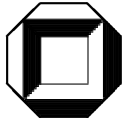


Diploma Thesis

Automation in Collaborative Business Process Instantiation



UNIVERSITÄT KARLSRUHE (TH)
INSTITUTE FOR PROGRAM STRUCTURES AND DATA ORGANIZATION
Prof. Dr.-Ing. Klemens Böhm

Automation in Collaborative Business Process Instantiation

Diploma Thesis

cand. inform. Ingo M. Weber

Karlsruhe, November 2005

Supervised by:

Prof. Dr.-Ing. Klemens Böhm
Dipl.-Math. Jochen Haller, SAP Research
Dipl.-Inform. Jutta Mülle

Diploma Thesis

Institute for Program Structures and Data Organization

at the Universität Karlsruhe (TH)

Title : Automation in Collaborative Business Process Instantiation

Author : cand. inform. Ingo M. Weber

email: imweber@imweber.de

Ich versichere hiermit wahrheitsgemäß, die Arbeit bis auf die dem Aufgabensteller bereits bekannte Hilfe selbständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderung entnommen wurde.

Karlsruhe, den 23.11.2005

.....

Ort, Datum

(Ingo Marc Weber)

Acknowledgements

This work was conducted in close cooperation with SAP Research, CEC Karlsruhe, Germany. Thanks to Jochen Haller, who does a great job as supervisor, and the whole Security & Trust team.

Also many thanks to Jutta Mülle, who gave me valuable feedback and accompanied me in writing this thesis from the university's side, namely the IPD.

Contents

Abstract	1
1 Introduction	3
1.1 Overview	3
1.2 Aims and Objectives	5
1.3 Outline of the Thesis	7
2 Problem Description	9
2.1 Business Process Modeling in the Context of Virtual Organizations	9
2.2 The Information Gap	12
2.3 Language Differences	13
2.4 Aims and Criteria for a Good Solution	14
2.5 Solution Overview	16
2.6 Chapter Summary	17
3 Technological Foundation	19
3.1 The Environment of Web Services	19
3.2 A Short Introduction to Business Process Modeling	22
3.2.1 Process Theory	22
3.2.2 WS-CDL	24
3.2.3 Vision of a Better Choreography Description Language	27
3.2.4 WSBPEL	29
3.3 Chapter Summary	33

4	Related Work	35
5	Conceptual Solution	39
5.1	Overview of the Conceptual Model	39
5.2	Architecture of the Derivation Components	41
5.2.1	The Knowledge Base	41
5.2.2	The CDL2BPEL Algorithm	44
5.2.3	Process Optimization	47
5.3	Translation Rules and Issues	48
5.3.1	Translation Tables	49
5.3.2	WSDL generation	57
5.3.3	Complex Cases of Mismatches in the Translation	57
5.4	Deployment and Confidentiality Implications	66
5.4.1	Deployment and Execution of Generated Processes	66
5.4.2	Integration With Other TrustCoM Services	67
5.5	Chapter Summary	71
6	Implementation Details	73
6.1	Overview	73
6.2	Details on the CDL2BPEL Service	76
6.2.1	Validity Check for Channel Usage	80
6.3	Status of the Implementation	81
6.4	Points for Improvement	82
6.5	Chapter Summary	82
7	Evaluation	83
8	Conclusions and Future Work	87
8.1	Summary	87
8.2	Future work	88
8.2.1	VO Evolution and Business Processes	88

8.2.2	Extending the Knowledge Base	89
A	Business Process Code Examples	93
A.1	Example 1	93
A.1.1	WS-CDL Code	93
A.1.2	WSBPEL Code	96
A.2	Example 2	104
A.2.1	WS-CDL Code	104
A.2.2	WSBPEL Code	111
B	Knowledge Base Contents	143
B.1	Pattern 1 : StoragePartner - Get Raw Data	143
B.2	Pattern 2 : StoragePartner - Store Result Data	144
B.3	Pattern 3 : AnalysisPartner - Analyze Data	146
B.4	Pattern 4 : Initiator - Combine Results	147
B.5	Pattern 5 : Initiator - Identify Need	147
B.6	Pattern 6 : StoragePartner - Store Raw Data	148
B.7	Pattern 7 : AnalysisPartner1 - Analyze Data	149
B.8	Pattern 8 : AnalysisPartner2 - Analyze Data	150
	Bibliography	153
	Index	158

List of Figures

1.1	The Analysis-Storage choreography part	4
1.2	The resulting BPEL processes, derived from the Analysis-Storage choreography part in Figure 1.1. (a) The process for the AnalysisPartner role. (b) The process for the StorageProvider role.	6
2.1	The phases in the VO life cycle.	10
3.1	The Web Services stack depicts the various layers of protocols and standards used in a SOA.	21
5.1	The resulting BPEL processes, derived from the Analysis-Storage choreography part in Figure 1.1. (a) The process for the AnalysisPartner role. (b) The process for the StorageProvider role.	40
5.2	The data model for the database tables behind the Knowledge Base.	42
5.3	The five steps of the CDL2BPEL algorithm	44
5.4	Graphical representation of the XPath query to an assignment in the XML tree of a CDL document. A stronger frame indicates the selection of an element. . . .	46
5.5	Graphical representation of the XPath query to an assignment in the XML tree of a BPEL document. A stronger frame indicates the selection of an element. . .	47
5.6	Visualization of the BPEL activities at one involved role resulting from the translation of a cdl:interaction that makes use of the attribute “align” and has a cdl:timeout and multiple cdl:record children.	58
5.7	CDL source code: Choice with both blocking and non-blocking Work Units. . . .	61

5.8	Visualization of the BPEL activities resulting from the translation of the CDL snippet in Figure 5.7 by the CDL2BPEL Service. This graph is a snapshot from ActiveWebflow TM Professional by ActiveEndpoints, a graphical designer for BPEL processes.	63
5.9	Setup of the relevant components in a VO, with communication from CDL2BPEL services to other components.	68
5.10	Sequence of calls in the transition from the formation to the operation phase of a VO	69
6.1	Data flow diagram with the relevant software components, divided into three partitions: External components, services of the prototype, and other TrustCoM services.	74
6.2	The class diagram overview over <i>org.pi4soa.cdl.*</i> . All shown elements are interfaces. <i>org.pi4soa.cdl.impl.*</i> has the exact same structure, where all elements are named as their respective interfaces, but ending with an additional “Impl”. All elements in <i>org.pi4soa.cdl.impl.*</i> implement their interface from <i>org.pi4soa.cdl.*</i>	78
6.3	The class diagram overview over selected parts of <i>org.activebpel.rt.bpel.def.*</i> . <i>IAeActivityContainerDef</i> , <i>IAeSingleActivityContainerDef</i> , and <i>IAeMultipleActivityContainerDef</i> are interfaces, all other elements are classes. All activity definitions in <i>org.activebpel.rt.bpel.def.activity</i> are inherited from <i>AeActivityDef</i>	79

Abstract

In today's business world, enterprises can gain a strategic advantage over competitors by fast, on-demand collaboration with other companies. The concept of Virtual Organizations (VOs) offers a framework for such collaborations, providing techniques for all stages of the life-cycle. In order to achieve the business objective of a VO, coordinated and coherent behavior of all parties has to characterize the operations. A standard way to specify coherency and synchronization is to define a collaborative business process from a global point of view.

The challenge addressed by this thesis is, how a high-level *choreography* description of a collaborative business process can serve as the business protocol and activity description, from which the partners can derive executable business process representations automatically. With a focus on VOs, a top-down approach with the highest degree of automation is pursued. The main issues in the derivation are caused by differences in languages and the differing levels of details: executable processes contain more information than the choreography. For the different languages chosen, a careful analysis of the semantics of each element led to a number of translation rules. The missing information is inserted into the executable processes via a specialized knowledge base. This knowledge base delivers process parts, which model detailed role internal activities, while avoiding their exposure to collaborating roles. The overall solution achieves the goal of a completely automated derivation and is well-suited for Virtual Organizations.

Chapter 1

Introduction

1.1 Overview

In today's business world, there is a strong need for information technology integration across organizational boundaries. Following the trend of Service Oriented Architectures (SOAs), enterprises will continue to expose well-defined communication interfaces to their respective business partners, in order to enable the information exchange and thus the cooperation of applications on geographically distributed information systems. Emerging standards mainly from OASIS¹ and W3C², especially in the Web service area, allow for a cross-domain business collaboration based on open standards. However, there have to be some explicit, harmonized facts and coherency in the interactions between the systems of several partner roles: For application interoperation, a non-legally binding contractual agreement has to be followed, guaranteeing a common understanding of which information has to be communicated and when. This contractual agreement can take the form of a *choreography*, specifying the interactions and local activities between all roles involved in a collaborative business process at a higher level. I.e., only the interaction points and their respective order are defined in the choreography. All local activities are mentioned only in a way that describes the facts relevant for collaboration.

A simple example from collaborative engineering is shown in Figure 1.1. The graph, modeled as an UML activity diagram³, shows an excerpt of a choreography that involves the roles Analysis (*AP*) and Storage Partner (*SP*). Assume the Analysis Partner receives the request to perform an analysis on engineering data stored within the Storage Partner's domain. The raw data is

¹Organization for the Advancement of Structured Information Standards (OASIS), <http://www.oasis-open.org/>

²World Wide Web Consortium (W3C), <http://www.w3.org>

³Unified Modeling Language, see <http://www.uml.org/>

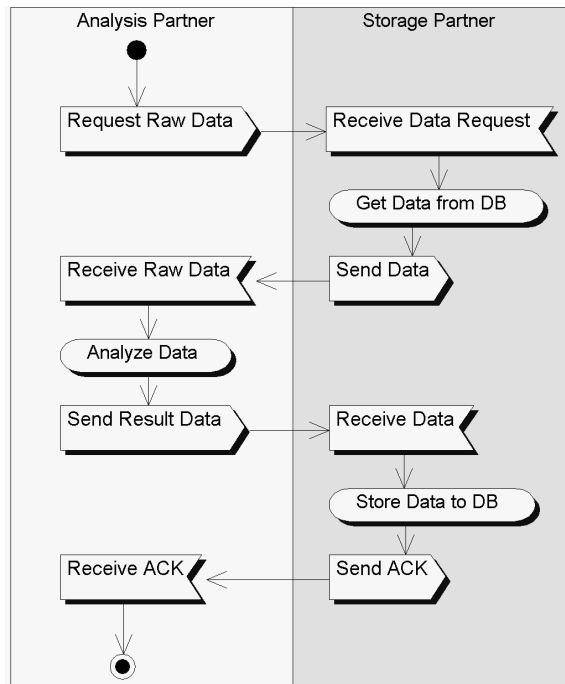


Figure 1.1: The Analysis-Storage choreography part

referenced uniquely, and AP holds the reference information. Now, AP requests data from SP , who retrieves it from the local storage facility and sends it to AP . AP then locally performs the actual analysis work. The results are transmitted back to SP , who in turn stores them in the local database. This example will be further developed throughout the remainder of the thesis. Other examples could be simple order and invoice exchange between collaborating partners' financial systems, or more complex interactions like connecting supply chains across the administrative boundaries of organizations.

At the choreography level, the partners specify the global view of their cooperation instances. That is, each time when a business objective arises, a new instance of a collaborative business process is created and executed by each collaborating role involved in the choreography template. However, a choreography can be seen as the combination of a set of public interfaces and is not executable as process: it only contains the public knowledge for all roles required to participate in a collaboration, but not the private knowledge specifying the detailed activities performed by each single role within its own domain. E.g., in Figure 1.1 the interactions between the two roles are stated, but the knowledge on how the activity 'Analyze Data' is to be done is of no relevance to the choreography and thus not modeled at this level. Therefore, an executable representation of the business process of each role or partner needs to be created. In any of the cases, the local

business process representations have to contain the local knowledge that is not present in the choreography. This thesis deals with the problem of relating the executable business processes from a choreography, in which not all information for execution is stated, in order to enable collaborations based on choreographies by automation.

1.2 Aims and Objectives

This research work was conducted in the context of *Virtual Organizations (VOs)* within the European Union funded IST project *TrustCoM*⁴, which imposes specific requirements: A VO is formed in response to a business objective that can not be addressed by just one partner alone. A swift reaction to the emerging business need, fast partner consortium formation, and quick, automatic adaptation of IT infrastructures are of essence. Thus, an automated solution deriving executable business processes from a choreography was desired, following a top-down approach which is aligned with the VO formation and partner selection procedures [36]. In TrustCoM, the usage of open standards is a requirement.

Business application logic is hereby encapsulated in service implementation. Therefore, a business process at one role can be seen as the ordering structure around local web service invocations, also called *orchestration*. Orchestration captures the local, role specific view on business processes, which orders the calls on available services and guarantees a defined execution order. In contrast, the global view encompasses the collaborative business process, which orders the interactions between the involved roles. Figures 1.1 and 1.2 emphasize the differences between the perspectives.

The overall goal of this work is thus a conceptual mapping that shows how the transformation of a choreography in a standardized⁵ language to a set of orchestrations in another standardized language can take place. The mapping forms the basis for a prototypical implementation, showing the validity of the developed concept. For a given choreography, the orchestrations generated by the prototype then need to be deployed in execution engines and are thereafter executed whenever the need for the collaboration arises. Deployment and execution are the requirements from the VO environment and put a high burden on the semantic correctness and completeness of the orchestrations.

Figure 1.2 shows the UML activity diagrams corresponding to the output of the prototype,

⁴<http://www.eu-trustcom.com>

⁵The usage of standardized languages is very important, due to the interoperability requirement: Various organizations involved have to be able to communicate, and should have the same understanding of the business processes on all levels.

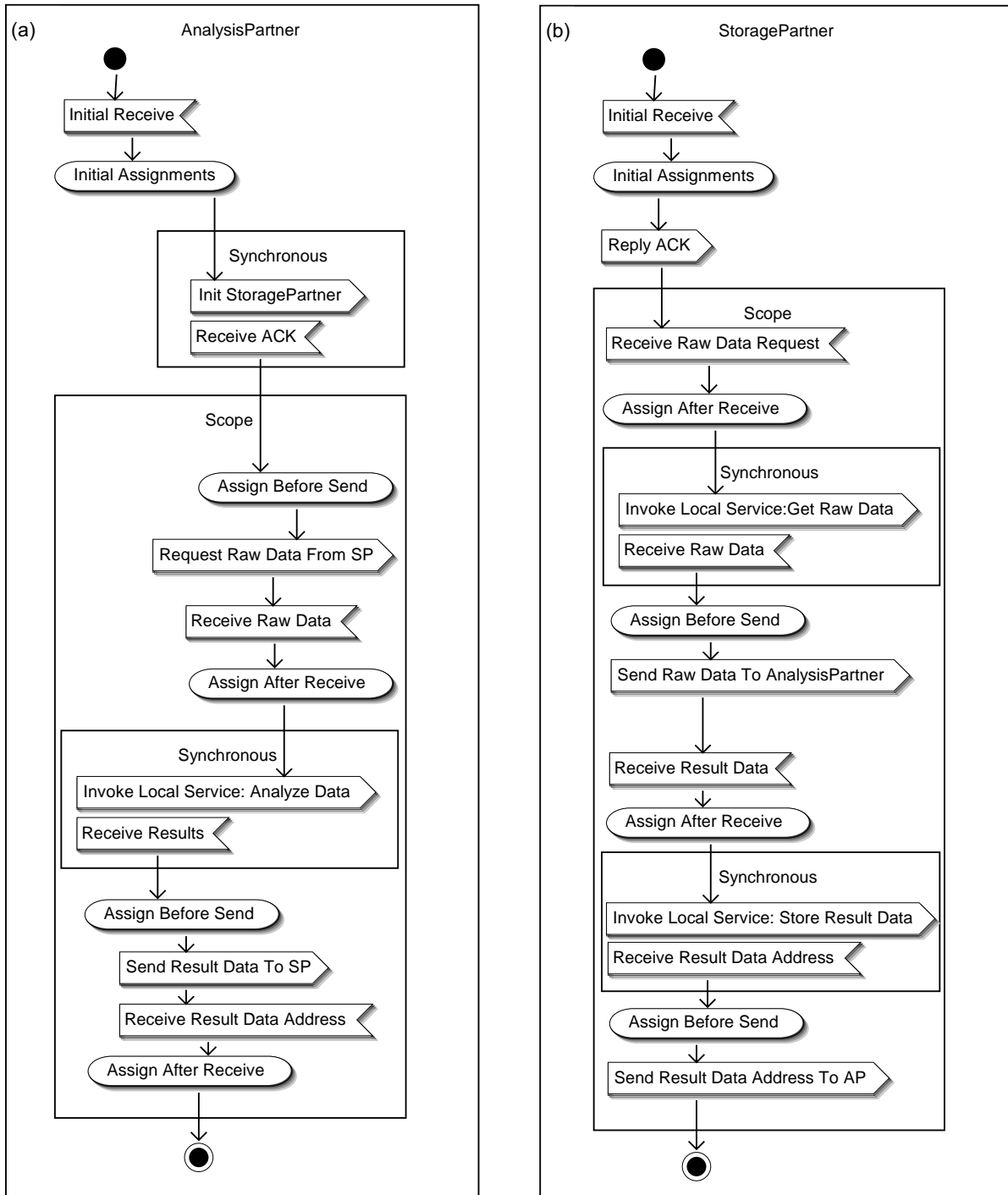


Figure 1.2: The resulting BPEL processes, derived from the Analysis-Storage choreography part in Figure 1.1. (a) The process for the AnalysisPartner role. (b) The process for the StorageProvider role.

where the input was the Analysis-Storage choreography part from Figure 1.1. The actual code behind all example process diagrams can be found in Appendix A

1.3 Outline of the Thesis

With respect to the goal and the methodology, the remainder of the thesis is structured as follows:

In the Problem Description, more on VOs, necessary definitions, and details on the actual problems are stated. A set of criteria for the evaluation emerges from the pursued objectives. A short overview over the chosen solution motivates the next chapters.

The Technical Foundation chapter goes into details about SOA, Web services, process theory, and actual business process languages chosen, also reasoning on the quality of the employed choreography language and the wish list for such a standard. Related Work outlines the relevant ongoing efforts on collaborative business processes and their relation to executable processes.

The Solution chapter is the centerpiece of this work, explaining the suggested modules and their interplay, stating the translation rules in a tabular form, alongside with some workarounds, not all of which resemble a perfect match.

Details on the prototypical implementation, such as programming language and existing code used, can be found in the Implementation Details chapter. The Evaluation follows, revisiting the quality of translations and weighting the results in terms of the criteria from the Problem Description. The main part of the thesis ends with a Conclusion and Future Work.

The two appendix chapters provide all the documents and database contents for diagrams shown in the thesis. The References list all sources of input, and the Index refers back to the points where the most important terms were introduced.

Chapter 2

Problem Description

This chapter formulates the problem addressed by the thesis. Since it is a part of a large research project that deals with Virtual Organizations, these serve as a motivation and codify the environment. After the general problem is stated, the two main sub-problems are outlined in more detail. A formulation of the aims and criteria for a good solution to the presented problem follows. An overview over the anticipated solution concludes the chapter.

2.1 Business Process Modeling in the Context of Virtual Organizations

Definition 1: A **Virtual Organization** is a combination of various parties (persons and/or organizations) located over a wide geographical area which are committed to achieving a collective goal by pooling their core competencies and resources. The partners in a Virtual Organization enjoy equal status and are dependent upon electronic connections (ICT¹ infrastructure) for the co-ordination of their activities [7]. The conducted research was motivated by the VO environment introduced within the EU-funded project *TrustCoM*². TrustCoM focuses on VOs tackling collaborative projects in swift reaction to an emerging business opportunity. Good summaries of real-world VO requirements are given in [34, 9], in the exemplary chemical environment.

The life cycle of a VO is typically divided into four phases [41]:

- During the **VO Identification Phase**, the opportunity is identified, evaluated, and selected.

¹Information and Communication Technology (ICT)

²<http://www.eu-trustcom.com>

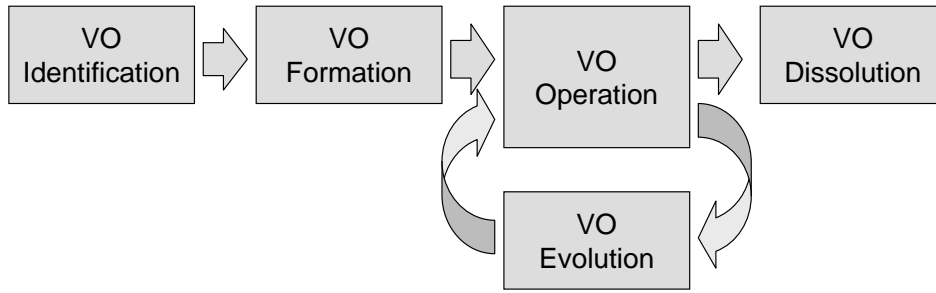


Figure 2.1: The phases in the VO life cycle.

- The **VO Formation Phase** comprises the partner identification, evaluation, and selection.
- In the **VO Operation & Evolution Phase**, services and resources of the single VO partners are integrated, and VO-wide collaborative processes aim at the achievement of shared business objectives. Membership and the structure of VOs may evolve over time in response to changes of objectives or to adapt to new opportunities in the business environment.
- The **VO Dissolution Phase** is initiated when the market opportunity is fulfilled or has ceased to exist. Here, the distribution of results and products takes place, along with billing and the like.

Initially, in Identification and Formation phase the VO is assembled. This involves that a VO initiator, e.g., an aerospace industry system integrator, selects suitable partners to fill the roles required to enact business processes during the operational phase. Such roles may range from simple storage providers to specialized experts in subsystem, for instance wing, fuel tank or antenna, design and manufacturing. The partners are not required to have an existing business relationship of any kind. Departing from the highest level, fitting real partner organizations to roles during identification phase, the subsequent VO formation becomes more detailed when ICT, security, trust, and various other infrastructures and systems of the partners have to be connected [36]. Consequently, the top-down approach also holds for the establishment of collaborative business processes. Departing from a high-level choreography description available to the VO initiator, it is required to derive executable business processes for each participating role. Since VOs are dynamic environments which are intended to act and form fast upon emerging business opportunities, an automatic derivation methodology is highly desired.

Definition 2: A **choreography** describes collaborations of parties by defining from a global

viewpoint their common and complementary observable behavior, where information exchanges occur, when the jointly agreed ordering rules are satisfied. The choreography offers a means by which the rules of participation within a collaboration can be clearly defined and agreed to, jointly. Each entity may then implement its portion of the choreography as determined by the common or global view [54].

Definition 3: A **choreography** defines the sequence and conditions under which multiple cooperating independent agents exchange messages in order to perform a task to achieve a goal state [50].

In TrustCoM, the view based collaborative business process model is used, as presented by Schulz et al in [39, 38], which, in turn, is based on the specifications of the WfMC³. There, the needs for confidentiality of entire processes or workflows of the respective partners and the integration of multiple private workflows into a global view are identified as critical for successful operation of virtual enterprises, extended enterprises, and virtual organizations. On the one hand, an organization may not be willing to share detailed information about a complete business process, since the information in it represents an asset to its owner. On the other hand, enough information has to be provided to the coalition (or the VO in our context) to allow for a correct collaborative, public workflow spanning all parties. Their approach introduces a coalition model with three tiers: private processes, public views of these processes, and a (global) public process. These three tiers correspond to the notions of private business processes, the interfaces of these processes, and choreographies, respectively.

Applying the collaborative business process model to the VO environment, the collaborating members are informed about the VO objective through the shared choreography. They can thus infer the behavior that is expected from them during the VO operation phase which corresponds to their public process. A partner is only required to expose the public process to the VO which serves as the interface for the confidential private process.

Following the Service Oriented Architecture (SOA) paradigm, implemented modular pieces of application logic are assumed to be available as services. Therefore, the private business process can be seen as a stateful wrapper around the services, guaranteeing the defined order of calls to the services. Thus, it is also called **orchestration**, providing a local, role specific view on a private/public process pair, in contrast to the choreography which captures the global view on a collaboration among different roles.

The overlapping and differing aspects of orchestrations and choreographies were identified in [33], with a focus on WSCI [49] instead of WS-CDL, resulting in the statement that an orchestration

³Workflow Management Coalition, among others see [55] and [56]

is the 'inside-out' perspective of one particular partner, while the choreography offers a global view.

The problem thus is to find a way to generate a set of executable, private processes and the public view on them from a choreography description of the overall collaborative process. Two sub-problems are of major importance for the achievement of the goal: The information gap between the levels of detail and the differences in the languages chosen for the description of choreographies and orchestrations. These two sub-problems are discussed in detail below.

2.2 The Information Gap

Choreographies model the interplay between the multiple parties in a collaborative business process and are not concerned about the details of each individual role's activities. Orchestrations, however, need to contain all details required for the execution of a single partner's business process. Thus, the sum of the orchestrations contains more knowledge than the respective choreography. The term *information gap* is used in this work for the issues related to these different levels of detail within the described collaborative business process model.

In detail, the information gap contains the following points of information differences between the choreography and the orchestrations:

1. The internal or private actions for each role, which are of no interest to the choreography.
2. Branching conditions in the orchestration, which are not observable on the collaborative level.
3. Extensions, e.g., annotating security requirements and transactional behavior.
4. Details in error and compensation handling, which are not treated at the choreography level.
5. Runtime details, such as initial and gluing activities, which relate to service endpoint behavior.

While the points 1, 2, and 5 are addressed by the work presented in Chapter 5, 3 is subject to future work. Point 4 could be addressed by best practices for choreography design.

2.3 Language Differences

From the extensive set of available languages⁴, the following three choices were made, addressing the requirements and constraints as presented above in this section. Note, that these choices were not part of this thesis. Instead, they were given from TrustCoM's side, where the usage of open, available standards is a requirement. Chapter 4 introduces related work and other approaches/languages.

- The **Web Service Choreography Description Language (WS-CDL)** [54] is used to specify choreographies. Although it is still in the process of standardization and severe structural critique was expressed (see Section 3.2.3), it represents the most promising and expressive approach currently available. Most other choreography languages state single-partner processes and connect them, at the cost of hard legibility and high risk of incoherency. WS-CDL always offers a combined view on all partners' activities, making it much easier to realize and observe coherence in the various parties' behavior.
- For executable (private) processes, the **Web Service Business Process Execution Language (WSBPEL)** [2] is employed. All generated private processes are executable BPEL processes. This language was chosen, because it is mature, widely known and used, and draws strong attention from industry. It combines interesting aspects from multiple theories for processes. While getting an overview over BPEL, as it is also referred to, is easy, there are many details in the form concepts that allow for flexible usage and sophisticated synchronization mechanisms. Therefore, BPEL is both comprehensible and expressive.
- The public views are expressed in the **Web Service Description Language (WSDL)** [51]. Other possibilities such as abstract BPEL processes were considered and would have been more flexible than static WSDL descriptions. This choice was taken, since WSDL is a rather mature and stable standard, at the same time supporting the choice of executable BPEL. Abstract BPEL is specified, but its intended usage and coherent runtime mapping to executable BPEL is still under discussion in the OASIS Technical Committee standardizing WSBPEL.

These languages are introduced in detail in Section 3.2, along with the general notion of business processes and the process theory that drove their design. The languages chosen here allow for a

⁴As for XML-based standard languages (or languages in the process of becoming standardized), there are multiple alternatives for each category, such as WSCI, WSCL, BPSS, and WfXML for choreographies and BPML, XPD, XLANG, WSFL, and BPDM for executable private processes. In [22], 15 different XML-based specifications for business process modeling are compared and the results were taken into account here.

solution with a pragmatic touch. A more idealistic solution, however, would most likely include another choice for the public processes: Either abstract BPEL or executable BPEL focussing on the interactions with business partners could be good choices. Abstract BPEL describes ordering constraints over the interactions - in contrast to WSDL. Executable BPEL could be used in the following way: When a private process is executed, its corresponding public process is executed as well. Only the public process may be invoked from other administrative domains, and it tunnels the calls to the private process, i.e., the public process only passes messages from or to the private process. While this clearly means overhead in message communication, the status of the public process could be made available to the other VO members for monitoring without revelation of confidential information. By far the most parts of the solution presented in Chapter 5 would still be applicable and valid for another choice of the public process language.

The problem of *language differences* thus is the question of how elements in WS-CDL can be translated to elements in WSBPEL and WSDL, in particular for elements in the source language having no perfect match of semantics in the elements in the target languages. In the next chapter, an overview of WS-CDL and WSBPEL is given. The differences in the control flow structures and basic activities are manifold and sometimes a boolean attribute setting in an element changes the meaning completely. It is thus a hard problem, to which the suggested solution is stated in Chapter 5.

In order to qualify elements and functions in the remainder of this thesis, the following prefixes are employed: *cdl* refers to language items from WS-CDL, *bpel* refers to WSBPEL, and *wSDL* to WSDL language elements.

2.4 Aims and Criteria for a Good Solution

During the Operation Phase of a VO, the goal of the VO is pursued by all of its members via local activities, where synchronization between the members is achieved through communication. Thus, there is a need to express a control flow over local and communication activities of the respective members in a standardized way and to enable the communication across organizational boundaries.

The anticipated procedure here relates to the VO life cycle shown in Figure 2.1 and is the following: From the high-level perspective of the whole VO, first the overall goal is specified, and an agreement about the coherent behavior of all partners during the operation phase is modeled as a choreography. From each VO member's perspective, the member commits to the choreography, which can be seen as a (non-legally binding) contract. After all members agreed to the

same protocol, each of them derives executable processes that correspond to the choreography. These executable processes expose Web service interfaces, enabling communication and thus synchronization among the different processes with respect to the required interoperability and in conformance with the choreography. The executable processes then orchestrate the various Web service calls - the calls to partners and the local calls to the actual business applications, thus aligning them and hiding the local calls from the other VO members. Enabling this procedure by automation is the goal of this thesis.

In more technical terms, the problem is to start with a choreography in WS-CDL and to derive WSBPEL and WSDL for each role in the choreography, followed by the automatic deployment and execution of the outcomes⁵. In order to find a solution, a conceptual mapping for the derivation will be constructed, and a proof-of-concept prototype of this mapping along with necessary extensions will be developed. The main issues are, again, the information gap and the language differences.

An ideal solution can take any valid WS-CDL document and derive a set of processes from it, which together are semantically equivalent to the original choreography. The following set of criteria in a decreasing priority order will serve as a bottom line for the evaluation of a solution to the problem:

1. **Executability:** Each choreography that could possibly result in executable processes actually will be translated to processes which are indeed executable.
2. **Completeness and standard compliance**, in the sense that each WS-CDL element is understood and a suitable translation is found. Thus, the criteria aims at having two equal sets: the set of meaningful⁶ WS-CDL documents and the set of choreographies the solution can deal with. In consequence, input documents should contain only elements anticipated by WS-CDL with the intended meaning.
3. **High degree of automation**, which manifests itself in as much automation as can be achieved: During the conversion the solution should expect user input at no point. Also, the user of an implementation of a solution to the problem has little or no touching points of with the output languages.

⁵The execution is achieved by an existing BPEL engine, of course making this engine and its specifics the target of deployment.

⁶The WS-CDL specification defines the semantics of all elements of the language. However, the set of restrictions for combination and usage of elements is not exhaustive, and thus, for instance, a valid WS-CDL document could be a choreography that always reaches a deadlock state. By meaningful, we here aim at choreographies whose execution is likely to terminate in finite time and after successful completion.

4. The solution can be **extrapolated** from the currently chosen languages to a meta model while maintaining the solution's validity
5. **Extensibility**, so that extensions, e.g., for security and transactional behavior, can be easily added to the approach.
6. **Optimization** of the output. Often, automated generation leaves visible artifacts, resulting in unnecessarily "bloated" output. Here, "slim" processes are targeted, containing little or no such artifacts. Ideally, the executable processes contain no more elements than a human designer would have used.
7. **Robustness** of the approach, so that small errors in the derivation are corrected and major errors do not result in erroneous output.

2.5 Solution Overview

In order to satisfy the above-stated criteria, the following solution is suggested:

Given a choreography, the approach presented in this thesis generates executable processes for each role along with public views on them. The underlying process model follows the view based process model as introduced in [39], where private, executable processes maintain their confidentiality by only exposing views in terms of public processes to collaborating roles.

The conceptual model for the transformation consists of three parts: the translation table with the rules on how to translate every single element from the choreography language to elements in the orchestration language; the *Knowledge Base (KB)* where each partner's local knowledge about internal implementations is made available as executable process parts, which are inserted at the respective spots during the transformation process; and the CDL2BPEL algorithm, which specifies - on the basis of the KB and the rules - the steps to be taken in order to achieve the derivation. In other words, the information gap is filled by partner-specific Knowledge Bases, the language differences are overcome by a conceptual mapping of elements, and the CDL2BPEL algorithm shows how to put these pieces together and apply them to choreographies.

Although the approach focusses on the specific needs of virtual organizations, it is applicable in many other scenarios as well. The VO environment imposes in that sense more challenges on the solution, since all aspects of executable processes, including the local knowledge of the partners, have to be available and specified already at derivation time. All parts of this solution are explained in Chapter 5 in detail.

2.6 Chapter Summary

In this chapter, the introduced terms were defined and the problems addressed by this thesis were stated. The big problem of deriving executable processes from a choreography has two major subproblems, namely the issue of the information gap as a result of the different points of view taken by choreographies and orchestrations, and the problem of language differences resulting from the chosen set of languages. While Chapter 5 is concerned with the solutions to these problems, the next chapter introduces the technology the solution builds on.

Chapter 3

Technological Foundation

This chapter gives an introduction into a number of concepts and languages which will be employed throughout the remainder of the work. The problem requires the usage of Web services and potentially extensions as well as business processes on several levels. Since the solution contains a conceptual mapping from a choreography language to an orchestration language, the chosen languages for these purposes in use are introduced shortly. Also, the suitability of WSCDL as a choreography language is discussed here, both, in general, and in the way it is used in TrustCoM specifically.

3.1 The Environment of Web Services

Nowadays, many languages rely on the Extensible Markup Language (XML)¹ as underlying data model. In short, XML is a linearization of a tree structure, where each node in the tree is called an element. An element can have a (character string) value, a set of attributes, and a number of child elements. In general, no values, attributes, or child elements have to be specified. An XML-based language essentially restricts the structure of a tree, e.g., by specifying required and allowed attributes of an element. An XML document represents a tree, and as such it always has exactly one root element.

A Web service is a piece of software that can be accessed in a standardized, platform-neutral way. Ideally, a Web service is independent, self-contained, stateless, and modular. Oftentimes a Web service encapsulates another piece of software like a complex business function, making it available in a new way with the above stated features. Its functionality follows a well-defined

¹<http://www.w3.org/XML/>

behavioral, not semantic, interface in form of a WSDL² document. Messages exchanged with a Web service are in most cases compliant to SOAP³, and Web services may be advertised in UDDI⁴ registries. All the three standards are XML-based and have clearly stated semantics. Therefore, Web service technology offers a high degree of interoperability - at the cost of higher bandwidth usage due to uncompressed XML messages.

WSDL interfaces contain all information necessary for the invocation of a Web service. A WSDL document contains port types, which are a superset for operations offered by the Web service. An operation has inbound, outbound, and fault messages. The message types for the operations are also defined in the WSDL document, and are referenced by the operations. Message types list a set of message parts, each of which is of a certain XML Schema⁵ type. For each port type, binding information may be specified, stating the used Web service “calling style” and how the deployed Web service is bound to underlying transmission protocols, e.g., HTTP⁶. Finally, a service element lists ports, which refer to binding information and thus to port types. All of the elements are named, and the respective elements in messages to this Web service have to match the names. Together, the WSDL elements define the available operations, how they can be invoked, and what information a message needs to contain. Besides the standard elements, WSDL documents are often used as the location to specify or include referenced XML Schema information, e.g., for complex data structure of message content. Also, WSBPEL adds certain elements to the WSDL interface of a process, namely partner link types and properties/property aliases. `bpel:partnerLinkTypes` are templates for bilateral relationships, required for interactions in BPEL, and bind port types to communication partners of a BPEL process. Properties and property aliases specify which parts of a message are used to identify the message conversation between processes. That is, based on the value of a `bpel:property`, an execution engine can decide to which instance of a process a certain message has to be routed, given there are multiple instances at the same point in time.

The following two frameworks aim at specifying the interplay of multiple Web services and standardizing a set of extensions.

