of global view and scalability. Subsequent to this, we explain how the solution approach is applied to this scenario, and how the solution realizes the defined use-cases. Finally, we argue if and to what extend the requirements are satisfied.

Scenario Description and Use-Cases

This section describes the first evaluation scenario shown in Figure 5.2. The transport process is represented by a hierarchical transport plan consisting of TEPs to illustrate the relations between the involved stakeholders. Each TEP thereby realizes one part of the overall transport process. Subsequently, we describe this transport plan and the corresponding use-cases.

The top-level agent at the root of the tree organizes the shipments in order to get the goods from the manufacturer to the final point of distribution. This cargo agent, or freight forwarder, is an expert in supply chain management. It is his/her responsibility to book carrier types for road and sea transport. The upper layer of TEPs describes the service agreements between the cargo agent and service providers. In a first step, the fish is delivered by a trucking company at the Port of Ålesund and stuffed into containers at the terminal. After finishing the subsequent sea transport, the cargo is unloaded at the terminal in Rotterdam in order to prepare the fish for oversea shipping.
In some cases, a service provider sub-contracts further third-party service providers. An example provides TEP1-3, which realizes the feedering by booking services that are realized by different TEPs. Feedering corresponds to the transshipment of goods or containers to an intermediate destination to combine small shipments into large oversea shipments. Thus, a feeder shipping line does the shipment from Ålesund to Rotterdam, where the cargo is transferred into bigger deep sea vessels. In the following, we focus on this feedering part. The first step of the feedering is to load the containers onto the ship at the terminal. In a second step, the port call is handled by the port control in Ålesund, in order to organize the ship departure. Finally, another port call is required to handle the ship arrival at Rotterdam, before the feedering phase is considered as

ID	Attribute	Description
Gv01	Actor	Feeder Agent
	Story	The feeder agent requires detailed information of its corresponding transport leg. This includes the actual state, what parts are finished, as well as all the data which is required for the specific task.
	Conditions	None
	Expected Behavior	The system responds that the feeding ship has already started, however does not yet arrived. The coordinates are provided and the way-bill is presented.
Gv02	Actor	Feeder Agent
	Story	The feeder Agent wants to know the status of the goods while these are being transported by the truck in TEP1-1.
	Conditions	The feeder agent has the access right to get this information.
	Expected Behavior	The feeder agent gets the good details, information about the trucker company, the location of the truck, and the information that the transport is in time.

Table 5.1: Use Cases for Global View (REQ1.1)

completed.
The challenges and deviation that arise in this scenario are summarized in [Ria12] for each of the involved roles. Common problems are the availability of information, and changes in booking which require re-planning. Based on this scenario, we define the following use-cases that are subject of validation. Table 5.1 contains selected use-cases related to the global view requirement. Each use-case has an identifier, an involved actor, as well as a description of the setting. Table 5.2 describes the use-cases for both types of scalability. The first use-case addresses functional scalability, whereby the second one describes a setting for a discussion on load scalability.

Realization

This section demonstrates the application of Collaboration Artifacts to the first evaluation scenario. We describe the design process on basis of Collaboration Artifacts for the fish export scenario.

A TEP is considered as the key domain artifact in this scenario, because it represents the explicit knowledge of one part of the process. Once instantiated, the TEP CA evolves, and passes several stages on its way to the operational business goal. TEP2-1 describes the loading at the terminal in Ålesund. TEP2-2 in contrast, is supposed to handle a port call. Although, this TEP has some data types in common with TEP2-1, it differs in the corresponding port call behavior. These differences in the required transport services and data attributes are discovered among all TEPs. As a consequence, the Collaboration Artifact model is defined by one central TEP artifact. This TEP Collaboration Artifact serves as the Artifact Template including those data and life-cycle attributes that are common among all possible instances. The resulting Collaboration Artifact Template is shown in Figure 5.3. It consists of a stage for initialization and a stage for finalizing the TEP. Since these default stages are part of the artifact skeleton, they are available for all instances.

ID	Attribute	Description
Sc01	Actor	Business / IT Consultant
	Story	New functionality should be integrated into the system. TEPs must support Warehousing. The functional capacity is extended. (Functional Scalability).
	Conditions	A process and data-model is already available.
	Expected Behavior	The warehousing model is implemented as a Feature of a TEP. The system must not be changed.
Sc02	Actor	Business / IT Consultant
	Story	It is decided to increase the performance by providing more resources. (Load Scalability).
	Conditions	The implementation provides parallel execution. Transaction conflicts are checked.
	Expected Behavior	The execution is distributed on several machines. The performance increases.

Table 5.2: Use Cases for Scalability (REQ3.1)

The differences between TEP instances are in the implemented services. Whereby TEP1-3 realizes the feedering transport, TEP2-1 is responsible for the loading, whereas TEP2-2 handles a port call. The e-Freight modeling framework therefore provides a list of service codes, identifying services such as loading/unloading, transport, stuffing, and others. Those services, as well as the identified top-level stages of the first modeling cycle are taken as a basis for feature type modeling. However, differences between instances of the same feature type must be considered. The truck loading process for instance requires different steps and document types than the loading of a ship requires. This distinction influences the creation of a feature type repository as well as the points of variation of a feature type.

TEP1-3 represents the feedering to transship goods from one point to another. The corresponding Transport Feature shown in Figure 5.4 consists of three stages: Departure, Ongoing Shipment, and Arrival. To each stage a task is associated which informs registered stakeholders of the current state. If a *ready for transport* event is received, the *Departure* stage is opened and a notification is sent to the registered users of this artifact. The task of the *Ongoing Shipment* stage is to inform the stakeholders about the concrete position. Finally, the system sends a notification as soon as the milestone *Transport Completed* is achieved. The data attribute *Transport Means*, which actually belongs to the e-Freight TEP model, is the central part of the Transport Feature data model. The TEP Collaboration Artifact type in contrast only contains data attributes that are common among all possible TEP instances. Since there are TEPs which replace services without any transport means, this data attribute is removed from the TEP template.

TEP2-1 provides an example with two features. On the one hand, an export license must be checked by an external service or agent, and on the other hand cargo must be loaded onto the ship. The features are illustrated in 5.5. The *Export License* Feature consists of two stages. One stage, which informs all registered stakeholders of an inserted export license, and another stage for license analysis. The license checking itself is an external service, accomplished by external web-services or humans, who assess the license for validity. One main data attribute hereby is the *Export License* document, which is specific to this feature. The second feature related to TEP2-1 is *Loading*, which contains all information related to the loading of the ship. Loading events are handled, and corresponding data such as a load list is checked for inconsistencies or missing information, once the *Loading* stage is opened. The related milestone finally is triggered by the external service. In a configuration step, both features are bound to the TEP template. The

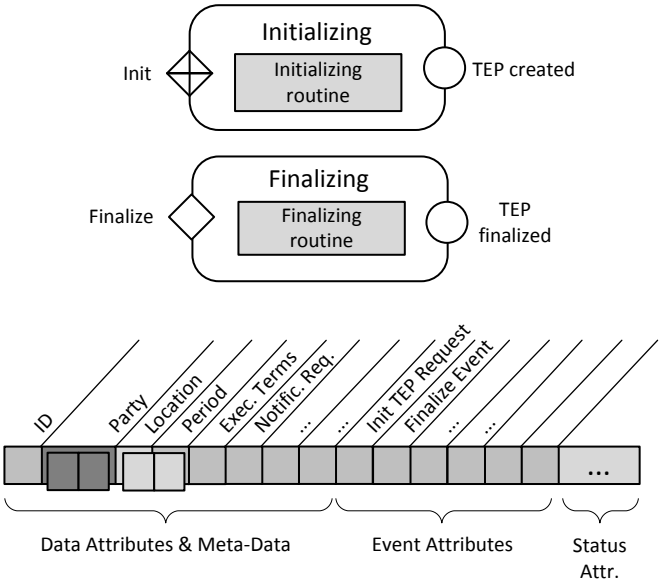

Figure 5.3: TEP Collaboration Artifact (Template)

instantiated Collaboration Artifact for TEP2-1 is presented in Figure 5.6. The remaining TEPs are modeled in the same way, until the whole transport plan is represented by configured Collaboration Artifacts.

Discussion

In this section we discuss the previously described application of the solution approach. The goal is to demonstrate the benefits by analyzing the use-cases of global view (Table 5.1) and scalability (Table 5.2), to discuss if and to what extend the corresponding requirements are satisfied.

Global View The artifact-centric modeling approach provides an end-to-end process view of how key conceptual entities evolve. This is achieved by including both, data and process information into a single artifact. The life-cycle of this element can be tracked to monitor how it evolves across a number of silos and services. Since Collaboration Artifacts are based on artifact-centric modeling, the same properties of an end-to-end process view hold for our solution approach. To what extend the solution provides a global view however depends on the concrete artifact type design. A very abstract, but overarching artifact for instance, provides information of the overall business process, however such a complex artifact type is not very flexible.

In the first use-case *(Table:5.1; Id:Gv01)*, the feeder agent wants to access the Collaboration Artifact TEP1-3. This artifact includes all information related to the transport, which are required to achieve the operational goal. The user has direct access to all status attributes, and thus knows exactly in which phase the TEP is. A status request for this TEP results in: *Departed Milestone* achieved, *Ongoing Shipment* stage active, *Ship Arrival* guard false. If the feeder agent however is interested in the status of the loading, its request must be re-directed to the corresponding artifact. This step is required, since a feeder agent is not directly responsible for the loading. Instead, the agent registers for the status change events, or the Artifact Manager coordinates the notification exchange using the governance model.