Definition 4: **Service-oriented architecture (SOA)** is the architectural style that supports loosely coupled services to enable business flexibility in an interoperable, technology-agnostic manner. SOA consists of a composite set of business-aligned services that support a flexible

²Web Service Description Language (WSDL), see [51]

³Formerly the abbreviation meant Simple Object Access Protocol, but this meaning is no longer intended. Different versions exist, e.g., SOAP 1.2 [48].

⁴Universal Description, Discovery and Integration (UDDI), see [28]

⁵<http://www.w3.org/XML/Schema>

⁶Hypertext Transfer Protocol (HTTP), <http://www.w3.org/Protocols/>

and dynamically re-configurable end-to-end business processes realization using interface-based service descriptions [5].

Another initiative of interest is the **Web Services Architecture (WSA)**, being “intended to provide a common definition of a Web service, and define its place within a larger Web services framework to guide the community” [50]. It thus aims at specifying the interplay of the numerous standards and languages around Web services, such as reliable messaging, discovery, and many more. Although it is focussed on the basic Web services and only touching the business process level, it can be influential for this work in the following respect. Given a well-understood WSA that enables the inclusion of reliability, transactions, and security, the choreography and the business processes should be aware of these extensions in order to make use of the features in a standardized way.

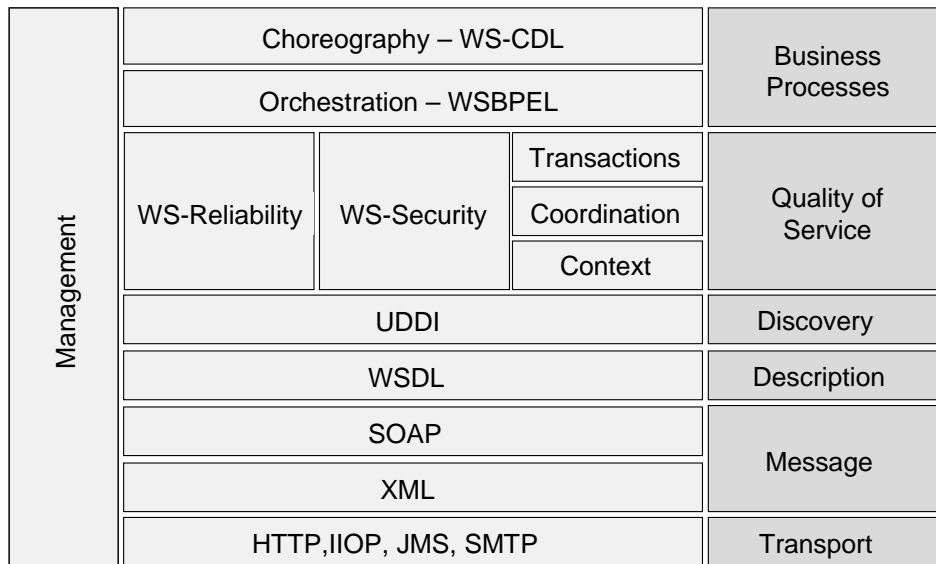


Figure 3.1: The Web Services stack depicts the various layers of protocols and standards used in a SOA.

Figure 3.1 shows the *Web Services Stack*⁷, which consists of six layers, each of which builds on the underlying layers: transport, messaging, description, discovery, quality of service, and business processes. The top layer subsumes orchestrations and choreographies, and the distinction between them is a main motivation for this work. In contrast to the (7-layered) OSI Reference Model in networking, where each layer uses solely the functions of the layer on which it is placed, the Web Services stack’s layers are permeable. E.g., the orchestration layer directly references

⁷Many versions of this diagram exist, including the WSA draft (above). The shown version differs from the WSA one, focussing on the upper layers instead of the lower ones.

elements specified in the description layer.

Based on the big picture introduced above, the languages of choice are introduced in the next section along with their underlying theoretic models.

3.2 A Short Introduction to Business Process Modeling

A *business process* is “a collection of related, structured activities – a chain of events – that produce a specific service or product for a particular customer or customers.”⁸ That is, data and control flow between input, activities of the process, and output are described in a business process. Business processes are well suited for tasks that are either stateful, long-running, or event-based, or a combination of the three. Business process definitions then are templates of how instances of the described processes are to be executed - not seldom they form a contingency plan, stating what behavior should be shown in which situation.

The abbreviation BPM is used to refer to *Business Process Modeling* as well as to *Business Process Management*, depending on the context. Where BPManagement deals with deployment and runtime management of processes, BPMModeling refers to theoretical models and languages built after these models. Research on BPManagement deals with interesting topics, such as on-the-fly replacement of running processes and much more, but this thesis mostly makes use of the basic deployment aspects of BPManagement. The focus here is on BPMModeling, namely two languages for modeling and their underlying theories. Thus, in the rest of this work, the abbreviation BPM stands for Business Process Modeling.

What today is referred to as a business process seems very similar to the before-known notion of a *workflow*. [15] argues that the name “workflow” is rather used for processes with a focus on human actions – and that is just not modern anymore – although the name “workflow” better hints at the meaning than “business process”

3.2.1 Process Theory

There are two main theoretical models on which most descriptive languages and graphical notations for business process modeling are built⁹: The π -calculus and Petri-Nets.

⁸From <http://www.gao.gov/policy/itguide/glossary.htm> as of November 8th, 2005

⁹Or at least claim to do so, as [46] criticizes.

π -calculus

The π -calculus [25, 24] is a Turing complete algebraic language that describes control flow based on communication channels in a formal way. Communication activities over the channels are arranged in the control flow of the processes. The control flow knows constructs for sequences, conditional branching, concurrency, repetition, and encapsulation¹⁰. Based on its description in π -calculus, a distributed system of several processes can be tested for guaranteed lockfreedom, that is, under no circumstances it will deadlock and livelock. When mapping it to a π -calculus representation, basically all details of a set of processes, such as the detailed branching conditions, are abstracted away. The remainder is the plain description of the control flow with data and channel variables, as well as communication and empty (i.e., no-op) activities.

Petri-Nets

In contrast, the *Petri-Nets*¹¹ are a graphical model with clearly defined semantics and a strong mathematical backing. Certain graph-theoretic results can be employed for reasoning about nets and their status. Classic Petri-Nets consist of places, transitions, arcs, and tokens. Places can in general contain any non-negative number of tokens. The arcs are directed links connecting transitions to places or places to transitions, but never places to places or transitions to transitions. A transition may 'fire', if all of the places that have inbound arcs to it have at least one token in it. When a transition fires, one token is removed from every place linked to the transition by an inbound arc, and one token is added to every place that is connected to the transition via an outbound arc. The state of the Petri-Net is the location and number of all tokens in it, events are modeled as firing transitions. Using this model, processes with sophisticated synchronization mechanisms and mutual exclusion can be modeled in a very elegant and comprehensible way.

Several extensions exist, the most important introduce hierarchy and color into Petri-Nets [16]. Color of tokens refers to the idea of tokens being distinguishable, in contrast to being just 'black' as in the classic nets, and enables conditional execution / branching. More appropriate distinction features in our context can be parameters or variable values. Hierarchy provides Petri-Nets with modularity, by allowing sub-nets, e.g., as a substitution of places. In this case, a firing transition that has an outbound arc to a sub-net adds a token to the sub-net's start place.

Using places, transitions, and arcs, the flow of tokens - and thus the state of a net or, in our terms, a process - can be controlled very flexibly. Besides sequential, parallel, and conditional

¹⁰Used for definition of private variables

¹¹See [45, 35] and many more.

branching constructs, arbitrary circles and 'GOTO'-commands can be expressed. In such Petri-Nets the *dead-path elimination* can be conducted, which finds all paths in a net that will never be taken again: the dead paths.

Differences Between the Models

One of the main differences between Petri-Nets and π -calculus is their respective representation of control flow. While the π -calculus arranges activities in explicit, structuring activities such as sequence or parallel, the Petri-Nets formulate the control flow in terms of arcs and transitions. Both approaches are valuable, although not equally expressive. In [44], about 20 workflow patterns are identified and different approaches are evaluated with respect to their ability to express the patterns. The basic patterns, as sequence, split/join, and the like cause no issues to the most approaches, but for more challenging patterns as “Multiple Instances Without Runtime Knowledge”, no elegant solution can be found. One pattern, “Arbitrary Circles” can be stated with Petri-Nets but not in the π -calculus.

In terms of actual business process modeling languages, XLANG¹² is inspired by the π -calculus, while WSFL¹³ in contrast is based on Petri-Nets. WSBPEL, being the blend of WSFL and XLANG, contains concepts of both, π -calculus and Petri-Nets. Since our approach aims at translating from the also π -calculus oriented WS-CDL to WSBPEL, the BPEL elements coming from Petri-Nets are rarely used in the generated processes. In essence, that is the `bpel:flow` with its links, join condition mechanisms, and dead-path elimination. More on WSBPEL and WS-CDL with the respective elements can be found in the next sections.

3.2.2 WS-CDL

Overview

The **Web Service Choreography Description Language (WS-CDL)** [54] is the main effort of W3C's WS Choreography Working Group. Still being on the way of becoming a standard, it offers the most promising, currently available way to describe business processes for multi-party collaborations from a high-level perspective. Similar to an abstract BPEL process, a choreography¹⁴ in WS-CDL only describes the externally observable aspects of a collaborative

¹²See [42]. XLANG is the business process language for Microsoft's BizTalk e-commerce suite. The process description is added to a WSDL definition of a Web service's interface.

¹³See [19] WSFL is IBM's business process language. It knows “Flow Models” for the control flow of processes, which arrange the activities in directed, potentially cyclic graphs.

¹⁴See the definition of choreography on p. 10

business process. It is important to keep in mind that a choreography is not meant for execution, but resembles a design artifact. Examples of WS-CDL documents can be found in Appendix A.

Main Elements

Multiple choreographies together form a `cdl:package`. One of them is marked as the “root choreography”, and thus is the starting point for a package. Having its roots in the π -calculus, a choreography in CDL describes the control flow around basic activities through structuring activities. A choreography can have variables, exception handlers, and finalizers, which define communication and the like at the end of a choreography. Due to the point of view taken by CDL, there are only few basic activities, with the interaction as the center piece, since the focus of choreographies is to describe the how and when of communication. All basic activities, conditional expressions, and variables can be defined for only a subset (sometimes of size one) of the available roles.

In CDL, the concept used for referring to one of the parties is always the role type (or, in short, the role). A party that wants to participate in a choreography can be required to play multiple roles by specifying a `cdl:participant` subsuming these roles. Note that each role can belong to zero or one participant. Also, a role can be defined to show more than one behavior. Each behavior can be refined in a WSDL document, and, if it is not, has no deeper meaning for the details of the choreography. However, both, a WSDL and a CDL document, describe the behavioral interface of entities, although a choreography includes far more information. Thus, our impression is that the redundancy in providing an additional WSDL per role behavior alongside with a choreography yields no significant advantage. Note, that CDL has a closed-world assumption, meaning that interactions are always bilateral between two roles specified in the choreography.

Starting with the structuring ones, the list of activities is shown below.

- sequence - Sequential order of activities.
- parallel - Parallel execution of activities.
- workUnit - As the most unusual structuring activity, the `workUnit` specifies conditions under which an enclosed activity is executed or repeated. Its guard condition is similar to an if-condition in standard programming languages, and can contain various XPath expressions or CDL supplied functions. The guard can be evaluated either immediately or deferred (e.g., when a variable becomes available) by setting the `block` attribute to `true`

or false, respectively. Furthermore, the repeat condition states if a workUnit is considered again after completion.

- choice - Exclusive branching. A `cdl:choice` is intended to contain workUnits as children, with a guard condition. If there are non-workUnit children in a choice, the branching condition is said to be non-observable or not relevant at the choreography.
- interaction - Used for communication between two roles. In data exchanges, the submitted variables are specified. Timeout conditions can be defined directly in an interaction, as well as assignments with reference to the data exchanges. If an interaction's "align" attribute is true, transactionality for an interaction is enabled, in the sense that the interaction only shows effect if the involved roles have the mutual understanding that the interaction completed successfully.
- noAction - Explicit "no operation" for a specified role.
- silentAction - Partner-internal action, whose details are of no interest to the choreography as a whole. The comment, by default in natural language, specifies what a partner is assumed to do at that instant, e.g., "analysis of aircraft antenna".
- assign - Variable value modification. Can cause exceptions.
- perform - Execution of another choreography. With CDL's binding mechanism, variable values from the outer choreography can be carried over to the inner choreography.

The link between WS-CDL and the π -calculus is strong, and also becomes apparent in the availability of channels in CDL. There, channel variables are of a channel type, which allows the definition of identity and reference tokens, restrictions on the channel usage, and the receiving role at the end of a channel.

The way channels can be used in CDL as well as certain activities and more allow for several points of critique. This critique is subject to Section 3.2.3, also suggesting solutions to deal with the issues.

Graphical Notation

UML Activity Diagrams offer a good visualization for choreographies, as justified in [11]. Where BPEL deals with only one party per process, in a choreography there are always multiple roles. The distinction between activities of the various roles is achieved by using a swimlane (large, rectangular boxes) per partner. In contrast to WS-CDL, UML Activity Diagrams do not know

a single activity for the interaction as a whole, so one `cdl:interactions` is represented by a pair of send and receive activities.

3.2.3 Vision of a Better Choreography Description Language

Much critique has been formulated against WS-CDL. During the work on this thesis, a few more points for argumentation about CDL became clear. Here, an overview over the published critique is given, followed by the additional observations from this work. Three approaches to overcome the issues are suggested.

Analysis of Shortfalls in WS-CDL

In [3], WS-CDL is introduced and strongly criticized. The main points of critique (p.16-18) are: the not explicitly stated link to a formalism as the π -calculus on the one hand, and the conceptual limits of linking WS-CDL to WSDL, WSDL-MEPs¹⁵, and WSBPEL on the other hand; the not anticipated runtime selection of participants; the restriction to binary interactions (see below); the dissimilarity of the sets of control flow constructs of WS-CDL and WSBPEL with respect to the fact that WSBPEL is the most promising orchestration language; and the discrepancy of WS-CDL being a design-level language and having no graphical representation. These are all very good points and - since they are deeply positioned in the concepts of the language - question the future of WS-CDL as a whole.

A description of 15 “Service Interaction Patterns”¹⁶ can be found in [4]. They can be seen as combined control flow and message exchange patterns, a collection of requirements the ideal choreography language should be able to satisfy. The abstract reads “beyond bilateral interactions, these patterns cover multilateral, competing, atomic and causally related interactions.” In WS-CDL, communication is always bilateral, and built-in transactionality is restricted to the guaranteed mutual agreement of single variable values at one point in time. Therefore, WS-CDL most likely is unable to express the majority of these patterns.

In the attempt to formalize the underlying semantics of the WS-CDL syntax, [13] reasons about three interaction patterns. In particular, they focus on the alignment property of interactions and find the message exchanges in CDL insufficient. In this work, the message exchanges in BPEL resulting from an aligned `cdl:interaction` contain additional acknowledgement exchanges. Our impression is that aligned interactions can be enabled quite simply by such acknowledgement

¹⁵WSDL 2.0 Message Exchange Patterns

¹⁶Animations visualizing the patterns are available at <http://www.serviceinteraction.com/>

exchanges, and the critique from [13] can be compensated by observing WS-CDL as a design-time language which is not suitable for execution by itself.

From the perspective of this work, there are a few more points of critique that deserve being mentioned. The different sets of available events are an issue to the derivation of executable WSBPEL processes from WS-CDL - regardless of the derivation being conducted automatically or manually. Also, the redundancy in certain WS-CDL elements makes writing a choreography with a general-purpose editor inconvenient. For instance, an attribute whose content has to be a variable, still needs to use the `cdl:getVariable` function.

The way channels can be used in WS-CDL seems inconsistent: If CDL allowed for multiple entities playing the same role in one choreography, the channel could be the reference to the actual entity to which a message should be sent. But then, there would have to be more elements for dealing with channels - and there should be the opportunity to send a message to all such entities playing the same role. Basically, that would come down to providing multicast and broadcast abilities.

Again, WS-CDL does not provide the means for complex interaction patterns, like the ones envisioned in [4]. It thus seems not suitable for modeling related use cases, like a broad request for proposals with unknown outcome. However, such means would be useful for the sole purpose of modeling, but translating such cases to executable business processes might overly stress their capabilities. Current business process execution languages offer a lot of opportunities, but the default case should be that process instances complete without faults. A process can catch and correct faults with minor impact. But if the list of potential errors gets to large, an executable process may be too inflexible. Potentially, a decision as complex as the choice of a proposal in a real-world request for proposal has high economical impact on an organization, and therefore has to be controlled by a human actor. Neither human interaction nor complex decision making are core features of executable business processes.

Suggested Solutions

The following three approaches could possibly solve the issues of WS-CDL's lack of support for complex messaging patterns:

- WS-CDL can be used in its current version, while other requirements are satisfied by applying a suitable combination of web service standards (e.g., WS-Coordination, WS-Federation or WS-Trust).

- Alternatively, WS-CDL could be extended with more complex interaction structures or notions for multiple instances of a role.
- The most radical approach would be to abandon WS-CDL replace it with a new choreography language. The other, currently available languages are intrinsically bottom-up approaches, and do not offer a holistic, consistent view. Currently, the group who published [4] works on the design of a new, graphical notation language for choreographies. Here, the focus is on the conceptual level with the emphasis on an implementation free specification, expressive enough to model the presented interaction patterns. In addition, [26] introduces an extension of BPEL named distBPEL for pervasive computing. distBPEL targets the execution of one BPEL process on multiple mobile devices, by searching a suitable device for each step of the process. Although the idea is not directly applicable in this environment, the approach could serve as a baseline for an extension of BPEL towards a collaborative process language.

The other critique could be addressed by carefully extending the chosen languages, e.g., by offering additional events in BPEL or defining a standard way to use WS-Transaction in conjunction with both, WS-CDL and WSBPEL.

Although an ideal, general choreography language would have a stronger standing if modeling complex interaction patterns with it was possible, interactions involving multiple participants would be hard to translate to BPEL. Potentially WS-Coordination would offer a good way to enable such interactions for execution.

To sum it up: WS-CDL is good for the way it is used in TrustCoM: It specifies the interactions between multiple roles from a high-level perspective, and before the choreographies execution, the roles are bound to VO members. A choreography language that allows for the modeling of complex interaction patterns would mostly be good for design, not for execution as a business process, because its execution should include executable business processes as well as more flexible programming models and human interaction for distinctive choice points with a high economical impact.

3.2.4 WSBPEL

Overview

The **Web Services Business Process Execution Language (WSBPEL)** originates in an initiative from Microsoft, Oracle, and BEA, and is currently under further development by a

large consortium within OASIS¹⁷, including SAP and many more. It started off as the improved and extended combination of XLANG and WSFL, thus containing the general model of basic and structuring activities derived from the π -calculus, as well as a structuring element called *Flow*, that allows for Petri-Net like links between the contents.

Main Elements

In this work, WSBPEL Version 1.1 [2] is considered, due to its maturity and acceptance in both science and industry. When talking about WSBPEL (or BPEL in short) without reference to another version, this version is meant. First, this version is described below. The vision of the next version as well as an extension follow thereafter. Refer to Appendix A for examples of (executable) BPEL processes.

There are structuring and basic activities in BPEL. Structuring activities contain themselves one or more activities, which in turn can be basic or structuring again. The hereby enabled nesting of structuring activities allows for flexible creation of processes with the required behavior. A BPEL process has variables and potentially event, fault, and compensation handlers. Variables can be of user-defined XML Schema type, of `wsdl:messageType`, or of simple type. Event handlers are a central point of reacting to incoming messages or timeouts, fault handlers catch a single specified fault or all faults, and compensation handlers are intended to be used as the location for the partial 'roll-back' of a long-running business process.

WSBPEL distinguishes between *abstract* and *executable* processes. Abstract processes are also referred to as *business protocols*. They describe the externally observable behavior of a process by setting the control flow around all interaction-related activities. In contrast, executable processes, as the name suggested, can be executed by a business process execution engine. Inherently, they contain more detail, namely data handling, termination, and the like.

The following set of activities is available for the actual process in WSBPEL, starting with the structuring activities:

- scope - Subprocess-like, encapsulating structure, allowing for the definition of own variables and event / fault / compensation handlers. Can also be used for synchronization, in that they can exclusively lock used variables for other activities during their execution.
- sequence - Sequential execution of enclosed activities.

¹⁷The Organization for the Advancement of Structured Information Standards (OASIS) WSBPEL Technical Consortium (TC), see <http://www.oasis-open.org>

- flow - Parallel execution of activities. Partial ordering and sophisticated synchronization in a Petri-Net-like way can be introduced via (directed) links. Links can have transition conditions. For multiple links with the same target activity, a join condition over the links' status can be defined. During execution, dead-path elimination (see page 24) is conducted, disabling the execution of elements whose (implicit or explicit) join condition evaluates to false. The completion of disabled activities is no longer necessary for the flow to complete. Since these links must not specify circles, there is a loss of expressiveness (and a gain in the robustness) of the language in comparison to Petri-Nets.
- switch - Branching / conditional execution of activities, where the conditions must be directly observable (e.g., variable value comparison). Achieved by `bpel:case` and `bpel:otherwise` constructs, which must contain one other activity.
- pick - Event-based (or deferred) branching, where at least one child of pick must be a `bpel:onMessage` event. Also, absolute and relative timeouts can be specified by `onAlarm` events.
- while - Repetition of enclosed activities.
- assign - Copy expressions or variable values to other variables.
- invoke - Synchronous and asynchronous Web Service calls, using message variables as input (and output in the case of synchronous calls).
- receive - Incoming Web Service calls, using a message variable as output.
- reply - Like asynchronous invoke, but only as a response on a synchronous call (started via a receive).
- wait - Waits until a specified deadline is reached or some duration passed by.
- empty - No operation (can be helpful as the source of links in a flow or similar).
- throw - Throws a fault (and thus causes the execution of a given fault handler).
- terminate - Exits the process immediately, setting its status to "faulted".
- compensate - Triggers a compensation handler.

There are two kinds of event handlers: `onMessage` and `onAlarm`. Where `onMessage` is like a receive that is enclosed in a `bpel:pick` or `bpel:eventHandler` and has child activities, the `onAlarm` enables timeout events and their treatment.

In order to allow the concurrent execution of multiple instances of one process, there must be a mechanism to route extended message exchange to one particular instance of a BPEL process. For this purpose, either the BPEL correlation sets or WS-Addressing [12] can be used. WS-Addressing achieves this by assigning a Universally Unique Identifier (UUID) to each communication, and `bpel:correlation` is based on the identifying parts of the messages, e.g., the social security number of a person or the order number in e-commerce.

Very recently, a Committee Draft of WSBPEL 2.0 [31] was released. It is not yet a new version of the standard, because it is neither released as a final version nor mature yet. However, it clearly shows the direction that is to be expected from the next version of WSBPEL. Most changes are minor, such as the renaming of `terminate` to `exit`, `switch` to `if`, and `onMessage` to `onEvent`. Also, there is another (slimmer) way to access variables, along with two other kinds of loops: `forEach` and `repeatUntil`. The `rethrow` activity simplifies fault handling. A new element, the `extensionActivity`, could be of great help by offering a way to insert self-defined activities into processes - but a way to distribute such `extensionActivities` within a VO would be required in our context.

None of the available WSBPEL versions knows an explicit subprocess in that respect, only scopes or separately defined stand-alone processes which could be used through Web Service invocation. Therefore, the translation in Section 5.3.1 of nested choreographies in CDL targets `bpel:scopes`. [18] suggests an extension for BPEL 2.0 that allows for convenient specification and use of subprocesses by automating the coupling and communication between the parent and the child process.

Graphical Notation

The standard visualization for business processes in this work is the *UML Activity Diagram*. In UML Activity Diagrams, the Partition element (rectangular container) is used for BPEL-specific elements as `bpel:scope`, `bpel:flow`, `bpel:switch`, `bpel:pick`, for grouping synchronous communication and more. When an even more BPEL specific representation is required, *ActiveWebFlow Professional's*¹⁸ notation¹⁹ is employed. The same notation is used in the *ActiveBPEL* engine to show the current progress of a process instance, where a check mark indicates the successful completion of an activity, “play” symbol refers to an activity currently being executed, and a

¹⁸ActiveEndpoints, Inc. (<http://www.active-endpoints.com>) maintains, among others, two products of relevance for this thesis: A WSBPEL graphical designer, called ActiveWebFlow Professional, and an open-source WSBPEL execution engine named ActiveBPEL (<http://www.activebpel.org>), licensed under the GNU General Public License (GPL).

¹⁹http://www.active-endpoints.com/products/tour/AWFPro_FlashTour_Slides.htm leads to a tutorial on the usage of ActiveWebflow, which explains the graphical notation implicitly.

red cross marks a faulted activity.

Examples of UML Activity Diagrams are shown in Figure 1.2, an exemplary ActiveWebflow screen-shot can be seen in Figure 5.8.

3.3 Chapter Summary

Motivated by the problem statement and solution overview from the previous chapter, this chapter laid the foundation in terms of technology used by the solution, which will be based on the here introduced models and languages. But before the solution is stated, an overview over related work is given in the subsequent chapter.

Chapter 4

Related Work

The main issue of this thesis is the automated derivation of executable business processes specified in WSBPEL and WSDL from WS-CDL modeled business choreographies in an interorganizational environment. There are several publications on issues related to interorganizational workflows in general [1, 8, 43, 47].

The overlapping and differing aspects of orchestrations and choreographies were already identified in [33], but focussing on WSCI [49] and not on WS-CDL. In [33], the author states that an orchestration is the 'inside-out' perspective of one particular partner, while the choreography offers a global view.

Again, our work is inspired by the research on cross-organizational/coalition workflow models [39, 38], which is based on the specifications of the WfMC¹. Their approach follows a bottom-up coalition model, where private processes are designed, public views on these processes are extracted (manually), and the public views are connected to form a public, collaborative process. Due to the environment of VOs here, a bottom-up approach would not be suitable.

Another approach to address coherency in behavior is envisioned in [37], where a harmonization of the message exchange is discussed. Harmonization here means that the message flow in a collaboration is regulated by a set of rules, with which many patterns of coordination, correlation, batching and the like can be modeled. Especially due to the coordination part, it could partially replace the choreographies here. However, the status of work is still early and purely conceptual.

The ebXML BPSS² offers a public view on business processes during design time which has some

¹Workflow Management Coalition, among others see [55] and [56]

²ebXML: Electronic Business using eXtensible Markup Language, see <http://www.ebxml.org/>, BPSS: Business Process Specification Schema. See [17], [20], and [27]

similarities to the choreographies used in our approach. In contrast to WS-CDL, the ebXML business processes can only specify binary relations, posing restrictions to the expressive power for ordering constraints of multi-party choreographies. Furthermore, ebXML does not explicitly address executable business processes, e.g., modeled in BPEL. Businesses mutually agree to follow a BPSS in a CPA³. Instead of striving for automation in setting up a collaboration, ebXML rather expects human beings to agree upon a CPA, taking existing executable processes into account. RosettaNet's⁴ Partner Interface Processes (PIP) with the RosettaNet Implementation Framework (RNIF) can be seen as predecessors to the before-mentioned standards.

The transformation of choreographies to WSBPEL is part of a comprehensive design methodology. Several approaches exist that propose such architectures for Business Process Modeling. Concerning private processes, that are essential in VOs, [40] presents a formal framework for private processes, process views, and integrated processes. The integrated process of one partner is the combined view of the private process and the public views of the other partners. The authors propose a bottom-up approach for design starting with the private processes.

In [10], a methodology for SOA design is proposed, that allows, among others, to model processes from choreographies in a top-down manner. Hereby, the designer(s) create static entities and model the processes that involve these entities as marked, labeled Petri nets. Since the choreography and orchestration viewpoints are unified, the orchestration for each role in a collaboration is inherently obvious. Just as WS-CDL, their meta-model cannot deal with complex message patterns involving multicast and broadcast exchanges with a unknown number of recipients⁵. The unified process model then represents the sum of the choreography and the orchestrations, containing all details of the orchestration. Due to the confidentiality requirement for internal implementations, this approach would not be suitable for the presented problem.

Also, [15] provides an exemplary BPM architecture, which is built on three standards: WS-CDL, the Business Process Modeling Notation (BPMN), and BPEL. This theoretical architecture envisions automated mappings from WS-CDL to BPMN (and compliance checks in the opposite direction), as well as from BPMN to BPEL. This approach is motivated as a good BPM solution for single companies, whose processes must comply to agreed-on choreographies for the interaction with business partners. Their architecture is more of a bottom-up approach, whereas our solution is meant for VOs, with intrinsic necessity for top-down solutions. Also, the author states that "BPMN has no behind-the-scenes open XML representation"⁶ which could be used for automated BPMN generation, based on a given WS-CDL definition.

³Collaboration Protocol Agreement, part of ebXML

⁴<http://www.rosettanet.org/>

⁵See Section 3.2.3 for a discussion about such message exchanges.

⁶[15], p. 41

A case study in e-commerce is presented in [6], with both a choreography in WS-CDL and the respective orchestrations. However, only a subset of the BPEL processes is shown in a proprietary graphical notation, omitting all details of the activities. Besides the incompleteness of the presented orchestrations, all processes were generated manually, and no translation rules were stated. In short, the paper’s contribution is in another area than the achievements of this thesis.

A conceptual model for the mapping from WSBPEL to WS-CDL is presented in [21]. Additionally, a proof-of-concept implementation of the mapping was realized with the Extensible Stylesheet Language Transformations (XSLT), enabling the generation of BPEL stubs from WS-CDL documents. Although the goal seems similar to our approach at a first glance, there are notable differences: our goal is to derive executable BPEL processes that can be deployed without human interaction and are executable in the sense that they really run through when called, and we provide a complete translation table not only a partial one as in [21]. Also, the prototypical implementation of our mapping is implemented as an extended compiler, offering dynamic translation opportunities and sophisticated consistency and correctness validation - qualities that XSLT cannot provide without extensions.

A more detailed look on the suggested mapping from [21] shows many differences in the mapping of single elements⁷: Mendling and Hafner suggest to relate blocking `cdl:workUnits`, that are waiting for a variable to become available, to `bpel:receives`, and assume that some BPEL engineer models the details from where the message comes. Here, such an approach is impossible, due to the goal of full automation. Thus, our solution is `bpel:links` from all points where the variable could be written - therefore making it available for usage. Also, their mapping of `cdl:choice` to `bpel:switch` without even converting the conditions of the switch cases ignores the fact, that `bpel:switch` is only appropriate for directly observable conditions. If all nested branches in a `cdl:choice` have directly observable conditions, then these conditions only have to be converted and copied. Otherwise, a `bpel:pick` has to be used for event-based branching - especially in a case where the conditions are only observable at one role, and the outcome of the branching is communicated via messages to other partners. Although the solution to this potentially very hard translation problem that this thesis suggests also is not perfect⁸, it goes far beyond the proposal of [21]. Last, Mendling and Hafner’s translation of `cdl:finalize` hints at too superficial study of the BPEL standard⁹, since their discussion misses BPEL’s opportunity of having a compensation handler with the attribute `enableInstanceCompensation` set to “true”.

⁷The comparison here is between the referred paper and the translation table in Section 5.3.1 from this work.

⁸Please see Section 5.3.3 and the evaluation in Chapter 7.

⁹Same is true for the suggested `bpel:while`, where they misunderstood the `bpel:while` as being a “do ... until”.

This construct in BPEL perfectly matches the `cdl:finalizer`, and, for once, allows GOTO-like commands in the form of `bpel:compensate` elements - corresponding to `cdl:finalize` commands. This is especially helpful if the process may take several defined endings, but always requires the same activities to be executed in the end.

In [23], the different focus of [21] and this work was emphasized. Different interpretations of the WS-CDL specification became obvious: while the focus of this work demands emphasizing the functional semantics of language elements, in [21] the focus was on the intended semantics of the elements. As an example of the differences, [21] states that `cdl:workUnits` are translated to `bpel:scopes`, because, as clarified in [23], both elements have transactional features. However, these transactional features are only intended in the WS-CDL specification, and do not match the defined functional features - that is, no error handling or compensation can be specified in a work unit. Thus, the suggested translation in this work is different. The generation of process stubs in [21] allows to shift the solution for language differences to an expert, who enriches the generated stubs with further detail. For this work, the generated processes have to be executable without further manual refinement.

Chapter 5

Conceptual Solution

In Chapter 2, the problem this thesis addresses is characterized: From a high-level choreography description of a collaborative business process, executable processes and public process views on them have to be derived. The two main challenges are the information gap and the language differences between the chosen languages for processes on differing levels of detail. This chapter suggests a solution in the given domain of Virtual Organizations in TrustCoM. Approaches to solve the main issues are outlined, together with the necessary integration with other parts of the TrustCoM framework. The conceptual part of the solution is subject to this chapter, while the prototypical implementation of the concept is described in the next chapter.

5.1 Overview of the Conceptual Model

In order to achieve an automated derivation of executable processes, the elements of a WSCDL document are translated in a depth-first search through the XML tree. For each role in the choreography, a private process and the public view on it are derived. Each element in the source document is added only to the processes of the roles for which it is relevant. If a part of the choreography cannot be translated, i.e., it falls into one of the categories of the information gap, it is sent as a request to a **Knowledge Base (KB)**. In the KB, the private, local, or confidential details of a private process are placed. If there are elements of a choreography which still cannot be translated, a list of these elements (and potentially other errors) is returned.

Figure 5.1 shows a graphical representation of the outcomes of the BPEL derivation from the choreography in Figure 1.1. Clearly, the shown orchestrations (or private processes) are far more detailed, and the local substitutes for the high-level, private activities from the choreography

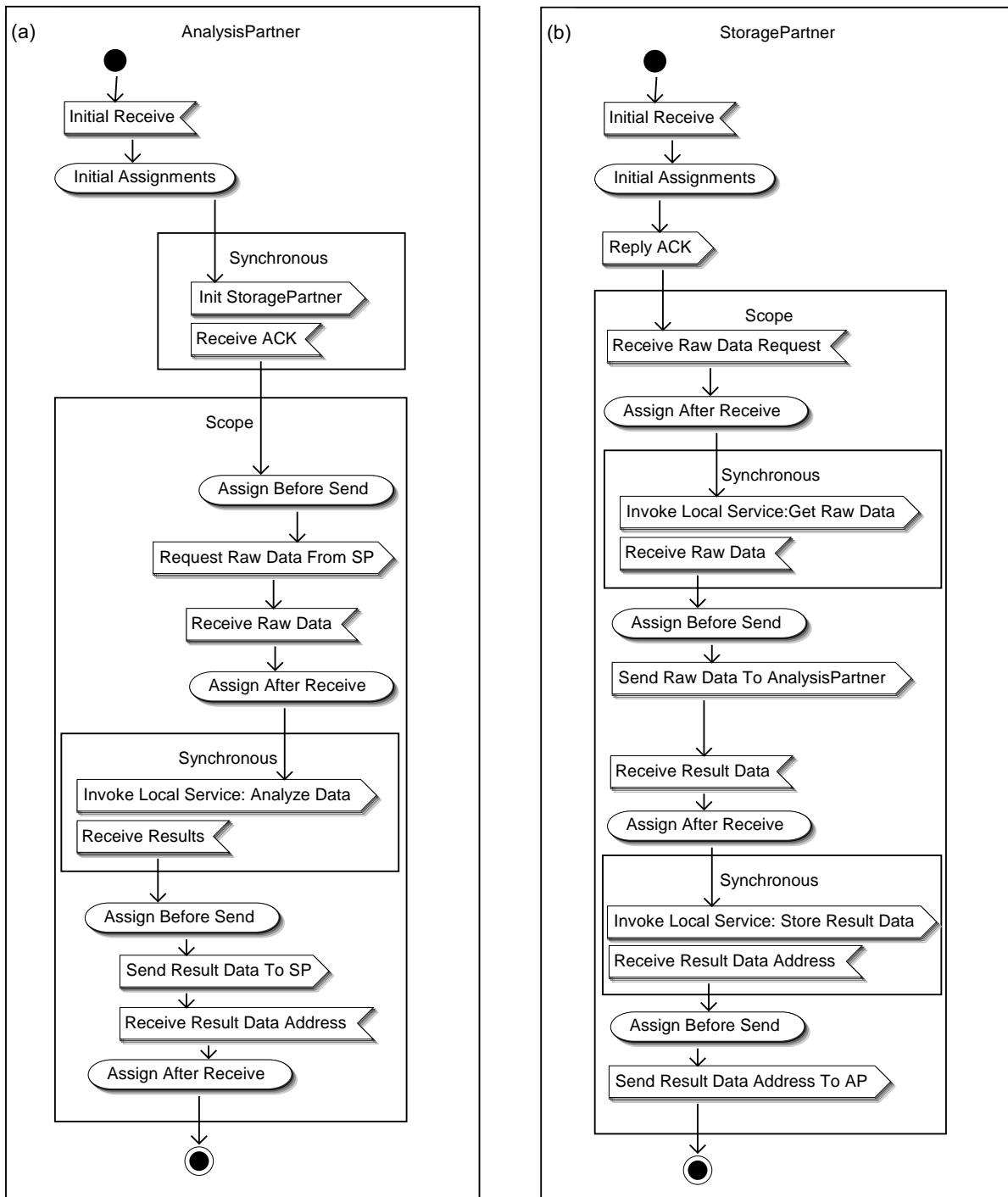


Figure 5.1: The resulting BPEL processes, derived from the Analysis-Storage choreography part in Figure 1.1. (a) The process for the AnalysisPartner role. (b) The process for the StorageProvider role.

are in place. These substitutes are local calls to business services and related activities, e.g., variable assignments, which are dependent on the organization that plays the respective role.