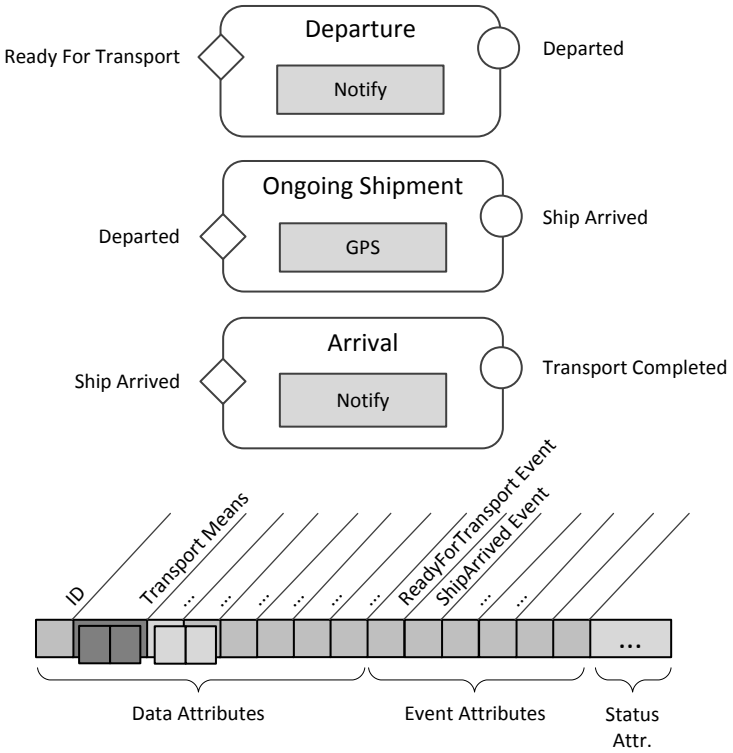

Figure 5.4: Transport Feature

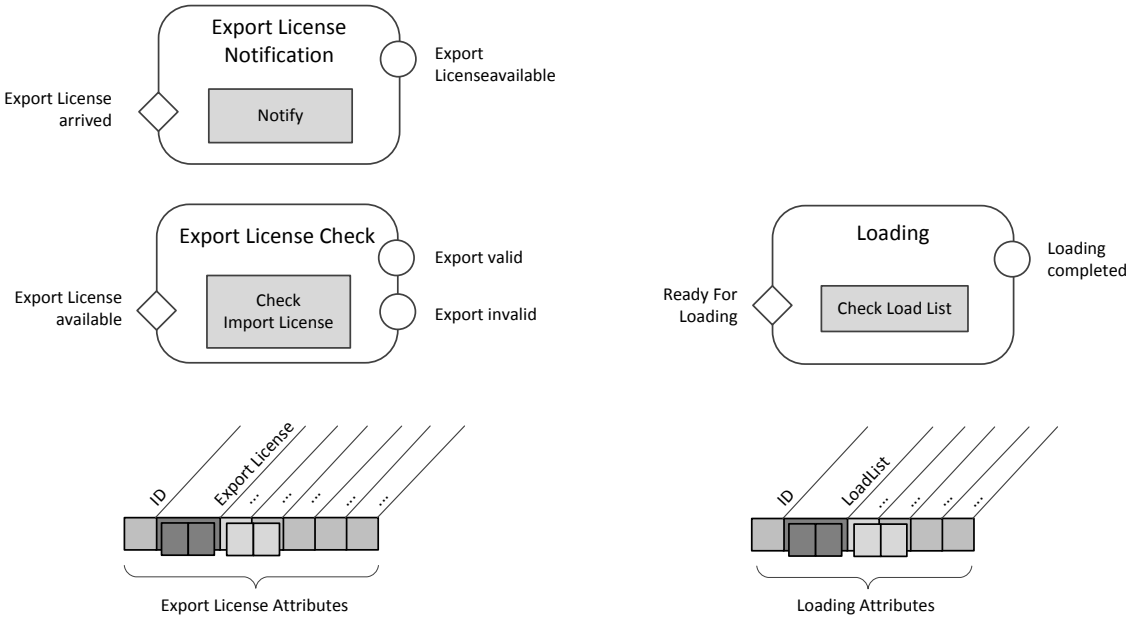

Figure 5.5: Export License Feature and Loading Feature

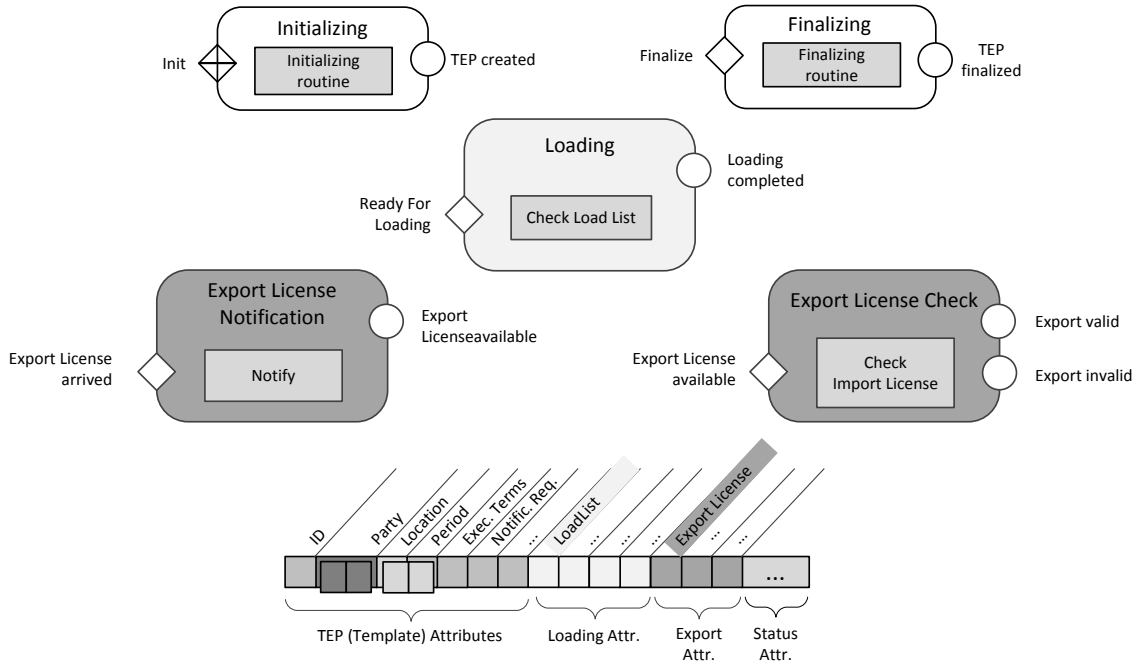

Figure 5.6: TEP 2-1

In the second use-case *(Table:5.1; Id:Gv02)*, the feeder agent is not interested in the status of its directly related TEP or corresponding sub-TEPs. The feeder agent requests information about the goods at the beginning of the chain. Notifications about delays in this first part of the transport are directly supported by Collaboration Artifacts. If the feeder agent however requests specific information, this is only possible with additional logic. Another solution is to represent the goods itself as a Collaboration Artifact, to which all stakeholders have direct access. This provides a complete end-to-end view, among silos, among all involved TEPs and their services, for all participants. However, this would result in a very complex artifact. A starting point for such artifact type provides the Goods Item Itinerary information model introduced by eFreight.

Scalability Scalability is a non-functional property of a system. Since we cannot measure such requirement for an abstract modeling approach, we discuss what concrete characteristics of our solution approach allows to build a scalable system.

Functional scalability *(Table 5.2, Id:Sc01)* is achieved by the modular Artifact Template and Feature concept. This allows the extension of artifact types and provides feature variants. As long as the configuration results in a valid Collaboration Artifact, additional functionality can be easily added, without big effort.

Load scalability (Table 5.2, Id:Sc02) is required to handle even complex scenarios. The artifact-centric approach is based on loosely-coupled design and message interaction. This architecture allows to distribute the artifact execution to different machines. The possible speedup of such scaling however depends on the given scenario and on the artifact design. These concerns are discussed in detail within the subsequent quantitative evaluation.

Conclusion

To conclude, both requirements which are described by the use-cases can be satisfied. By design, Collaboration Artifacts support a global view on a process by combining data- and life-cycle information within a single artifact. The extend of transparency however depends on the artifact design, since it determines what kind of data attributes and process information is encapsulated in a Collaboration Artifact. Although, modularization distributes information in the first place, the overall process can be determined by the collection of states of all involved Collaboration Artifacts. Since the modular design of Collaboration Artifacts allows for concurrent execution and message-based interaction, different artifacts can be distributed over several machines. This distribution is further supported by the autonomous, self-describing design of Collaboration Artifacts, which in the end leads to a scalable system.

5.2.2 Scenario II - Air freight transport

This scenario aims at demonstrating deviation management and re-planning by Collaboration Artifacts. The properties which are verified in this section focus on requirements of flexible process modeling.

Scenario Description and Use-Cases

This scenario is based on the motivating example of Section 2.1.1. It covers the air transport of machinery parts from a shipper in Kiev to a consignee in Amsterdam. The first part describes the road transport to the airport in Kiev. The trucker receives pick-up information and a load list, which are created by the shipper. Customs documents are confirmed, and statistics data is collected, before the shipment starts. At the airport, the truck is unloaded to finalize the first transport leg. The second transport leg then covers the air carrier process. Containers are registered, and finally loaded regarding security regulations. Further, an air way-bill document is required before the plane is allowed for take-off. When the aircraft arrives at the destination airport, the same procedure of unloading and road transport starts, until the goods are delivered at the consignee. With regard to the transport scenario of Section 2.1.1, in the following we abstract from the distinction of an import freight forwarder and an export freight forwarder. The resulting TEP-based transport plan is shown in Figure 5.7.

The root TEP represents the contract between the manufacturer and the freight forwarder. The TEPs in TCP-2 represent the actual shipment of the goods, this is the shipper-to-carrier transport leg, the carrier process, and finally the carrier-to-consignee transport leg. The global view property, which is evaluated within the fish transport scenario is a basic requirement to enable re-planning, since it guarantees that all information for deviation management is accessible. The goal of this use-case scenario is to describe how deviation management is supported by using Collaboration Artifacts. The corresponding use-cases are illustrated in Table 5.3. A typical case of re-planning is to add, or to remove TEPs or sub-trees of TEPs of the transport plan. One deviation event thereby affects the overall TEP/TCP structure. The solution approach thus must allow to effectively change parts of the transport model. The transport plan presented in Figure 5.8 is the result of re-planning. Another type of flexibility is to adapt single transport legs to new requirements. TEP2-1 for instance might suddenly require the ability to support export licensing, which is described as the extension of a Collaboration Artifact. Another, even more fine-grained

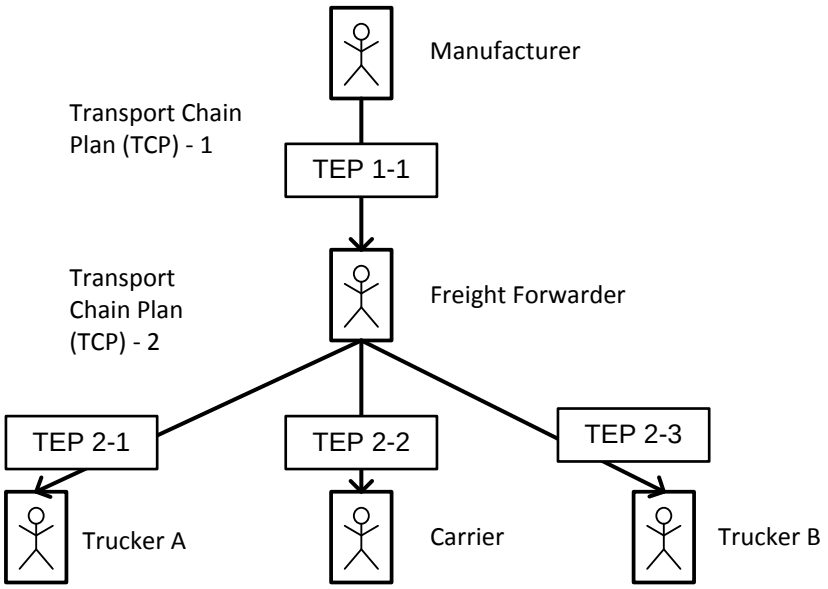

Figure 5.7: Air-freight Transport Plan

adaption is to customize the loading process regarding the sequence of activities, or the required documents.

Realization

The realization of this scenario is based on the same artifact types used for the fish transport scenario. This means, TEPs are considered as the key domain entities, and therefore are implemented as Collaboration Artifacts. In this case however the artifacts are configured with a different set of Features to provide the functionality for an air transport. The goal of this section is to demonstrate how the model allows to deal with uncertainty. This requirement on flexibility is subsequently discussed for the runtime behavior of artifacts, and for the overall artifact system. This covers realization of runtime artifact exchange, artifact extension, and variation as different abstraction levels of flexibility. In the following, we describe for each kind of flexibility represented in Table 5.3, how this it is realized by the solution approach.

The first use-case covers re-planning, which consists of various steps of structural changes by adding or removing TEPs. As described in the conceptual architecture, instances of Collaboration Artifacts are managed by a runtime container named Collaboration Artifact Manager. This manager is responsible for the registration and unregistration of Collaboration Artifacts at runtime. Once an artifact is added to the Artifact Manager, the guards and milestones are registered as event listeners. Such artifact registration results in a system, which is capable to handle new events by providing new functionality implemented by the added artifact. The management of Collaboration Artifacts is based on message exchange, which leads to a loosely-coupled system. This modularization and coordination of elements allows to flexible change the set of running artifact instances, without stopping the system or changing hard-wired references. The same way, in which a Collaboration Artifact is added, it can also be deleted.

The changed transport plan after re-planning is shown in Figure 5.8. Thereby, the re-planning of the initial transport plan consists of the exchange of a single TEP with a new sub-tree of TEPs, which is accomplished by several steps. First, TEP2-1 is removed from the transport plan. This

ID	Attribute	Description
FI01	Actor	Freight Forwarder
	Story	The following deviation occurs: The double amount of machinery parts must be transported, with the result that the booking changes. This requires transport re-planning.
	Conditions	Notifications about the changed booking are sent immediately by system.
	Expected Behavior	The TEP for the transport is easily exchanged by another TEP or a TEP-subtree to handle the changed amount (agility).
FI02	Actor	Freight Forwarder / Business Consultant
	Story	The plan must be varied at runtime. TEP2-1* requires the unforeseen step of customs review.
	Conditions	The customs review Feature already exists in the Feature type repository.
	Expected Behavior	Instead replacing TEP2-1* with a new TEP, which supports the required functionality, this Feature is added to the existing TEP2-1* at runtime (re-configuration).
FI03	Actor	Business / IT Consultant
	Story	The loading Feature in TEP3-1 must be varied. The loading process requires an additional pre-condition in order to execute the associated business task.
	Conditions	Loading Feature already exists in the Feature repository.
	Expected Behavior	The required variation is simple to integrate.

Table 5.3: Use Cases for Flexibility (REQ2.1, REQ2.2, REQ2.3)

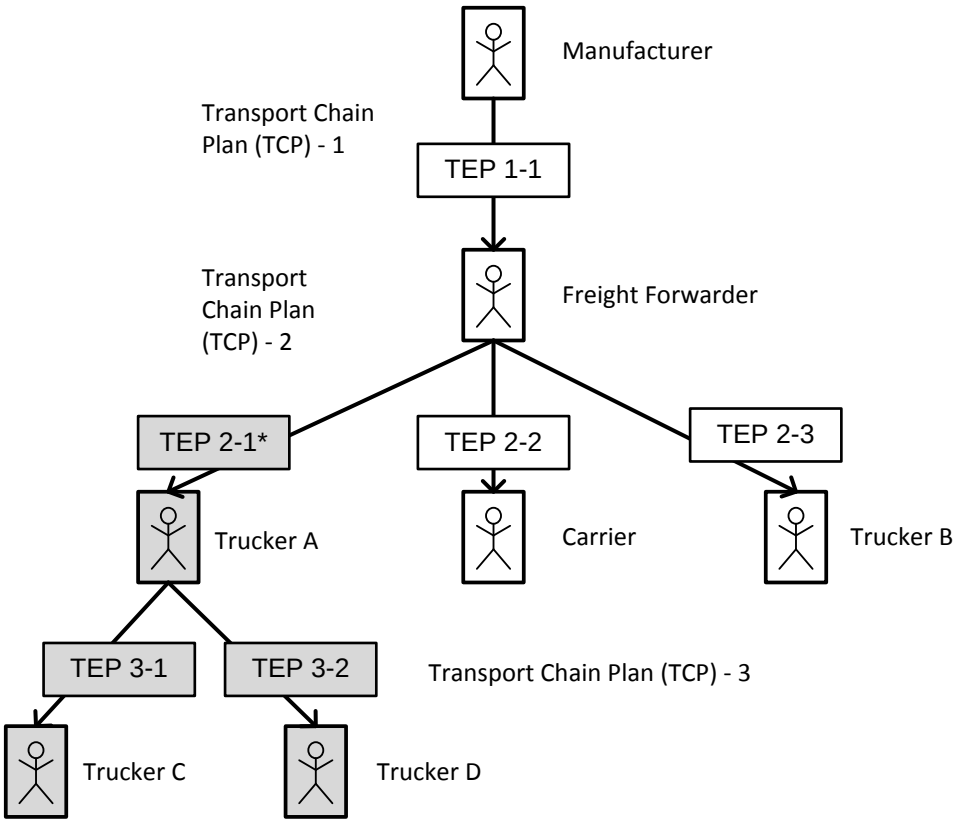

Figure 5.8: Applied Transport Re-Planning

task includes removing the TEP from TCP-2, and unregister it from the Collaboration Artifact Manager. This unregistration results in a changed artifact system, which is not capable to deal with a specific type of events anymore. Second, the configured artifact TEP2-1* is added to TCP-2 and to the artifact system. Thus, TEP2-1* represents the changed transport process agreement to replace TEP2-1. The new TEP is capable of the changed booking, while other constraints and destination data are not affected. In this use-case the involved stakeholder of TEP2-1* is the same as of TEP2-1, however in the new plan, Trucker A becomes a transport service client, which sub-contracts stakeholders for the actual transport. Finally, a new TCP-3 which includes TEP3-1 and TEP3-2 is added to the transport plan, both in charge to transport the machinery parts to the destination airport.