Also, the executable processes are concerned with initialization of the processes, which always means incoming messages in BPEL¹, as well as gluing activities. The latter include variable and message initialization, and apparent assignment statements such as wrapping variable values into messages before communicating them, and unwrapping afterwards.

5.2 Architecture of the Derivation Components

The derivation of BPEL and WSDL from WS-CDL is achieved in a 5-step algorithm, which is outlined in section 5.2.2. In short, the algorithm is an extended compiler, in that it reads a source document and generates an object tree for it, performs validation and transformation on the tree, and serializes the resulting object trees to a set of documents in the target languages. We call it 'extended', because it includes a dynamic part - the Knowledge Base - in addition to the static program code.

5.2.1 The Knowledge Base

The unobservable or private parts of a choreography, summarized as the information gap, are activities that are only mentioned in the choreography. That is, they only say what needs to be done when, but not how. A translation aiming at BPEL stubs for further manual specification would copy the comment from the choreography. But the goal here is full automation, therefore, a way to insert the actual activities into the private processes had to be found. This mechanism is the *Knowledge Base (KB)*.

The KB contains CDL patterns and their respective replacements in terms of BPEL activities as well as optional WSDL elements and deployment artifacts for all roles of interest. When queried, the KB tries to find a pattern matching the WS-CDL part from the request. If such a pattern is found, the respective BPEL and WSDL parts and the deployment information are retrieved. Since the patterns can be generic in that they contain placeholders for variable or partner names, the KB then replaces these placeholders with the instances from the query. Subsequently, the results are returned to the CDL2BPEL algorithm, which weaves them into the output documents of the corresponding roles.

¹Note, that here only the AnalysisPartner's process needs to be invoked from outside. It then calls the StorageProvider's process synchronously, so that one can be sure, that all processes are available before starting the work.

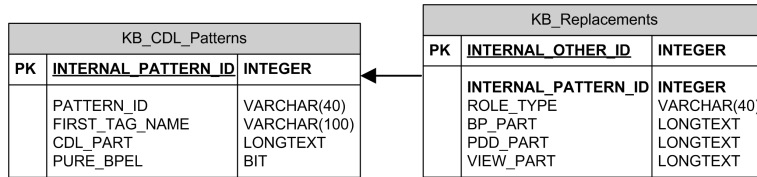


Figure 5.2: The data model for the database tables behind the Knowledge Base.

Here, an implementation based on a relational database system is envisioned. Figure 5.2 shows a possible data model, where each CDL pattern is represented by a data set in KB.CDL_Patterns. One pattern may be relevant for multiple roles, thus its ID is the foreign key of the table KB.Replacements, whose data sets each correspond to the replacement of a CDL pattern for the role in the field ROLE_TYPE. Examples for Knowledge Base patterns from the implementation can be found in Appendix B.

Using the Knowledge Base encourages the reuse of both choreographies and patterns and introduces a dynamic element into the derivation. That means that one choreography can be used for many VOs, and the pattern for a specific business service often requires only little change to make it apply to a certain choreography. If the changes are only in the form of variable or role names, that can be done automatically by adapting the names in the replacements to the names from the query to the KB. Also, standardized choreographies can be bound into other choreographies to solve different purposes, thus offering a way - comparable to design patterns - to rely on often used choreographies for standard parts of a choreography. This technique is enabled by the KB, because the private activity substitutions change with the respective usage and partners of the current choreography. Additionally, pre-specified sub-processes² could be bound into BPEL via the KB.

When the Knowledge Base is deployed and accessed solely locally at each member of a VO, it helps satisfying the confidentiality requirement that comes with optimized process parts and internal implementation. The KB also enables a late binding-like way to connect local services to a process. In that sense, the Knowledge Base can be seen as the material filling up the information gap.

The KB strongly depends on its contents, which must first be inserted and then maintained over time. If the patterns do not fit the rest of the process, runtime behavior is undefined and will most likely result in a fault. Thus, test-runs of the generated processes are highly encouraged. The usage of the KB could take place in the following way: A set of standard patterns are

²See [18] for a suggestion on how sub-processes may be specified in BPEL.

specified over time and made available via a centralized service, e.g., at the host. These patterns are not complete, only their CDL parts are specified. The standard CDL parts are then used in choreographies, whenever possible. Each potential VO member copies the standard patterns to its local KB and adds the private, executable parts. If a particular choreography requires patterns which are unavailable, the respective member needs to add these patterns. Currently, the private process parts need to be created manually, although standard BPEL graphical design tools can be used. Knowledge Base maintenance should be supported by a suitable front-end application. Future work could explore the opportunity of a learning Knowledge Base or semantic pattern descriptions, as suggested in Section 8.2.2.

There are many alternatives to the Knowledge Base, some of which are mentioned below:

- A CDL document could be enriched with the necessary information, e.g., by writing BPEL code as a comment into the document directly. This means that the private process parts can be seen by any member of the VO, forbidding the usage of confidential parts.
- One could also use another approach than instead of a choreography language like WS-CDL, which would be no choreography but a collaborative business process that contains all details. Here, too, the confidential process parts would be exposed to all VO members. Also, this information is partner-specific and would mean that the collaborative process needs to be designed or altered after the partners are selected - which is not the demand of TrustCoM.
- Another realistic approach would be to only generate process stubs and match them against existing private processes. A major issue for such a hybrid approach (top-down and bottom-up with intermediate matching) are the criteria for the match-making. Communication patterns could serve as a criteria, but the details of a hybrid approach are considered out of scope for this work. The hybrid approach has high potential, because optimized, tested private processes are very likely to be correct and stable.
- If the goal was not full automation, only the stubs could be generated and the local parts could be inserted and aligned by an expert.

Since none of the alternatives seems convincing for the requirements here, the Knowledge Base is the preferred approach and forms a centerpiece of the presented solution.

5.2.2 The CDL2BPEL Algorithm

The CDL2BPEL algorithm is responsible for applying the KB from the preceding section and the translation rules from Section 5.3.1. It shows how the CDL document is traversed and how the translation is applied to the elements. Figure 5.3 shows the various steps of the algorithm graphically. In detail, the five steps are:

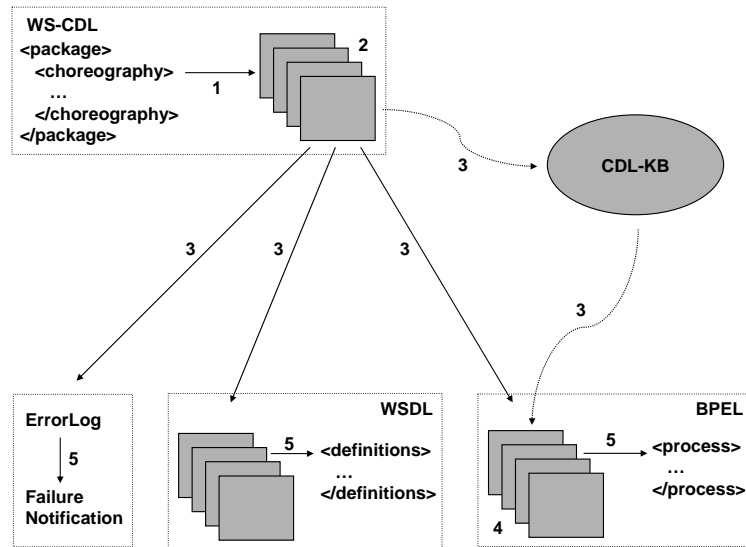


Figure 5.3: The five steps of the CDL2BPEL algorithm

1. Parse the choreography to memory objects and initialize these objects.
2. Validate the choreography, i.e., perform checks on correct variable and channel usage, the correct number of child elements with appropriate attributes, and more.
3. Translate from CDL to BPEL and WSDL
 - Traverse the XML tree from the root choreography in a depth-first search through the object tree, translating and adding each activity to the BPEL process of involved roles. Structuring activities, such as sequence or parallel, are added to the processes of all roles. The translations are performed according to the translation tables.
 - If the current element cannot be translated directly, try a KB lookup with the non-translatable part as input
 - For activities or sets of activities which still cannot be transformed, make an error note. (Report all errors back after traversing the whole tree, see step 5.)

- Extract WSDL files from interactions and tokens / token locators. Generate operations, port types, message schemas, `bpel:partnerLinkTypes`, and `bpel:properties`.
4. Validate and optimize the generated BPEL processes from the holistic perspective, i.e. after the translation is completed.
 - Add necessary gluing activities where obvious, for instance assignments.
 - Optimization and cleanup of the generated processes.
 5. Generate the BPEL and WSDL code from the objects OR return the error log.

The results of this algorithm are a set of documents, namely for each `cdl:roleType` a private, executable BPEL process and the public view on it as a WSDL definition. Note that WS-CDL knows an element called *participant*, which groups together multiple role types that, during execution, must be played by a single entity. Therefore, another approach could be to generate one process per participant. However, we decided on the above solution, because one organization representing a participant with multiple roles could work with multiple BPEL engines for the differing purposes of the roles. E.g., in a buyer-seller-shipper choreography where the seller and shipper roles have to be played by one entity, the company who plays these roles could have distinct departments for sales and shipping, each maintaining an own BPEL engine. Thus, this decision allows for more flexibility, but shifts the enforcement of the participant-role constraints to another spot.

The third step of the algorithm is the actual translation. Here, the translation rules are applied to each element. Although some translations depend on the context, most rules only focus on the current element and its attributes and children. These translations draw their validity from the fact that the location of an activity defines when and if the activity is executed, both, in WS-CDL and in WSBPEL: A structuring element contains activities, and these activities can themselves be structuring or basic. An activity then most of the time describes what is done when this element is reached, e.g., in a `bpel:assign`, the values copied to a variable do not depend on the question if the assign operation is the child of a `bpel:sequence` or a `bpel:flow`. Therefore, many elements can be translated by one rule, independent of their context. The important point is, that each activity's translation relies on the validity of the translations of its parents. Thus, the challenge for the algorithm is to place each output element at the right location in the output process. This is achieved by the interplay of paying attention to the current parents in the choreography, and the depth-first search manner of the translation.

Take the example from Figures 5.4 and 5.5, where a `cdl:assign` is translated, whose XPath³ query expression is `“/package/choreography/parallel/sequence/assign[3]”`. Then, it is translated to a `bpel:assign` at the respective role, with the query expression `“/process/scope/flow/sequence/assign[2]”` in the process. Note, that in the CDL the current assignment was the third assignment in this sequence, while in BPEL it is the second. Such differences come from the fact that only the relevant basic activities are added to the process of each role. Here, the `cdl:sequence` could enclose two assignments for role *A* and one for role *B*. Thus, in the process of role *A*, the current assignment is the second of its kind with the mentioned XPath expression. With structuring activities, CDL2BPEL deals differently: these activities are translated to each role, because the translation expects all such activities to be present at each role when the children of a structuring activity are translated. In the example above, CDL2BPEL adds the new `bpel:assign` to the process of role *A* as the next child of the present `bpel:sequence`.

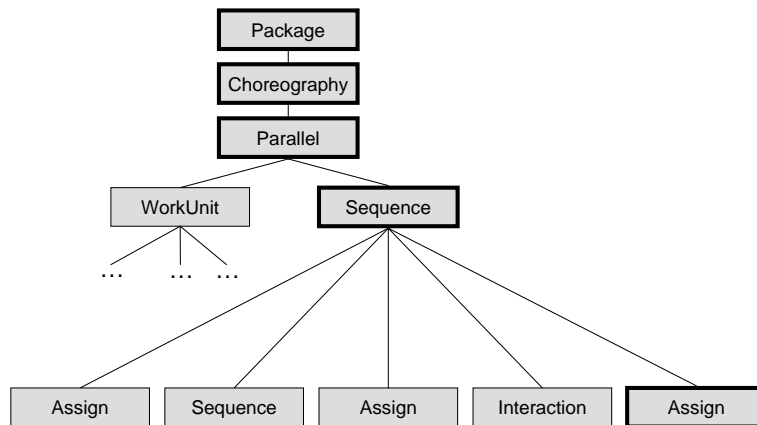


Figure 5.4: Graphical representation of the XPath query to an assignment in the XML tree of a CDL document. A stronger frame indicates the selection of an element.

Due to this mechanism, most elements can be translated without closer examination of the context. However, it often causes superfluous existence of structuring activities. In order to get valid BPEL documents, these structures need to be removed, which is done in the fourth step of the algorithm. Also, some optimization can be done here, which is described in the next section.

We did not attempt using XSLT⁴ for the transformations due to its limitations. XSLT is designed for generating XML result documents out of XML source documents in a straightforward manner. Here, we need more sophisticated mechanisms, e.g., for validation: WS-CDL

³XML Path Language (XPath), see [52]

⁴Extensible Stylesheet Language Transformations (XSLT), see XSLT 1.0 [53] and XSLT 2.0 (<http://www.w3.org/TR/xslt20/>), work in progress)

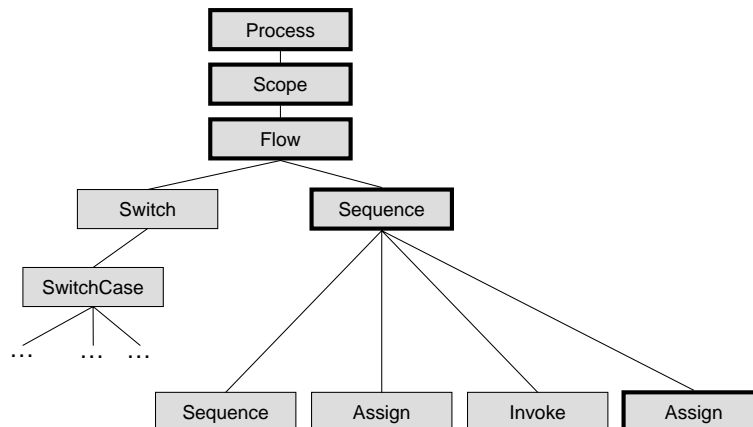


Figure 5.5: Graphical representation of the XPath query to an assignment in the XML tree of a BPEL document. A stronger frame indicates the selection of an element.

channel variables with a usage set to 'once' may actually be used more than once. Unrolling all the possible contingent execution paths that a choreography can take is a virtually impossible quest for XSLT: It is a rule-based translation language and not a fully-featured programming language like Java or C++. However, the implementation needs to be able to detect such errors in the choreography.

The details on the translation of WS-CDL elements are listed in the translation table in Section 5.3.1. For most of the WS-CDL elements, a match in BPEL was found. Problems and complex workarounds in the translation are described in Section 5.3.3.

5.2.3 Process Optimization

Automatically generated processes tend to be “inflated”, in that they contain more elements than necessary. A human designer often would spare out these elements, perhaps except for non-functional elements serving better comprehension, given they do not affect the performance. This section describes a set of optimization strategies that lead the anticipated optimization in the fourth step of the CDL2BPEL algorithm from above. These strategies can be applied generally and are not specific to CDL2BPEL.

In the broader sense, there could be four levels of possible optimization:

0. **No optimization.** In CDL2BPEL, that essentially means that the outputs are process stubs, and not executable processes.

1. **Cleanup:** Superfluously structuring activities are removed, i.e., structures without children. These structuring elements are not valid in BPEL, and represent artifacts of the automatic derivation which have to be removed if the processes are to be executed.
2. **Medium optimization** then touches the structure, in that `bpel:empty` elements with no attached links are removed. Also, constructs like a `bpel:sequence` with one child are resolved: The sequence can be replaced with its child, because there is no need to define an order over the execution of a single element.
3. **High optimization** goes further in that direction. A `bpel:flow` inside another flow can be taken out, when the child flow's links and child activities are moved to the parent flow. Same is true for sequences. This approach is valid, because the respective child structure defines no further semantics. E.g., a process with $Sequence(A, B, Sequence(C, D), E)$ is equivalent to $Sequence(A, B, C, D, E)$. Also, a scope without fault, event, or compensation handlers and no variable definitions can be taken out. In all of these points it is important that attached `bpel:links` also define a partial order and have to be kept. In the case of a sequence, all inbound links to the sequence become inbound links to its first child, while all outbound links now origin the last sequence child.

All proposed optimization above cleanup are purely syntactic. They are not semantic-aware or -changing, in that neither variable values nor control flow are affected by them. Although the proposals are rather simple, applying them at the same time yields major benefit in the number of elements and therefore most likely in the legibility of the processes. However, the optimizing changes remain cosmetic, because any good execution engine implementation should not spend a significant amount of time on the non-functional structures that are removed. Cleanup, in contrast, really makes a difference in terms of execution behavior: without cleanup, the processes will most likely be neither valid BPEL nor ready for deployment.

5.3 Translation Rules and Issues

Part of the whole derivation of executable orchestrations from choreographies is a translation, which follows a set of rules. These rules are described in two translation tables, which show the conceptual mapping for all WS-CDL elements and functions to WSBPEL and partly WSDL. The tables are followed by a description for the WSDL derivation, which is less complex than the one for processes. Several elements cannot be translated to BPEL directly, in part because of the information gap, and in part due to language differences. For the former, CDL-KB lookups

are necessary; for the latter, workarounds are suggested, although they do not always perform in perfect accordance with the choreography. These mismatches in the languages' concepts are outlined and their effects are analyzed at the end of this section.

5.3.1 Translation Tables

The following tables contain the rules for the derivation of BPEL processes (and parts of their WSDL interfaces, where applicable). All WS-CDL elements and supplied functions are contained, but the finer details of the translation are omitted here.

These tables were created by listing all elements and functions of WS-CDL and finding a suitable translation for each of them. Certain child activities and attributes demand for a case distinction with a different translation or additional elements in the output. Some of the translations need to be context-aware, e.g., `bpel:links` can only be used with child activities of a flow, and the translation of functions often needs to take the usage location into account. But for most of the CDL elements, the concept of nesting applies and means that the context does affect the translation.

Nesting here means that elements are children of other elements – in both, WS-CDL and BPEL. Therefore, a `cdl:sequence` can be translated to a `bpel:sequence` independent of the context. The BPEL element corresponding to the CDL parent of the `cdl:sequence` then becomes the parent of the newly created `bpel:sequence`. A restriction here is, that the BPEL translation of a certain CDL element must be able to contain at least as many children as the CDL element. For instance, a `cdl:parallel` may contain an arbitrary number of children, and may not be translated to a `bpel:switchCase`, since this BPEL element can only take one child. Of course, the semantics of CDL need to be matched in all detail by the translation to BPEL. Here, the fact that CDL has its roots only in the π -calculus, and BPEL knows constructs from both, the π -calculus and Petri-Nets, as stated in Chapter 3, helps a lot. Thus, many of the structuring elements from CDL in their basic form have a good, semantically equivalent yet simple match in BPEL.

At this point, the CDL2BPEL algorithm comes into play: Since it performs a depth-first search through the object tree directly corresponding to the XML tree in the CDL document, in the example the order of the sequence's children is kept, and each of them is translated after the sequence. The translations of the `cdl:sequence` then are appended to the `bpel:sequence` in the same order. The location where a BPEL element is added is derived from the respective source element's location in the CDL document. Also, here it becomes clear why all structuring elements are translated to all roles' processes: This technique, followed by an optimization with cleanup is faster and more robust than checking each CDL element with respect to its relevancy for the

roles.

For a number of translations, there would be alternatives in BPEL with the same functionality. The tables present the most simple constructs⁵ in BPEL with the respective semantics, given there were alternatives. An example is the translation of the non-blocking `cdl:workUnits` outside of `cdl:choices` with a guard condition (*cond*) but no repetition condition: such a work unit specifies a behavior equivalent to an *if cond then ...* in many programming languages. In BPEL, there are two suitable sets of elements with the same behavior: a switch with a switch case, having the condition *cond*; or a flow with a link from a `bpel:empty` to the translation of the child activity of the `cdl:workUnit`, where the link has the transition condition *cond*. Both translations are suitable, but the former one has a simpler structure, and thus was chosen here. More complex translations and issues are discussed in detail in Section 5.3.3.

⁵No formal proof for this claim will be given, the choices rely on thought, in-depth analysis, and intuition.

WS-CDL Elements

Each WS-CDL element and the BPEL elements which result from it are listed in the table below. If the children or certain attributes of an element require another translation rule, the distinctive entities are listed in the second column. E.g., a `cdl:assign` is usually translated to a `bpel:assign`, but if the `cdl:causeException` attribute is true a `bpel:sequence`, containing a `bpel:assign` and a `bpel:throw`, is added at the corresponding location in the process.

WS-CDL	Attributes & Children	BPEL	Remarks
Package		Process	A <code>cdl:package</code> has a root choreography, whose child activity becomes the main activity of the <code>bpel:process</code> .
Choreography	possibly with Error Handlers / Finalizers	Scopes with Fault / Compensation Handlers	Note that uncaught <code>cdl:exceptions</code> are propagated up to the enclosing <code>cdl:choreographies</code> - as is the case with faults in <code>bpel:scopes</code> . Uncaught faults at the <code>bpel:process</code> will terminate its execution.
	Coordination	Explicit communication	in the process' Compensation Handler with <code>enableInstanceCompensation=true</code> . Given its availability, a WS-Coordination protocol should be used to solve this issue.
Information Type		<code>xsd:complexType</code> in the WSDL	Use this type in the respective <code>bpel:variables</code> and <code>wSDL:message</code> parts.
Token		Correlation Set and Property	Correlation set references at usage locations of the corresponding variables, the property is added to the WSDL.
Token Locator		Property Alias	in the WSDL
Role Type		One Process per role type	Also a <code>bpel:role</code> in a <code>bpel:partnerLinkType</code> .

Relationship Type		Partner Link & Type	The <code>bpel:partnerLinkType</code> is specified in the WSDL.
Participant Type		—	Only the roles of the <code>cdl:participant</code> type that are relevant to a certain partner need to be included via <code>bpel:partnerLinks</code> in the respective <code>bpel:partner</code> element. See Section 5.2.2 for a justification of this translation.
Channel Type		—	Not translated, only used for reference to the identity token. See Section 5.3.3 for details.
Variable		Variable	Replace the <code>cdl:informationType</code> as described above, specify additional message variables in BPEL for <code>cdl:variables</code> used in interactions.
Interaction		Invoke and Receive for each <code>cdl:exchange</code> at the respective roles	If the <code>cdl:action</code> is <code>''request''</code> , then add a <code>bpel:invoke</code> at the <code>cdl:fromRole</code> and a <code>bpel:receive</code> at the <code>cdl:toRole</code> , if the <code>cdl:action</code> is <code>''respond''</code> switch the to- and from-roles. Also add the variables needed.
	timeout or align	scope around the invoke / receive	Also acknowledgement exchange, fault and compensation handlers realizing timeout and rollback
	record	assign	The <code>bpel:assign</code> is a sibling inside a sequence, if the value of the <code>bpel:when</code> is either <code>''before''</code> or <code>''after''</code> , if it is <code>''timeout''</code> , the <code>bpel:assign</code> is a child of the <code>bpel:onAlarm</code> inside the <code>bpel:pick</code> .
	more than two exchanges	CDL-KB lookup	More than two exchanges means that one of several responses may be sent back. However, the condition which one will be chosen is not stated at the choreography level.

Sequence		Sequence	Since the sequence in BPEL only orders the activities of one partner, the ordering of activities of multiple partners can only be guaranteed if there are intermediary interactions ⁶ .
Parallel		Flow	
	Work Units inside with certain guards	Links in the Flow	E.g., a <code>cdl:isVariableAvailable</code> (with <code>cdl:block=true</code>) can define a partial order of two activities in a <code>cdl:parallel</code> .
Choice		Switch or Pick	depending on the child elements.
	only non-blocking Work Units	Switch	<code>bpel:switch</code> is designed for directly observable conditions, which is the case for <code>cdl:workUnits</code> with <code>block=false</code> . The <code>cdl:workUnits</code> translate to <code>bpel:switchCases</code> .
	only non-blocking Work Units with an enclosed interaction ⁷ as child element for all cases	Switch at the sender, pick at the receiver	The receiving role learns which branch is chosen through incoming messages ⁸ , which is done by a <code>bpel:pick</code> with <code>onMessage</code> events. The sender can observe the conditions directly, thus a switch is appropriate.
	only blocking Work Units as children	Pick	<code>bpel:pick</code> offers an event-based, deferred choice, which corresponds to blocking Work Units.
	blocking and non-blocking Work Units as children		Such hybrid cases are subject to Section 5.3.3.
	children of any other type than WorkUnit	CDL-KB lookup	The branching criteria are non-observable at the choreography level ⁹ .

⁶These interactions could be inserted automatically where necessary. An implementation could provide this ability, e.g., as a configuration setting.

⁷Any set of structuring activities with an interaction as starting point of the activities at the receiving role.

⁸Each case must use a different message type.

⁹The WS-CDL specification [54] explains for this case: “..., it is assumed that the selection criteria for the activities are non-observable.”

Work Units	guard≠' ', repeat=false, block=false	Switch with Switch Case enclosing the WU children and an Otherwise containing an Empty	Only valid in non-effecting contexts of the Work Unit (not inside a choice, an exception block or similar). The bpel:switchCase's condition is the (converted) cdl:guard condition for the respective role(s).
	repeat ≠ false	While	Set the condition with respect to the semantics ¹⁰ .
	guard=' ', repeat=false	—	Such WorkUnits are executed exactly once and have no effect. Of course, the child activities are translated.
Silent Action		CDL-KB look-up	Partner-internal actions.
No Action		Empty	
Assign		Assign	
	causeException=true	Throw	A bpel:sequence of an assign followed by a throw.
Perform	bind	assign	Two bpel:assigns inside a bpel:sequence in the bpel:scope: One at the beginning from the enclosing to the enclosed scope, and one at the end in the opposite direction.
Exception Block		Fault Handlers	cdl:exceptions must be propagated to all parties of the Choreography, therefore the bpel:catch triggers the communication for the current scope explicitly ¹¹ .

¹⁰Note, that the bpel:while is a *while cond do ...* where *cond* is the content of the bpel:condition attribute, whereas the cdl:workUnit is a *if cond1 do ... while (cond1 && cond2) do ...* with *cond1* being the cdl:guard condition and *cond2* being the cdl:repeat condition. Thus, the translation in BPEL includes a boolean variable *bVarX* which is set to true before the bpel:while and to false inside the while. The bpel:condition then becomes *cond1' && (cond2' || bVarX)*, where *cond1'* and *cond2'* are the translated boolean conditions of *cond1* and *cond2*, respectively.

¹¹WS-Transaction and/or WS-Coordination can help here, given they are available in the runtime environment. Otherwise, the communication is specified in a compensation handler, which is called from the various fault handlers.

Exception Work Unit		Catch inside a Fault Handler	Exception WUs have to be non-blocking and non-repeating, allowing for this translation. All exceptions must be communicated to all parties. Use WS-Coordination for that purpose, if available.
Finalizer Block		Compensation Handler	at the process with enableInstanceCompensation=true
Finalizer Work Units		Switch or Pick	in the process' Compensation Handler
Finalize		Compensate	There must not be any other <code>bpel:compensate</code> activity in a fault handler that points to the compensation handler, so that it is not executed in any other case but a normal completion of the process.

Note that all children of a flow which are the target of conditional `bpel:links` require the `suppressJoinFailure` attribute set to true in order to elide the exception disabled links would cause otherwise.

WS-CDL Supplied Functions

Below is the conceptual translation for the WS-CDL supplied functions.

WS-CDL Function	BPEL	Remarks
All CDL supplied functions: Role Attribute		Add the respective activities only to the referenced roles.
<code>getCurrentTime</code>	—	No match ¹² (only in certain engine implementations).

¹²Could be realized by a supporting Web Service at each partner, or, depending on the execution engine, user-defined XPath functions could implement the desired behavior.

getCurrentDate	—	No match ¹² (only in certain engine implementations).
getCurrentDateTime	—	No match ¹² (only in certain engine implementations).
hasDurationPassed	for	In a bpel:wait or bpel:onAlarm event, depending on the context.
hasDeadlinePassed	until	In a bpel:wait or bpel:onAlarm event, depending on the context.
createNewID	—	No match ¹² .
getVariable	getVariableData() or Variable, Part, Query	In a copy-to, the function translates to the variable, part, and query attributes in BPEL, in all other cases the bpel:getVariableData() suffices.
isVariableAvailable	while with enclosed wait or static false	Replace with a static false at any role for which the variable is undefined. For all other roles: Initialize the variable with NULL, then add a bpel:while with condition <code>''variable = NULL''</code> and an enclosed wait, e.g., for 5 seconds, to the process and make sure, it precedes the targeted activity (e.g., a link in a flow or a surrounding sequence). More complex cases are studied in Section 5.3.3.
variablesAligned	Communication and check	Requires to obtain the variable value of the second variable via a bpel:invoke, followed by comparing it to the value of the first variable.
getChannelReference	—	No match (and none needed).
getChannelIdentity	—	No match (and none needed).
globalizedTrigger		The respective expression for each role only.
hasExceptionOccurred		For each role and fault type used, a boolean variable can be introduced that is set to true, as soon as the exception occurs. If the function is used in a blocking cdl:workUnit, then it is treated analogous to the isVariableAvailable function.

5.3.2 WSDL generation

Per `cdl:package` and `cdl:role` a `wsdl:service` is generated. The corresponding `wsdl:operations` and `wsdl:portTypes` are generated from `cdl:interactions` and CDL-KB patterns with `bpel:receive`, where the current role receives messages. A `cdl:interaction` always produces an operation (with respective port type and partner link type) at the receiver, independent of the action being "request" or "respond". Therefore, all interactions are asynchronous and will not cause faults due to timeouts of http requests.

There has to be a `bpel:partnerLinkTypes` for each `cdl:relationship`, but the partner link types are already represented by using the relationships in `cdl:interactions`. In contrast to `wsdl:portTypes` and `wsdl:operations`, `bpel:partnerLinkTypes` have to be present at the calling and the called role, and are therefore generated at each of the involved roles of a `cdl:interaction` as well as at CDL-KB patterns with `bpel:invoke/receive/reply` activities or `bpel:onMessage` events. All `bpel:partnerLinkTypes` that are used by role are defined in its own WSDL file.

5.3.3 Complex Cases of Mismatches in the Translation

There were several points in the translation table, where the concepts of CDL and BPEL diverged a lot. These cases are repeated here, together with the suggested solution.

Interaction-Related Issues

A `cdl:interaction` in its simplest form states that a message is sent from role *A* to role *B*. However, there are numerous settings which complicate the translation, of which the outcome for the "sender" role is shown in Figure 5.6. As such, the attribute "align" offers transactionality for interactions and attached `cdl:records`. A `cdl:record` is very similar to a `cdl:assign`, only that it is specified inside an interaction and its outcomes are protected by the alignment attribute. Therefore, the translation to BPEL needs to have activities for making backups of variable values, and respective variables that can serve this function. In the figure, the scope surrounding all activities contains the backup variable definitions and fault handler: any fault leads to a message being sent to the partner as well as the execution of the compensation handler, where the initial backup of variable values is copied back to the variables. Since a fault at the partner requires the rollback as well, there is an event handler, listening for a message and, when executed, also copies the values back to the original variables.

The sequence of activities on the left-hand side of the scope thus starts with an assignment,

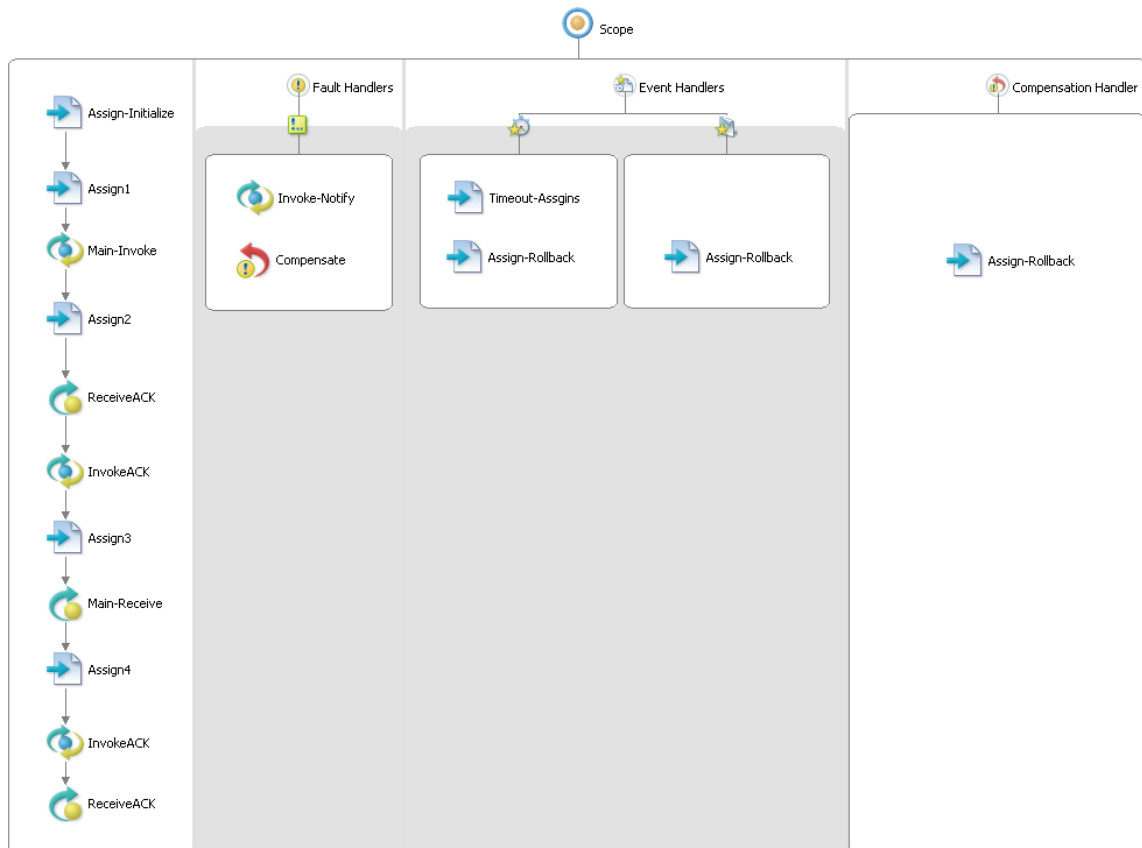


Figure 5.6: Visualization of the BPEL activities at one involved role resulting from the translation of a `cdl:interaction` that makes use of the attribute “align” and has a `cdl:timeout` and multiple `cdl:record` children.

where the original values of the variables used are copied to the backup variables. This set of BPEL activities corresponds to a `cdl:interaction` with two `cdl:exchange` elements, meaning that there is a request message followed by the respective response. Thus, the `bpel:sequence` contains a “Main-Invoke” and a “Main-Receive”. If both of the `cdl:exchanges` each have one `cdl:record` that is executed before and one after the exchange at the role of interest, these records are mapped to “Assign-1” through “Assign-4”. After each exchange and its related records finished, acknowledgements are sent, but only if needed. These acknowledgements are the two sets of “ReceiveACK” and “InvokeACK”.

Last but not least, the `cdl:timeout` is translated to another event handler with a `bpel:onAlarm` event. If this event is triggered because the interaction times out, there can be specific `cdl:records`, which are translated to the “Timeout-Assigns” in the event handler. Also, the interaction does not complete successfully and thus the variable values from the backup need to be restored. Since the timeout comes from the choreography, the event will be triggered at the same instant in both involved roles - making a message exchange in this case unnecessary.