Another type of flexibility, which is supported by the approach covers the extensibility of Collaboration Artifacts. This means the ability of an artifact to react to new requirements without the need to replace the whole artifact by a newly created one. This ability is realized by declared extension points of Collaboration Artifacts, which can be bound by one or more Features. The second use-case FI02 addresses this requirement. TEP2-1 is extended with a feature for export licensing, which results in the TEP2-1*. The extended TEP then is capable of handling new types of events, conditions, and data for export licensing. The solution approach realizes this extension by transparently adding new event listeners, which belong to the new Feature.

The third type of flexibility is about the variation of a single Feature (FI03). This is realized by explicitly declared variation points, which can be bound at runtime. These variation points are defined in advance, and cannot be varied. The Collaboration Feature thus is restricted to a certain level of variability.

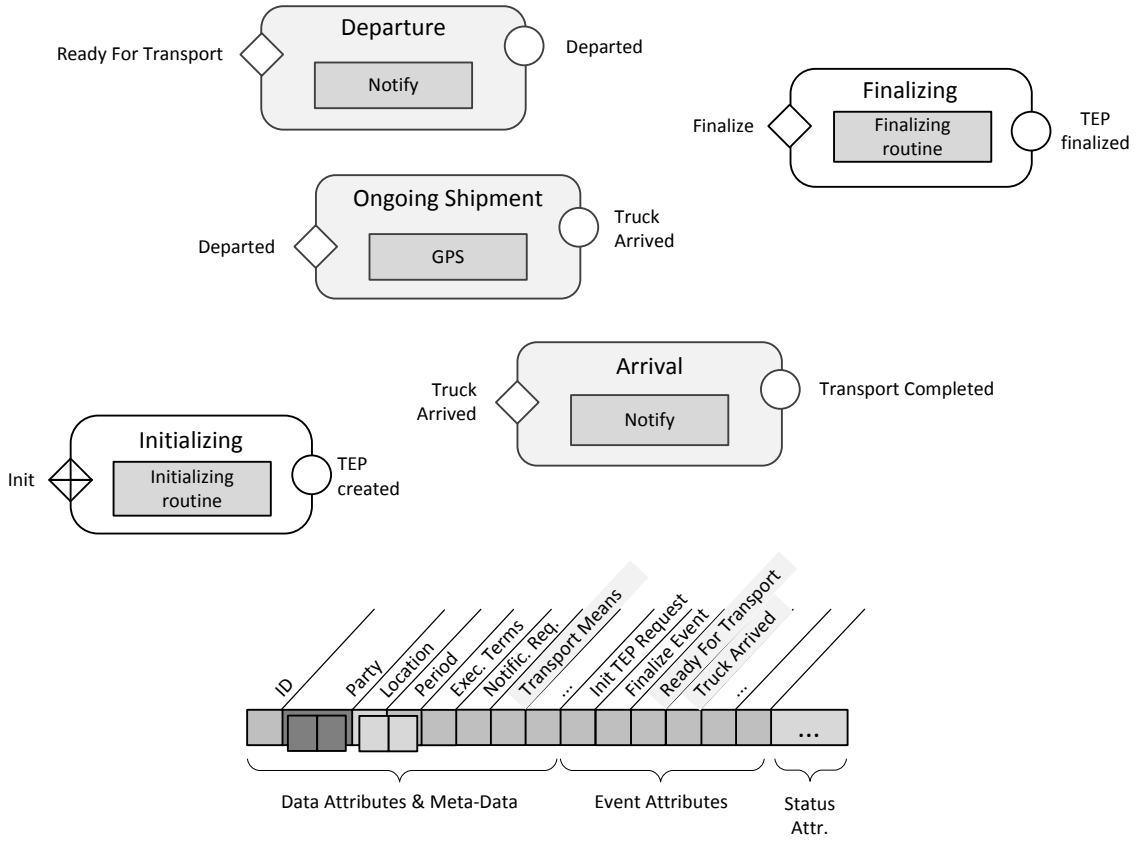

Figure 5.9: TEP 2-1 (Air-freight Scenario)

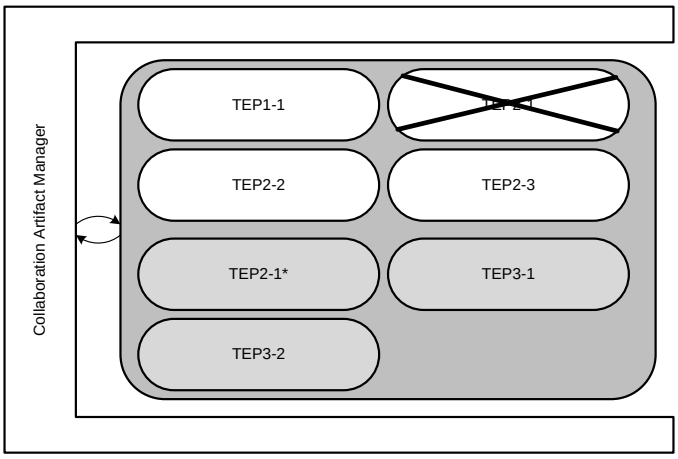

Figure 5.10: Re-Planning in Collaboration Artifact Manager

Discussion

This section discusses the previously described use-case realization. The aim is to evaluate if the requirements of flexibility are satisfied by the Configurable Collaboration Artifact concept.

Flexibility The term flexibility covers a group of requirements of different aspects and abstraction levels. This includes for instance the ability to support process variants, or to allow structural changes at runtime. In this paragraph we discuss aspects of flexibility which relate to process re-planning, extension, and variation.

The first use-case *(Table 5.3, Id:Fl01)* covers re-planning. This type of flexibility is basically provided by the artifact-centric modeling concept of a loosely-coupled artifact system. In general, re-planning can be performed without stopping the system. However, the effort which is required to change the structure of a transport plan, depends on the artifact system design. Completely autonomous Collaboration Artifacts can be easily exchanged, whereby artifacts that in some way depend on each other require additional logic. If an artifact is removed, which hierarchically depends on other artifacts in a parent-child relationship, then these artifacts must be removed as well, to prevent from inconsistent behavior.

The second use-case *(Table 5.3, Id:Fl02)* addresses flexibility for extension. This is achieved by declared extension points which are bound by a theoretically infinite number of Features. The extend of flexibility however depends on the available, pre-defined set of Features. In case, a certain Feature is not included in the Feature repository, the desired artifact extension cannot be realized without additional implementation effort. On the other hand, a Feature repository with many pre-defined elements results in an increased effort for verification, since the possibility for inconsistencies increases as well. Besides that, extending an artifact at runtime can only be performed if the artifact has not yet started.

The third use-case *(Table 5.3, Id:Fl03)* addresses the ability to adapt single Features. This type of flexibility is provided by declared variation points. The set of defined variation points however limits flexibility to a certain degree. This design allows for a minimization of hard-coded behavior and leads to more generic Feature definitions, which constitute the basis for creating Feature variants.

Conclusion

To conclude, all required facets of flexibility are supported by the concept of Configurable Collaboration Artifacts. The solution approach allows to react to unforeseen events and supports process re-planning by exchanging artifacts at runtime. The management unit to control these changes is the Artifact Manager with its registry of currently running artifacts. Further, the solution approach provides the extension and variation of a single Collaboration Artifact. This is realized by the template-hook concept, and by declared extension and variation points. In general, the solution provides flexibility by a consequent modular design.
However, flexibility highly depends on the chosen artifact design. In a first design cycle, we distinguished between four artifact types, one for each means of transport (Rail, Road, Air, Sea). Each artifact template then was pre-defined with a set of stages. The Road-TEP artifact type for instance consisted of pre-defined stages for booking, documenting, loading, and unloading. As it turned out, such artifact design is not flexible enough. According to domain experts, this manifestation of a Road-TEP artifact is not suitable to realize different kinds of transport scenarios, because there are specific needs for different road-transport scenarios, involved stakeholders, or country regulations.
Summarizing, the more high-level and modular a model is designed, the more flexible it is. The solution design however states always a compromise between flexibility and complexity.

5.3 PERFORMANCE AND SCALABILITY ANALYSIS

This section provides a quantitative evaluation of the developed solution. The aim is to evaluate the performance of the implemented Collaboration Management component, and to discuss the scalability of the Configurable Collaboration Artifact approach. In the first part, we analyze the performance of the implemented prototype. Therefore, we measure basic artifact operations at deploy- and at runtime, and discuss its influence on the overall performance of the system. In a second part, we describe different aspects apart from the basic artifact operations, which significantly influence the performance.

5.3.1 Experiment Setup

This section defines the test environment. First, we define the tested operations, which are considered as crucial for performance and scalability. Besides that, we define the system on which the prototype is implemented. Subsequent to this, we summarize the concrete input data which is used for the performance tests. At the end, we discuss the results with regard to system scalability.