Problems With Tokens and Channels

In CDL, tokens and token locators used for channel identity match BPEL’s properties and property aliases with correlation sets fairly well: tokens describe that a certain part of a message is used to identify the message conversation, and consequently the addressed process instance, while token locators describe where the token part is found in a message. But in contrast to CDL, in WSBPEL the activity that initializes a correlation set has to be defined. Usually, that is no issue: the first usage of a correlation set initializes it. However, there are cases for which the first usage is unclear, e.g., if there are three potential first usage points inside a `cdl:parallel` without further ordering constraints. At design time it cannot be determined which one will happen first. Using an un-initialized correlation set causes a fault in BPEL - just as initializing and already initialized correlation set does. Thus, the translation here is not obvious.

Since one instance of the collaborative process corresponds to exactly one instance of each executable process, only one identifier for the whole message conversation for an instance of the collaborative process would be sufficient. This identifier could be a simple counter value, or the date and time the instance was started. An assumption here is that the namespace of the choreography is unique to it. This can be guaranteed via namespace conventions in TrustCoM. Thus, a simple identifier may serve as the content of the single one correlation set that is generated for each process. The solution could generate this correlation set automatically, or leave it to the choreography designer to specify only one token which is used for correlation of

all channels. In the latter case, a small extension of WS-CDL could map `cdl:informationTypes` to XML Schema message types, so that an information type can specify which XML Schema type should be used for normal variables and of which type message variables for it are, allowing for token locators on the messages. The resulting single correlation set could then be initialized by the messages used for executable process initialization. This pragmatic solution solves the issues related to token translation.

CDL's channels are not translated in the translation tables above. In general, they are used as a reference to the endpoint of communication and have their origin in the π -calculus. There, an unknown entity's endpoint can be communicated by transmitting its channel information to another partner. E.g., role *A* knows the channel to role *B*, and passes this channel information to *C*. Then, *C* can send messages to *B* as well. In CDL, all roles are known, and there is no notion of having multiple instances of a role. In the example choreography in Figure 1.1, there is one `AnalysisPartner` role, which means there will be exactly one `AnalysisPartner` during execution. Thus, the channels do not serve for the distinction of multiple instances of a role. Due to the environment of VOs in TrustCoM, where all partners and the roles they play are known before the derivation takes place¹³, there is no need for dynamically transmitting endpoint information during the execution of the collaborative process. Therefore, channels do not have to be translated. If all channels use the same token for conversation identification, the channels do not even have to be used for the reference to this token during derivation.

Issues With Hybrid Choices in CDL

Especially in hybrid cases, where both blocking and non-blocking `cdl:workUnits` occur in the same `cdl:choice` element, the translation to BPEL is hard. A very challenging example is shown in Figure 5.7. There are nine work units in the exemplary choice, three non-blocking and six blocking ones. The guard conditions of the six blocking work units cover all CDL supplied functions, of which the usage makes sense in a blocking work unit: `hasDeadlinePassed`, `hasDurationPassed`, `isVariableAvailable`, `hasExceptionOccurred`, and `variablesAligned`.

Explanation of the resulting BPEL activities

The BPEL activities that result from the `cdl:choice` in Figure 5.7 are depicted in Figure 5.8. Although the structure of the process part is visualized in a rather intuitive way, details like the transition conditions of the links or the contents of the assignments cannot be seen. These

¹³Please see Section 5.4.2 for details.

```

01 <choice>
02   <workUnit guard="var1 = x" block="false">
03     <cdl:activity1/>
04   </workUnit>
05   <workUnit guard="var2 = y" block="false">
06     <cdl:activity2/>
07   </workUnit>
08   <workUnit guard="hasDeadlinePassed('dt3', '')" block="true">
09     <cdl:activity3/>
10   </workUnit>
11   <workUnit guard="hasDurationPassed('dur4', '')" block="true">
12     <cdl:activity4/>
13   </workUnit>
14   <workUnit guard="isVariableAvailable('var5', '')" block="true">
15     <cdl:activity5/>
16   </workUnit>
17   <workUnit guard="isVariableAvailable('var6', '')" block="true">
18     <cdl:activity6/>
19   </workUnit>
20   <workUnit guard="hasExceptionOccurred('exception7')" block="true">
21     <cdl:activity7/>
22   </workUnit>
23   <workUnit guard="variablesAligned('var8.1', 'var8.2', 'RelAB')" block="true">
24     <cdl:activity8/>
25   </workUnit>
26   <workUnit guard="otherRolesDecision" block="false">
27     <sequence>
28       <interaction>...</interaction>
29       <cdl:activity9/>
30     </sequence>
31   </workUnit>
32 </choice>

```

Figure 5.7: CDL source code: Choice with both blocking and non-blocking Work Units.

details are explained here.

The nine wild cards for CDL child elements of the work units, marked as `cdl:activity1` through `cdl:activity9` in Figure 5.7, are depicted as nine `bpel:empty` activities, A1 through A9. The cases for which they stand can be divided into three main groups: The directly observable cases, the message or time-related events and the variable value-related cases. There are other activities surrounding them, which are responsible for the control flow.

The first activity is a `bpel:assign`, in which the variables are initialized. That means all variables whose availability might be checked throughout the process are set to `NULL`. A variable is considered available as soon as its value differs from `NULL`. The assign activity will be among the first activities of the whole process. Most likely, other activities follow the assign, shown as a `bpel:empty` with the name “OtherActivities”, before the `bpel:flow` is reached.

The `bpel:flow` corresponds to the `cdl:choice` element. Its first child, a `bpel:empty`, serves as the source of conditional links to other activities. The links to A1 and A2 correspond to the first two `cdl:workUnits`, in that the transition conditions represent the guard conditions of the work units. But since the work units in a `cdl:choice` are mutually exclusive whereas the links in a flow can be enabled concurrently, the second transition condition contains the inverted first condition as well as the second work unit’s guard. In the example, the link from the `bpel:empty` to A1 has the condition “`var1 = x`”, whereas the link to A2 is only enabled when “`not(var1 = x) and (var2 = y)`”.

Also for the reason of mutual exclusivity, the link to the other cases, i.e. between the `bpel:empty` and the `bpel:assign`, is only enabled if “`not(var1 = x) and not(var2 = y)`”. So, if either A1 or A2 is executed, all of the other cases are deactivated.

The assign activity initializes the variables for the `bpel:while`, where the variable availability and other criteria are checked regularly. Among others, a boolean variable named “`terminateWhile`” is set to false. It serves as a break statement in the while. As mentioned earlier, the `bpel:while` is a *while cond do* The condition *cond* consists of several boolean expressions, all connected with an “or”. One of these expressions is the value of “`terminateWhile`”. The assignment activities enclosed in the while and named “`TerminateWhile`” each set “`terminateWhile`” to true, thus ending the repetition of the loop. The links having the source in the while all are deactivated if this variable is true. This construction makes sure that if either one of the activities A3, A4, or A9 is executed, no other activity in the flow is pursued.

Inside the `bpel:while`, there is a `bpel:pick` with four event handlers: three `onAlarm` and one `onMessage` handler. The right-most `onAlarm` handler contains an assign activity and a synchro-

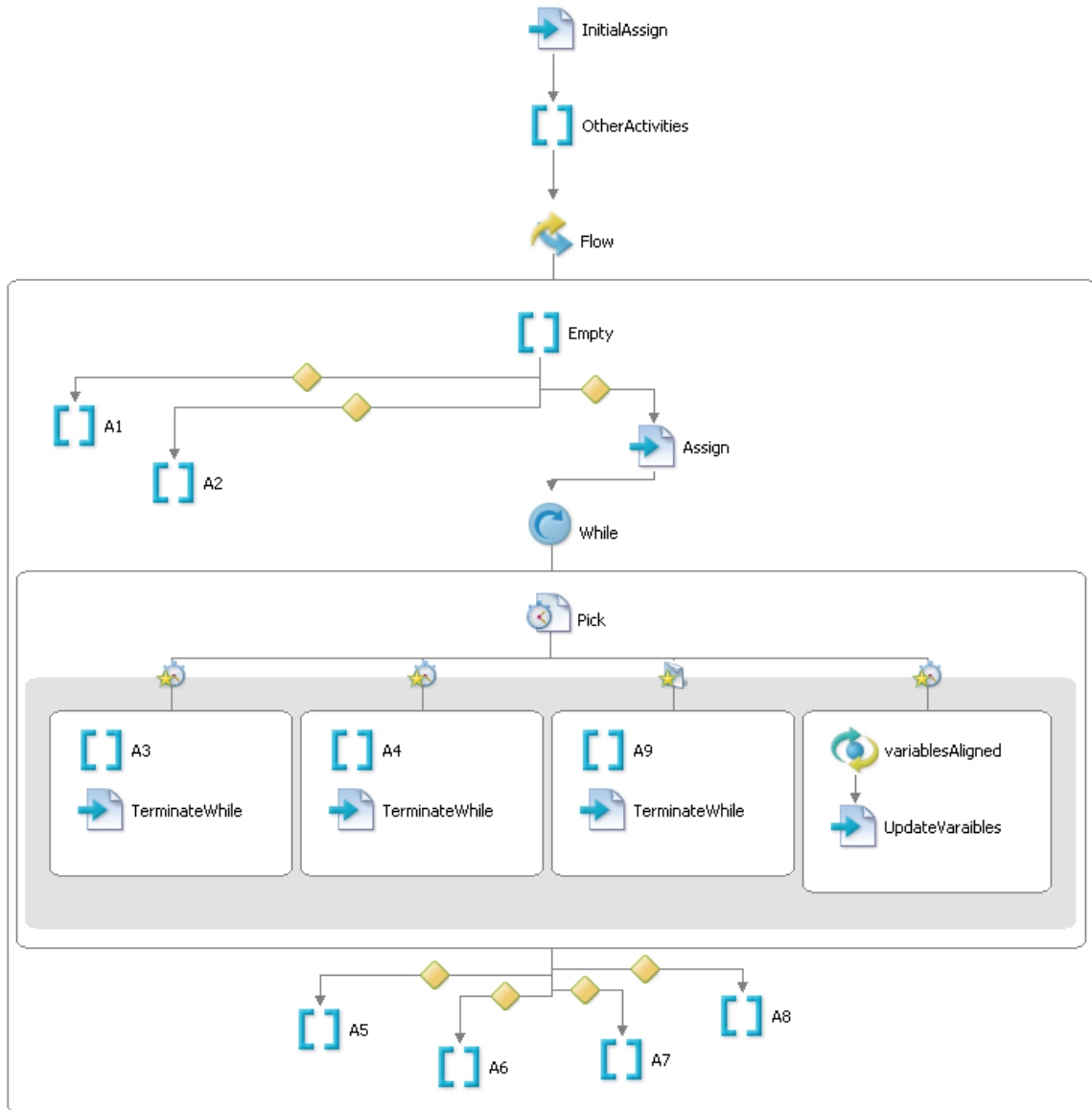


Figure 5.8: Visualization of the BPEL activities resulting from the translation of the CDL snippet in Figure 5.7 by the CDL2BPEL Service. This graph is a snapshot from ActiveWebflowTM Professional by ActiveEndpoints, a graphical designer for BPEL processes.

nous invoke for retrieving the value of "var8.2" from the respective partner. It is executed after a small amount of time¹⁴ and usually causes the while to repeat. In the assign, the value of the local variable "var8.2-local" is set from the content of the response message of the synchronous invoke.

The other conditions for the loop repetition are the values of variables, in this example "(var5 != NULL) or (var6 != NULL) or (exception7occurred) or (var8.1 = var8.2-local)". In the interplay with the above-mentioned onAlarm handler, the availability of the variables "var5" and "var6", the occurrence of the exception¹⁵, and the equality of "var8.1" and "var8.2" from the partner role are checked every n seconds.

A3 and A4 are inside of `bpel:onAlarm` event blocks. The parent of A3 has the parameter `until="dt3"`, which represents the guard attribute "`hasDeadlinePassed('dt3', '')`" of the respective `cdl:workUnit`. In contrast, the onAlarm surrounding A4 has the attribute `for="dur4WorkCopy"`. This variable is initialized with the value of "dur4". Each time, the other timeout occurs and the while is repeated, the value is decreased by the delay of one loop execution. E.g., if the value of "dur4" is 35s and the while is repeated every ten seconds, then "dur4WorkCopy" takes on the values 35s, 25s, 15s, 5s and is executed, since the event occurs before the other timeout event.

The fourth child of the `bpel:pick` is an `onMessage` handler. It corresponds to the ninth work unit in the `cdl:choice`, whose guard condition is directly observable by another role, but not this one. However, the work unit contains a sequence, whose first child is an interaction. Assume that the role for which the BPEL activities are shown in Figure 5.8 corresponds to the role that receives a message in the mentioned interaction. Therefore, this case is modeled as an `onMessage` event here.

Leaving out several of the cases still allows for a solution constructed in the suggested way to function. However, some cases can be solved easier, as stated in the table in Section 5.3.1.

Altogether, this set of BPEL activities allows the execution of each of the cases and guarantees that the execution of each single case prevents all other events from taking place. In other words: Each case can occur, but the execution of more than one case is precluded. Still, the solution is not perfect, in that there are a number of differences between the anticipated behavior of the CDL vs. the actual behavior of the generated BPEL, as discussed in the following.

¹⁴A duration of ten seconds seems to be reasonable for many domains.

¹⁵The variable "exception7occurred" will be set to true by the respective event handler, which can be triggered concurrently, e.g., by an incoming notification from a partner.

Differences between the theoretical behavior of the CDL vs. the generated BPEL's behavior

If the executable BPEL process, of which the shown BPEL snippet is part of, is deployed in a BPEL engine, the runtime behavior may differ slightly from the behavior anticipated by the choreography. The following list contains points of known divergence:

- The order in which events are processed may differ: Where a `cdl:choice` is supposed to react to the event occurring first, the shown BPEL suggestion checks for `isVariableAvailable`, `hasExceptionOccurred`, and `variablesAligned` only every five seconds. Therefore, for example a `hasDurationPassed` event may be handled instead of a `isVariableAvailable` event that happened three seconds earlier.
- Due to communication delays, the default timeout block of the `bpel:pick` may cause the duration event to occur later than expected, since the duration is recalculated and reset at each loop cycle and not measured from when the flow is entered.
- Also because of the communication delay, we cannot make sure an arriving message for the `onMessage` event is processed. That could actually result in missing an event at a certain role completely.
- The `variablesAligned` event will most likely not take place at the exact same instant in the two involved roles.
- All events are prepared only after the cases with the immediately observable conditions are checked. If, for example, a work unit realizing a deadline event precedes the other cases in the order of the elements in the CDL document, and the deadline already passed by, this case should be executed. Due to the differing notation in BPEL, the directly observable cases are checked first and possibly executed, essentially switching the priorities of the cases.

These differences emerge from the differing event models of WS-CDL and WSBPEL. There are only two kinds of events in BPEL: incoming messages and time-related events. Additionally, `bpel:links` offer a means to define a partial order over two activities inside a `bpel:flow`. In contrast, WS-CDL knows message and time-related events as well as blocking work units in general. A blocking work unit is executed when its guard condition evaluates to true. The guard condition can be one of several functions, such as `cdl:isVariableAvailable` or `cdl:variablesAligned`. These functions are hard to match in BPEL and the here suggested workaround suffers from the language differences.

5.4 Deployment and Confidentiality Implications

Automatic deployment and execution pose high burdens on the generated processes and demand some infrastructure in terms of available services. This section describes how deployment and execution further affect the derivation as well as the integration of the presented solution into the TrustCoM framework.

5.4.1 Deployment and Execution of Generated Processes

As outlined in the problem statement for this thesis, the goal are deployable, executable processes. There are certain criteria which need to be met by a process in order to fulfill these requirements. Apparently, the processes have to be valid and thus all the WSDL elements have to be specified at the right spot and the types, names and namespaces need to match with the information in the process definition. This applies more to a valid implementation than to the concept, but without a prototypical implementation it would be hard to show that all the required information is either available or can be generated automatically.

For process initialization, BPEL requires an initial receive operation. This means, that some outside entity has to send a message to the process in order to produce a new instance of the process¹⁶. For the presented solution, that means that either there has to be some role in the function of that outside sender, which should not be translated to BPEL¹⁷; or another way has to be specified for instance creation. Another way, in that sense, would be to specify a certain variable name, and assume that the variable is present in any choreography that is passed as input to CDL2BPEL. The collaborative process then always starts when the variable is initialized by an incoming message at the role for which the variable is defined. Both solutions require a convention on a CDL2BPEL-specific startup mechanism. This semantic-changing extension is necessary, but in conflict with the CDL standard in general and its closed-world assumption in particular.

In any of the cases, the partners' process instances need to be triggered as well. From a global point of view, one instance of the collaborative business process relates to exactly one process instance at each of the VO members. Here, we favor a concerted startup, such that one can be sure

¹⁶Again, the BPEL code represents a process definition, basically a template, from which instances may be run. These instances may be long-running and follow the definition, e.g., in that the current data of the instance is used when validating a branching condition. Since there may be multiple instances, the message conversation needs to identify the respective instance to which it belongs - otherwise the behavior is undefined.

¹⁷This role's activities cannot be translated to a BPEL process, because this process then would have to be started via a receive as well. Thus, some other program would specify the activities of this role - most likely only the sending of the initial message.

that the actual work part of the collaborative process only begins if there are running instances of all executable, private processes. This preference is related to the message communication across enterprise boundaries: Say, the collaborative process starts at roles *A* and *B*, and they do much work before the expertise of role *C* is required. But then the process instance of *C* cannot be created, because the firewall of the respective VO member forbids the message passing due to false configuration. Then, roles *A* and *B* need to catch the related fault and roll back to the state at the beginning of the process. Potentially that rollback is not even possible, e.g., because the production of various goods already begun. It is thus far more handy if simple faults like the one mentioned are caught before productive activities are started. Nevertheless, that does not help if the firewall is reconfigured while the processes are already running, especially if they are running for days or even weeks.

In terms of deployment, each executable process that is deployed requires its BPEL definition; the matching WSDL with BPEL's extensions for partner link types and properties/property aliases, but without binding information; some engine-specific deployment description; and finally the WSDLs of the other VO members with whom it communicates, this time without the BPEL extensions but with binding and service information. The information about endpoint addresses is retrieved from other TrustCoM services, as explained below. A proxy should be used, and the actual endpoint information should be registered with it during deployment - which requires altering the WSDLs' binding information. The proxy should be security-aware, relying on standards such as WS-Security [30], WS-Reliability [29], and more.

5.4.2 Integration With Other TrustCoM Services

There are several other components which are needed in the CDL2BPEL algorithm or thereafter. Figure 5.9 shows them and the communication between the CDL2BPEL service and the other components. In the shown VO, there are two members and a host. The host is the central administrative instance in a VO, knowing about the members and state of the VO, and more. The members are the participants in WS-CDL. Note, that there is no restriction on the number of `cdl:participants` that a member may represent. Another way to think about it is that there are distinct software packages, one for members, one for hosts, and one for initiators¹⁸ (which is a special kind of member, but the distinction is not relevant here). Each package brings along several components and, thus, features the software owner may use. The example in the figure

¹⁸It is very important to distinguish between the VO concepts and choreography roles. E.g., a choreography may know an "Initiator" role, and each VO knows such a member as well. However, the choreography is enacted only during the Operation phase, and its initiator is the one that starts this phase - which in general can be any member, in particular not the initiator of the VO.

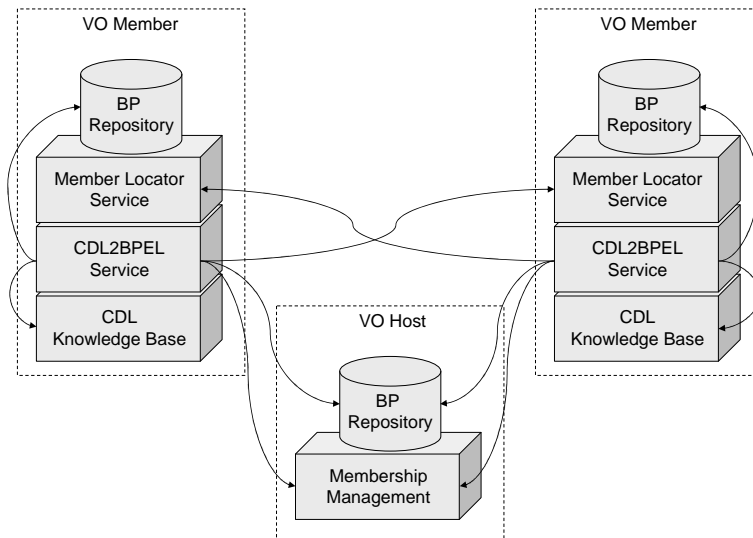


Figure 5.9: Setup of the relevant components in a VO, with communication from CDL2BPEL services to other components.

shows only the relevant components for the host and two members.

In order to get deployable, executable processes, the shown services are used by CDL2BPEL. First, the choreography is retrieved from the host's BP Repository. The BP Repositories are basically databases with business process content and a TrustCoM-specific identity model for data addressing. When the choreography is present, the member's CDL2BPEL algorithm derives the private processes for the roles the member plays by querying the local CDL-KB for private process parts whenever necessary.

For the purpose of deployment, the deployment descriptor document and the WSDLs of the partners' processes need to contain endpoint references, under which the processes of the partners may be called. This is not necessary for all partners, only for the ones with which a process actually will communicate. Here, the assumption is made that all partners use a known BPEL execution engine¹⁹. That means, that the way the engine makes deployed processes available is known, i.e., the role name, namespace of the choreography, and the engine's type and endpoint reference are sufficient information to construct the service name and the endpoint reference of the deployed process. Say, the process for role *A* will communicate with the process of role *B*, then the latter two pieces of information about the member playing role *B* are required for the deployment *A*'s process. Before writing out the generated documents, CDL2BPEL requests from

¹⁹This assumption is not too hard, since VOs in TrustCoM are enabled by a certain software. This software needs to contain a large set of modules and services, and most likely all VO members will use the exact same software package. Thus, the assumption comes down to the execution engine being part of this software package.

Membership Management at the host which member plays which role in the current VO and gets back their identities. With the member identity, more details on the member are retrieved, again from the Membership Management service. One of the details is the Main Endpoint Reference, the endpoint information under which the Member Locator Service of the respective member can be called. From the Member Locator Service, the type and endpoint reference of the BPEL engine in use can be requested. Besides, role name and namespace are present in the choreography.

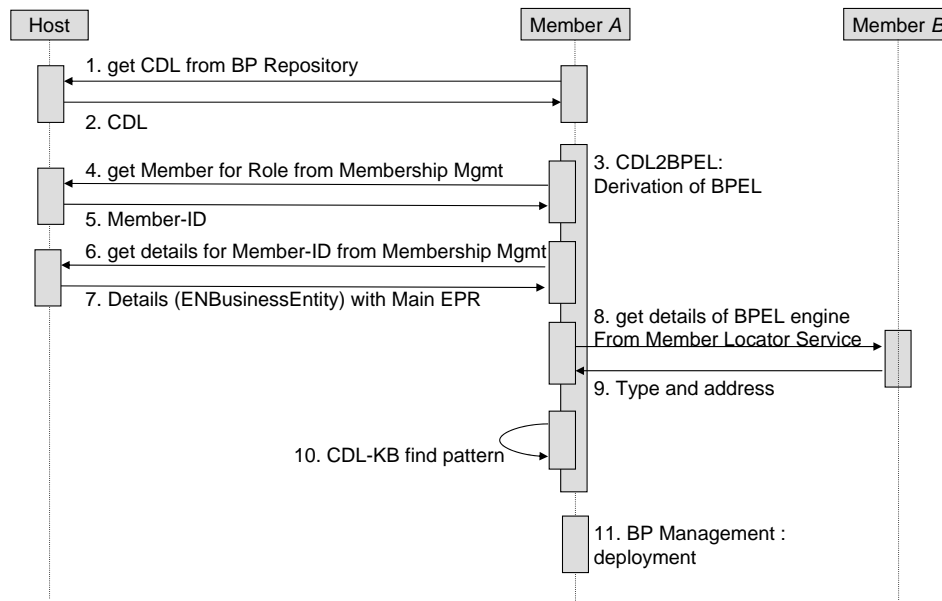


Figure 5.10: Sequence of calls in the transition from the formation to the operation phase of a VO

From the chain of queries and the information in the choreography, the endpoint reference of the partners' processes can be constructed, even before they are actually deployed. Note, that this, too, is a major benefit of constructing BPEL and WSDL from a choreography: Take the example where the processes of roles *A* and *B* need to call each other. For *A*'s deployment, the WSDL with the binding information of *B* is required and needs to be transmitted from *B* to *A*. This WSDL is usually generated by the execution engine after *B*'s process is deployed. But the deployment of *B*'s process in turn needs the WSDL of *A*'s deployed process. The presented approach thus makes the transmission of the WSDLs unnecessary and solves the chicken-and-egg problem of the mutual dependency elegantly.

After all information is retrieved, the output documents are written and stored at the local BP Repository, which hands back an ID for each document. These IDs are sent to the VO

management suite, which called CDL2BPEL. With this information, the local BP Management service is invoked, which retrieves the documents from the BP Repository, packages them in the engine-specific way and deploys them to the execution engine. If this procedure is finished at all partners, the operation phase of the VO is entered, and an instance of the collaborative process can be triggered each time the need for it arises. The sequence of these calls explained above is visualized in Figure 5.10.

Ideally, there should be dynamic, SOAP aware messaging subsystems in each of the partners' domains, so that during deployment the actual endpoint of a partner is registered with the local proxy, and messages are sent to it instead of the partner during execution. The proxies would also be helpful for VO evolution, e.g., if a VO member was replaced with another candidate, because, say, its performance violates the Service-Level Agreement (SLA) of the VO. The details of these mechanisms are outside the scope of this work, but deserve being mentioned since the solution presented allows for their usage. Note, that VO evolution poses a challenging problem to business processes, because the replacement of member X with Y requires to bring Y 's process instance to a state from which it can replace X , and, most likely, the other involved members need to roll their processes back to a defined state, in order to bring the collaborative process to an overall consistent state. In the worst case, the processes need to be terminated completely, and a new instance of the collaborative process has to be started. The cost of termination may be high, especially if e.g, the production of physical goods was initiated already, and should be accounted for in the decision on the replacement of a member. Again, these problems are outside of the scope of this thesis, but form a challenging starting point for future work.

The public views for the negotiation during the VO's formation phase are derived by the initiator's CDL2BPEL service and only require its local BP Repository. Membership Management, Member Locator Service, and the CDL Knowledge Base are not needed, because neither endpoint information nor local knowledge is present in the public views.

Due to the fact that the installation of the services containing private knowledge is local, the confidentiality requirements can be easily met by simply allowing outside access only to the Member Locator Service. The CDL Knowledge Base and the BP Repository as well as the execution engine should be only accessible by authorized users in the respective member's organization.

Other related services in TrustCoM are UML2CDL and the Notification Service. The Notification Service offers a publish-subscribe mechanism for dynamic message routing in a VO. For instance, VO members report their readiness for transition to the operation phase over it. UML2CDL transforms annotated UML Activity Diagrams in XMI²⁰ exchange format to

²⁰XML Metadata Interchange (XMI), see [32]

WS-CDL descriptions, and thus enables graphical modeling in a well-known modeling notation. The user should still know WS-CDL, in order to design valid choreographies. Nevertheless, UML2CDL and CDL2BPEL together mean that there are no direct touching points of a user to WS-CDL, except if the user wants to alter the choreography manually before applying CDL2BPEL. However, UML2CDL also increases the chance that invalid choreographies are designed. Thus, the choreography should be validated before the VO relies on its correctness, i.e., before the formation phase is entered.

5.5 Chapter Summary

This chapter suggests a conceptual solution to the problem of this thesis. The CDL2BPEL algorithm describes how the different parts of the concept work together: It applies the static mapping component from the translation tables as well as the dynamic component in terms of the Knowledge Base. The generated processes are optimized and enriched with deployment information. For the latter and other purposes, the solution is integrated with other components of the TrustCoM framework. The requirements of executability of the output processes and confidentiality of internal implementations can be satisfied with the described solution.

The presented conceptual model was implemented in a prototype, which is subject to the next chapter.

Chapter 6

Implementation Details

This chapter gives an overview over the prototypical implementation of the solution presented in the preceding chapter, discusses details about the implementation of the CDL2BPEL Service, and outlines how the prototype could be improved. Since the implementation became very time-consuming, it could not be finished yet. However, the prototype still shows that the solution is suitable for the problem in the given domain.

6.1 Overview

For the CDL2BPEL algorithm, the Knowledge Base, and the BPManagement Service, proof-of-concept prototypes have been implemented as Java Web Services. The business processes in Figure 5.1 demonstrate graphically the result of applying the implementation to the CDL document belonging to the choreography from Figure 1.1. The XML source code used for the diagrams can be found in the appendices.

The prototype subsumes and requires a set of software components, for which Figure 6.1 shows the relations via the data flow. Although all of these services are explained in short below, the implementation for this thesis only consists of the CDL2BPEL, CDL-KB, and BPManagement services.

- A **UML Modeler** tool is used to model the choreography graphically as a UML Activity Diagram.
- The **UML2CDL Service** translates the diagram to a WS-CDL representation.

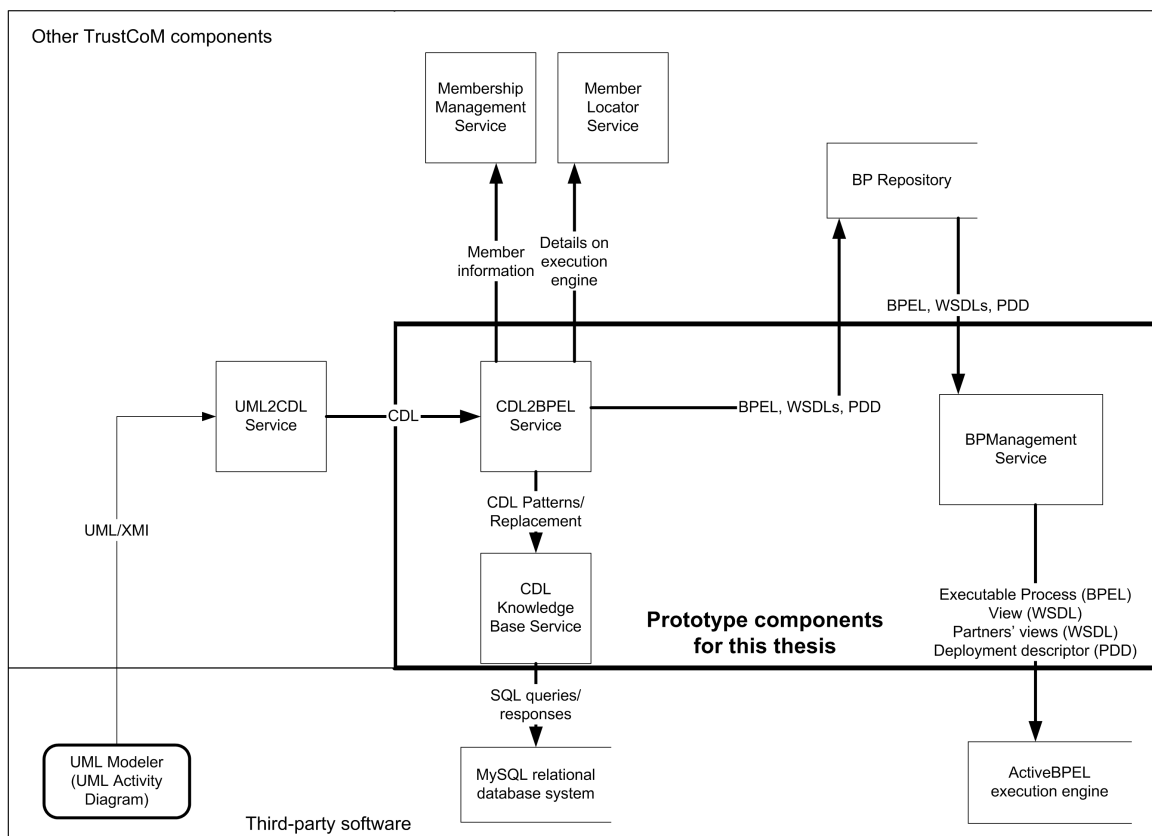


Figure 6.1: Data flow diagram with the relevant software components, divided into three partitions: External components, services of the prototype, and other TrustCoM services.

- The **CDL2BPEL Service** takes the choreography as an input¹ and derives the executable, private processes, the public views on them, and deployment descriptors from it. During the derivation, information is requested from the KB, Membership Management, and Member Locator Services. The resulting documents are stored in the BP Repository.
- In the **CDL Knowledge Base**, CDL patterns are stored along with the respective replacements. For the purpose of storage, a relational database system is used.
- **Membership Management** knows the mapping from choreography roles to VO members and details about the members.
- The **Member Locator Service** can answer queries on the services available at the partner it belongs to, among others, the type and address of the BPEL execution engine in use.
- The **BP Repository** takes all kinds of business process-related documents and has a specific identity system for the data sets it contains.
- To a degree, the **BPManagement Service** abstracts from the used BPEL execution engine. When invoked for the deployment of processes, it gets the required documents from the BP Repository, packages them in the engine-specific way and hands the package over to the execution engine.
- **ActiveBPEL** is the execution engine used in this prototype. Deployed process definitions are ready for execution. That is, when an initializing Web service call to a deployed process arrives, the engine generates a new instance of the process and executes the instance by following the process definition. During execution, other Web service calls are made, both, between BPEL engines and from the engine to other Web services that implement business or supporting functions.

The whole set of components ultimately will be controlled by a VO Management Suite, which has functions for all phases of the VO life cycle (Figure 2.1). Until the presented components are integrated into this suite, they are controlled via a java servlet. This servlet is a front-end, which invokes Web services with the data a user provides. It thus provides control over the following operations: deriving views and processes for WS-CDL descriptions (CDL2BPEL); deploying the generated views (BPManagement); invoking the Web service interfaces to the deployed processes, and thus triggering the execution of instances of the processes.

¹The choreography could also be transmitted to the BP Repository, from where CDL2BPEL could then retrieve it.

The preference in TrustCoM is to use open standards and software, which is met with Java Web services and open-source implementations of the relational database system and the BPEL engine.

6.2 Details on the CDL2BPEL Service

The CDL2BPEL Service implements the CDL2BPEL algorithm from Section 5.2.2. Its first step is to parse a WS-CDL document and generate a corresponding tree of java objects from it, on which the derivation is conducted. This step is mostly done using code from sourceforge's Pi4SOA open source project². From these CDL objects, BPEL objects representing processes are generated. The latter objects are in the end serialized to WSBPEL code. This task is done using the classes from another open source project, namely the before-mentioned ActiveBPEL engine³.

In more detail, the method `CDL2BPELServiceBindingImpl.deriveViews` parses the choreography and generates a BPEL and a WSDL Handler per role. These are used for adding elements to the respective documents. The `BpelHandler` has a number of Hashtables, one of which has as values the added activities, the keys are the position in the CDL object tree as an `ArrayList`. This Hashtable enables the insertion of activities at the right location in the process of the current `BpelHandler`. If an activity is to be added by a CDL document, the element determines its own position in the object tree by building an `ArrayList` with the Hash code of each of its parents. The `add`-method of the `BpelHandler` is called with the BPEL element to be added as well as the mentioned `ArrayList`. It can then get the parent of this element by dropping the last element of the `ArrayList` and querying the Hashtable with this `ArrayList`. The value for that key is a container BPEL element, in which the new activity can be added. The new activity is stored with its own identifying `ArrayList` in the Hashtable.

By performing a depth-first search through the CDL object tree (which represents the XML element tree from the CDL source document) and translating all structuring activities to all roles, this system works. However, a number of roles then carries structuring elements with no children - which is incorrect in BPEL. These empty structures are removed in the cleanup of the BPEL code, where also optimization takes place.

To all CDL elements, a `convertToBpel` method was added, by altering the `org.pi4soa.cdl.CDLType` interface, which is implemented by all `org.pi4soa.cdl.impl.*` activities and other elements. Also, a

²Version 1.0.0, May 2005, see <http://sourceforge.net/projects/pi4soa/>

³Version 1.0.9, March 2005, see <http://www.activebpel.org/>

cleanupAutogenBpel method was added to all BPEL activities (via *org.activebpel.rt.bpel.def.AeNamedDef*) as well as some other necessary classes, all in *org.activebpel.rt.bpel.def* , *org.activebpel.rt.bpel.def.activity* , or *org.activebpel.rt.bpel.def.activity.support*). Both methods, *convertToBpel* and *cleanupAutogenBpel* realize a depth-first search through the respective object trees. *convertToBpel* takes as input two Hashtables, which contain the Bpel and WsdHandlers for all roles as values and the role names as keys, as well as a ModelListener for reporting errors and warnings. *cleanupAutogenBpel* takes a BpelHandler as input, which is necessary for keeping an overview over *bpel:links* in the process, and request the optimization level from *org.activebpel.rt.bpel.def.AeActivityDef* if required. Figures 6.2 and 6.3 show the class diagram overviews⁴ for *org.pi4soa.cdl.** and *org.activebpel.rt.bpel.def.** , respectively.