Test System

In order to evaluate the performance of the prototype, we first define the central operations for Collaboration Artifacts management. We consider the following operations as suitable for performance tests:

- Instantiation and initial configuration of CAs
- Re-planning (runtime exchange of CAs)
- Artifact execution and messaging

These operations are performed for each scenario. The instantiation of the Collaboration Artifact system is the most complex task. On the basis of a given transport plan, the Collaboration Artifacts are instantiated and configured with the corresponding Features. Afterwards, the CAs are registered at the Artifact Manager, and the governance model is constructed.
The re-planning operation is required each time changes of the business process require structural modifications of the artifact-based transport plan. These structural changes of the process are handled by exchanging CAs at runtime. Each time a CA is added or removed, it must be (un-)registered at the Collaboration Manager.
The runtime execution of Collaboration Artifacts is based on message exchange and of the execution of associated business tasks. These operations are crucial for the performance of the artifact system, because the number of transmitted messages increases with the complexity of the scenario. Messaging operations basically consist of the creation, delegation, and finally the processing by the target artifact. The prototype was implemented, and tested on the following system:

- Processor: Intel(R) Core(TM)2 Duo @ 3,00 GHz
- RAM: 4 GB
- Operating System: Microsoft Windows 7 Enterprise 64-bit Edition; Service Pack 1
- Java(TM) SE Runtime Environment (build 1.7.0_02-b13)

Test Data

The test data for the instantiation of a process with Collaboration Artifacts, is provided by a binary, hierarchical TEP/TCP transport plan. Each TCP thereby represents a node of the tree and consists of two TEPs. Each TCP contains two child TCPs, which are associated to the corresponding TEPs. The first test scenario for the instantiation provides a transport plan with 7 TCPs, which result in a tree with 14 Collaboration Artifacts. The root of the tree consists of one TCP, the first layer of two, and the third layer of 4 TCPs. This transport plan serves as the governance model for the Artifact Manager.
The second test scenario is generated by a transport plan with the depth of 10. The resulting $2^{10} - 1$ TCPs, require the instantiation and configuration of 2046 Collaboration Artifacts.
The performance tests for re-planning are based on an artifact system with already 100 CA instances. The test scenario consists of operations for replacing one or more artifacts by the Artifact Manager. Within these test scenarios, we assume that all artifacts are independent from each other, whereby the exchange of one artifact has not influence to the others. Further, it is assumed, that the new artifact is already instantiated and configured.
The execution of a Collaboration Artifact is tested by the simulation of a request-response message interaction with the CA Manager. One CA message is inserted into the system. The CA Manager then identifies the target artifact and forwards it to the corresponding instance. This message triggers the guard of the artifact and finally activates the stage. The associated business task updates the artifact state and sends a notification back to the CA Manager. This message interaction is evaluated, since it states a crucial task for the performance of a system with numerous Collaboration Artifacts and stakeholders.

Operation	Number of affected CAs	Elapsed Time (millis)
Instantiation	14	348
	2046	1493
Re-Planning	1	2
	100	14
Artifact Messaging	1 (Request-response)	83

Table 5.4: Performance Test Results

Results

This section presents the performance results for the execution of the tested operations. The experiments are implemented with the *JUnit* [1] testing framework. Each test is repeated 50-times to obtain statistically valid results. The results of the performance tests are shown in Table 5.4. The table summarizes the different experiments and the time which is required to perform the operations.

The instantiation of the initial transport process is most time-consuming. However, this is considered as not crucial for the scalability or the overall performance of the runtime system, because it is only performed once during deployment. Nevertheless, the system guarantees performance even for the configuration of a high number of CAs. However, this mainly depends on the complexity of the artifact configuration, and the complexity of the implemented data and life-cycle model. In addition to that, the required time for instantiation depends on the complexity of the governance model. If we increase the depth of the TEP-tree, the number of nodes increases exponentially, which leads to a long instantiation process.
With regard to the performance results, re-planning is the most efficient task. This results from the small number of operations which are required. Since the new Collaboration Artifact is already configured, the Artifact Manager updates the artifact registry and its governance model. However, re-planning is crucial for the performance, because in dynamic environments, such structural changes are very common. Additional verification tasks would decrease the performance of re-planning significantly.
Artifact interaction by message exchange is the most influencing task on the performance of the system, because the number of the exchanged messages increases with the number of involved stakeholders and with the number of registered CAs. A request-response message interaction and the execution of the associated task, requires various sequential operations. A message is created, inserted into the system, forwarded by the CA Manager, and finally processed by the Collaboration Artifact. If the message triggers an associated task, a notification is created and sent back to the CA Manager. We consider the required time of 83 milliseconds for a messaging cycle as adequate. In contrast to the re-planning operations, the effort for messaging does not increase for additional verification or consistency checking behavior. Even for an increasing number of stakeholders and artifacts, the number of messages is manageable, since the TEP artifacts are almost autonomous in their behavior, without strong interdependencies.

[1] http://www.junit.org

5.3.2 Discussion

The previously performed performance test demonstrate, that the prototypical implementation serves a basis for an efficient management of Collaboration Artifacts. However, the results are limited in their information value applied to other scenarios, because the system performance depends on many other aspects. Besides the computational effort of basic Collaboration Artifact operations, additional aspects which influence the overall system performance are discussed in the following.

Performance Influencing Factors

System performance has multiple dimensions such as short response times or high throughput along with low utilization. In the following we describe several aspects which determine those performance dimensions; more important, we argue in which case a certain decision has a benefit.

- Available Resources
  - Number of Processing Units
  - Memory
- Artifact Model / Design
  - Granularity / Abstraction
  - Complexity of atomic Tasks
  - Level of Inter-Artifact Dependency
- Scenario Complexity (Load)
  - Number of concurrent Transport Plans / Artifacts

Available Resources At first, we consider what impact additional resources have to the performance of the system. We discuss this aspect for additional machines, and for additional memory separately.

Machines. The idea is to increase the performance of a system by setting up more servers and to distribute the runtime model. However, more machines do not necessarily result in more performance. In general, the performance increases, if the execution model allows for concurrency. Besides that, the impact depends on the additional effort for synchronizing and coordinating the distributed model elements. For highly interactive, and collaborative models, this additional effort might even decrease the performance.

Memory. The second important parameter is the available memory. In complex scenarios with long living processes, which are implemented in Collaboration Artifacts, it is not efficient to hold all data in memory. Access operations to a hard disk however are cost intensive, especially for frequent read/write operations on a Collaboration Artifact. As a consequence, the amount of available memory increases the performance of a Collaboration Artifact system.

Artifact Design Apart from the number of available resources, the design of the Collaboration Artifacts has fundamental influence to the performance. The type of identified Collaboration Artifacts and the corresponding Features, as well as the implementation of the associated artifact tasks have fundamental influence on the performance. In the following we discuss those aspects which must be decided at design time.

Granularity. This aspect is about the chosen level of abstraction for artifact and feature types. In general, there is a trade-off between expressiveness and flexibility on the one hand, and complexity on the other hand. This applies for instance to the additional effort for artifact verification. The complexity of modularization influences the required effort for re-planning and re-configuration. Very coarse-grain, monolithic artifact types might allow for high performance, however provide less flexibility and scalability.

Artifact Dependencies. Another aspect, which is directly related to granularity, is the number of artifact dependencies. A design with almost completely autonomous artifacts implementing independent tasks, requires less artifact messages for synchronization and thus has a positive influence on the performance.

Complexity of Tasks. The third aspect we mention, is the kind of tasks implemented by a Collaboration Artifact. Complex, time-consuming reasoning and decision making tasks decrease the performance, whereby simple status checks or notification tasks are less resource- and time-consuming.

Scenario Complexity (Load) The complexity of the overall process scenario typically cannot be influenced. However, it has direct impact on the performance of the system.

Number of Transport Plans / Artifacts. The number of transport plans and Collaboration Artifacts that are managed by the artifact system determine the number of required resources. Thus, this complexity of a certain scenario influences the performance, which may degrade significantly under load. The load thereby is determined by the number of involved stakeholders sending or requesting information at the same time.