⁴Note, that the diagrams are no UML Class Diagrams, rather overviews over the actual class diagrams. Italic names stand for interfaces, not slanted names for classes. Continuous arrows indicate inheritance, interface implementations are shown with dashed arrows. Other relations are not expressed for reasons of legibility.

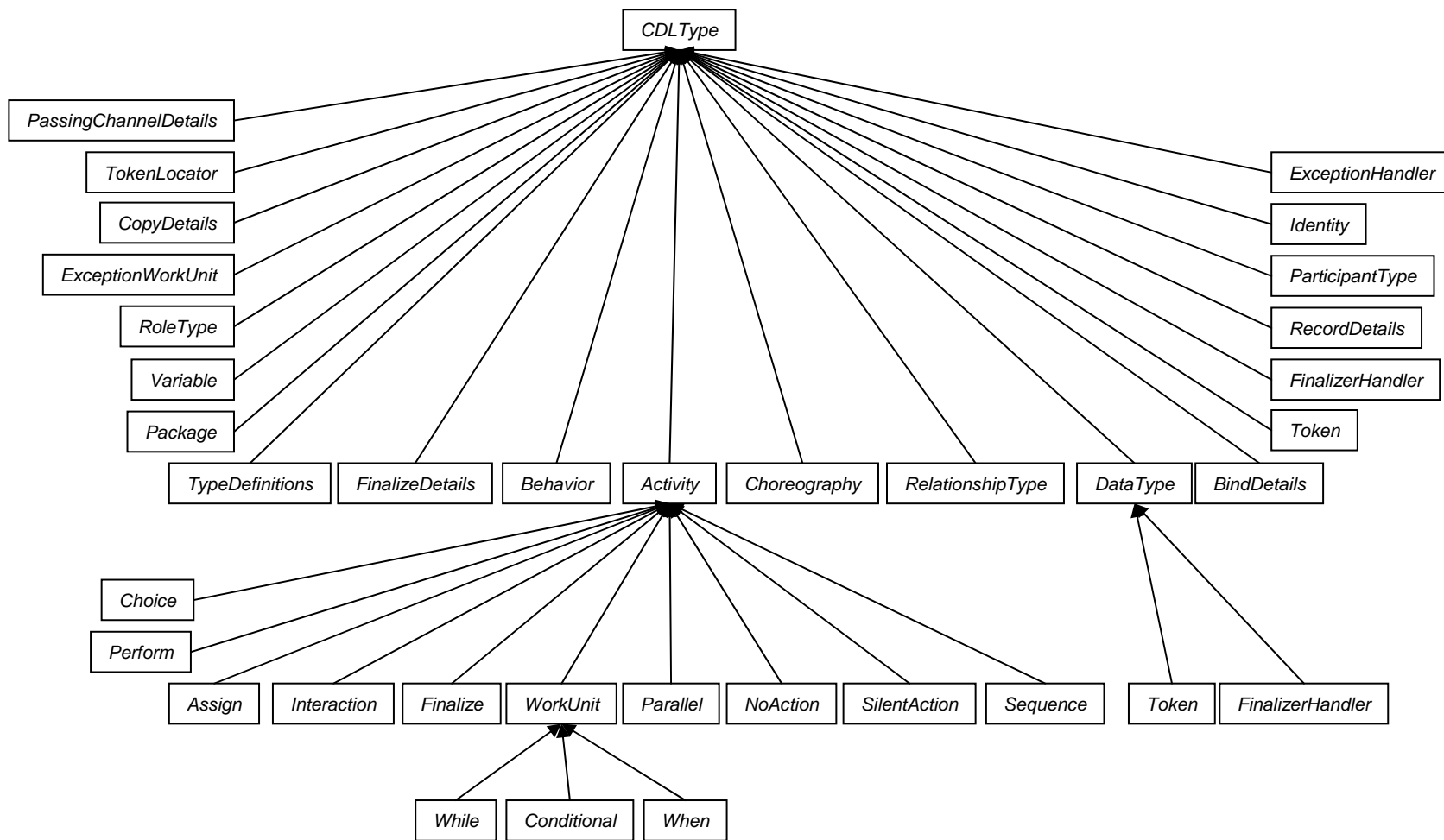


Figure 6.2: The class diagram overview over *org.pi4soa.cdl.**. All shown elements are interfaces. *org.pi4soa.cdl.impl.** has the exact same structure, where all elements are named as their respective interfaces, but ending with an additional “Impl”. All elements in *org.pi4soa.cdl.impl.** implement their interface from *org.pi4soa.cdl.**.

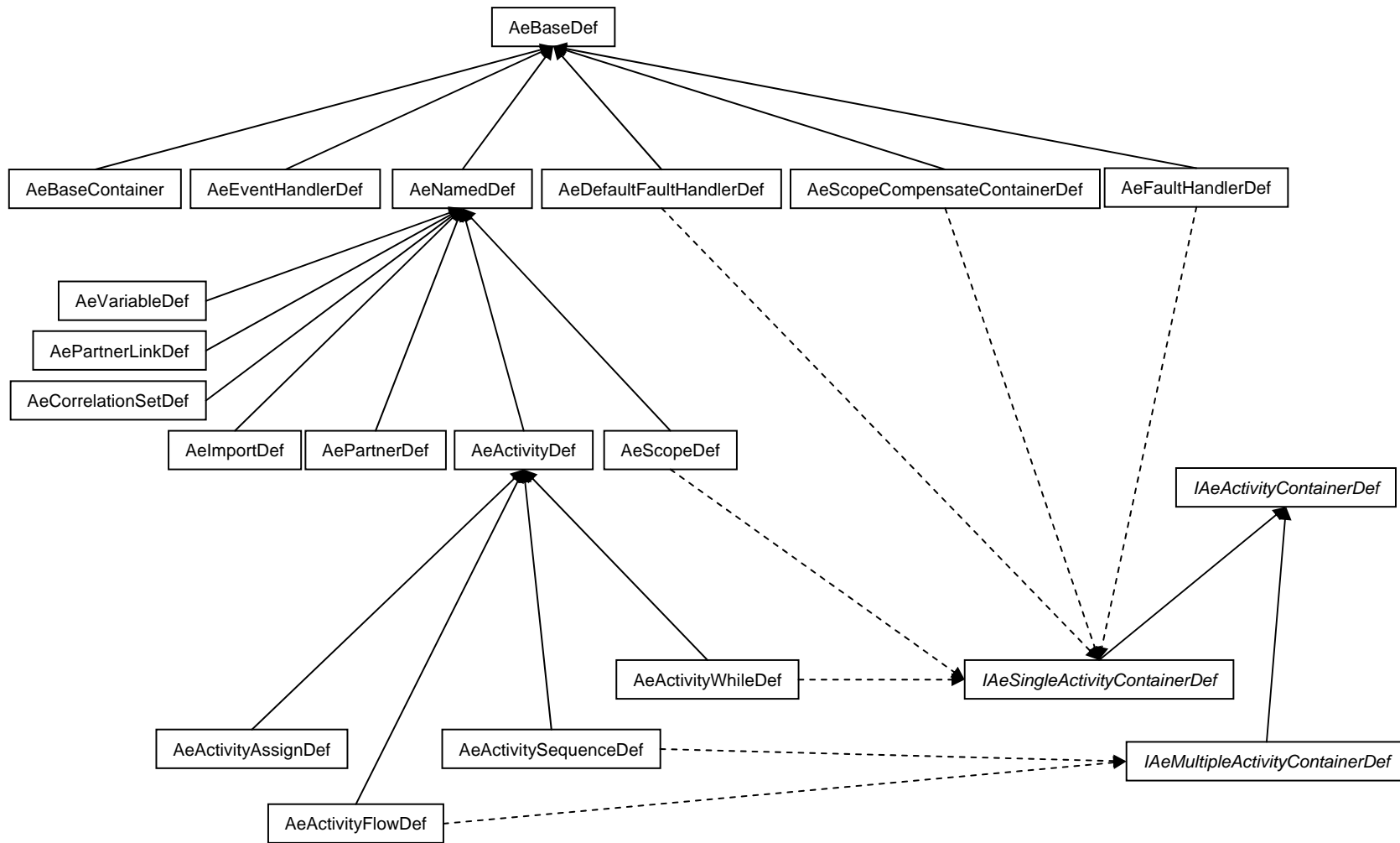


Figure 6.3: The class diagram overview over selected parts of `org.activebpel.rt.bpel.def.*`. `IAeActivityContainerDef`, `IAeSingleActivityContainerDef`, and `IAeMultipleActivityContainerDef` are interfaces, all other elements are classes. All activity definitions in `org.activebpel.rt.bpel.def.activity` are inherited from `AeActivityDef`.

6.2.1 Validity Check for Channel Usage

Channel usage in WS-CDL can be either limited to once or unlimited. In the case where the usage is limited to once, the CDL2BPEL-Service checks, whether this constraint could or will be violated during the execution of a choreography instance. But since the service is called at design-time, the check cannot guarantee constraint conformance in all cases. E.g., the data that is being evaluated at the guard conditions of workunits is not known a priori. In a slightly more complex scenario, there could be two workunits with the same guard condition and containing an interaction on the same channel with a limited usage, but other activities in between. Usually we would assume that these two interactions both occurred when the guard condition evaluates to true. However, at design time we cannot be sure that the variables used in the guard conditions are not altered between the workunits, and thus both guard conditions will evaluate to the same outcome at run time. The channel **could** be used twice, but this is not a **must**.

Based on this example and other insights, we classify three usage cases for channels: **MUST** cases, **COULD** cases, and **CHOICE** cases. If the number of **MUST** cases exceeds the allowed number of usages, an error is reported and the service stops. If there are more than allowed **COULD** cases, a warning is triggered, but the service continues. **CHOICE** cases can be mutually exclusive, in which case they are only counted once. Depending on the context of the usage, they are counted as **MUST** or as **COULD** cases.

The check whether two **CHOICE** cases are mutually exclusive compares the paths to the interaction where the channel is used. If two paths are equal up to a node X , where X is a choice node, then the two channel usages are mutually exclusive. If X is not a choice node, they are not. In order to know the relationship between every pair of **CHOICE** cases, each pair of paths would have to be checked. Due to the structure of the object tree, which is gained from the XML (or DOM) tree, it is only necessary to check each pair of subsequent choice paths when going through the tree in a depth-first manner. While this approach is not complete in terms of the relationships between paths, it suffices our needs in that each violation of channel usage constraints will be detected.

Visiting each node in the choreography subtree in a depth-first search falls short in three cases of activities: perform, interaction with positive causeException attribute, and finalize. In these cases, the references to choreographies, exception handlers, and finalizers have to be resolved. Choreographies enclosed in a perform activity cannot use channel variables from the enclosing choreography⁵, therefore the referenced choreography has to be checked for valid channel usage

⁵The perform element can bind variables from the enclosing choreography to the choreography to be performed, but they have to belong to a certain roleType. This is not the case for channel variables, and thus they cannot

for channels defined within itself, only. In contrast, the finalizers and exception handlers are seen as children elements⁶ of the respective origin of interaction and then searched in the mentioned depth-first manner.

6.3 Status of the Implementation

Currently, there are some constraints on the CDL documents that are converted correctly by the CDL2BPEL implementation. Most parts of the conversion work fine, but runtime behavior was only tested⁷ for the example choreographies in Appendix A and few other choreographies. The implementation converts most of the CDL elements, only a small subset is not converted currently. Also, only `xsd:simple` types should be used in the `cdl:informationType` specification for variables. The latter issue comes from incorrect handling of complex type variables in the ActiveBPEL engine: ActiveBPEL, version 1.2.0, cannot deal with regular variables of a complex schema type. Complex types are only handled correctly in message variables.

The Knowledge Base service has been implemented as well, and is based on a relational database system. Pattern matching in the current implementation requires equal strings for element and attribute names, as well as for attribute and element values, and an equal structure. That does not mean that the strings have to be equal – indentation, line breaks, and way an element is closed⁸ do not play a role. Currently, the patterns can be inserted either via SQL (locally) or Web service invocation. A front-end supporting pattern insertion and maintenance is highly desired. Exemplary KB patterns can be found in Appendix B.

The deployment via BPManagement Service is implemented and works. The BP Repository is not integrated, because for the current status of the implementation it is easier to use simple files in the file system: it allows for easier debugging than storing the documents in a database. The integration should be fairly simple, though, because the current system already works with identity strings like the BP Repository.

The stated restrictions do not invalidate the concept at any point. Rather, the implementation should be completed in order to be a full implementation of the presented solution, which is only a question of investing time. It still serves as a proof-of-concept, because the implemented parts already allow the generation, deployment, and execution of private processes and public

be employed in variable binding.

⁶For this purpose, we clone the referenced nodes and their children, and set the parent references of these clones accordingly.

⁷Please see the evaluation in the next chapter for details on performed test.

⁸In XML, an element without child elements and values can either be closed directly, as for instance in “<content/>”, or by a closing tag, e.g., “<content></content>”. Both ways represent the same structure.

views. This shows that the concepts of the CDL2BPEL algorithm, the Knowledge Base, and the translation tables work in practice. In particular, the nesting technique from the algorithm together with the optimization yields executable processes.

6.4 Points for Improvement

The prototype could be improved and further optimized at several points. The methods *convertToBpel* and *cleanupAutogenBpel* could have been implemented using the Visitor pattern from Software Engineering. This change would lead to slightly more flexibility in terms of ease in maintaining the code, and seems more appealing from an engineering point of view. Of course, the most obvious improvement would be to implement the remaining parts of the translation.

Currently, the optimization knows four levels. A fifth level could be implemented for generation of abstract BPEL processes, leading to a more sophisticated language for public views. The changes necessary for this extension are minor, since this level only has to remove all process parts that are related to execution - such as assignments, variables, and the like. Please refer to the BPEL specification [2] for an exhaustive list of differences between abstract and executable BPEL processes.

6.5 Chapter Summary

This chapter described the details of the prototypical implementation. The three components implemented are the CDL2BPEL Service, the CDL Knowledge Base Service, and the BPMManagement Service. In the TrustCoM framework, they make use of a set of other services, and rely on external software for data storage and process execution. Although the prototype does not implement the solution from Chapter 5 completely, it proves the validity of general concept with respect to the goal.

Chapter 7

Evaluation

The goal from the problem statement in Chapter 2 was to derive executable processes and their public views for each role in a choreography, followed by the automatic deployment and execution of the outcomes. The main issues of the derivation were the information gap and the language differences. This chapter evaluates the solution suggested in Chapter 5 according to the criteria formulated in Section 2.4.

Section 5.3.1 gives a conceptual mapping for all WS-CDL elements to BPEL and WSDL. The CDL2BPEL algorithm at each member uses this mapping and the local Knowledge Base in order to derive executable processes and their interfaces from choreographies. The prototype proves the validity of this approach for the addressed problem by applying the implementation to a set of collaborative engineering choreographies within a TrustCoM VO. For two of those choreographies, the WS-CDL and BPEL code is shown in Appendix A. Appendix B shows the Knowledge Base patterns used. The outputs are processes which are indeed executable. Besides the real-world choreographies from TrustCoM, extended versions of them and artificial examples making use of other elements were tested. The valid execution was only shown for the available collaborative engineering choreographies. As soon as other use cases become available from the respective partners in TrustCoM, their modeling and testing will be direct future work.

However, there were a couple of problems in the translation table, as outlined in Section 5.3.3. Correlation sets in BPEL must be initialized, making the translation of `cdl:tokens` hard. A problem with no real-world meaning is the usage of `cdl:workUnits` outside of parallel constructs. The requirement of extending information types with mappings to message content structure, however, means that a choreography has to be designed for the purpose of using it as input to CDL2BPEL. Also, for certain cases of the `cdl:choice` element, no semantically equivalent

construct was found. That is, a possible workaround could differ in its behavior as follows:

- Timing issues can switch the order of condition evaluation, since there are no event-based BPEL constructs matching the `cdl:isVariableAvailable` and `cdl:variablesAligned` functions.
- Communication delays in Web Service invocations can cause a duration-based timeout to happen later than intended or even missing an event completely.

A solution to these timing- and event-related issues basically requires changes or extensions in BPEL, thus being postponed until the next version¹ of BPEL becomes available. Some of these issues could also be solved with suitable coordination protocols in WS-Coordination.

Besides the problems with the hybrid choice in its most complicated form, the issues can be overcome very well. The token initialization problem can be neglected in the environment of VOs, and the problem with work units outside of a parallel will not be present in any meaningful choreography, as argued in Section 5.3.3. Altogether, the problem with the hybrid choice remains, but the suggested solution solves the problem.

In Section 2.4, a set of criteria was stated. These are mentioned here, together with an examination on how far they are met by the suggested solution.

1. **Executability.** Conceptually, this criteria is satisfied, as proved by the prototype. However, the constraints on executable output are high, for instance in that all variables must be initialized before they are used. That has to be assured via initialization, data exchange, and Knowledge Base patterns and should be tested with generated processes.
2. **Completeness and standard compliance.** The translation table is complete, in that there is a suggested translation for each CDL element and function. However, some workarounds are not perfect, as stated above. Since only three serious issues are identified, i.e., issues where CDL's respective semantics cannot be mapped perfectly, and these issues either are minor in practice or the environment of VOs, or they are hard to trigger, the overall impact of these issues to the solution is considered minor. In essence, choreographies have to be designed or altered with respect to the usage as a source for CDL2BPEL. Note, that BPEL's initialization technique required for a semantically specialized variable or role in the choreography, as stated in Section 5.4.1. The variable-based initialization breaks the closed-world assumption of WS-CDL: the variable content is transmitted from somewhere outside the choreography.

¹The OASIS WSBPEL Technical Committee plans two further versions of the BPEL standard, before handing over the control to W3C.

3. **High degree of automation.** The solution runs fully automated and no intermediate user input is required during the conversion of valid, meaningful choreographies. Incorrect input is not accepted or will not lead to executable processes. But, as with any compiler, the requirement of correct input is comprehensible. Once the Knowledge Base contains the necessary content, the user does not have to deal with any of the target languages. Due to the information gap, the one touching point with the target languages, the Knowledge Base, is inevitable.
4. **Extrapolation** from the chosen languages. The language-independent concepts are valid for similar languages or meta models. These concepts are the CDL2BPEL algorithm and the notion of the Knowledge Base. Obviously, the translation table then loses its applicability and has to be adapted to the new setup. Even so, certain aspects from it may be re-used, as for instance the way transactional interactions are modeled in BPEL (with acknowledgement exchange and variable value backup with respective rollback in the case of a fault). The same is true for the implementation: the language-specific parts lose their applicability, but the overall algorithm and the Knowledge Base would only require little adaptation. If BPEL was still the goal language, the optimization implementation remains valid.
5. **Extensibility.** As a next step in TrustCoM, security, trust, and contract enforcement will be added to choreographies and must be present in the output as well. These extensions can be inserted into CDL via XML extension or annotation at respective spots. Doing so should not interfere with the concept of this solution, because most of these extensions seem to be orthogonal to the regular translation. For other extensions it is highly unclear, how they are modeled in the execution language, making it hard to predict how their insertion will interfere with the shown concept. Small extensions for specific purposes could simply get translated through the Knowledge Base.
6. **Optimization.** With the available and anticipated optimization levels, a high flexibility in terms of the output is given: Processes can be left completely unoptimized, making them stubs that cannot be executed; or they can be optimized up to a point where the remaining structure is minimal², leaving few wishes open.
7. **Robustness.** The solution should not produce erroneous output in any case. Instead, it returns an error note if there are issues. As most compilers, it is not forgiving and does not correct errors by itself. Although the output should be valid BPEL - given output

²No formal proof of this claim is given here. But since the highest optimization level only keeps elements that serve a purpose, the minimality in that sense is apparent. Please refer to Section 5.2.3 for details on the optimization.

is produced - the approach does not validate if the BPEL code then is executable. To a certain degree, that could be done, e.g., by assuring that all variables are initialized before their first usage. But there would still be many cases, for which no guarantees can be made at design time - the validity depends on both, actual data and possibly timing, information that is in general only known during runtime. Thus, the robustness could be improved, but not overly so.

The approach requires that choreographies are modeled with respect to the mentioned limitations. But since they are not too many, especially in the context of VOs, this solution will supposedly perform well for most scenarios. If WS-CDL's design better fitted the event model of BPEL³, the translation would not suffer from these issues.

Besides limitations and additional requirements for an input choreography, the solution yields great benefit: Designing executable processes is a time-consuming task. Aligning multiple processes and making them work together is even more complex, and arguably gets harder exponentially with the number of involved processes. The shown approach knows no limitation in the number of roles in a choreography and inherently causes the output processes to fit together perfectly. Thus, the advantages of following this top-down approach by far outweigh the limitations.

³For a discussion on the suitability of WS-CDL as a choreography language, please refer to Section 3.2.3.

Chapter 8

Conclusions and Future Work

8.1 Summary

This thesis describes the task of deriving executable public and private processes from a high-level choreography in the environment of self-managed Virtual Organizations. The TrustCoM project demands the usage of open standards and a top-down approach for this task. The main challenges are initially pointed out, namely the language differences between the chosen languages for process definitions on different levels, and the information gap between high-level choreography descriptions and executable processes. The technological foundation outlines techniques and standards used, by summarizing the area of Web services and the relevant parts of business processing. A discussion about the quality of the chosen choreography language is provided there as well. Related work on collaborative, interorganizational business processes is analyzed and compared, with emphasis on the opportunities for mapping the different levels of processes.

Motivated by the problem, technology in use, and related work, a possible solution is presented, which is based on the three centerpieces: the CDL2BPEL algorithm, the translation tables, and the Knowledge Base. The translation tables show a static mapping from the source to the target language. Since some elements of the source language are not intended to provide enough information for their execution, these elements are translated via the Knowledge Base. The Knowledge Base thus allows for filling the information gap in a partner-specific way, satisfying the confidentiality requirement for private process parts. The CDL2BPEL algorithm brings these and other pieces together and states, how input documents are processed. Integration with other services in the VO environment of TrustCoM is outlined together with a deployment plan of the solution, which satisfies the confidentiality requirement of internal implementations.

For the suggested solution, a proof-of-concept prototype has been developed, which will be integrated into the TrustCoM VO Management Suite. Based on the translation results of the implementation, the evaluation shows, that in most practically relevant cases the algorithm yields executable processes. Also, the criteria for a good solution are evaluated and confirm the suitability of the presented solution for the given domain. While the CDL2BPEL algorithm and the Knowledge Base are language-independent concepts, the translation tables depend on the current choice of languages.

8.2 Future work

Future work is planned to build incrementally upon the presented derivation approach. As an immediate step, more evaluation based on other real-world examples is planned as soon as other use cases are developed by the respective partners in TrustCoM. Furthermore, TrustCoM aims at provisioning of secure collaborative business processes. The confidentiality of private processes can be enforced through their deployment environment. Process execution at runtime needs to be reactive to security subsystem decisions taken outside the process execution environment, but within the same administrative domain. A security control concept for collaborative business processes is already available and was published by Jochen Haller [14]. It will be implemented in TrustCoM based on the presented work. Such points of security enforcement could be marked by annotation of particular CDL activities, e.g., the `cdl:noAction`.

8.2.1 VO Evolution and Business Processes

The evolution of a VO can take place during the operation phase, and is triggered by a number of reasons. A few examples are the following: a member could quit the collaboration and needs to be replaced; a member does not satisfy the Service Level Agreement (SLA) of the VO, e.g., by responding with a latency time over a certain threshold, and is replaced; the objective of the VO may have to be adapted to a changed market situation. All of these cases and more should be accounted for by the business processes. Section 5.4.2 introduces the related problems, such as reaching a coherent state of the collaborative process after the replacement of a member, and describes first steps towards a solution. The enactment of VO Evolution by business processes may be crucial for the success of the whole notion of using business processes for the operation phase.

8.2.2 Extending the Knowledge Base

Future work could also include basing the Knowledge Base on business process-focussed semantic descriptions of the tasks to be fulfilled. Therefore, the patterns could be more generic, in that they would be compared with a functional description of a part of the choreography.

Besides, in many cases an organization being a VO member will already have a complete, optimized private process for a similar purpose as the derived private process from a choreography. Thus, the opportunity of taking the existing processes into account should be investigated, and was already mentioned in Section 5.2.1. Potentially, a good matching algorithm between the two private processes could replace the Knowledge Base in its role of inserting member-internal details into the derived private process. Or, more likely, such comparison could be used for Knowledge Base maintenance: The KB could learn from the matches and generalize the CDL patterns, add new ones, or refine the replacements.

Appendix

Appendix A

Business Process Code Examples

Below, there is the code for two collaborative engineering example choreographies, together with the respective executable WBPEL code, which the prototypical implementation automatically generated.

In the first example, an AnalysisPartner retrieves technical data from a StoragePartner, analyzes it, and transmits the results back to the StoragePartner. The code corresponds to the UML Activity Diagrams in Figures 1.1 and 5.1. The second, more complete example has four roles: Two AnalysisPartners (*AP1* and *AP2*), a StoragePartner (*SP*), and an Initiator (*I*). First, *I* identifies the need for collaboration and asks *SP* to store the raw design data. Then, in two parallel streams of work, *AP1* and *AP2* are both instructed to provide their expertise for the design data. For that purpose, they collect the data from *SP*, analyze it, ask *SP* to store the results, and also transmit the results back to *I*. After *I* received both answers, the results are combined and the process ends.

Since the CDL2BPEL algorithm generates one BPEL process per role, the first choreography leads to two BPEL processes, and the second example to four such processes. The automatic derivation of the processes below made use of the Knowledge Base pattern examples shown in Appendix B.

A.1 Example 1

A.1.1 WS-CDL Code

```
<?xml version="1.0" encoding="UTF-8"?>
<package name="SAC_Package" version="1.0"
```

```

    targetNamespace="http://cdl2bpel.cecka.sap.com/SAC1"
    xmlns="http://www.w3.org/2004/12/ws-chor/cdl">
<informationType name="stringType" type="xsd:string"/>
<informationType name="uriType" type="xsd:anyURI"/>
<token name="dummyToken" informationType="stringType"/>
<roleType name="AnalysisPartner">
  <behavior name="Analyzer"/>
</roleType>
<roleType name="StoragePartner">
  <behavior name="StorageProvider"/>
</roleType>
<relationshipType name="AnalysisStorageRel">
  <roleType typeRef="AnalysisPartner" behavior="Analyzer"/>
  <roleType typeRef="StoragePartner" behavior="StorageProvider"/>
</relationshipType>
<participantType name="Analyzer">
  <roleType typeRef="AnalysisPartner"/>
</participantType>
<participantType name="StorageProvider">
  <roleType typeRef="StoragePartner"/>
</participantType>
<channelType name="getData" usage="once">
  <roleType typeRef="StoragePartner" behavior="StorageProvider"/>
  <reference>
    <token name="dummyToken"/>
  </reference>
</channelType>
<channelType name="storeData" usage="once">
  <roleType typeRef="StoragePartner" behavior="StorageProvider"/>
  <reference>
    <token name="dummyToken"/>
  </reference>
</channelType>
<choreography name="SAC_Choreography" root="true">
  <relationship type="AnalysisStorageRel"/>
  <variableDefinitions>
    <!-- Variable that is used for outside calls to the choreography
      IMPORTANT: The role for which it is specified defines the process which
      is started first! HAS to be specified at the root choreography, for
      the current CDL2BPEL implementation to work -->
    <variable name="initVOChorVar" informationType="uriType"
      roleTypes="AnalysisPartner"/>
    <!-- The other variables -->
    <variable name="varRawDataAddr_Ana" informationType="uriType"
      roleTypes="AnalysisPartner"/>
    <variable name="varRawDataAddr_Sto" informationType="uriType"
      roleTypes="StoragePartner"/>
    <variable name="varResultDataAddr_Ana" informationType="uriType"
      roleTypes="AnalysisPartner"/>
    <variable name="varResultDataAddr_Sto" informationType="uriType"
      roleTypes="StoragePartner"/>
    <variable name="varRawData_Sto" informationType="stringType"

```

```

    roleTypes="StoragePartner"/>
<variable name="varResultData_Sto" informationType="stringType"
  roleTypes="StoragePartner"/>
<variable name="varRawData_Ana" informationType="stringType"
  roleTypes="AnalysisPartner"/>
<variable name="varResultData_Ana" informationType="stringType"
  roleTypes="AnalysisPartner"/>
<variable name="chVarGet" channelType="getData"/>
<variable name="chVarStore" channelType="storeData"/>
</variableDefinitions>
<sequence>
  <assign roleType="AnalysisPartner">
    <copy name="initial-assign">
      <source variable="cdl:getVariable('initVOChorVar','','')"/>
      <target variable="cdl:getVariable('varRawDataAddr_Ana','','')"/>
    </copy>
  </assign>
  <interaction name="getRawDataReq" operation="getRawDataOp"
    channelVariable="chVarGet">
    <participate relationshipType="AnalysisStorageRel"
      fromRoleTypeRef="AnalysisPartner" toRoleTypeRef="StoragePartner"/>
    <exchange name="exRawDataAddr" informationType="uriType"
      action="request">
      <send variable="cdl:getVariable('varRawDataAddr_Ana','','')"/>
      <receive variable="cdl:getVariable('varRawDataAddr_Sto','','')"/>
    </exchange>
  </interaction>
  <silentAction roleType="StoragePartner">
    <description type="documentation">getRawDataFromDB
      varRawDataAddr_Sto varRawData_Sto</description>
  </silentAction>
  <interaction name="getRawDataResp" operation="getRawDataOp"
    channelVariable="chVarGet">
    <participate relationshipType="AnalysisStorageRel"
      fromRoleTypeRef="AnalysisPartner" toRoleTypeRef="StoragePartner"/>
    <exchange name="exRawData" informationType="stringType"
      action="respond">
      <send variable="cdl:getVariable('varRawData_Sto','','')"/>
      <receive variable="cdl:getVariable('varRawData_Ana','','')"/>
    </exchange>
  </interaction>
  <silentAction roleType="AnalysisPartner">
    <description type="documentation">analyzeData varRawData_Ana
      varResultData_Ana</description>
  </silentAction>
  <interaction name="storeResultDataReq" operation="storeResultDataOp"
    channelVariable="chVarStore">
    <participate relationshipType="AnalysisStorageRel"
      fromRoleTypeRef="AnalysisPartner" toRoleTypeRef="StoragePartner"/>
    <exchange name="exResultData" informationType="stringType"
      action="request">
      <send variable="cdl:getVariable('varResultData_Ana','','')"/>

```

```

    <receive variable="cdl:getVariable('varResultData_Sto','','')"/>
  </exchange>
</interaction>
<silentAction roleType="StoragePartner">
  <description type="documentation">storeDataToDB varResultData_Sto
    varResultDataAddr_Sto</description>
</silentAction>
<interaction name="storeResultDataResp" operation="storeResultDataOp"
  channelVariable="chVarStore">
  <participate relationshipType="AnalysisStorageRel"
    fromRoleTypeRef="AnalysisPartner" toRoleTypeRef="StoragePartner"/>
  <exchange name="exResultAddr" informationType="uriType"
    action="respond">
    <send variable="cdl:getVariable('varResultDataAddr_Sto','','')"/>
    <receive variable="cdl:getVariable('varResultDataAddr_Ana','','')"/>
  </exchange>
</interaction>
</sequence>
</choreography>
</package>

```

A.1.2 WSBPEL Code

AnalysisPartner Role

```

<?xml version="1.0" encoding="UTF-8"?>
<process name="BP_AnalysisPartner" suppressJoinFailure="yes"
  targetNamespace="http://cdl2bpel.cecka.sap.com/SAC1/AnalysisPartner"
  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:ns0="http://cdl2bpel.cecka.sap.com/SAC1/StoragePartner"
  xmlns:tns="http://cdl2bpel.cecka.sap.com/SAC1/AnalysisPartner">
  <partnerLinks>
    <partnerLink myRole="AnalysisPartnerRole_InitProcessPL" name="InitProcessPL"
      partnerLinkType="tns:InitProcessPLType"/>
    <partnerLink name="InitProcessStoragePartnerPL"
      partnerLinkType="tns:InitProcessStoragePartnerPLType"
      partnerRole="StoragePartnerRole_InitProcessStoragePartnerPL"/>
    <partnerLink myRole="AnalysisPartnerRole_AnalysisStoragePL"
      name="AnalysisStoragePL" partnerLinkType="tns:AnalysisStoragePLType"
      partnerRole="StoragePartnerRole_AnalysisStoragePL"/>
  </partnerLinks>
  <variables>
    <variable messageType="tns:InitProcessOpMsg" name="InitProcessOpMsgVar"/>
    <variable messageType="ns0:InitProcessStoragePartnerOpMsg"
      name="InitProcessStoragePartnerOpMsgVar"/>
    <variable messageType="ns0:InitProcessStoragePartnerOpRespMsg"
      name="InitProcessStoragePartnerOpRespMsgVar"/>
    <variable name="initVOChorVar" type="tns:uriType"/>
    <variable name="varRawDataAddr_Ana" type="tns:uriType"/>
  </variables>

```

```

<variable name="varResultDataAddr_Ana" type="tns:uriType"/>
<variable name="varRawData_Ana" type="tns:stringType"/>
<variable name="varResultData_Ana" type="tns:stringType"/>
<variable messageType="ns0:getRawDataOpReqMsg"
  name="getRawDataOpReqMsgVar"/>
<variable messageType="tns:getRawDataOpRespMsg"
  name="getRawDataOpRespMsgVar"/>
<variable messageType="ns0:storeResultDataOpReqMsg"
  name="storeResultDataOpReqMsgVar"/>
<variable messageType="tns:storeResultDataOpRespMsg"
  name="storeResultDataOpRespMsgVar"/>
</variables>
<sequence>
  <receive createInstance="yes" name="InitProcess" operation="InitProcessOp"
    partnerLink="InitProcessPL" portType="tns:InitProcess"
    variable="InitProcessOpMsgVar"/>
  <assign>
    <copy>
      <from part="initVOChorVar" variable="InitProcessOpMsgVar"/>
      <to variable="initVOChorVar"/>
    </copy>
    <copy>
      <from expression="true()"/>
      <to part="Start" variable="InitProcessStoragePartnerOpMsgVar"/>
    </copy>
  </assign>
  <flow>
    <invoke inputVariable="InitProcessStoragePartnerOpMsgVar"
      name="InitStoragePartnerProcess"
      operation="InitProcessStoragePartnerOp"
      outputVariable="InitProcessStoragePartnerOpRespMsgVar"
      partnerLink="InitProcessStoragePartnerPL"
      portType="ns0:InitProcessStoragePartner"/>
  </flow>
  <scope variableAccessSerializable="no">
    <sequence>
      <assign>
        <copy>
          <from variable="initVOChorVar"/>
          <to variable="varRawDataAddr_Ana"/>
        </copy>
      </assign>
      <scope name="Scope-1" variableAccessSerializable="no">
        <sequence>
          <assign>
            <copy>
              <from variable="varRawDataAddr_Ana"/>
              <to part="payload" variable="getRawDataOpReqMsgVar"/>
            </copy>
          </assign>
          <invoke inputVariable="getRawDataOpReqMsgVar"
            operation="getRawDataOpReq"

```

```

        partnerLink="AnalysisStoragePL"
        portType="ns0:StoragePartner_AnalysisStoragePT"/>
    </sequence>
</scope>
<scope name="Scope-2" variableAccessSerializable="no">
    <sequence>
        <receive operation="getRawDataOpResp" partnerLink="AnalysisStoragePL"
            portType="tns:AnalysisPartner_AnalysisStoragePT"
            variable="getRawDataOpRespMsgVar"/>
        <assign>
            <copy>
                <from part="payload" variable="getRawDataOpRespMsgVar"/>
                <to variable="varRawData_Ana"/>
            </copy>
        </assign>
    </sequence>
</scope>
<assign>
    <copy>
        <from expression="'someResData'"/>
        <to variable="varResultData_Ana"/>
    </copy>
</assign>
<scope name="Scope-3" variableAccessSerializable="no">
    <sequence>
        <assign>
            <copy>
                <from variable="varResultData_Ana"/>
                <to part="payload" variable="storeResultDataOpReqMsgVar"/>
            </copy>
        </assign>
        <invoke inputVariable="storeResultDataOpReqMsgVar"
            operation="storeResultDataOpReq"
            partnerLink="AnalysisStoragePL"
            portType="ns0:StoragePartner_AnalysisStoragePT"/>
    </sequence>
</scope>
<scope name="Scope-4" variableAccessSerializable="no">
    <sequence>
        <receive operation="storeResultDataOpResp" partnerLink="AnalysisStoragePL"
            portType="tns:AnalysisPartner_AnalysisStoragePT"
            variable="storeResultDataOpRespMsgVar"/>
        <assign>
            <copy>
                <from part="payload" variable="storeResultDataOpRespMsgVar"/>
                <to variable="varResultDataAddr_Ana"/>
            </copy>
        </assign>
    </sequence>
</scope>
</sequence>
</scope>