Summarizing, there are various aspects which influence the performance of a Collaboration Artifact system. Especially when resources are limited, there must be some kind of trade-off favoring some of the requirements while neglecting others. Since Collaboration Artifacts support concurrency of tasks, more processing units seem to be an appropriate technique to increase performance. Collaboration Artifacts are collaborative themselves, however they represent encapsulated units of work, with minimal dependencies among each other. Fully independent components like transport plans can be distributed without any disadvantages. Besides a performance increase by using additional resources, decisions regarding the artifact design have a direct influence. Therefore, it is important that domain experts are directly involved into the design process to find an appropriate level, which allows to intuitively represent the domain, and provides an efficient execution. Beyond these aspects there are parameters, which cannot be influenced. This applies to the kind of business tasks which are implemented in the system. Some tasks are fundamentally serial, whereby others are concurrent. Amdahl's law 5.1 [Pei+05] describes how much a program can theoretically be sped up by additional computing resources, based on the proportion of parallel and serial tasks. Let F be the fraction of parts that must be executed serially, then the law says that on a machine with N processors, the maximum speedup

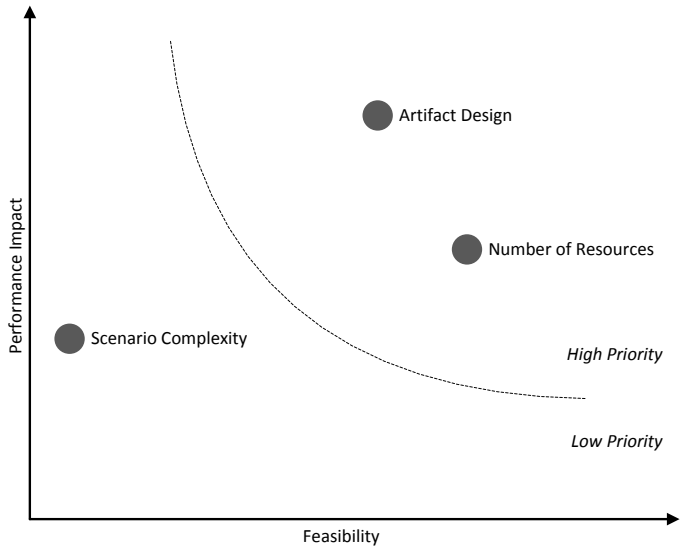

Figure 5.11: Performance Factors (Estimated Impact and Feasibility)

is:

$$speedup \leq \frac{1}{F + \frac{1-F}{N}} \tag{5.1}$$

This means, that the maximum speedup that can be achieved converges to $1/F$. Applied to a Collaboration Artifact system in which fifty percent of the processing must be executed serially, the system can be sped up only by a factor of two.
Although it is not a trivial task to identify the aspect with the highest impact on performance, we think an appropriate artifact design is one of the most fundamental requirements in order to enable efficient system execution. As a consequence this should have the highest priority for the system design. The properties given by the scenario cannot be influenced at all; moreover we think this is less crucial for the the performance. Therefore this aspect has a low priority for a system design. This rating is illustrated in Figure 5.11, which says that the aspects in the upper right corner can be influenced and do have a relatively high impact on the performance.

5.4 SUMMARY

In this section we summarize the results of the evaluation presented in the preceding sections. Thereby we validate the solution against the requirements of Section 2.1. In the first part, we summarize the results of the qualitative, scenario-based evaluation. Afterwards, we summarize the results of the quantitative evaluation.

The qualitative evaluation concludes that all requirements and defined use-cases can be satisfied by an appropriate design of Collaboration Artifacts. The first use-case scenario aims at evaluating aspects of visibility and scalability. A global process view (REQ1.1) is provided by the combination of data- and life-cycle information in a fundamental way. This allows the stakeholders to get all relevant information, which are required to accomplish their tasks. Since a Collaboration Artifact includes the end-to-end life-cycle of the represented domain element, all data which influence the internal business process of a stakeholder is available.

The requirement of scalability (REQ3.1) is satisfied by a consequent modular design, whereby Collaboration Artifacts are loosely-coupled with well-defined communication interfaces for message exchange.
The second use-case scenario validates the requirements of flexibility. The requirement for process agility (REQ2.2) is satisfied by the concept of autonomous, self-contained artifacts, which are coordinated by the Artifact Manager. This design of combining relatively independent process parts allows for efficient process re-planning.
Process adaption (REQ2.3) is accomplished by the applied template-hook concept. On the one hand, this concept allows to extend Collaboration Artifacts with a set of Features, which are required for a specific scenario. Besides that, the Features itself are adaptable to new scenarios by the parametrization of declared variation points.
The ability of process adaption is directly related to the requirement of supporting process variants (REQ2.1). How many scenarios are possible off-the-shelf, depends on the extend of the library of pre-defined variations in form of feature definitions and Artifact Templates. By choosing TEPs as key domain artifacts, we are able to represent two different transport scenarios, which cover sea, road, and air transport processes. A TEP Collaboration Artifact is abstract enough to represent different kinds of transport and logistics processes. This however is possible due to the ability of process adaption and configuration. The general concept of Collaboration Artifact templates and configurable Collaboration Features is very high-level, and thus provides certain flexibility. Due to the artifact-centric design, we are not restricted to a pre-defined process flow. The pre-defined sets of artifact and feature types finally determine the number of possible scenarios, which can be represented by this model instance.

The quantitative evaluation verifies properties of performance and scalability. Performance tests demonstrate that the prototypical implementation is an example of an efficient realization of a Collaboration Artifact system. It is concluded that the basic operations for process management are time-efficient. This applies primarily to the implemented notification and messaging tasks. It is concluded that this prototype system guarantees performance even for complex scenarios. However, the significance of those measurements is limited, since the performance significantly depends on the implementation of security and persistence concepts, which are out of scope of the prototypical implementation. Therefore, a discussion is conducted to estimate additional performance aspects. We conclude, that additional resources increase the performance, which however depends on the chosen Collaboration Artifact design in the first place.

6 CONCLUSION

This chapter concludes the thesis by analyzing the results and by giving a future perspective on the work. The goal is to describe the added value, and the implications of the developed solution approach, with respect to the identified problems.

Various research efforts target at improving the management of collaborative business processes. However, there are still open problems with respect to efficient data exchange and information management. On the one hand, the existing approaches lack of transparency, which however is required to provide the involved stakeholders with necessary information about data, processes, conditions, and critical events. On the other hand, existing solutions can hardly manage frequent environmental changes, which require re-planning and adaptations of the initial process. To solve these problems, additional management solutions on top of existing technologies and infrastructures are required. The problem thereby is to modify certain parts of the process, without interrupting its execution.
The thesis addresses the research question of how the modeling of inter-organizational business processes can be improved to get a more flexible, and scalable system, which in addition provides sufficient information transparency to execute the process efficiently. With regard to this research question, the solution approach introduces an adaptive modeling approach, based on *Configurable Collaboration Artifacts*. The main idea is to extend Business Artifacts with the ability of (re-)configuration with pre-defined, but variable Features. This approach enforces modularization and as a consequence increases reuse and flexibility. A prototypical implementation illustrates technical feasibility of the investigated concepts, and provides an infrastructure for runtime configuration, runtime management, and real-time notification support. A final evaluation shows that the solution can satisfy all declared requirements. The working hypothesis formulated in Section 1.4 is verified as follows:

- Existing approaches either do not provide sufficient flexibility and adaptability (Business Artifacts), or they do not support a global process view or scalability requirements (process-centric modeling approaches). These shortcomings are not derived from infrastructures or technologies in the first place, but from the underlying modeling concept (H1, H2).
- A set of loosely-coupled Collaboration Artifacts enables re-planning by exchanging artifacts without modifying the the overall process structure. The decomposition of Business Artifacts into Templates and variable Features represents a modular design, which allows for consistent variation and extension of single Collaboration Artifacts (H3).

- The underlying artifact-centric modeling paradigm provides a global view on the process, by combining artifact life-cycle and the corresponding information model in a fundamental way (H4).
- The realization of a Collaboration Manager allows the execution of Collaboration Artifacts and thus demonstrates technical feasibility (H5).
- The solution approach allows for the representation of various concrete transport plans for different logistics scenarios, since two different scenarios (fish transport, and air-freight transport) are realized by one concrete model with a pre-defined set of artifact types and features (H6). In general, the model is characterized by under-specification, which allows to re-use and adapt a model for different use-cases within the same domain.
- The evaluation verifies properties of visibility and flexibility by simulating common user scenarios as well as deviations which result in process re-planning. Performance is verified by a quantitative evaluation. Scalability is satisfied by the message-based interaction of loosely-coupled, autonomous Collaboration Artifacts (H7).

Summarizing, the thesis is able to answer the overall research question. The management of inter-organizational business process models can be improved using key domain artifacts. These artifacts are configurable and exchangeable at runtime, in order to react to process deviations and changing environments. Involved stakeholders share a global process view on the information which is required to accomplish the business goal. The technical architecture enables runtime management of a scalable Collaboration Artifact system. The developed concept model can be used as a meta-model for instantiating an adaptive design-, and execution model for business collaboration management systems.
However, there is still place for improvement. Future work is needed to increase reliability and usability. To provide a big picture of the content of the thesis, we subsequently summarize the previous chapters and their main results. This summary states the basis for the discussion on possible future work.

6.1 SUMMARY

This section gives an overview of the previous chapters of this work. A brief summary for each chapter describes the content, the results, and its scientific contribution.

The first chapter of the thesis (Chapter 2) identifies the basic requirements of the solution approach, based on a detailed description of a motivating example. According to this, relevant research areas are determined, which subsequently states the basis for a state-of-the-art analysis. Appropriate developments of each research area are described, and finally assessed with regard to the identified requirements. The research approach of Business Artifacts for data-centric modeling of business processes seems to be most promising. However, none of the existing approaches is capable to satisfy all of the declared requirements. At the end, it is concluded that a combination of artifact-centric modeling with configurative mechanisms is most suitable for the solution approach.

In Chapter 3 we introduce the concept of Configurable Collaboration Artifacts as the central term of the solution approach. This concept is based on the idea of artifact-centric modeling, however with the ability of data and process adaption. The conceptual design introduces extension- and variation points in order to configure an artifact for a specific scenario. The central idea behind

this concept is in separating fixed parts of the business process from flexible parts. The result of this decomposition is a set of predefined, extensible artifact types, which are called Collaboration Artifact Templates. On the other hand, the exchangeable parts are represented by a collection of artifact fragments, also referred to as Collaboration Features. This chapter further describes the process of configuring Collaboration Artifacts. At the end, the need for verification is motivated, as well as a decision problem is formulated, which must be solved in order to guarantee correctness of artifact configuration. The last part describes the runtime behavior of Collaboration Artifacts, and proposes the conceptual design for a management system.