```

```

</sequence>
</process>

```

StoragePartner Role

```

<?xml version="1.0" encoding="UTF-8" ?>
<process name="BP_StoragePartner" suppressJoinFailure="yes"
  targetNamespace="http://cdl2bpel.cecka.sap.com/SAC1/StoragePartner"
  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:choreons="http://cdl2bpel.cecka.sap.com/SAC1"
  xmlns:ns0="http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage"
  xmlns:ns1="http://cdl2bpel.cecka.sap.com/SAC1/AnalysisPartner"
  xmlns:tns="http://cdl2bpel.cecka.sap.com/SAC1/StoragePartner"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <partnerLinks>
    <partnerLink myRole="StoragePartnerRole_InitProcessStoragePartnerPL"
      name="InitProcessStoragePartnerPL"
      partnerLinkType="tns:InitProcessStoragePartnerPLType" />
    <partnerLink myRole="StoragePartnerRole_AnalysisStoragePL"
      name="AnalysisStoragePL" partnerLinkType="tns:AnalysisStoragePLType"
      partnerRole="AnalysisPartnerRole_AnalysisStoragePL" />
    <partnerLink name="PrivateStoPL" partnerLinkType="tns:PrivateStoPLType"
      partnerRole="Role_PrivateStoPL" />
  </partnerLinks>
  <variables>
    <variable messageType="tns:InitProcessStoragePartnerOpMsg"
      name="InitProcessStoragePartnerOpMsgVar" />
    <variable name="ChoreographyIdVariable" type="choreons:stringID" />
    <variable messageType="tns:InitProcessStoragePartnerOpRespMsg"
      name="InitProcessStoragePartnerOpRespMsgVar" />
    <variable name="varRawDataAddr_Sto" type="xsd:anyURI" />
    <variable name="varResultDataAddr_Sto" type="xsd:anyURI" />
    <variable name="varRawData_Sto" type="xsd:string" />
    <variable name="varResultData_Sto" type="xsd:string" />
    <variable messageType="tns:getRawDataOpReqMsg" name="getRawDataOpReqMsgVar" />
    <variable messageType="ns0:getDataRequest" name="getDataRequest" />
    <variable messageType="ns0:getDataResponse" name="getDataResponse" />
    <variable messageType="ns1:getRawDataOpRespMsg" name="getRawDataOpRespMsgVar" />
    <variable messageType="tns:storeResultDataOpReqMsg"
      name="storeResultDataOpReqMsgVar" />
    <variable messageType="ns0:storeDataRequest" name="storeDataRequest" />
    <variable messageType="ns0:storeDataResponse" name="storeDataResponse" />
    <variable messageType="ns1:storeResultDataOpRespMsg"
      name="storeResultDataOpRespMsgVar" />
  </variables>
  <correlationSets>
    <correlationSet name="choreographyCorrelationSet"
      properties="tns:stringIDToken" />
  </correlationSets>
</sequence>

```

```

<receive createInstance="yes" name="InitProcess"
  operation="InitProcessStoragePartnerOp"
  partnerLink="InitProcessStoragePartnerPL"
  portType="tns:InitProcessStoragePartner"
  variable="InitProcessStoragePartnerOpMsgVar">
  <correlations>
    <correlation initiate="yes" set="choreographyCorrelationSet" />
  </correlations>
</receive>
<assign>
  <copy>
    <from>
      <InitProcessStoragePartnerOpRespMsgElem
        xmlns="http://cdl2bpel.cecka.sap.com/SAC1">
        <IDElem>
          <ID />
        </IDElem>
      </InitProcessStoragePartnerOpRespMsgElem>
    </from>
    <to part="ACK" variable="InitProcessStoragePartnerOpRespMsgVar" />
  </copy>
  <copy>
    <from part="Start" query="descendant::choreons:ID"
      variable="InitProcessStoragePartnerOpMsgVar" />
    <to variable="ChoreographyIdVariable" />
  </copy>
  <copy>
    <from variable="ChoreographyIdVariable" />
    <to part="ACK" query="descendant::choreons:ID"
      variable="InitProcessStoragePartnerOpRespMsgVar" />
  </copy>
</assign>
<reply name="InitResponse" operation="InitProcessStoragePartnerOp"
  partnerLink="InitProcessStoragePartnerPL"
  portType="tns:InitProcessStoragePartner"
  variable="InitProcessStoragePartnerOpRespMsgVar">
  <correlations>
    <correlation set="choreographyCorrelationSet" />
  </correlations>
</reply>
<scope variableAccessSerializable="no">
  <sequence>
    <scope name="Scope-1" variableAccessSerializable="no">
      <sequence>
        <receive operation="getRawDataOpReq" partnerLink="AnalysisStoragePL"
          portType="tns:StoragePartner_AnalysisStoragePT"
          variable="getRawDataOpReqMsgVar">
          <correlations>
            <correlation set="choreographyCorrelationSet" />
          </correlations>
        </receive>
        <assign>

```

```

    <copy>
      <from part="payload" query="/choreons:getRawDataOpReqMsgElem
        /choreons:content/choreons:content"
        variable="getRawDataOpReqMsgVar" />
      <to variable="varRawDataAddr_Sto" />
    </copy>
  </assign>
</sequence>
</scope>
<sequence>
  <assign>
    <copy>
      <from>
        <getDataReqElem
          xmlns="http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage">
          <content />
        </getDataReqElem>
      </from>
      <to part="getDataReqElem" variable="getDataRequest" />
    </copy>
    <copy>
      <from variable="varRawDataAddr_Sto" />
      <to part="getDataReqElem" query="/ns0:getDataReqElem/ns0:content"
        variable="getDataRequest" />
    </copy>
  </assign>
  <invoke inputVariable="getDataRequest" operation="getData"
    outputVariable="getDataResponse" partnerLink="PrivateStoPL"
    portType="ns0:PrivateStoragePT">
    <correlations>
      <correlation set="choreographyCorrelationSet" />
    </correlations>
  </invoke>
  <assign>
    <copy>
      <from part="getDataRespElem" query="/ns0:getDataRespElem/ns0:content"
        variable="getDataResponse" />
      <to variable="varRawData_Sto" />
    </copy>
  </assign>
</sequence>
<scope name="Scope-2" variableAccessSerializable="no">
  <sequence>
    <assign>
      <copy>
        <from>
          <getRawDataOpRespMsgElem
            xmlns="http://cdl2bpel.cecka.sap.com/SAC1">
            <content>
              <content />
              <IDElem>
                <ID />
            </content>
          </getRawDataOpRespMsgElem>
        </from>
      </copy>
    </assign>
  </sequence>
</scope>

```

```

        </IDElem>
      </content>
    </getRawDataOpRespMsgElem>
  </from>
  <to part="payload" variable="getRawDataOpRespMsgVar" />
</copy>
<copy>
  <from variable="ChoreographyIdVariable" />
  <to part="payload" query="descendant::choreons:ID"
    variable="getRawDataOpRespMsgVar" />
</copy>
<copy>
  <from variable="varRawData_Sto" />
  <to part="payload" query="/choreons:getRawDataOpRespMsgElem
    /choreons:content/choreons:content"
    variable="getRawDataOpRespMsgVar" />
</copy>
</assign>
<invoke inputVariable="getRawDataOpRespMsgVar"
  operation="getRawDataOpResp" partnerLink="AnalysisStoragePL"
  portType="ns1:AnalysisPartner_AnalysisStoragePT">
  <correlations>
    <correlation set="choreographyCorrelationSet" />
  </correlations>
</invoke>
</sequence>
</scope>
<scope name="Scope-3" variableAccessSerializable="no">
  <sequence>
    <receive operation="storeResultDataOpReq" partnerLink="AnalysisStoragePL"
      portType="tns:StoragePartner_AnalysisStoragePT"
      variable="storeResultDataOpReqMsgVar">
      <correlations>
        <correlation set="choreographyCorrelationSet" />
      </correlations>
    </receive>
    <assign>
      <copy>
        <from part="payload" query="/choreons:storeResultDataOpReqMsgElem
          /choreons:content/choreons:content"
          variable="storeResultDataOpReqMsgVar" />
        <to variable="varResultData_Sto" />
      </copy>
    </assign>
  </sequence>
</scope>
<sequence>
  <assign>
    <copy>
      <from>
        <storeDataReqElem
          xmlns="http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage">

```

```

        <content />
    </storeDataReqElem>
</from>
<to part="storeDataReqElem" variable="storeDataRequest" />
</copy>
<copy>
    <from variable="varResultData_Sto" />
    <to part="storeDataReqElem" query="/ns0:storeDataReqElem/ns0:content"
        variable="storeDataRequest" />
</copy>
</assign>
<invoke inputVariable="storeDataRequest" operation="storeData"
    outputVariable="storeDataResponse"
    partnerLink="PrivateStoPL" portType="ns0:PrivateStoragePT">
    <correlations>
        <correlation set="choreographyCorrelationSet" />
    </correlations>
</invoke>
<assign>
    <copy>
        <from part="storeDataRespElem"
            query="/ns0:storeDataRespElem/ns0:content"
            variable="storeDataResponse" />
        <to variable="varResultDataAddr_Sto" />
    </copy>
</assign>
</sequence>
<scope name="Scope-4" variableAccessSerializable="no">
    <sequence>
        <assign>
            <copy>
                <from>
                    <storeResultDataOpRespMsgElem
                        xmlns="http://cdl2bpel.cecka.sap.com/SAC1">
                        <content>
                            <content />
                            <IDElem>
                                <ID />
                            </IDElem>
                        </content>
                    </storeResultDataOpRespMsgElem>
                </from>
                <to part="payload" variable="storeResultDataOpRespMsgVar" />
            </copy>
            <copy>
                <from variable="ChoreographyIdVariable" />
                <to part="payload" query="descendant::choreons:ID"
                    variable="storeResultDataOpRespMsgVar" />
            </copy>
            <copy>
                <from variable="varResultDataAddr_Sto" />
                <to part="payload" query="/choreons:storeResultDataOpRespMsgElem

```

```

        /choreons:content/choreons:content"
        variable="storeResultDataOpRespMsgVar" />
    </copy>
</assign>
<invoke inputVariable="storeResultDataOpRespMsgVar"
    operation="storeResultDataOpResp" partnerLink="AnalysisStoragePL"
    portType="ns1:AnalysisPartner_AnalysisStoragePT">
    <correlations>
        <correlation set="choreographyCorrelationSet" />
    </correlations>
</invoke>
</sequence>
</scope>
</sequence>
</scope>
</sequence>
</process>

```

A.2 Example 2

A.2.1 WS-CDL Code

```

<?xml version="1.0" encoding="UTF-8" ?>
<package name="SAC_Package" version="2.0"
    targetNamespace="http://cdl2bpel.cecka.sap.com/SAC2"
    xmlns:tns="http://cdl2bpel.cecka.sap.com/SAC2"
    xmlns:cdl2bpelns="http://cdl2bpel.cecka.sap.com"
    xmlns:choreons="http://cdl2bpel.cecka.sap.com/SAC2"
    xmlns="http://www.w3.org/2004/12/ws-chor/cdl"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:schema attributeFormDefault="qualified"
        elementFormDefault="qualified"
        targetNamespace="http://cdl2bpel.cecka.sap.com/SAC2">
    <xsd:complexType name="stringAndIDType">
        <xsd:sequence>
            <xsd:element name="content" type="xsd:string" />
            <xsd:element name="IDElem" type="tns:stringID" />
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="uriAndIDType">
        <xsd:sequence>
            <xsd:element name="content" type="xsd:anyURI" />
            <xsd:element name="IDElem" type="tns:stringID" />
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="stringID">
        <xsd:sequence>
            <xsd:element name="ID" type="xsd:string" />
        </xsd:sequence>
    </xsd:complexType>

```

```

    </xsd:complexType>
</xsd:schema>
<informationType name="stringType" type="xsd:string">
  <cdl2bpelns:map schemaType="tns:stringAndIDType" query="/choreons:content" />
</informationType>
<informationType name="stringIDType" type="tns:stringID" />
<informationType name="uriType" type="xsd:anyURI">
  <cdl2bpelns:map schemaType="tns:uriAndIDType" query="/choreons:content" />
</informationType>
<token name="stringIDToken" informationType="stringIDType" />
<tokenLocator tokenName="stringIDToken" informationType="stringType"
  query="descendant::choreons:ID" />
<tokenLocator tokenName="stringIDToken" informationType="uriType"
  query="descendant::choreons:ID" />
<roleType name="Initiator">
  <behavior name="Initiation" />
</roleType>
<roleType name="AnalysisPartner1">
  <behavior name="Analyzer" />
</roleType>
<roleType name="AnalysisPartner2">
  <behavior name="Analyzer" />
</roleType>
<roleType name="StoragePartner">
  <behavior name="StorageProvider" />
</roleType>
<relationshipType name="Analysis1StorageRel">
  <roleType typeRef="AnalysisPartner1" behavior="Analyzer" />
  <roleType typeRef="StoragePartner" behavior="StorageProvider" />
</relationshipType>
<relationshipType name="Analysis2StorageRel">
  <roleType typeRef="AnalysisPartner2" behavior="Analyzer" />
  <roleType typeRef="StoragePartner" behavior="StorageProvider" />
</relationshipType>
<relationshipType name="InitiatorStorageRel">
  <roleType typeRef="Initiator" behavior="Initiation" />
  <roleType typeRef="StoragePartner" behavior="StorageProvider" />
</relationshipType>
<relationshipType name="InitiatorAnalysis1Rel">
  <roleType typeRef="Initiator" behavior="Initiation" />
  <roleType typeRef="AnalysisPartner1" behavior="Analyzer" />
</relationshipType>
<relationshipType name="InitiatorAnalysis2Rel">
  <roleType typeRef="Initiator" behavior="Initiation" />
  <roleType typeRef="AnalysisPartner2" behavior="Analyzer" />
</relationshipType>
<participantType name="Analyzer1">
  <roleType typeRef="AnalysisPartner1" />
</participantType>
<participantType name="Analyzer2">
  <roleType typeRef="AnalysisPartner2" />
</participantType>

```

```

<participantType name="InitiatorP">
  <roleType typeRef="Initiator" />
</participantType>
<participantType name="StorageP">
  <roleType typeRef="StoragePartner" />
</participantType>
<channelType name="StorageChType" usage="unlimited">
  <roleType typeRef="StoragePartner" behavior="StorageProvider" />
  <identity>
    <token name="stringIDToken" />
  </identity>
</channelType>
<channelType name="Analysis1ChType" usage="unlimited">
  <roleType typeRef="AnalysisPartner1" behavior="Analyzer" />
  <identity>
    <token name="stringIDToken" />
  </identity>
</channelType>
<channelType name="Analysis2ChType" usage="unlimited">
  <roleType typeRef="AnalysisPartner2" behavior="Analyzer" />
  <identity>
    <token name="stringIDToken" />
  </identity>
</channelType>
<choreography name="SAC_Choreography" root="true">
  <relationship type="Analysis1StorageRel" />
  <relationship type="Analysis2StorageRel" />
  <relationship type="InitiatorStorageRel" />
  <relationship type="InitiatorAnalysis1Rel" />
  <relationship type="InitiatorAnalysis2Rel" />
  <variableDefinitions>
    <!-- Variable that is used for outside calls to the choreography
      IMPORTANT: The role for which it is specified defines the process which
      is started first! HAS to be specified at the root choreography, for
      the current CDL2BPEL implementation to work -->
    <variable name="initVOChorVar" informationType="stringType"
      roleTypes="Initiator" />
    <!-- The other variables -->
    <variable name="varRawDataAddr_Init" informationType="uriType"
      roleTypes="Initiator" />
    <variable name="varRawData_Init" informationType="stringType"
      roleTypes="Initiator" />
    <variable name="varResultData1_Init" informationType="stringType"
      roleTypes="Initiator" />
    <variable name="varResultData2_Init" informationType="stringType"
      roleTypes="Initiator" />
    <variable name="varOverallResultData_Init" informationType="stringType"
      roleTypes="Initiator" />
    <variable name="varRawDataAddr_Ana1" informationType="uriType"
      roleTypes="AnalysisPartner1" />
    <variable name="varResultDataAddr_Ana1" informationType="uriType"
      roleTypes="AnalysisPartner1" />
  </variableDefinitions>
</choreography>

```

```

<variable name="varRawData_Ana1" informationType="stringType"
  roleTypes="AnalysisPartner1" />
<variable name="varResultData_Ana1" informationType="stringType"
  roleTypes="AnalysisPartner1" />
<variable name="varRawDataAddr_Sto" informationType="uriType"
  roleTypes="StoragePartner" />
<variable name="varResultDataAddr_Sto" informationType="uriType"
  roleTypes="StoragePartner" />
<variable name="varRawData_Sto" informationType="stringType"
  roleTypes="StoragePartner" />
<variable name="varResultData_Sto" informationType="stringType"
  roleTypes="StoragePartner" />
<variable name="varRawDataAddr_Ana2" informationType="uriType"
  roleTypes="AnalysisPartner2" />
<variable name="varResultDataAddr_Ana2" informationType="uriType"
  roleTypes="AnalysisPartner2" />
<variable name="varRawData_Ana2" informationType="stringType"
  roleTypes="AnalysisPartner2" />
<variable name="varResultData_Ana2" informationType="stringType"
  roleTypes="AnalysisPartner2" />
<variable name="chVarAna1" channelType="Analysis1ChType" />
<variable name="chVarAna2" channelType="Analysis2ChType" />
<variable name="chVarStorage" channelType="StorageChType" />
</variableDefinitions>
<sequence>
  <assign roleType="Initiator">
    <copy name="initial-assign">
      <source variable="cdl:getVariable('initVOChorVar','','')" />
      <target variable="cdl:getVariable('varRawData_Init','','')" />
    </copy>
  </assign>
  <silentAction roleType="Initiator">
    <description type="documentation">IdentifyNeed</description>
  </silentAction>
  <interaction name="storeRawData" operation="storeToDB"
    channelVariable="chVarStorage">
    <participate relationshipType="InitiatorStorageRel"
      fromRoleTypeRef="Initiator" toRoleTypeRef="StoragePartner" />
    <exchange name="sto1" informationType="stringType" action="request">
      <send variable="cdl:getVariable('varRawData_Init','','')" />
      <receive variable="cdl:getVariable('varRawData_Sto','','')" />
    </exchange>
  </interaction>
  <silentAction roleType="StoragePartner">
    <description type="documentation">storeDataToDB varRawData_Sto
      varRawDataAddr_Sto</description>
  </silentAction>
  <interaction name="storeRawDataResp" operation="storeToDB"
    channelVariable="chVarStorage">
    <participate relationshipType="InitiatorStorageRel"
      fromRoleTypeRef="Initiator" toRoleTypeRef="StoragePartner" />
    <exchange name="sto2" informationType="uriType" action="respond">

```

```

    <send variable="cdl:getVariable('varRawDataAddr_Sto','','')" />
    <receive variable="cdl:getVariable('varRawDataAddr_Init','','')" />
  </exchange>
</interaction>
<parallel>
  <sequence>
    <interaction name="analyzeDataReq1" operation="AnalyzeOp1"
      channelVariable="chVarAna1">
      <participate relationshipType="InitiatorAnalysis1Rel"
        fromRoleTypeRef="Initiator" toRoleTypeRef="AnalysisPartner1" />
      <exchange name="exAnalyze1" informationType="uriType" action="request">
        <send variable="cdl:getVariable('varRawDataAddr_Init','','')" />
        <receive variable="cdl:getVariable('varRawDataAddr_Ana1','','')" />
      </exchange>
    </interaction>
    <interaction name="getRawDataReq1" operation="getRawDataOp1"
      channelVariable="chVarStorage">
      <participate relationshipType="Analysis1StorageRel"
        fromRoleTypeRef="AnalysisPartner1" toRoleTypeRef="StoragePartner" />
      <exchange name="exRawDataAddr1" informationType="uriType"
        action="request">
        <send variable="cdl:getVariable('varRawDataAddr_Ana1','','')" />
        <receive variable="cdl:getVariable('varRawDataAddr_Sto','','')" />
      </exchange>
    </interaction>
    <silentAction roleType="StoragePartner">
      <description type="documentation">getRawDataFromDB varRawDataAddr_Sto
        varRawData_Sto</description>
    </silentAction>
    <interaction name="getRawDataResp1" operation="getRawDataOp1"
      channelVariable="chVarStorage">
      <participate relationshipType="Analysis1StorageRel"
        fromRoleTypeRef="AnalysisPartner1" toRoleTypeRef="StoragePartner" />
      <exchange name="exRawData1" informationType="stringType"
        action="respond">
        <send variable="cdl:getVariable('varRawData_Sto','','')" />
        <receive variable="cdl:getVariable('varRawData_Ana1','','')" />
      </exchange>
    </interaction>
    <silentAction roleType="AnalysisPartner1">
      <description type="documentation">analyzeData varRawData_Ana1
        varResultData_Ana1</description>
    </silentAction>
    <interaction name="storeResultDataReq1" operation="storeResultDataOp1"
      channelVariable="chVarStorage">
      <participate relationshipType="Analysis1StorageRel"
        fromRoleTypeRef="AnalysisPartner1" toRoleTypeRef="StoragePartner" />
      <exchange name="exResultData1" informationType="stringType"
        action="request">
        <send variable="cdl:getVariable('varResultData_Ana1','','')" />
        <receive variable="cdl:getVariable('varResultData_Sto','','')" />
      </exchange>

```

```

</interaction>
<silentAction roleType="StoragePartner">
  <description type="documentation">storeDataToDB varResultData_Sto
    varResultDataAddr_Sto</description>
</silentAction>
<interaction name="storeResultDataResp1" operation="storeResultDataOp1"
  channelVariable="chVarStorage">
  <participate relationshipType="Analysis1StorageRel"
    fromRoleTypeRef="AnalysisPartner1" toRoleTypeRef="StoragePartner" />
  <exchange name="exResultAddr1" informationType="uriType"
    action="respond">
    <send variable="cdl:getVariable('varResultDataAddr_Sto','','')" />
    <receive variable="cdl:getVariable('varResultDataAddr_Ana1','','')" />
  </exchange>
</interaction>
<interaction name="analyzeDataResp1" operation="AnalyzeOp1"
  channelVariable="chVarAna1">
  <participate relationshipType="InitiatorAnalysis1Rel"
    fromRoleTypeRef="Initiator" toRoleTypeRef="AnalysisPartner1" />
  <exchange name="exAnalyzeResp1" informationType="stringType"
    action="respond">
    <send variable="cdl:getVariable('varResultData_Ana1','','')" />
    <receive variable="cdl:getVariable('varResultData1_Init','','')" />
  </exchange>
</interaction>
</sequence>
<sequence>
  <interaction name="analyzeDataReq2" operation="AnalyzeOp2"
    channelVariable="chVarAna2">
    <participate relationshipType="InitiatorAnalysis2Rel"
      fromRoleTypeRef="Initiator" toRoleTypeRef="AnalysisPartner2" />
    <exchange name="exAnalyze2" informationType="uriType" action="request">
      <send variable="cdl:getVariable('varRawDataAddr_Init','','')" />
      <receive variable="cdl:getVariable('varRawDataAddr_Ana2','','')" />
    </exchange>
  </interaction>
  <interaction name="getRawDataReq2" operation="getRawDataOp2"
    channelVariable="chVarStorage">
    <participate relationshipType="Analysis2StorageRel"
      fromRoleTypeRef="AnalysisPartner2" toRoleTypeRef="StoragePartner" />
    <exchange name="exRawDataAddr2" informationType="uriType"
      action="request">
      <send variable="cdl:getVariable('varRawDataAddr_Ana2','','')" />
      <receive variable="cdl:getVariable('varRawDataAddr_Sto','','')" />
    </exchange>
  </interaction>
<silentAction roleType="StoragePartner">
  <description type="documentation">getRawDataFromDB varRawDataAddr_Sto
    varRawData_Sto</description>
</silentAction>
<interaction name="getRawDataResp2" operation="getRawDataOp2"
  channelVariable="chVarStorage">

```

```

    <participate relationshipType="Analysis2StorageRel"
      fromRoleTypeRef="AnalysisPartner2" toRoleTypeRef="StoragePartner" />
    <exchange name="exRawData2" informationType="stringType"
      action="respond">
      <send variable="cdl:getVariable('varRawData_Sto','','')" />
      <receive variable="cdl:getVariable('varRawData_Ana2','','')" />
    </exchange>
  </interaction>
  <silentAction roleType="AnalysisPartner2">
    <description type="documentation">analyzeData varRawData_Ana2
      varResultData_Ana2</description>
  </silentAction>
  <interaction name="storeResultDataReq2" operation="storeResultDataOp2"
    channelVariable="chVarStorage">
    <participate relationshipType="Analysis2StorageRel"
      fromRoleTypeRef="AnalysisPartner2" toRoleTypeRef="StoragePartner" />
    <exchange name="exResultData2" informationType="stringType"
      action="request">
      <send variable="cdl:getVariable('varResultData_Ana2','','')" />
      <receive variable="cdl:getVariable('varResultData_Sto','','')" />
    </exchange>
  </interaction>
  <silentAction roleType="StoragePartner">
    <description type="documentation">storeDataToDB varResultData_Sto
      varResultDataAddr_Sto</description>
  </silentAction>
  <interaction name="storeResultDataResp2" operation="storeResultDataOp2"
    channelVariable="chVarStorage">
    <participate relationshipType="Analysis2StorageRel"
      fromRoleTypeRef="AnalysisPartner2" toRoleTypeRef="StoragePartner" />
    <exchange name="exResultAddr2" informationType="uriType"
      action="respond">
      <send variable="cdl:getVariable('varResultDataAddr_Sto','','')" />
      <receive variable="cdl:getVariable('varResultDataAddr_Ana2','','')" />
    </exchange>
  </interaction>
  <interaction name="analyzeDataResp2" operation="AnalyzeOp2"
    channelVariable="chVarAna2">
    <participate relationshipType="InitiatorAnalysis2Rel"
      fromRoleTypeRef="Initiator" toRoleTypeRef="AnalysisPartner2" />
    <exchange name="exAnalyzeResp2" informationType="stringType"
      action="respond">
      <send variable="cdl:getVariable('varResultData_Ana2','','')" />
      <receive variable="cdl:getVariable('varResultData2_Init','','')" />
    </exchange>
  </interaction>
</sequence>
</parallel>
<silentAction roleType="Initiator">
  <description type="documentation">combineResults varResultData1_Init
    varResultData2_Init varOverallResultData_Init</description>
</silentAction>

```

```

    </sequence>
  </choreography>
</package>

```

A.2.2 WSBPEL Code

Initiator Role

```

<?xml version="1.0" encoding="UTF-8" ?>
<process name="BP_Initiator" suppressJoinFailure="yes"
  targetNamespace="http://cdl2bpel.cecka.sap.com/SAC2/Initiator"
  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:choreons="http://cdl2bpel.cecka.sap.com/SAC2"
  xmlns:ns0="http://cdl2bpel.cecka.sap.com/SAC2/AnalysisPartner1"
  xmlns:ns1="http://cdl2bpel.cecka.sap.com/SAC2/AnalysisPartner2"
  xmlns:ns2="http://cdl2bpel.cecka.sap.com/SAC2/StoragePartner"
  xmlns:tns="http://cdl2bpel.cecka.sap.com/SAC2/Initiator"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <partnerLinks>
    <partnerLink myRole="InitiatorRole_InitProcessPL" name="InitProcessPL"
      partnerLinkType="tns:InitProcessPLType" />
    <partnerLink name="InitProcessAnalysisPartner1PL"
      partnerLinkType="tns:InitProcessAnalysisPartner1PLType"
      partnerRole="AnalysisPartner1Role_InitProcessAnalysisPartner1PL" />
    <partnerLink name="InitProcessAnalysisPartner2PL"
      partnerLinkType="tns:InitProcessAnalysisPartner2PLType"
      partnerRole="AnalysisPartner2Role_InitProcessAnalysisPartner2PL" />
    <partnerLink name="InitProcessStoragePartnerPL"
      partnerLinkType="tns:InitProcessStoragePartnerPLType"
      partnerRole="StoragePartnerRole_InitProcessStoragePartnerPL" />
    <partnerLink myRole="InitiatorRole_InitiatorStoragePL" name="InitiatorStoragePL"
      partnerLinkType="tns:InitiatorStoragePLType"
      partnerRole="StoragePartnerRole_InitiatorStoragePL" />
    <partnerLink myRole="InitiatorRole_InitiatorAnalysis1PL"
      name="InitiatorAnalysis1PL" partnerLinkType="tns:InitiatorAnalysis1PLType"
      partnerRole="AnalysisPartner1Role_InitiatorAnalysis1PL" />
    <partnerLink myRole="InitiatorRole_InitiatorAnalysis2PL"
      name="InitiatorAnalysis2PL" partnerLinkType="tns:InitiatorAnalysis2PLType"
      partnerRole="AnalysisPartner2Role_InitiatorAnalysis2PL" />
  </partnerLinks>
  <variables>
    <variable messageType="tns:InitProcessOpMsg" name="InitProcessOpMsgVar" />
    <variable name="ChoreographyIdVariable" type="choreons:stringID" />
    <variable messageType="ns0:InitProcessAnalysisPartner1OpMsg"
      name="InitProcessAnalysisPartner1OpMsgVar" />
    <variable messageType="ns0:InitProcessAnalysisPartner1OpRespMsg"
      name="InitProcessAnalysisPartner1OpRespMsgVar" />
    <variable messageType="ns1:InitProcessAnalysisPartner2OpMsg"
      name="InitProcessAnalysisPartner2OpMsgVar" />
  </variables>

```

```

<variable messageType="ns1:InitProcessAnalysisPartner2OpRespMsg"
  name="InitProcessAnalysisPartner2OpRespMsgVar" />
<variable messageType="ns2:InitProcessStoragePartnerOpMsg"
  name="InitProcessStoragePartnerOpMsgVar" />
<variable messageType="ns2:InitProcessStoragePartnerOpRespMsg"
  name="InitProcessStoragePartnerOpRespMsgVar" />
<variable name="initVOChorVar" type="xsd:string" />
<variable name="varRawDataAddr_Init" type="xsd:anyURI" />
<variable name="varRawData_Init" type="xsd:string" />
<variable name="varResultData1_Init" type="xsd:string" />
<variable name="varResultData2_Init" type="xsd:string" />
<variable name="varOverallResultData_Init" type="xsd:string" />
<variable messageType="ns2:storeToDBReqMsg" name="storeToDBReqMsgVar" />
<variable messageType="tns:storeToDBRespMsg" name="storeToDBRespMsgVar" />
<variable messageType="ns0:AnalyzeOp1ReqMsg" name="AnalyzeOp1ReqMsgVar" />
<variable messageType="tns:AnalyzeOp1RespMsg" name="AnalyzeOp1RespMsgVar" />
<variable messageType="ns1:AnalyzeOp2ReqMsg" name="AnalyzeOp2ReqMsgVar" />
<variable messageType="tns:AnalyzeOp2RespMsg" name="AnalyzeOp2RespMsgVar" />
</variables>
<correlationSets>
  <correlationSet name="choreographyCorrelationSet"
    properties="tns:stringIDToken" />
</correlationSets>
<sequence>
  <receive createInstance="yes" name="InitProcess" operation="InitProcessOp"
    partnerLink="InitProcessPL" portType="tns:InitProcess"
    variable="InitProcessOpMsgVar">
    <correlations>
      <correlation initiate="yes" set="choreographyCorrelationSet" />
    </correlations>
  </receive>
  <assign>
    <copy>
      <from part="initVOChorVar" query="/choreons:InitProcessOpMsgElem
        /choreons:content/choreons:content" variable="InitProcessOpMsgVar" />
      <to variable="initVOChorVar" />
    </copy>
    <copy>
      <from part="initVOChorVar" query="descendant::choreons:ID"
        variable="InitProcessOpMsgVar" />
      <to variable="ChoreographyIdVariable" />
    </copy>
    <copy>
      <from>
        <InitProcessAnalysisPartner1OpMsgElem
          xmlns="http://cdl2bpel.cecka.sap.com/SAC2">
          <IDElem>
            <ID />
          </IDElem>
        </InitProcessAnalysisPartner1OpMsgElem>
      </from>
      <to part="Start" variable="InitProcessAnalysisPartner1OpMsgVar" />
    </copy>
  </assign>
</sequence>

```

```

</copy>
<copy>
  <from variable="ChoreographyIdVariable" />
  <to part="Start" query="descendant::choreons:ID"
    variable="InitProcessAnalysisPartner1OpMsgVar" />
</copy>
<copy>
  <from>
    <InitProcessAnalysisPartner2OpMsgElem
      xmlns="http://cdl2bpel.cecka.sap.com/SAC2">
      <IDElem>
        <ID />
      </IDElem>
    </InitProcessAnalysisPartner2OpMsgElem>
  </from>
  <to part="Start" variable="InitProcessAnalysisPartner2OpMsgVar" />
</copy>
<copy>
  <from variable="ChoreographyIdVariable" />
  <to part="Start" query="descendant::choreons:ID"
    variable="InitProcessAnalysisPartner2OpMsgVar" />
</copy>
<copy>
  <from>
    <InitProcessStoragePartnerOpMsgElem
      xmlns="http://cdl2bpel.cecka.sap.com/SAC2">
      <IDElem>
        <ID />
      </IDElem>
    </InitProcessStoragePartnerOpMsgElem>
  </from>
  <to part="Start" variable="InitProcessStoragePartnerOpMsgVar" />
</copy>
<copy>
  <from variable="ChoreographyIdVariable" />
  <to part="Start" query="descendant::choreons:ID"
    variable="InitProcessStoragePartnerOpMsgVar" />
</copy>
</assign>
<flow>
  <invoke inputVariable="InitProcessAnalysisPartner1OpMsgVar"
    name="InitAnalysisPartner1Process"
    operation="InitProcessAnalysisPartner1Op"
    outputVariable="InitProcessAnalysisPartner1OpRespMsgVar"
    partnerLink="InitProcessAnalysisPartner1PL"
    portType="ns0:InitProcessAnalysisPartner1">
    <correlations>
      <correlation set="choreographyCorrelationSet" />
    </correlations>
  </invoke>
  <invoke inputVariable="InitProcessAnalysisPartner2OpMsgVar"
    name="InitAnalysisPartner2Process" operation="InitProcessAnalysisPartner2Op"

```

```

    outputVariable="InitProcessAnalysisPartner2OpRespMsgVar"
    partnerLink="InitProcessAnalysisPartner2PL"
    portType="ns1:InitProcessAnalysisPartner2">
    <correlations>
      <correlation set="choreographyCorrelationSet" />
    </correlations>
  </invoke>
  <invoke inputVariable="InitProcessStoragePartnerOpMsgVar"
    name="InitStoragePartnerProcess"
    operation="InitProcessStoragePartnerOp"
    outputVariable="InitProcessStoragePartnerOpRespMsgVar"
    partnerLink="InitProcessStoragePartnerPL"
    portType="ns2:InitProcessStoragePartner">
    <correlations>
      <correlation set="choreographyCorrelationSet" />
    </correlations>
  </invoke>
</flow>
<scope variableAccessSerializable="no">
  <sequence>
    <assign>
      <copy>
        <from variable="initVOChorVar" />
        <to variable="varRawData_Init" />
      </copy>
    </assign>
    <scope name="Scope-1" variableAccessSerializable="no">
      <sequence>
        <assign>
          <copy>
            <from>
              <storeToDBReqMsgElem xmlns="http://cdl2bpel.cecka.sap.com/SAC2">
                <content>
                  <content />
                  <IDElem>
                    <ID />
                  </IDElem>
                </content>
              </storeToDBReqMsgElem>
            </from>
            <to part="payload" variable="storeToDBReqMsgVar" />
          </copy>
          <copy>
            <from variable="ChoreographyIdVariable" />
            <to part="payload" query="descendant::choreons:ID"
              variable="storeToDBReqMsgVar" />
          </copy>
          <copy>
            <from variable="varRawData_Init" />
            <to part="payload" query="/choreons:storeToDBReqMsgElem
              /choreons:content/choreons:content"
              variable="storeToDBReqMsgVar" />
          </copy>
        </assign>
      </sequence>
    </scope>
  </sequence>
</scope>

```

```

        </copy>
    </assign>
    <invoke inputVariable="storeToDBReqMsgVar" operation="storeToDBReq"
        partnerLink="InitiatorStoragePL"
        portType="ns2:StoragePartner_InitiatorStoragePT">
        <correlations>
            <correlation set="choreographyCorrelationSet" />
        </correlations>
    </invoke>
</sequence>
</scope>
<scope name="Scope-2" variableAccessSerializable="no">
    <sequence>
        <receive operation="storeToDBResp"
            partnerLink="InitiatorStoragePL"
            portType="tns:Initiator_InitiatorStoragePT"
            variable="storeToDBRespMsgVar">
            <correlations>
                <correlation set="choreographyCorrelationSet" />
            </correlations>
        </receive>
        <assign>
            <copy>
                <from part="payload" query="/choreons:storeToDBRespMsgElem
                    /choreons:content/choreons:content"
                    variable="storeToDBRespMsgVar" />
                <to variable="varRawDataAddr_Init" />
            </copy>
        </assign>
    </sequence>
</scope>
<flow>
    <sequence>
        <scope name="Scope-3" variableAccessSerializable="no">
            <sequence>
                <assign>
                    <copy>
                        <from>
                            <AnalyzeOp1ReqMsgElem
                                xmlns="http://cdl2bpel.cecka.sap.com/SAC2">
                                <content>
                                    <content />
                                    <IDElem>
                                        <ID />
                                    </IDElem>
                                </content>
                            </AnalyzeOp1ReqMsgElem>
                        </from>
                        <to part="payload" variable="AnalyzeOp1ReqMsgVar" />
                    </copy>
                    <copy>
                        <from variable="ChoreographyIdVariable" />

```

```

        <to part="payload" query="descendant::choreons:ID"
            variable="AnalyzeOp1ReqMsgVar" />
    </copy>
    <copy>
        <from variable="varRawDataAddr_Init" />
        <to part="payload" query="/choreons:AnalyzeOp1ReqMsgElem
            /choreons:content/choreons:content"
            variable="AnalyzeOp1ReqMsgVar" />
    </copy>
</assign>
<invoke inputVariable="AnalyzeOp1ReqMsgVar"
    operation="AnalyzeOp1Req" partnerLink="InitiatorAnalysis1PL"
    portType="ns0:AnalysisPartner1_InitiatorAnalysis1PT">
    <correlations>
        <correlation set="choreographyCorrelationSet" />
    </correlations>
</invoke>
</sequence>
</scope>
<scope name="Scope-4" variableAccessSerializable="no">
    <sequence>
        <receive operation="AnalyzeOp1Resp"
            partnerLink="InitiatorAnalysis1PL"
            portType="tns:Initiator_InitiatorAnalysis1PT"
            variable="AnalyzeOp1RespMsgVar">
            <correlations>
                <correlation set="choreographyCorrelationSet" />
            </correlations>
        </receive>
        <assign>
            <copy>
                <from part="payload" query="/choreons:AnalyzeOp1RespMsgElem
                    /choreons:content/choreons:content"
                    variable="AnalyzeOp1RespMsgVar" />
                <to variable="varResultData1_Init" />
            </copy>
        </assign>
    </sequence>
</scope>
</sequence>
<sequence>
    <scope name="Scope-5" variableAccessSerializable="no">
        <sequence>
            <assign>
                <copy>
                    <from>
                        <AnalyzeOp2ReqMsgElem
                            xmlns="http://cdl2bpel.cecka.sap.com/SAC2">
                            <content>
                                <content />
                                <IDElem>
                                    <ID />
                                </IDElem>
                            </content>
                        </AnalyzeOp2ReqMsgElem>
                    </from>
                </copy>
            </assign>
        </sequence>
    </scope>
</sequence>

```

```

        </IDElem>
      </content>
    </AnalyzeOp2ReqMsgElem>
  </from>
  <to part="payload" variable="AnalyzeOp2ReqMsgVar" />
</copy>
<copy>
  <from variable="ChoreographyIdVariable" />
  <to part="payload" query="descendant::choreons:ID"
    variable="AnalyzeOp2ReqMsgVar" />
</copy>
<copy>
  <from variable="varRawDataAddr_Init" />
  <to part="payload" query="/choreons:AnalyzeOp2ReqMsgElem
    /choreons:content/choreons:content"
    variable="AnalyzeOp2ReqMsgVar" />
</copy>
</assign>
<invoke inputVariable="AnalyzeOp2ReqMsgVar" operation="AnalyzeOp2Req"
  partnerLink="InitiatorAnalysis2PL"
  portType="ns1:AnalysisPartner2_InitiatorAnalysis2PT">
  <correlations>
    <correlation set="choreographyCorrelationSet" />
  </correlations>
</invoke>
</sequence>
</scope>
<scope name="Scope-6" variableAccessSerializable="no">
  <sequence>
    <receive operation="AnalyzeOp2Resp"
      partnerLink="InitiatorAnalysis2PL"
      portType="tns:Initiator_InitiatorAnalysis2PT"
      variable="AnalyzeOp2RespMsgVar">
      <correlations>
        <correlation set="choreographyCorrelationSet" />
      </correlations>
    </receive>
    <assign>
      <copy>
        <from part="payload" query="/choreons:AnalyzeOp2RespMsgElem
          /choreons:content/choreons:content"
          variable="AnalyzeOp2RespMsgVar" />
        <to variable="varResultData2_Init" />
      </copy>
    </assign>
  </sequence>
</scope>
</sequence>
</flow>
<assign>
  <copy>
    <from expression="concat('Result1: ',

```

```

        bpws:getVariableData('varResultData1_Init'), ' -- ',
        'Result2: ', bpws:getVariableData('varResultData2_Init'))" />
    <to variable="varOverallResultData_Init" />
  </copy>
</assign>
</sequence>
</scope>
</sequence>
</process>

```

AnalysisPartner1 Role

```

<?xml version="1.0" encoding="UTF-8" ?>
<process name="BP_AnalysisPartner1" suppressJoinFailure="yes"
  targetNamespace="http://cdl2bpel.cecka.sap.com/SAC2/AnalysisPartner1"
  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:choreons="http://cdl2bpel.cecka.sap.com/SAC2"
  xmlns:ns0="http://cdl2bpel.cecka.sap.com/SAC2/StoragePartner"
  xmlns:ns1="http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis"
  xmlns:ns2="http://cdl2bpel.cecka.sap.com/SAC2/Initiator"
  xmlns:tns="http://cdl2bpel.cecka.sap.com/SAC2/AnalysisPartner1"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <partnerLinks>
    <partnerLink myRole="AnalysisPartner1Role_InitProcessAnalysisPartner1PL"
      name="InitProcessAnalysisPartner1PL"
      partnerLinkType="tns:InitProcessAnalysisPartner1PLType" />
    <partnerLink myRole="AnalysisPartner1Role_InitiatorAnalysis1PL"
      name="InitiatorAnalysis1PL" partnerLinkType="tns:InitiatorAnalysis1PLType"
      partnerRole="InitiatorRole_InitiatorAnalysis1PL" />
    <partnerLink myRole="AnalysisPartner1Role_Analysis1StoragePL"
      name="Analysis1StoragePL" partnerLinkType="tns:Analysis1StoragePLType"
      partnerRole="StoragePartnerRole_Analysis1StoragePL" />
    <partnerLink name="PrivateAnaPL" partnerLinkType="tns:PrivateAnaPLType"
      partnerRole="Role_PrivateAnaPL" />
  </partnerLinks>
  <variables>
    <variable messageType="tns:InitProcessAnalysisPartner1OpMsg"
      name="InitProcessAnalysisPartner1OpMsgVar" />
    <variable name="ChoreographyIdVariable" type="choreons:stringID" />
    <variable messageType="tns:InitProcessAnalysisPartner1OpRespMsg"
      name="InitProcessAnalysisPartner1OpRespMsgVar" />
    <variable name="varRawDataAddr_Ana1" type="xsd:anyURI" />
    <variable name="varResultDataAddr_Ana1" type="xsd:anyURI" />
    <variable name="varRawData_Ana1" type="xsd:string" />
    <variable name="varResultData_Ana1" type="xsd:string" />
    <variable messageType="tns:AnalyzeOp1ReqMsg" name="AnalyzeOp1ReqMsgVar" />
    <variable messageType="ns0:getRawDataOp1ReqMsg" name="getRawDataOp1ReqMsgVar" />
    <variable messageType="tns:getRawDataOp1RespMsg" name="getRawDataOp1RespMsgVar"/>
    <variable messageType="ns1:analyzeDataRequest" name="analyzeDataRequest" />
    <variable messageType="ns1:analyzeDataResponse" name="analyzeDataResponse" />
  </variables>

```

```

<variable messageType="ns0:storeResultDataOp1ReqMsg"
  name="storeResultDataOp1ReqMsgVar" />
<variable messageType="tns:storeResultDataOp1RespMsg"
  name="storeResultDataOp1RespMsgVar" />
<variable messageType="ns2:AnalyzeOp1RespMsg" name="AnalyzeOp1RespMsgVar" />
</variables>
<correlationSets>
  <correlationSet name="choreographyCorrelationSet"
    properties="tns:stringIDToken" />
</correlationSets>
<sequence>
  <receive createInstance="yes" name="InitProcess"
    operation="InitProcessAnalysisPartner1Op"
    partnerLink="InitProcessAnalysisPartner1PL"
    portType="tns:InitProcessAnalysisPartner1"
    variable="InitProcessAnalysisPartner1OpMsgVar">
    <correlations>
      <correlation initiate="yes" set="choreographyCorrelationSet" />
    </correlations>
  </receive>
  <assign>
    <copy>
      <from>
        <InitProcessAnalysisPartner1OpRespMsgElem
          xmlns="http://cdl2bpel.cecka.sap.com/SAC2">
          <IDElem>
            <ID />
          </IDElem>
        </InitProcessAnalysisPartner1OpRespMsgElem>
      </from>
      <to part="ACK" variable="InitProcessAnalysisPartner1OpRespMsgVar" />
    </copy>
    <copy>
      <from part="Start" query="descendant::choreons:ID"
        variable="InitProcessAnalysisPartner1OpMsgVar" />
      <to variable="ChoreographyIdVariable" />
    </copy>
    <copy>
      <from variable="ChoreographyIdVariable" />
      <to part="ACK" query="descendant::choreons:ID"
        variable="InitProcessAnalysisPartner1OpRespMsgVar" />
    </copy>
  </assign>
  <reply name="InitResponse" operation="InitProcessAnalysisPartner1Op"
    partnerLink="InitProcessAnalysisPartner1PL"
    portType="tns:InitProcessAnalysisPartner1"
    variable="InitProcessAnalysisPartner1OpRespMsgVar">
    <correlations>
      <correlation set="choreographyCorrelationSet" />
    </correlations>
  </reply>
</scope variableAccessSerializable="no">

```

```

<sequence>
  <flow>
    <sequence>
      <scope name="Scope-1" variableAccessSerializable="no">
        <sequence>
          <receive operation="AnalyzeOp1Req"
            partnerLink="InitiatorAnalysis1PL"
            portType="tns:AnalysisPartner1_InitiatorAnalysis1PT"
            variable="AnalyzeOp1ReqMsgVar">
            <correlations>
              <correlation set="choreographyCorrelationSet" />
            </correlations>
          </receive>
          <assign>
            <copy>
              <from part="payload" query="/choreons:AnalyzeOp1ReqMsgElem
                /choreons:content/choreons:content"
                variable="AnalyzeOp1ReqMsgVar" />
              <to variable="varRawDataAddr_Anal1" />
            </copy>
          </assign>
        </sequence>
      </scope>
      <scope name="Scope-2" variableAccessSerializable="no">
        <sequence>
          <assign>
            <copy>
              <from>
                <getRawDataOp1ReqMsgElem
                  xmlns="http://cdl2bpel.cecka.sap.com/SAC2">
                  <content>
                    <content />
                    <IDElem>
                      <ID />
                    </IDElem>
                  </content>
                </getRawDataOp1ReqMsgElem>
              </from>
              <to part="payload" variable="getRawDataOp1ReqMsgVar" />
            </copy>
            <copy>
              <from variable="ChoreographyIdVariable" />
              <to part="payload" query="descendant::choreons:ID"
                variable="getRawDataOp1ReqMsgVar" />
            </copy>
            <copy>
              <from variable="varRawDataAddr_Anal1" />
              <to part="payload" query="/choreons:getRawDataOp1ReqMsgElem
                /choreons:content/choreons:content"
                variable="getRawDataOp1ReqMsgVar" />
            </copy>
          </assign>
        </sequence>
      </scope>
    </flow>
  </sequence>

```

```

    <invoke inputVariable="getRawDataOp1ReqMsgVar"
      operation="getRawDataOp1Req" partnerLink="Analysis1StoragePL"
      portType="ns0:StoragePartner_Analysis1StoragePT">
      <correlations>
        <correlation set="choreographyCorrelationSet" />
      </correlations>
    </invoke>
  </sequence>
</scope>
<scope name="Scope-3" variableAccessSerializable="no">
  <sequence>
    <receive operation="getRawDataOp1Resp"
      partnerLink="Analysis1StoragePL"
      portType="tns:AnalysisPartner1_Analysis1StoragePT"
      variable="getRawDataOp1RespMsgVar">
      <correlations>
        <correlation set="choreographyCorrelationSet" />
      </correlations>
    </receive>
    <assign>
      <copy>
        <from part="payload" query="/choreons:getRawDataOp1RespMsgElem
          /choreons:content/choreons:content"
          variable="getRawDataOp1RespMsgVar" />
        <to variable="varRawData_Ana1" />
      </copy>
    </assign>
  </sequence>
</scope>
<sequence>
  <assign>
    <copy>
      <from>
        <analyzeDataReqElem
          xmlns="http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis">
          <content />
        </analyzeDataReqElem>
      </from>
      <to part="analyzeDataReqElem" variable="analyzeDataRequest" />
    </copy>
    <copy>
      <from variable="varRawData_Ana1" />
      <to part="analyzeDataReqElem"
        query="/ns1:analyzeDataReqElem/ns1:content"
        variable="analyzeDataRequest" />
    </copy>
  </assign>
  <invoke inputVariable="analyzeDataRequest"
    operation="analyzeData" outputVariable="analyzeDataResponse"
    partnerLink="PrivateAnaPL" portType="ns1:PrivAnalysisPT">
    <correlations>
      <correlation set="choreographyCorrelationSet" />
    </correlations>
  </invoke>
</sequence>

```

```

    </correlations>
  </invoke>
  <assign>
    <copy>
      <from part="analyzeDataRespElem"
        query="/ns1:analyzeDataRespElem/ns1:content"
        variable="analyzeDataResponse" />
      <to variable="varResultData_Ana1" />
    </copy>
  </assign>
</sequence>
<scope name="Scope-4" variableAccessSerializable="no">
  <sequence>
    <assign>
      <copy>
        <from>
          <storeResultDataOp1ReqMsgElem
            xmlns="http://cdl2bpe1.cecka.sap.com/SAC2">
            <content>
              <content />
              <IDElem>
                <ID />
              </IDElem>
            </content>
          </storeResultDataOp1ReqMsgElem>
        </from>
        <to part="payload" variable="storeResultDataOp1ReqMsgVar" />
      </copy>
      <copy>
        <from variable="ChoreographyIdVariable" />
        <to part="payload" query="descendant::choreons:ID"
          variable="storeResultDataOp1ReqMsgVar" />
      </copy>
      <copy>
        <from variable="varResultData_Ana1" />
        <to part="payload" query="/choreons:storeResultDataOp1ReqMsgElem
          /choreons:content/choreons:content"
          variable="storeResultDataOp1ReqMsgVar" />
      </copy>
    </assign>
    <invoke inputVariable="storeResultDataOp1ReqMsgVar"
      operation="storeResultDataOp1Req" partnerLink="Analysis1StoragePL"
      portType="ns0:StoragePartner_Analysis1StoragePT">
      <correlations>
        <correlation set="choreographyCorrelationSet" />
      </correlations>
    </invoke>
  </sequence>
</scope>
<scope name="Scope-5" variableAccessSerializable="no">
  <sequence>
    <receive operation="storeResultDataOp1Resp"

```

```

partnerLink="Analysis1StoragePL"
portType="tns:AnalysisPartner1_Analysis1StoragePT"
variable="storeResultDataOp1RespMsgVar">
<correlations>
  <correlation set="choreographyCorrelationSet" />
</correlations>
</receive>
<assign>
  <copy>
    <from part="payload"
      query="/choreons:storeResultDataOp1RespMsgElem
        /choreons:content/choreons:content"
      variable="storeResultDataOp1RespMsgVar" />
    <to variable="varResultDataAddr_Ana1" />
  </copy>
</assign>
</sequence>
</scope>
<scope name="Scope-6" variableAccessSerializable="no">
  <sequence>
    <assign>
      <copy>
        <from>
          <AnalyzeOp1RespMsgElem
            xmlns="http://cdl2bpel.cecka.sap.com/SAC2">
            <content>
              <content />
              <IDElem>
                <ID />
              </IDElem>
            </content>
          </AnalyzeOp1RespMsgElem>
        </from>
        <to part="payload" variable="AnalyzeOp1RespMsgVar" />
      </copy>
      <copy>
        <from variable="ChoreographyIdVariable" />
        <to part="payload" query="descendant::choreons:ID"
          variable="AnalyzeOp1RespMsgVar" />
      </copy>
      <copy>
        <from variable="varResultData_Ana1" />
        <to part="payload" query="/choreons:AnalyzeOp1RespMsgElem
          /choreons:content/choreons:content"
          variable="AnalyzeOp1RespMsgVar" />
      </copy>
    </assign>
    <invoke inputVariable="AnalyzeOp1RespMsgVar"
      operation="AnalyzeOp1Resp" partnerLink="InitiatorAnalysis1PL"
      portType="ns2:Initiator_InitiatorAnalysis1PT">
    <correlations>
      <correlation set="choreographyCorrelationSet" />

```

```

        </correlations>
    </invoke>
</sequence>
</scope>
</sequence>
</flow>
</sequence>
</scope>
</sequence>
</process>

```

AnalysisPartner2 Role

```

<?xml version="1.0" encoding="UTF-8" ?>
<process name="BP_AnalysisPartner2" suppressJoinFailure="yes"
  targetNamespace="http://cdl2bpel.cecka.sap.com/SAC2/AnalysisPartner2"
  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:choreons="http://cdl2bpel.cecka.sap.com/SAC2"
  xmlns:ns0="http://cdl2bpel.cecka.sap.com/SAC2/StoragePartner"
  xmlns:ns1="http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis"
  xmlns:ns2="http://cdl2bpel.cecka.sap.com/SAC2/Initiator"
  xmlns:tns="http://cdl2bpel.cecka.sap.com/SAC2/AnalysisPartner2"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <partnerLinks>
    <partnerLink myRole="AnalysisPartner2Role_InitProcessAnalysisPartner2PL"
      name="InitProcessAnalysisPartner2PL"
      partnerLinkType="tns:InitProcessAnalysisPartner2PLType" />
    <partnerLink myRole="AnalysisPartner2Role_InitiatorAnalysis2PL"
      name="InitiatorAnalysis2PL" partnerLinkType="tns:InitiatorAnalysis2PLType"
      partnerRole="InitiatorRole_InitiatorAnalysis2PL" />
    <partnerLink myRole="AnalysisPartner2Role_Analysis2StoragePL"
      name="Analysis2StoragePL" partnerLinkType="tns:Analysis2StoragePLType"
      partnerRole="StoragePartnerRole_Analysis2StoragePL" />
    <partnerLink name="PrivateAnaPL" partnerLinkType="tns:PrivateAnaPLType"
      partnerRole="Role_PrivateAnaPL" />
  </partnerLinks>
  <variables>
    <variable messageType="tns:InitProcessAnalysisPartner2OpMsg"
      name="InitProcessAnalysisPartner2OpMsgVar" />
    <variable name="ChoreographyIdVariable" type="choreons:stringID" />
    <variable messageType="tns:InitProcessAnalysisPartner2OpRespMsg"
      name="InitProcessAnalysisPartner2OpRespMsgVar" />
    <variable name="varRawDataAddr_Ana2" type="xsd:anyURI" />
    <variable name="varResultDataAddr_Ana2" type="xsd:anyURI" />
    <variable name="varRawData_Ana2" type="xsd:string" />
    <variable name="varResultData_Ana2" type="xsd:string" />
    <variable messageType="tns:AnalyzeOp2ReqMsg" name="AnalyzeOp2ReqMsgVar" />
    <variable messageType="ns0:getRawDataOp2ReqMsg" name="getRawDataOp2ReqMsgVar" />
    <variable messageType="tns:getRawDataOp2RespMsg" name="getRawDataOp2RespMsgVar"/>
    <variable messageType="ns1:analyzeDataRequest" name="analyzeDataRequest" />
  </variables>

```

```

<variable messageType="ns1:analyzeDataResponse" name="analyzeDataResponse" />
<variable messageType="ns0:storeResultDataOp2ReqMsg"
  name="storeResultDataOp2ReqMsgVar" />
<variable messageType="tns:storeResultDataOp2RespMsg"
  name="storeResultDataOp2RespMsgVar" />
<variable messageType="ns2:AnalyzeOp2RespMsg" name="AnalyzeOp2RespMsgVar" />
</variables>
<correlationSets>
  <correlationSet name="choreographyCorrelationSet"
    properties="tns:stringIDToken" />
</correlationSets>
<sequence>
  <receive createInstance="yes" name="InitProcess"
    operation="InitProcessAnalysisPartner2Op"
    partnerLink="InitProcessAnalysisPartner2PL"
    portType="tns:InitProcessAnalysisPartner2"
    variable="InitProcessAnalysisPartner2OpMsgVar">
    <correlations>
      <correlation initiate="yes" set="choreographyCorrelationSet" />
    </correlations>
  </receive>
  <assign>
    <copy>
      <from>
        <InitProcessAnalysisPartner2OpRespMsgElem
          xmlns="http://cdl2bpel.cecka.sap.com/SAC2">
          <IDElem>
            <ID />
          </IDElem>
        </InitProcessAnalysisPartner2OpRespMsgElem>
      </from>
      <to part="ACK" variable="InitProcessAnalysisPartner2OpRespMsgVar" />
    </copy>
    <copy>
      <from part="Start" query="descendant::choreons:ID"
        variable="InitProcessAnalysisPartner2OpMsgVar" />
      <to variable="ChoreographyIdVariable" />
    </copy>
    <copy>
      <from variable="ChoreographyIdVariable" />
      <to part="ACK" query="descendant::choreons:ID"
        variable="InitProcessAnalysisPartner2OpRespMsgVar" />
    </copy>
  </assign>
  <reply name="InitResponse" operation="InitProcessAnalysisPartner2Op"
    partnerLink="InitProcessAnalysisPartner2PL"
    portType="tns:InitProcessAnalysisPartner2"
    variable="InitProcessAnalysisPartner2OpRespMsgVar">
    <correlations>
      <correlation set="choreographyCorrelationSet" />
    </correlations>
  </reply>

```

```

<scope variableAccessSerializable="no">
  <sequence>
    <flow>
      <sequence>
        <scope name="Scope-1" variableAccessSerializable="no">
          <sequence>
            <receive operation="AnalyzeOp2Req"
              partnerLink="InitiatorAnalysis2PL"
              portType="tns:AnalysisPartner2_InitiatorAnalysis2PT"
              variable="AnalyzeOp2ReqMsgVar">
              <correlations>
                <correlation set="choreographyCorrelationSet" />
              </correlations>
            </receive>
            <assign>
              <copy>
                <from part="payload" query="/choreons:AnalyzeOp2ReqMsgElem
                  /choreons:content/choreons:content"
                  variable="AnalyzeOp2ReqMsgVar" />
                <to variable="varRawDataAddr_Ana2" />
              </copy>
            </assign>
          </sequence>
        </scope>
        <scope name="Scope-2" variableAccessSerializable="no">
          <sequence>
            <assign>
              <copy>
                <from>
                  <getRawDataOp2ReqMsgElem
                    xmlns="http://cdl2bpel.cecka.sap.com/SAC2">
                    <content>
                      <content />
                      <IDElem>
                        <ID />
                      </IDElem>
                    </content>
                  </getRawDataOp2ReqMsgElem>
                </from>
                <to part="payload" variable="getRawDataOp2ReqMsgVar" />
              </copy>
              <copy>
                <from variable="ChoreographyIdVariable" />
                <to part="payload" query="descendant::choreons:ID"
                  variable="getRawDataOp2ReqMsgVar" />
              </copy>
              <copy>
                <from variable="varRawDataAddr_Ana2" />
                <to part="payload" query="/choreons:getRawDataOp2ReqMsgElem
                  /choreons:content/choreons:content"
                  variable="getRawDataOp2ReqMsgVar" />
              </copy>
            </assign>
          </sequence>
        </scope>
      </flow>
    </sequence>
  </scope>

```

```

</assign>
<invoke inputVariable="getRawDataOp2ReqMsgVar"
  operation="getRawDataOp2Req" partnerLink="Analysis2StoragePL"
  portType="ns0:StoragePartner_Analysis2StoragePT">
  <correlations>
    <correlation set="choreographyCorrelationSet" />
  </correlations>
</invoke>
</sequence>
</scope>
<scope name="Scope-3" variableAccessSerializable="no">
  <sequence>
    <receive operation="getRawDataOp2Resp"
      partnerLink="Analysis2StoragePL"
      portType="tns:AnalysisPartner2_Analysis2StoragePT"
      variable="getRawDataOp2RespMsgVar">
      <correlations>
        <correlation set="choreographyCorrelationSet" />
      </correlations>
    </receive>
    <assign>
      <copy>
        <from part="payload" query="/choreons:getRawDataOp2RespMsgElem
          /choreons:content/choreons:content"
          variable="getRawDataOp2RespMsgVar" />
        <to variable="varRawData_Ana2" />
      </copy>
    </assign>
  </sequence>
</scope>
<sequence>
  <assign>
    <copy>
      <from>
        <analyzeDataReqElem
          xmlns="http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis">
          <content />
        </analyzeDataReqElem>
      </from>
      <to part="analyzeDataReqElem" variable="analyzeDataRequest" />
    </copy>
    <copy>
      <from variable="varRawData_Ana2" />
      <to part="analyzeDataReqElem"
        query="/ns1:analyzeDataReqElem/ns1:content"
        variable="analyzeDataRequest" />
    </copy>
  </assign>
  <invoke inputVariable="analyzeDataRequest" operation="analyzeData"
    outputVariable="analyzeDataResponse"
    partnerLink="PrivateAnaPL" portType="ns1:PrivAnalysisPT">
    <correlations>

```

```

        <correlation set="choreographyCorrelationSet" />
    </correlations>
</invoke>
<assign>
    <copy>
        <from part="analyzeDataRespElem"
            query="/ns1:analyzeDataRespElem/ns1:content"
            variable="analyzeDataResponse" />
        <to variable="varResultData_Ana2" />
    </copy>
</assign>
</sequence>
<scope name="Scope-4" variableAccessSerializable="no">
    <sequence>
        <assign>
            <copy>
                <from>
                    <storeResultDataOp2ReqMsgElem
                        xmlns="http://cdl2bpe1.cecka.sap.com/SAC2">
                        <content>
                            <content />
                            <IDElem>
                                <ID />
                            </IDElem>
                        </content>
                    </storeResultDataOp2ReqMsgElem>
                </from>
                <to part="payload" variable="storeResultDataOp2ReqMsgVar" />
            </copy>
            <copy>
                <from variable="ChoreographyIdVariable" />
                <to part="payload" query="descendant::choreons:ID"
                    variable="storeResultDataOp2ReqMsgVar" />
            </copy>
            <copy>
                <from variable="varResultData_Ana2" />
                <to part="payload" query="/choreons:storeResultDataOp2ReqMsgElem
                    /choreons:content/choreons:content"
                    variable="storeResultDataOp2ReqMsgVar" />
            </copy>
        </assign>
        <invoke inputVariable="storeResultDataOp2ReqMsgVar"
            operation="storeResultDataOp2Req" partnerLink="Analysis2StoragePL"
            portType="ns0:StoragePartner_Analysis2StoragePT">
            <correlations>
                <correlation set="choreographyCorrelationSet" />
            </correlations>
        </invoke>
    </sequence>
</scope>
<scope name="Scope-5" variableAccessSerializable="no">
    <sequence>

```

```

<receive operation="storeResultDataOp2Resp"
  partnerLink="Analysis2StoragePL"
  portType="tns:AnalysisPartner2_Analysis2StoragePT"
  variable="storeResultDataOp2RespMsgVar">
  <correlations>
    <correlation set="choreographyCorrelationSet" />
  </correlations>
</receive>
<assign>
  <copy>
    <from part="payload"
      query="/choreons:storeResultDataOp2RespMsgElem
        /choreons:content/choreons:content"
      variable="storeResultDataOp2RespMsgVar" />
    <to variable="varResultDataAddr_Ana2" />
  </copy>
</assign>
</sequence>
</scope>
<scope name="Scope-6" variableAccessSerializable="no">
  <sequence>
    <assign>
      <copy>
        <from>
          <AnalyzeOp2RespMsgElem
            xmlns="http://cdl2bpel.cecka.sap.com/SAC2">
            <content>
              <content />
              <IDElem>
                <ID />
              </IDElem>
            </content>
          </AnalyzeOp2RespMsgElem>
        </from>
        <to part="payload" variable="AnalyzeOp2RespMsgVar" />
      </copy>
      <copy>
        <from variable="ChoreographyIdVariable" />
        <to part="payload" query="descendant::choreons:ID"
          variable="AnalyzeOp2RespMsgVar" />
      </copy>
      <copy>
        <from variable="varResultData_Ana2" />
        <to part="payload" query="/choreons:AnalyzeOp2RespMsgElem
          /choreons:content/choreons:content"
          variable="AnalyzeOp2RespMsgVar" />
      </copy>
    </assign>
    <invoke inputVariable="AnalyzeOp2RespMsgVar"
      operation="AnalyzeOp2Resp" partnerLink="InitiatorAnalysis2PL"
      portType="ns2:Initiator_InitiatorAnalysis2PT">
    <correlations>

```

```

        <correlation set="choreographyCorrelationSet" />
    </correlations>
</invoke>
</sequence>
</scope>
</sequence>
</flow>
</sequence>
</scope>
</sequence>
</process>

```

StoragePartner Role

```

<?xml version="1.0" encoding="UTF-8" ?>
<process name="BP_StoragePartner" suppressJoinFailure="yes"
  targetNamespace="http://cdl2bpel.cecka.sap.com/SAC2/StoragePartner"
  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:choreons="http://cdl2bpel.cecka.sap.com/SAC2"
  xmlns:ns0="http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage"
  xmlns:ns1="http://cdl2bpel.cecka.sap.com/SAC2/Initiator"
  xmlns:ns2="http://cdl2bpel.cecka.sap.com/SAC2/AnalysisPartner1"
  xmlns:ns3="http://cdl2bpel.cecka.sap.com/SAC2/AnalysisPartner2"
  xmlns:tns="http://cdl2bpel.cecka.sap.com/SAC2/StoragePartner"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <partnerLinks>
    <partnerLink myRole="StoragePartnerRole_InitProcessStoragePartnerPL"
      name="InitProcessStoragePartnerPL"
      partnerLinkType="tns:InitProcessStoragePartnerPLType" />
    <partnerLink myRole="StoragePartnerRole_InitiatorStoragePL"
      name="InitiatorStoragePL" partnerLinkType="tns:InitiatorStoragePLType"
      partnerRole="InitiatorRole_InitiatorStoragePL" />
    <partnerLink name="PrivateStoPL" partnerLinkType="tns:PrivateStoPLType"
      partnerRole="Role_PrivateStoPL" />
    <partnerLink myRole="StoragePartnerRole_Analysis1StoragePL"
      name="Analysis1StoragePL" partnerLinkType="tns:Analysis1StoragePLType"
      partnerRole="AnalysisPartner1Role_Analysis1StoragePL" />
    <partnerLink myRole="StoragePartnerRole_Analysis2StoragePL"
      name="Analysis2StoragePL" partnerLinkType="tns:Analysis2StoragePLType"
      partnerRole="AnalysisPartner2Role_Analysis2StoragePL" />
  </partnerLinks>
  <variables>
    <variable messageType="tns:InitProcessStoragePartnerOpMsg"
      name="InitProcessStoragePartnerOpMsgVar" />
    <variable name="ChoreographyIdVariable" type="choreons:stringID" />
    <variable messageType="tns:InitProcessStoragePartnerOpRespMsg"
      name="InitProcessStoragePartnerOpRespMsgVar" />
    <variable name="varRawDataAddr_Sto" type="xsd:anyURI" />
    <variable name="varResultDataAddr_Sto" type="xsd:anyURI" />
    <variable name="varRawData_Sto" type="xsd:string" />
  </variables>

```

```

<variable name="varResultData_Sto" type="xsd:string" />
<variable messageType="tns:storeToDBReqMsg" name="storeToDBReqMsgVar" />
<variable messageType="ns0:storeDataRequest" name="storeDataRequest" />
<variable messageType="ns0:storeDataResponse" name="storeDataResponse" />
<variable messageType="ns1:storeToDBRespMsg" name="storeToDBRespMsgVar" />
<variable messageType="tns:getRawDataOp1ReqMsg" name="getRawDataOp1ReqMsgVar" />
<variable messageType="ns0:getDataRequest" name="getDataRequest" />
<variable messageType="ns0:getDataResponse" name="getDataResponse" />
<variable messageType="ns2:getRawDataOp1RespMsg" name="getRawDataOp1RespMsgVar"/>
<variable messageType="tns:storeResultDataOp1ReqMsg"
  name="storeResultDataOp1ReqMsgVar" />
<variable messageType="ns2:storeResultDataOp1RespMsg"
  name="storeResultDataOp1RespMsgVar" />
<variable messageType="tns:getRawDataOp2ReqMsg" name="getRawDataOp2ReqMsgVar" />
<variable messageType="ns3:getRawDataOp2RespMsg" name="getRawDataOp2RespMsgVar"/>
<variable messageType="tns:storeResultDataOp2ReqMsg"
  name="storeResultDataOp2ReqMsgVar" />
<variable messageType="ns3:storeResultDataOp2RespMsg"
  name="storeResultDataOp2RespMsgVar" />
</variables>
<correlationSets>
  <correlationSet name="choreographyCorrelationSet"
    properties="tns:stringIDToken" />
</correlationSets>
<sequence>
  <receive createInstance="yes" name="InitProcess"
    operation="InitProcessStoragePartnerOp"
    partnerLink="InitProcessStoragePartnerPL"
    portType="tns:InitProcessStoragePartner"
    variable="InitProcessStoragePartnerOpMsgVar">
    <correlations>
      <correlation initiate="yes" set="choreographyCorrelationSet" />
    </correlations>
  </receive>
  <assign>
    <copy>
      <from>
        <InitProcessStoragePartnerOpRespMsgElem
          xmlns="http://cdl2bpel.cecka.sap.com/SAC2">
          <IDElem>
            <ID />
          </IDElem>
        </InitProcessStoragePartnerOpRespMsgElem>
      </from>
      <to part="ACK" variable="InitProcessStoragePartnerOpRespMsgVar" />
    </copy>
    <copy>
      <from part="Start" query="descendant::choreons:ID"
        variable="InitProcessStoragePartnerOpMsgVar" />
      <to variable="ChoreographyIdVariable" />
    </copy>
    <copy>