Chapter 4 describes the technical realization of the solution approach. Based on the conceptual design, the technical architecture is described, which is supposed to manage runtime Collaboration Artifacts. The focus thereby is on describing why a certain design is chosen to implement flexible behavior. In particular, this includes extension mechanisms of artifact templates, and variation of Artifact Features. Subsequent to this, the implementation for the runtime management is described. This section demonstrates the technical feasibility of the solution approach.

In Chapter 5 we perform a scenario-based evaluation. Based on a fish transport scenario, we discuss to what extend the requirements on a global process view and scalability are satisfied. An air-freight use-case is then used to evaluate solution properties of flexibility. In a second part, the quantitative aspects of the solution approach, which are performance and scalability are evaluated. This chapter demonstrates, that all requirements can be satisfied, however the design of a Collaboration Artifact system is characterized by trade-offs between flexibility and complexity.

6.2 FUTURE WORK

After concluding that the developed solution approach satisfies the given requirements and thus seems to be suitable to solve the overall research problem, this section describes possibilities for future work to leverage the results, or to improve the proposed solution.

The prototypical implementation demonstrates technical feasibility. However, not all discussed concepts are realized in code. A possible extension would be the enhancement of the Collaboration Manager with fully automated support for artifact configuration. A transport plan, which is inserted into the system, must be analyzed in order to identify the required artifact types and features. This process of artifact instantiation could be further enhanced and controlled by additional configuration rules.
The quantitative evaluation of performance and scalability criteria reveals possibilities for improvement. It is concluded for instance, that the distribution of independent artifacts on several machines might increase the performance of the system. In future work, concurrency could be considered in the implementation, and parallel transport plans could be distributed on several processing units in order to increase the performance.
Another possible extension would be the integration of persistence and of security concepts and frameworks. In the introduced technical architecture, these concerns are placed in the middleware. However, the underlying Collaboration Artifact model must enable these aspects in the first place.

BIBLIOGRAPHY

[AB01] W. M. P. van der Aalst and P. J. S. Berens. "Beyond Workflow Management". In: *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work - GROUP '01*. New York, USA: ACM Press, 2001, pp. 42–51. ISBN: 1581132948. DOI: 10.1145/500286.500296. URL: http://portal.acm.org/citation.cfm?doid=500286.500296.

[AWG05] Wil M.P. van der Aalst, Mathias Weske, and Dolf Grünbauer. "Case handling: a new paradigm for business process support". In: *Data & Knowledge Engineering* 53.2 (May 2005), pp. 129–162. ISSN: 0169023X. DOI: 10.1016/j.datak.2004.07.003. URL: http://linkinghub.elsevier.com/retrieve/pii/S0169023X04001296.

[Aal+09] W. M. P. Aalst et al. "Flexibility as a Service". In: *Database Systems for Advanced Applications*. 2009, pp. 319 –333. ISBN: 978-3-642-04204-1. DOI: 10.1007/978-3-642-04205-8_27. URL: http://dx.doi.org/10.1007/978-3-642-04205-8_27.

[Ada+06] Michael Adams et al. "Worklets : A Service-Oriented Implementation of Dynamic Flexibility in Workflows". In: *Proceedings of the 2006 Confederated international conference on: On the Move to Meaningful Internet Systems: CoopIS, DOA, GADA, and ODBASE - Volume Part I*. 2006, pp. 291–308. DOI: 10.1007/11914853_18. URL: http://dx.doi.org/10.1007/11914853_18.

[Ada07] Michael James Adams. "Facilitating Dynamic Flexibility and Exception Handling for Workflows". PhD thesis. Queensland University of Technology, Brisbane, Australia, 2007.

[Am03] Uwe Aß mann. *Invasive Software Composition*. Springer, 2003, pp. I–XII, 1–334. URL: http://www.springer.com/computer/swe/book/978-3-540-44385-8.

[BDO05] Alistair Barros, Marlon Dumas, and Phillipa Oaks. "A Critical Overview of the Web Services Choreography Description Language". In: *BPTrends* March (2005), pp. 1–24.

[BHS09] Kamal Bhattacharya, Richard Hull, and Jianwen Su. "A Data-Centric Design Methodology for Business Processes". English. In: *Handbook of Research on Business Process Modeling*. Ed. by Jorge Cardoso and Wil van der Aalst. IGI Global, Apr. 2009. Chap. XXIII, pp. 503–531. ISBN: 9781605662886. DOI: 10.4018/978-1-60566-288-6. URL: http://www.igi-global.com/chapter/handbook-research-business-process-modeling/19707/.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008. ISBN: 9780262026499.

[BLB12] Francesco Belardinelli, Alessio Lomuscio, and Nick Bezhanishvili. *Artifact verification and synthesis techniques (ACSI Deliverable 2.1.2)*. Tech. rep. 257593. 2012.

[BMR09] Lars Baacke, Tobias Mettler, and Peter Rohner. "Component-Based Process Modelling in Health Care". In: *17th European Conference on Information Systems, ECIS 2009*. Verona, Italy, 2009, pp. 430–441. URL: http://www.alexandria.unisg.ch/export/DL/54003.pdf.

[BWH08] Adam Barker, Jon B. Weissman, and Jano Van Hemert. "Orchestrating Data-Centric Workflows". In: *Eighth IEEE International Symposium on Cluster Computing and the Grid* (May 2008), pp. 210–217. DOI: 10.1109/CCGRID.2008.50. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4534221.

[Bha+07a] K. Bhattacharya et al. "Artifact-centered operational modeling: Lessons from customer engagements". In: *IBM Systems Journal* 46.4 (2007), pp. 703–721. ISSN: 0018-8670. DOI: 10.1147/sj.464.0703. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5386578.

[Bha+07b] Kamal Bhattacharya et al. "Towards Formal Analysis of Artifact-Centric Business Process Models". In: *Proceedings of the 5th international conference on Business process management*. Springer-Verlag, Berlin, Heidelberg, 2007, pp. 288–304. URL: http://dl.acm.org/citation.cfm?id=1793114.1793141.

[CH09] David Cohn and Richard Hull. "Business Artifacts : A Data-centric Approach to Modeling Business Operations and Processes". In: *IEEE Data Eng. Bull.* (2009), pp. 3–9.

[CL04] Martin Christopher and Hau Lee. "Mitigating supply chain risk through improved confidence". In: *International Journal of Physical Distribution & Logistics Management* 34.5 (2004), pp. 388–396. ISSN: 0960-0035. DOI: 10.1108/09600030410545436. URL: http://www.emeraldinsight.com/10.1108/09600030410545436.

[Dad+10] Peter Dadam et al. "From ADEPT to AristaFlow BPM Suite : A Research Vision has become Reality". In: *Business Process Management Workshops*. 43rd ed. Springer-Verlag, Berlin, Heidelberg, 2010, pp. 529–531. ISBN: 978-3-642-12186-9. URL: http://dx.doi.org/10.1007/978-3-642-12186-9_50.

[Den+04] ShuiGuang Deng et al. "Enhancement of workflow flexibility by composing activities at run-time". In: *Proceedings of the 2004 ACM symposium on Applied computing - SAC '04* (2004), p. 667. DOI: 10.1145/967900.968041. URL: http://portal.acm.org/citation.cfm?doid=967900.968041.

[Deu+09] Alin Deutsch et al. "Automatic verification of data-centric business processes". In: *Proceedings of the 12th International Conference on Database Theory - ICDT '09*. New York, USA: ACM Press, 2009, p. 252. ISBN: 9781605584232. DOI: 10.1145/1514894.1514924. URL: http://portal.acm.org/citation.cfm?doid=1514894.1514924.

[Dre+05] Alexander Dreiling et al. "Model-Driven Process Configuration of Enterprise Systems". 2005.

[ENS10] Stefan Eicker, Annett Nagel, and Peter M Schuler. "Operationalization of Flexibility in Business Process Management". In: *Social Psychology* (2010), pp. 431–436.

[FHS09] Christian Fritz, Richard Hull, and Jianwen Su. "Automatic construction of simple artifact-based business processes". In: *Proceedings of the 12th International Conference on Database Theory - ICDT '09*. New York, USA: ACM Press, 2009, p. 225. ISBN: 9781605584232. DOI: 10.1145/1514894.1514922. URL: http://portal.acm.org/citation.cfm?doid=1514894.1514922.

[FKM09] Agata Filipowska, Monika Kaczmarek, and Ivan Markovic. "Organizational Ontologies to Support Semantic Business Process Management". In: *Proceedings of the 4th International Workshop on Semantic Business Process Management*. 2009, pp. 35–42.

[Fre05] Lars-Ake Fredlund. "Implementing WS-CDL". 2005.

[GD11] Fahad R Golra and Fabien Dagnat. "Using Component-oriented Process Models for Multi-Metamodel Applications". In: *Proceedings of the 2011 Frontiers of Information Technology*. 2011, pp. 218–223.

[GS05] Jennifer Greene and Andrew Stellman. *Applied Software Project Management*. First Edit. O'Reilly Media, Inc., 2005. ISBN: 0596009488.

[GS07] Cagdas E. Gerede and Jianwen Su. "Specification and Verification of Artifact Behaviors in Business Process Models". In: *Proceedings of the 5th international conference on Service-Oriented Computing, ICSOC '07*. Vienna, Austria: Springer-Verlag, Berlin, Heidelberg, 2007, pp. 181–192.

[Gam+95] Erich Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995. ISBN: 9780201633610.

[HNN09] Richard Hull, Nanjangud C. Narendra, and Anil Nigam. "Facilitating Workflow Interoperation Using Artifact-Centric Hubs". In: *Proceedings of the 7th International Joint Conference on Service-Oriented Computing*. Springer-Verlag, Berlin, Heidelberg, 2009, pp. 1–18. DOI: 10.1007/978-3-642-10383-4_1. URL: http://dx.doi.org/10.1007/978-3-642-10383-4_1.

[Hav09] Michael Havey. *Essential Business Process Modeling*. O'Reilly Media, Inc., 2009. ISBN: 0596008430.

[Hen+00] Clemens Hensinger et al. "ADEPT workflow - Advanced Workflow Technology for the Efficient Support of Adaptive , Enterprise-wide Processes". In: *Proceedings of the Conference on Extending Database Technology*. 2000, pp. 1–2.

[Hep+05] Martin Hepp et al. "Semantic Business Process Management : A Vision Towards Using Semantic Web Services for Business Process Management". In: *Proceedings of the IEEE ICEBE*. 1. 2005.

[Hil+04] Hal Hilderbrad et al. "Process-Centric Realization of SOA". In: *J2EE Journal* (2004). URL: http://mohamadafshar.sys-con.com/node/46870.

[Hul+11a] Richard Hull et al. "A Formal Introduction to Business Artifacts with Guard-Stage-Milestone Lifecycles". 2011.

[Hul+11b] Richard Hull et al. "Business Artifacts with Guard-Stage-Milestone Lifecycles : Managing Artifact Interactions with Conditions and Events". In: *Proceedings of the 5th ACM international conference on Distributed event-based system*. 2011, pp. 51–62.

[Hul+11c] Richard Hull et al. "Introducing the Guard-Stage-Milestone Approach for Specifying Business Entity Lifecycles". In: *Proceedings of the 7th international conference on Web services and formal methods*. 257593. Springer-Verlag, Berlin, Heidelberg, 2011, pp. 1–24. URL: http://dl.acm.org/citation.cfm?id=1987781.1987782.

[Hul08] Richard Hull. "Artifact-centric business process models: Brief survey of research results and challenges". In: *On the Move to Meaningful Internet Systems: OTM 2008* 5332 (2008), pp. 1152–1163. DOI: 10.1007/978-3-540-88873-4_17. URL: http://dx.doi.org/10.1007/978-3-540-88873-4_17http://www.springerlink.com/index/7G7610532J3V6787.pdf.

[KO05] Agnes Koschmider and Andreas Oberweis. "Ontology Based Business Process Description". In: *Proceedings of the CAiSE05 WORKSHOPS*. 2005, pp. 321–333.

[LBW07] Rong Liu, Kamal Bhattacharya, and Frederick Y. Wu. "Modeling Business Contexture and Behavior Using Business Artifacts". In: *Proceedings of the 19th international conference on Advanced information systems engineering*. Trondheim, Norway: Springer-Verlag, Berlin, Heidelberg, 2007, pp. 324–339. DOI: 10.1007/978-3-540-72988-4_23. URL: http://dl.acm.org/citation.cfm?id=1768029.1768061.

[LR10] Kung-kiu Lau and Tauseef Rana. "A Taxonomy of Software Composition Mechanisms". In: *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE, Sept. 2010, pp. 102–110. ISBN: 978-1-4244-7901-6. DOI: 10.1109/SEAA.2010.36. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5598084.

[Loo+08] Nik Looker et al. "Scenario Based Evaluation". In: *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing* (May 2008), pp. 148–154. DOI: 10.1109/ISORC.2008.56. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4519572.

[Mei+11] Theo Dirk Meijler et al. "Coordinating Variable Collaboration Processes in Logistics". In: *Proceedings of the 13th International Conference on Modern Information Technology in the Innovation Processes of Industrial Enterprises, Trondheim, Norway*. 2011.

[NC03] A. Nigam and N. S. Caswell. "Business artifacts: An approach to operational specification". In: *IBM Systems Journal* 42.3 (2003), pp. 428–445. ISSN: 0018-8670. DOI: 10.1147/sj.423.0428. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5386806.

[Nan+10] Prabir Nandi et al. "Data4BPM, Part 1: Introducing Business Entities and the Business Entity Definition Language (BEDL)". 2010.

[OAS07] OASIS. *Business Process Execution Language 2.0*. 2007. URL: https://www.oasis-open.org/committees/download.php/23964/wsbpel-v2.0-primer.htm (visited on Sept. 16, 2012).

[OMG11] OMG. *Business Process Model and Notation Version 2.0*. 2011. URL: http://www.omg.org/spec/BPMN/2.0/.

[OR03] By Martin Owen and Jog Raj. "BPMN and Business Process Management: Introduction to the New Business Process Modeling Standard". 2003. URL: http://www.omg.org/bpmn/Documents/6AD5D16960.BPMN_and_BPM.pdf.

[PBL05] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag, New York, Inc., 2005. ISBN: 3540243720.

[Pei+05] Tim Peierls et al. *Java Concurrency in Practice*. Addison-Wesley Professional, 2005. ISBN: 0321349601.

[Pre95] Wolfgang Pree. *Design Patterns for Object-Oriented Software Development*. Addison Wesley Longman, 1995. ISBN: 0-201-42294-8.

[RD08] Marcello La Rosa and Marlon Dumas. "Configurable Process Models : How to Adopt Standard Practices in Your Own Way?" In: *BPTrends* (2008), pp. 1–10.

[RRA03] Hajo Reijers, Jaap Rigter, and Wil M.P. van der Aalst. "The Case Handling Case". In: *International Journal of Cooperative Information Systems* 12 (2003), pp. 365–391. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.4951.

[RRD03] Manfred Reichert, Stefanie Rinderle, and Peter Dadam. "ADEPT Workflow Management System : Flexible Support for Enterprise-Wide Business Processes". In: *Proceedings of the 2003 international conference on Business process management*. Eindhoven, The Netherlands: Springer-Verlag, 2003, pp. 370–379. URL: http://dl.acm.org/citation.cfm?id=1761141.1761169.

[Ria12] Agathe Rialland. *High Level Specification of Use Case Scenarios (FInest Deliverable D2.2)*. Tech. rep. 285598. 2012. URL: http://www.finest-ppp.eu/project-results/deliverables.

[Ros08] Mike Rosen. "BPM and SOA Orchestration or Choreography ?" In: *BPTrends* (2008), pp. 1–3.

[SAP07] SAP. *Technical Architecture Modeling Standard Version 1.0*. 2007. URL: http://www.fmc-modeling.org/fmc-and-tam (visited on Aug. 17, 2012).

[Sch+08] M. H. Schonenberg et al. "Towards a Taxonomy of Process Flexibility". 2008.

[Sch08] Thomas Schlegel. "Laufzeit-Modellierung objektorientierter interaktiver Prozesse in der Produktion". PhD thesis. Universität Stuttgart, 2008. ISBN: 9783939890355.

[W3C05] W3C. *Web Services Choreography Description Language Version 1.0*. 2005. URL: http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/.

[WHM08] Ingo Weber, Jörg Hoffmann, and Jan Mendling. "Beyond Soundness : On the Verification of Semantic Business Process Models". 2008.

[Wal11] Craig Walls. *Spring in Action*. Manning Publications, 2011. ISBN: 9781935182351.

[Wyn+09] M.T. Wynn et al. "Business process verification: finally a reality!" In: *Business Process Management Journal* 15.1 (2009), pp. 74–92. ISSN: 1463-7154. DOI: 10.1108/14637150910931479.

[YLZ11] Sira Yongchareon, Chengfei Liu, and Xiaohui Zhao. "An Artifact-Centric View-Based Approach to Modeling Inter-organizational Business Processes". In: *Web Information System Engineering, WISE 2011*. Springer-Verlag, Berlin, Heidelberg, 2011, pp. 273–281. DOI: 10.1007/978-3-642-24434-6_22.

[Yon+10] Sira Yongchareon et al. "An Artifact-Centric Approach to Generating Web-Based Business Process Driven User Interfaces". In: *Proceedings of the 11th international conference on Web information systems engineering*. Springer-Verlag, Berlin, Heidelberg, 2010, pp. 419–427. URL: http://dl.acm.org/citation.cfm?id=1991336.1991383.

[Zha+09] Xiangpeng Zhao et al. "Enforcing Constraints on Life Cycles of Business Artifacts". In: *In Proceedings of the Third IEEE International Symposium on Theoretical Aspects of Software Engineering*. IEEE, July 2009, pp. 111–118. ISBN: 978-0-7695-3757-3. DOI: 10.1109/TASE.2009.46. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5198493.

[ACS10] ACSI Consortium. *Artifact-Centric Service Interoperation Project*. 2010. URL: http://www.acsi-project.eu/ (visited on May 31, 2012).

[BMT10] BMT Group. *Reference Solutions for Next Generation National Single Windows (e-Freight Deliverable D3.2)*. 2010. URL: http://www.efreightproject.eu (visited on Sept. 25, 2012).