```

```

    <from variable="ChoreographyIdVariable" />
    <to part="ACK" query="descendant::choreons:ID"
        variable="InitProcessStoragePartnerOpRespMsgVar" />
    </copy>
</assign>
<reply name="InitResponse" operation="InitProcessStoragePartnerOp"
    partnerLink="InitProcessStoragePartnerPL"
    portType="tns:InitProcessStoragePartner"
    variable="InitProcessStoragePartnerOpRespMsgVar">
    <correlations>
        <correlation set="choreographyCorrelationSet" />
    </correlations>
</reply>
<scope variableAccessSerializable="no">
    <sequence>
        <scope name="Scope-1" variableAccessSerializable="no">
            <sequence>
                <receive operation="storeToDBReq" partnerLink="InitiatorStoragePL"
                    portType="tns:StoragePartner_InitiatorStoragePT"
                    variable="storeToDBReqMsgVar">
                    <correlations>
                        <correlation set="choreographyCorrelationSet" />
                    </correlations>
                </receive>
                <assign>
                    <copy>
                        <from part="payload" query="/choreons:storeToDBReqMsgElem
                            /choreons:content/choreons:content"
                            variable="storeToDBReqMsgVar" />
                        <to variable="varRawData_Sto" />
                    </copy>
                </assign>
            </sequence>
        </scope>
        <sequence>
            <assign>
                <copy>
                    <from>
                        <storeDataReqElem
                            xmlns="http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage">
                            <content />
                        </storeDataReqElem>
                    </from>
                    <to part="storeDataReqElem" variable="storeDataRequest" />
                </copy>
                <copy>
                    <from variable="varRawData_Sto" />
                    <to part="storeDataReqElem" query="/ns0:storeDataReqElem/ns0:content"
                        variable="storeDataRequest" />
                </copy>
            </assign>
            <invoke inputVariable="storeDataRequest" operation="storeData"

```

```

    outputVariable="storeDataResponse"
    partnerLink="PrivateStoPL" portType="ns0:PrivateStoragePT">
    <correlations>
      <correlation set="choreographyCorrelationSet" />
    </correlations>
  </invoke>
<assign>
  <copy>
    <from part="storeDataRespElem"
      query="/ns0:storeDataRespElem/ns0:content"
      variable="storeDataResponse" />
    <to variable="varRawDataAddr_Sto" />
  </copy>
</assign>
</sequence>
<scope name="Scope-2" variableAccessSerializable="no">
  <sequence>
    <assign>
      <copy>
        <from>
          <storeToDBRespMsgElem xmlns="http://cdl2bpel.cecka.sap.com/SAC2">
            <content>
              <content />
              <IDElem>
                <ID />
              </IDElem>
            </content>
          </storeToDBRespMsgElem>
        </from>
        <to part="payload" variable="storeToDBRespMsgVar" />
      </copy>
      <copy>
        <from variable="ChoreographyIdVariable" />
        <to part="payload" query="descendant::choreons:ID"
          variable="storeToDBRespMsgVar" />
      </copy>
      <copy>
        <from variable="varRawDataAddr_Sto" />
        <to part="payload" query="/choreons:storeToDBRespMsgElem
          /choreons:content/choreons:content"
          variable="storeToDBRespMsgVar" />
      </copy>
    </assign>
    <invoke inputVariable="storeToDBRespMsgVar" operation="storeToDBResp"
      partnerLink="InitiatorStoragePL"
      portType="ns1:Initiator_InitiatorStoragePT">
      <correlations>
        <correlation set="choreographyCorrelationSet" />
      </correlations>
    </invoke>
  </sequence>
</scope>

```

```

<flow>
  <sequence>
    <scope name="Scope-3" variableAccessSerializable="no">
      <sequence>
        <receive operation="getRawDataOp1Req"
          partnerLink="Analysis1StoragePL"
          portType="tns:StoragePartner_Analysis1StoragePT"
          variable="getRawDataOp1ReqMsgVar">
          <correlations>
            <correlation set="choreographyCorrelationSet" />
          </correlations>
        </receive>
        <assign>
          <copy>
            <from part="payload" query="/choreons:getRawDataOp1ReqMsgElem
              /choreons:content/choreons:content"
              variable="getRawDataOp1ReqMsgVar" />
            <to variable="varRawDataAddr_Sto" />
          </copy>
        </assign>
      </sequence>
    </scope>
    <sequence>
      <assign>
        <copy>
          <from>
            <getDataReqElem
              xmlns="http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage">
              <content />
            </getDataReqElem>
          </from>
          <to part="getDataReqElem" variable="getDataRequest" />
        </copy>
        <copy>
          <from variable="varRawDataAddr_Sto" />
          <to part="getDataReqElem" query="/ns0:getDataReqElem/ns0:content"
            variable="getDataRequest" />
        </copy>
      </assign>
      <invoke inputVariable="getDataRequest" operation="getData"
        outputVariable="getDataResponse"
        partnerLink="PrivateStoPL" portType="ns0:PrivateStoragePT">
      <correlations>
        <correlation set="choreographyCorrelationSet" />
      </correlations>
    </invoke>
    <assign>
      <copy>
        <from part="getDataRespElem"
          query="/ns0:getDataRespElem/ns0:content"
          variable="getDataResponse" />
        <to variable="varRawData_Sto" />
      </copy>
    </assign>
  </sequence>
</flow>

```

```

    </copy>
  </assign>
</sequence>
<scope name="Scope-4" variableAccessSerializable="no">
  <sequence>
    <assign>
      <copy>
        <from>
          <getRawDataOp1RespMsgElem
            xmlns="http://cdl2bpel.cecka.sap.com/SAC2">
            <content>
              <content />
              <IDElem>
                <ID />
              </IDElem>
            </content>
          </getRawDataOp1RespMsgElem>
        </from>
        <to part="payload" variable="getRawDataOp1RespMsgVar" />
      </copy>
      <copy>
        <from variable="ChoreographyIdVariable" />
        <to part="payload" query="descendant::choreons:ID"
          variable="getRawDataOp1RespMsgVar" />
      </copy>
      <copy>
        <from variable="varRawData_Sto" />
        <to part="payload" query="/choreons:getRawDataOp1RespMsgElem
          /choreons:content/choreons:content"
          variable="getRawDataOp1RespMsgVar" />
      </copy>
    </assign>
    <invoke inputVariable="getRawDataOp1RespMsgVar"
      operation="getRawDataOp1Resp" partnerLink="Analysis1StoragePL"
      portType="ns2:AnalysisPartner1_Analysis1StoragePT">
      <correlations>
        <correlation set="choreographyCorrelationSet" />
      </correlations>
    </invoke>
  </sequence>
</scope>
<scope name="Scope-5" variableAccessSerializable="no">
  <sequence>
    <receive operation="storeResultDataOp1Req"
      partnerLink="Analysis1StoragePL"
      portType="tns:StoragePartner_Analysis1StoragePT"
      variable="storeResultDataOp1ReqMsgVar">
      <correlations>
        <correlation set="choreographyCorrelationSet" />
      </correlations>
    </receive>
    <assign>

```

```

    <copy>
      <from part="payload"
        query="/choreons:storeResultDataOp1ReqMsgElem/choreons:content
          /choreons:content"
        variable="storeResultDataOp1ReqMsgVar" />
      <to variable="varResultData_Sto" />
    </copy>
  </assign>
</sequence>
</scope>
<sequence>
  <assign>
    <copy>
      <from>
        <storeDataReqElem
          xmlns="http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage">
          <content />
        </storeDataReqElem>
      </from>
      <to part="storeDataReqElem" variable="storeDataRequest" />
    </copy>
    <copy>
      <from variable="varResultData_Sto" />
      <to part="storeDataReqElem"
        query="/ns0:storeDataReqElem/ns0:content"
        variable="storeDataRequest" />
    </copy>
  </assign>
  <invoke inputVariable="storeDataRequest" operation="storeData"
    outputVariable="storeDataResponse" partnerLink="PrivateStoPL"
    portType="ns0:PrivateStoragePT">
    <correlations>
      <correlation set="choreographyCorrelationSet" />
    </correlations>
  </invoke>
  <assign>
    <copy>
      <from part="storeDataRespElem"
        query="/ns0:storeDataRespElem/ns0:content"
        variable="storeDataResponse" />
      <to variable="varResultDataAddr_Sto" />
    </copy>
  </assign>
</sequence>
<scope name="Scope-6" variableAccessSerializable="no">
  <sequence>
    <assign>
      <copy>
        <from>
          <storeResultDataOp1RespMsgElem
            xmlns="http://cdl2bpel.cecka.sap.com/SAC2">
            <content>

```

```

        <content />
        <IDElem>
            <ID />
        </IDElem>
    </content>
</storeResultDataOp1RespMsgElem>
</from>
<to part="payload" variable="storeResultDataOp1RespMsgVar" />
</copy>
<copy>
    <from variable="ChoreographyIdVariable" />
    <to part="payload" query="descendant::choreons:ID"
        variable="storeResultDataOp1RespMsgVar" />
</copy>
<copy>
    <from variable="varResultDataAddr_Sto" />
    <to part="payload" query="/choreons:storeResultDataOp1RespMsgElem
        /choreons:content/choreons:content"
        variable="storeResultDataOp1RespMsgVar" />
</copy>
</assign>
<invoke inputVariable="storeResultDataOp1RespMsgVar"
    operation="storeResultDataOp1Resp" partnerLink="Analysis1StoragePL"
    portType="ns2:AnalysisPartner1_Analysis1StoragePT">
    <correlations>
        <correlation set="choreographyCorrelationSet" />
    </correlations>
</invoke>
</sequence>
</scope>
</sequence>
<sequence>
    <scope name="Scope-7" variableAccessSerializable="no">
        <sequence>
            <receive operation="getRawDataOp2Req"
                partnerLink="Analysis2StoragePL"
                portType="tns:StoragePartner_Analysis2StoragePT"
                variable="getRawDataOp2ReqMsgVar">
                <correlations>
                    <correlation set="choreographyCorrelationSet" />
                </correlations>
            </receive>
            <assign>
                <copy>
                    <from part="payload" query="/choreons:getRawDataOp2ReqMsgElem
                        /choreons:content/choreons:content"
                        variable="getRawDataOp2ReqMsgVar" />
                    <to variable="varRawDataAddr_Sto" />
                </copy>
            </assign>
        </sequence>
    </scope>

```

```

<sequence>
  <assign>
    <copy>
      <from>
        <getDataReqElem
          xmlns="http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage">
          <content />
        </getDataReqElem>
      </from>
      <to part="getDataReqElem" variable="getDataRequest" />
    </copy>
    <copy>
      <from variable="varRawDataAddr_Sto" />
      <to part="getDataReqElem" query="/ns0:getDataReqElem/ns0:content"
        variable="getDataRequest" />
    </copy>
  </assign>
  <invoke inputVariable="getDataRequest" operation="getData"
    outputVariable="getDataResponse" partnerLink="PrivateStoPL"
    portType="ns0:PrivateStoragePT">
    <correlations>
      <correlation set="choreographyCorrelationSet" />
    </correlations>
  </invoke>
  <assign>
    <copy>
      <from part="getDataRespElem"
        query="/ns0:getDataRespElem/ns0:content"
        variable="getDataResponse" />
      <to variable="varRawData_Sto" />
    </copy>
  </assign>
</sequence>
<scope name="Scope-8" variableAccessSerializable="no">
  <sequence>
    <assign>
      <copy>
        <from>
          <getRawDataOp2RespMsgElem
            xmlns="http://cdl2bpel.cecka.sap.com/SAC2">
            <content>
              <content />
              <IDElem>
                <ID />
              </IDElem>
            </content>
          </getRawDataOp2RespMsgElem>
        </from>
        <to part="payload" variable="getRawDataOp2RespMsgVar" />
      </copy>
      <copy>
        <from variable="ChoreographyIdVariable" />

```

```

        <to part="payload" query="descendant::choreons:ID"
            variable="getRawDataOp2RespMsgVar" />
    </copy>
</copy>
<copy>
    <from variable="varRawData_Sto" />
    <to part="payload" query="/choreons:getRawDataOp2RespMsgElem
        /choreons:content/choreons:content"
        variable="getRawDataOp2RespMsgVar" />
    </copy>
</assign>
<invoke inputVariable="getRawDataOp2RespMsgVar"
    operation="getRawDataOp2Resp" partnerLink="Analysis2StoragePL"
    portType="ns3:AnalysisPartner2_Analysis2StoragePT">
    <correlations>
        <correlation set="choreographyCorrelationSet" />
    </correlations>
</invoke>
</sequence>
</scope>
<scope name="Scope-9" variableAccessSerializable="no">
    <sequence>
        <receive operation="storeResultDataOp2Req"
            partnerLink="Analysis2StoragePL"
            portType="tns:StoragePartner_Analysis2StoragePT"
            variable="storeResultDataOp2ReqMsgVar">
            <correlations>
                <correlation set="choreographyCorrelationSet" />
            </correlations>
        </receive>
        <assign>
            <copy>
                <from part="payload"
                    query="/choreons:storeResultDataOp2ReqMsgElem/choreons:content
                        /choreons:content"
                    variable="storeResultDataOp2ReqMsgVar" />
                <to variable="varResultData_Sto" />
            </copy>
        </assign>
    </sequence>
</scope>
<sequence>
    <assign>
        <copy>
            <from>
                <storeDataReqElem
                    xmlns="http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage">
                    <content />
                </storeDataReqElem>
            </from>
            <to part="storeDataReqElem" variable="storeDataRequest" />
        </copy>
    </assign>
    <copy>

```

```

    <from variable="varResultData_Sto" />
    <to part="storeDataReqElem"
        query="/ns0:storeDataReqElem/ns0:content"
        variable="storeDataRequest" />
    </copy>
</assign>
<invoke inputVariable="storeDataRequest" operation="storeData"
    outputVariable="storeDataResponse" partnerLink="PrivateStoPL"
    portType="ns0:PrivateStoragePT">
    <correlations>
        <correlation set="choreographyCorrelationSet" />
    </correlations>
</invoke>
<assign>
    <copy>
        <from part="storeDataRespElem"
            query="/ns0:storeDataRespElem/ns0:content"
            variable="storeDataResponse" />
        <to variable="varResultDataAddr_Sto" />
    </copy>
</assign>
</sequence>
<scope name="Scope-10" variableAccessSerializable="no">
    <sequence>
        <assign>
            <copy>
                <from>
                    <storeResultDataOp2RespMsgElem
                        xmlns="http://cdl2bpel.cecka.sap.com/SAC2">
                        <content>
                            <content />
                            <IDElem>
                                <ID />
                            </IDElem>
                        </content>
                    </storeResultDataOp2RespMsgElem>
                </from>
                <to part="payload" variable="storeResultDataOp2RespMsgVar" />
            </copy>
            <copy>
                <from variable="ChoreographyIdVariable" />
                <to part="payload" query="descendant::choreons:ID"
                    variable="storeResultDataOp2RespMsgVar" />
            </copy>
            <copy>
                <from variable="varResultDataAddr_Sto" />
                <to part="payload" query="/choreons:storeResultDataOp2RespMsgElem
                    /choreons:content/choreons:content"
                    variable="storeResultDataOp2RespMsgVar" />
            </copy>
        </assign>
    </sequence>
</scope>
<invoke inputVariable="storeResultDataOp2RespMsgVar"

```

```
        operation="storeResultDataOp2Resp" partnerLink="Analysis2StoragePL"
        portType="ns3:AnalysisPartner2_Analysis2StoragePT">
        <correlations>
            <correlation set="choreographyCorrelationSet" />
        </correlations>
    </invoke>
</sequence>
</scope>
</sequence>
</flow>
</sequence>
</scope>
</sequence>
</process>
```


Appendix B

Knowledge Base Contents

There are eight patterns in the Knowledge Base of the prototype. The first three are used by Example 1 from Appendix A.1, while all patterns except for the third one are used by Example 2 in Appendix A.2.

The patterns are divided into four parts for the presentation here: a CdlPart with the WS-CDL elements which are replaced by this pattern in a choreography; a BpPart which is inserted into the executable process (WSBPEL); a ViewPart for the public view (WSDL), which can also be the inclusion of an external WSDL file; and a PddPart for the Process Deployment Descriptor, where the shown examples contain the WS-Addressing information for an external Web service, if any.

In general, there could be multiple Bp-, View-, and PddParts - one set of these for each relevant role - per CdlPart. In the presented examples, the CdlPart is always a cdl:silentAction, which is intended to be executed at one single role. Therefore, there is always only set of replacements in the patterns. The language choices in the examples have no impact on the concept of the Knowledge Base, only on its content.

B.1 Pattern 1 : StoragePartner - Get Raw Data

```
<CdlPart>
  <silentAction roleType="StoragePartner">
    <description type="documentation">getRawDataFromDB varRawDataAddr_Sto
      varRawData_Sto</description>
  </silentAction>
</CdlPart>
<BpPart roleType="StoragePartner">
```

```

<variable
  messageType="{http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage}getDataRequest"
  name="getDataRequest" />
<variable
  messageType="{http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage}getDataResponse"
  name="getDataResponse" />
<sequence>
  <assign>
    <copy>
      <from>
        <getDataReqElem xmlns="http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage">
          <content />
        </getDataReqElem>
      </from>
      <to part="getDataReqElem" variable="getDataRequest" />
    </copy>
    <copy>
      <from variable="varRawDataAddr_Sto" />
      <to part="getDataReqElem" variable="getDataRequest"
        query="/ns0:getDataReqElem/ns0:content" />
    </copy>
  </assign>
  <invoke inputVariable="getDataRequest" name="InvokeGetData" operation="getData"
    outputVariable="getDataResponse" partnerLink="PrivateStoPL"
    portType="{http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage}PrivateStoragePT"
  />
  <assign>
    <copy>
      <from part="getDataRespElem" variable="getDataResponse"
        query="/ns0:getDataRespElem/ns0:content" />
      <to variable="varRawData_Sto" />
    </copy>
  </assign>
</sequence>
</BpPart>
<ViewPart roleType="StoragePartner" viewId="WsdlIdSubPrivSto"
  namespace="http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage"></ViewPart>
<PddPart roleType="StoragePartner">
  <wsa:EndpointReference xmlns:s="http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
    <wsa:Address>http://localhost:8080/axis/services/PrivateStorageServicePort
    </wsa:Address>
    <wsa:ServiceName PortName="PrivateStorageServicePort">s:PrivateStorageService
    </wsa:ServiceName>
  </wsa:EndpointReference>
</PddPart>

```

B.2 Pattern 2 : StoragePartner - Store Result Data

```

<Cd1Part>

```

```

<silentAction roleType="StoragePartner">
  <description type="documentation">storeDataToDB varResultData_Sto
    varResultDataAddr_Sto</description>
</silentAction>
</CdlPart>
<BpPart roleType="StoragePartner">
  <variable messageType="{http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage}
    storeDataRequest" name="storeDataRequest" />
  <variable messageType="{http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage}
    storeDataResponse" name="storeDataResponse" />
  <sequence>
    <assign>
      <copy>
        <from>
          <storeDataReqElem
            xmlns="http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage">
              <content />
            </storeDataReqElem>
          </from>
          <to part="storeDataReqElem" variable="storeDataRequest" />
        </copy>
        <copy>
          <from variable="varResultData_Sto" />
          <to part="storeDataReqElem" variable="storeDataRequest"
            query="/ns0:storeDataReqElem/ns0:content" />
        </copy>
      </assign>
      <invoke inputVariable="storeDataRequest" name="InvokeStoreData"
        operation="storeData" outputVariable="storeDataResponse"
        partnerLink="PrivateStoPL"
        portType="{http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage}PrivateStoragePT"
        />
      <assign>
        <copy>
          <from part="storeDataRespElem" variable="storeDataResponse"
            query="/ns0:storeDataRespElem/ns0:content" />
          <to variable="varResultDataAddr_Sto" />
        </copy>
      </assign>
    </sequence>
  </BpPart>
  <ViewPart roleType="StoragePartner" viewId="WsdllIdSubPrivSto"
    namespace="http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage"></ViewPart>
  <PddPart roleType="StoragePartner">
    <wsa:EndpointReference xmlns:s="http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage"
      xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
      <wsa:Address>http://localhost:8080/axis/services/PrivateStorageServicePort
      </wsa:Address>
      <wsa:ServiceName PortName="PrivateStorageServicePort">s:PrivateStorageService
      </wsa:ServiceName>
    </wsa:EndpointReference>
  </PddPart>

```

B.3 Pattern 3 : AnalysisPartner - Analyze Data

```

<CdlPart>
  <silentAction roleType="AnalysisPartner">
    <description type="documentation">analyzeData varRawData_Ana varResultData_Ana
    </description>
  </silentAction>
</CdlPart>
<BpPart roleType="AnalysisPartner">
  <variable messageType="{http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis}
  analyzeDataRequest" name="analyzeDataRequest" />
  <variable messageType="{http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis}
  analyzeDataResponse" name="analyzeDataResponse" />
  <sequence>
    <assign>
      <copy>
        <from>
          <analyzeDataReqElem
            xmlns="http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis">
              <content />
            </analyzeDataReqElem>
          </from>
          <to part="analyzeDataReqElem" variable="analyzeDataRequest" />
        </copy>
        <copy>
          <from variable="varRawData_Ana" />
          <to part="analyzeDataReqElem" variable="analyzeDataRequest"
            query="/ns1:analyzeDataReqElem/ns1:content" />
        </copy>
      </assign>
      <invoke inputVariable="analyzeDataRequest" name="InvokeAnalyze"
        operation="analyzeData" outputVariable="analyzeDataResponse"
        partnerLink="PrivateAnaPL"
        portType="{http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis}PrivAnalysisPT"
        />
      <assign>
        <copy>
          <from part="analyzeDataRespElem" variable="analyzeDataResponse"
            query="/ns1:analyzeDataRespElem/ns1:content" />
          <to variable="varResultData_Ana" />
        </copy>
      </assign>
    </sequence>
  </BpPart>
  <ViewPart roleType="AnalysisPartner" viewId="WsdllIdSubPrivAna"
    namespace="http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis"></ViewPart>
  <PddPart roleType="AnalysisPartner">
    <wsa:EndpointReference

```

```

xmlns:s="http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
<wsa:Address>http://localhost:8080/axis/services/PrivAnalysisServicePort
</wsa:Address>
<wsa:ServiceName PortName="PrivAnalysisServicePort">s:PrivAnalysisService
</wsa:ServiceName>
</wsa:EndpointReference>
</PddPart>

```

B.4 Pattern 4 : Initiator - Combine Results

```

<CdlPart>
  <silentAction roleType="Initiator">
    <description type="documentation">combineResults varResultData1_Init
      varResultData2_Init varOverallResultData_Init</description>
  </silentAction>
</CdlPart>
<BpPart roleType="Initiator">
  <assign>
    <copy>
      <from expression="concat('Result1: ',
        bpws:getVariableData('varResultData1_Init'), ' -- ', 'Result2: ',
        bpws:getVariableData('varResultData2_Init'))" />
      <to variable="varOverallResultData_Init" />
    </copy>
  </assign>
</BpPart>
<ViewPart roleType="Initiator"></ViewPart>
<PddPart roleType="Initiator" />

```

B.5 Pattern 5 : Initiator - Identify Need

```

<CdlPart>
  <silentAction roleType="Initiator">
    <description type="documentation">IdentifyNeed</description>
  </silentAction>
</CdlPart>
<BpPart roleType="Initiator">
  <empty />
</BpPart>
<ViewPart roleType="Initiator"></ViewPart>
<PddPart roleType="Initiator" />

```

B.6 Pattern 6 : StoragePartner - Store Raw Data

```

<CdlPart>
  <silentAction roleType="StoragePartner">
    <description type="documentation">storeDataToDB varRawData_Sto varRawDataAddr_Sto
    </description>
  </silentAction>
</CdlPart>
<BpPart roleType="StoragePartner">
  <variable messageType="{http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage}
    storeDataRequest" name="storeDataRequest" />
  <variable messageType="{http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage}
    storeDataResponse" name="storeDataResponse" />
  <sequence>
    <assign>
      <copy>
        <from>
          <storeDataReqElem
            xmlns="http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage">
              <content />
            </storeDataReqElem>
          </from>
          <to part="storeDataReqElem" variable="storeDataRequest" />
        </copy>
        <copy>
          <from variable="varRawData_Sto" />
          <to part="storeDataReqElem" variable="storeDataRequest"
            query="/ns0:storeDataReqElem/ns0:content" />
        </copy>
      </assign>
    <invoke inputVariable="storeDataRequest" name="InvokeStoreData"
      operation="storeData" outputVariable="storeDataResponse"
      partnerLink="PrivateStoPL"
      portType="{http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage}PrivateStoragePT"
    />
    <assign>
      <copy>
        <from part="storeDataRespElem" variable="storeDataResponse"
          query="/ns0:storeDataRespElem/ns0:content" />
        <to variable="varRawDataAddr_Sto" />
      </copy>
    </assign>
  </sequence>
</BpPart>
<ViewPart roleType="StoragePartner" viewId="Wsd1IdSubPrivSto"
  namespace="http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage"></ViewPart>
<PddPart roleType="StoragePartner">
  <wsa:EndpointReference xmlns:s="http://trustcom.cecka.sap.com/cdl2bpel/PrivStorage"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
    <wsa:Address>http://localhost:8080/axis/services/PrivateStorageServicePort
    </wsa:Address>
    <wsa:ServiceName PortName="PrivateStorageServicePort">s:PrivateStorageService
  </wsa:EndpointReference>

```

```

    </wsa:ServiceName>
  </wsa:EndpointReference>
</PddPart>

```

B.7 Pattern 7 : AnalysisPartner1 - Analyze Data

```

<CdlPart>
  <silentAction roleType="AnalysisPartner1">
    <description type="documentation">analyzeData varRawData_Ana1 varResultData_Ana1
    </description>
  </silentAction>
</CdlPart>
<BpPart roleType="AnalysisPartner1">
  <variable messageType="{http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis}"
    analyzeDataRequest" name="analyzeDataRequest" />
  <variable messageType="{http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis}"
    analyzeDataResponse" name="analyzeDataResponse" />
  <sequence>
    <assign>
      <copy>
        <from>
          <analyzeDataReqElem
            xmlns="{http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis}"
            <content />
          </analyzeDataReqElem>
        </from>
        <to part="analyzeDataReqElem" variable="analyzeDataRequest" />
      </copy>
      <copy>
        <from variable="varRawData_Ana1" />
        <to part="analyzeDataReqElem" variable="analyzeDataRequest"
          query="/ns1:analyzeDataReqElem/ns1:content" />
        </copy>
      </assign>
      <invoke inputVariable="analyzeDataRequest" name="InvokeAnalyze"
        operation="analyzeData" outputVariable="analyzeDataResponse"
        partnerLink="PrivateAnaPL"
        portType="{http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis}PrivAnalysisPT"
        />
      <assign>
        <copy>
          <from part="analyzeDataRespElem" variable="analyzeDataResponse"
            query="/ns1:analyzeDataRespElem/ns1:content" />
          <to variable="varResultData_Ana1" />
        </copy>
      </assign>
    </sequence>
  </BpPart>
  <ViewPart roleType="AnalysisPartner1" viewId="WsdIdSubPrivAna"
    namespace="http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis"></ViewPart>

```

```

<PddPart roleType="AnalysisPartner1">
  <wsa:EndpointReference
    xmlns:s="http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
    <wsa:Address>http://localhost:8080/axis/services/PrivAnalysisServicePort
    </wsa:Address>
    <wsa:ServiceName PortName="PrivAnalysisServicePort">s:PrivAnalysisService
    </wsa:ServiceName>
  </wsa:EndpointReference>
</PddPart>

```

B.8 Pattern 8 : AnalysisPartner2 - Analyze Data

```

<CdlPart>
  <silentAction roleType="AnalysisPartner2">
    <description type="documentation">analyzeData varRawData_Ana2 varResultData_Ana2
    </description>
  </silentAction>
</CdlPart>
<BpPart roleType="AnalysisPartner2">
  <variable messageType="{http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis}
    analyzeDataRequest" name="analyzeDataRequest" />
  <variable messageType="{http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis}
    analyzeDataResponse" name="analyzeDataResponse" />
  <sequence>
    <assign>
      <copy>
        <from>
          <analyzeDataReqElem
            xmlns="http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis">
            <content />
          </analyzeDataReqElem>
        </from>
        <to part="analyzeDataReqElem" variable="analyzeDataRequest" />
      </copy>
      <copy>
        <from variable="varRawData_Ana2" />
        <to part="analyzeDataReqElem" variable="analyzeDataRequest"
          query="/ns1:analyzeDataReqElem/ns1:content" />
      </copy>
    </assign>
    <invoke inputVariable="analyzeDataRequest" name="InvokeAnalyze"
      operation="analyzeData" outputVariable="analyzeDataResponse"
      partnerLink="PrivateAnaPL"
      portType="{http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis}PrivAnalysisPT"
    />
    <assign>
      <copy>
        <from part="analyzeDataRespElem" variable="analyzeDataResponse"
          query="/ns1:analyzeDataRespElem/ns1:content" />

```

```
        <to variable="varResultData_Ana2" />
    </copy>
</assign>
</sequence>
</BpPart>
<ViewPart roleType="AnalysisPartner2" viewId="WsdIdSubPrivAna"
  namespace="http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis"></ViewPart>
<PddPart roleType="AnalysisPartner2">
  <wsa:EndpointReference
    xmlns:s="http://trustcom.cecka.sap.com/cdl2bpel/PrivAnalysis"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
    <wsa:Address>http://localhost:8080/axis/services/PrivAnalysisServicePort
    </wsa:Address>
    <wsa:ServiceName PortName="PrivAnalysisServicePort">s:PrivAnalysisService
    </wsa:ServiceName>
  </wsa:EndpointReference>
</PddPart>
```


Bibliography

- [1] G. Alonso, F. Casati, and et al. *Web Services - Concepts, Architectures and Applications*. Springer, 2004. 35
- [2] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana. *Business Process Execution Language for Web Services*. 2nd public draft release, Version 1.1, 5 May 2003. 13, 30, 82
- [3] Alistair Barros, Marlon Dumas, and Phillipa Oaks. A Critical Overview of the Web Services Choreography Description Languages (WS-CDL). BPTrends Newsletter, Vol. 3, March 2005. 27
- [4] Alistair Barros, Marlon Dumas, and Arthur H.M. ter Hofstede. Service Interaction Patterns: Towards a Reference Framework for Service-Based Business Process Interconnection. <http://sky.fit.qut.edu.au/dumas/ServiceInteractionPatterns.pdf>, April 2005. 27, 28, 29
- [5] Bernhard Borges, Kerrie Holley, and Ali Arsanjani. Delving into Service-Oriented Architecture. <http://www.developer.com/java/ent/article.php/3409221>, 2004. 21
- [6] Mario Bravetti, Claudio Guidi, Roberto Lucchi, and Gianluigi Zavattaro. Supporting e-commerce systems formalization with choreography languages. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 831–835, New York, NY, USA, 2005. ACM Press. 37
- [7] René Bultje and Jacoliene van Wijk. Taxonomy of Virtual Organisations, based on definitions, characteristics and typology. VoNet: The Netwslter @ <http://www.virtual-organization.net/>, 1998. 9
- [8] L.F. Cabrera, G. Copeland, J. Johnson, and D. Langworthy. Coordinating Web Services Activities with WS-Coordination, WS-AtomicTransaction, and WS-BusinessActivity, 2004. 35

- [9] Adrian Conlin, Philip English, Hugo Hiden, Julian Morris, Rob Smith, and Allen Wright. A Computer Architecture to Support the Operation of Virtual Organisations for the Chemical Development Lifecycle. In *Proceedings of the 15th European Symposium on Computer Aided Process Engineering (ESCAPE)*, 2005. 9
- [10] Remco Dijkman and Marlon Dumas. Service-Oriented Design: A Multi-Viewpoint Approach. *Intl. J. Cooperative Information Systems*, 13(4):337–368, 2004. 36
- [11] Marlon Dumas and Arthur H. M. ter Hofstede. UML Activity Diagrams as a Workflow Specification Language. In *UML '01: Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools*, pages 76–90, London, UK, 2001. Springer-Verlag. 26
- [12] Steve Graham et al. *Web Services Addressing*. BEA, IBM, Microsoft, SAP, Sun Microsystems, Inc., 10 August 2004. 32
- [13] Roberto Gorrieri, Claudio Guidi, and Roberto Lucchi. Reasoning about interaction patterns in Choreography. In *Proceedings of the 2nd International Workshop on Web Services and Formal Methods (WS-FM '05)*, 2005. 27, 28
- [14] Jochen Haller, Philip Robinson, and Yucel Karabulut. Security Controls in Collaborative Business Processes. In *6th IFIP Working Conference on VIRTUAL ENTERPRISES (PROVE'05)*, 2005. 88
- [15] Mike Havey. *Essential Business Process Modeling*. Copyright 2005 O'Reilly Media, Inc, 2005. 22, 36
- [16] Peter Huber, Kurt Jensen, and Robert M. Shapiro. Hierarchies in coloured Petri nets. In *Proceedings of the 10th International Conference on Applications and Theory of Petri Nets*, pages 313–341, London, UK, 1991. Springer-Verlag. 23
- [17] Romin Irani. Collaborative Electronic Business is here to stay: An Introduction to ebXML. <http://www.webservicesarchitect.com/content/articles/irani02.asp>, 2001. 35
- [18] Matthias Kloppmann, Dieter Koenig, Frank Leymann, Gerhard Pfau, Alan Rickayzen, Claus von Riegen, Patrick Schmidt, and Ivana Trickovic. WS-BPEL Extension for Sub-Processes: BPEL-SPE. White Paper, September 2005. 32, 42
- [19] Frank Leymann. Web Services Flow Language (WSFL) 1.1, May 2001. 24
- [20] Benoit Marchal. An Introduction to the ebXML CPP. <http://www.developer.com/xml/article.php/2247851>, 2003. 35

- [21] Jan Mendling and Michael Hafner. From Inter-Organizational Workflows to Process Execution: Generating BPEL from WS-CDL. In *Proceedings of OTM 2005 Workshops. Lecture Notes in Computer Science 3762*, pages 506–515. Springer Verlag, October 2005. 37, 38
- [22] Jan Mendling, Markus Nüttgens, and Gustaf Neumann. A Comparison of XML Interchange Formats for Business Process Modelling, 2004. 13
- [23] Jan Mendling and Ingo Weber. Email discussion, August 2005. 38
- [24] Robin Milner. Functions as Processes. In *Proceedings of the seventeenth international colloquium on Automata, languages and programming*, pages 167–180, New York, NY, USA, 1990. Springer-Verlag New York, Inc. 23
- [25] Robin Milner. The Polyadic π -Calculus. In *CONCUR '92: Proceedings of the Third International Conference on Concurrency Theory*, page 1, London, UK, 1992. Springer-Verlag. 23
- [26] Frédéric Montagut and Refik Molva. Enabling Pervasive Execution of Workflows. In *CollaborateCom 2005, 1st IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*,, December 2005. 29
- [27] Duane Nickull, Jean-Jacques Dubray, Colleen Evans, Pim van der Eijk, Vivek Chopra, David A Chappell, Betty Harvey, Marcel Noordzij, Jan Vegtand, Tim McGrath, and Bruce Peat. *Professional ebXML Foundations*. Wrox Press, 2001. 35
- [28] OASIS. *UDDI Version 3.0.2*. OASIS Standard 19 October 2004. <http://uddi.org/pubs/uddi-v3.0.2-20041019.htm>. 20
- [29] OASIS Web Services Reliable Messaging TC. *WS-Reliability 1.1*. OASIS Standard, 15 November 2004. 67
- [30] OASIS Web Services Security TC. *Web Services Security v1.0*, March 2004. (WS-Security 2004), OASIS Standard 200401. 67
- [31] OASIS WSBPEL Technical Committee. *Web Services Business Process Execution Language Version 2.0*, 21 September 2005. Committee Draft, work in progress. 32
- [32] Object Management Group, Inc. *MOF 2.0/XMI Mapping Specification, v2.1*, September 2005. 70
- [33] C. Peltz. Web Services Orchestration and Choreography. *Computer*, 36(10):46–52, 2003. 11, 35

- [34] Panayiotis Periorellis, Christopher J. W. Townson, and Philip English. Structural Concepts for Trust, Contract and Security Management for a Virtual Chemical Engineering. In *Proceedings of the 2nd Annual Conference on Privacy, Security and Trust*, October 2004. 9
- [35] Robert-Christoph Riemann. *Modelling of Concurrent Systems: Structural and Semantical Methods in the High Level Petri Net Calculus*. PhD thesis, Univ. Paris XI, 1999. Herbert Utz Verlag, ISBN 3-89675-629-X. 23
- [36] Philip Robinson, Yuecel Karabulut, and Jochen Haller. Dynamic Virtual Organization Management for Service Oriented Enterprise Applications. In *to appear: The First International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2005)*, 2005. 5, 10
- [37] Shazia W. Sadiq, Maria E. Orlowska, Wasim Sadiq, and Karsten A. Schulz. Facilitating Business Process Management with Harmonized Messaging. In *ICEIS (1)*, pages 30–36, 2004. 35
- [38] Karsten A. Schulz and Maria E. Orlowska. Architectural Issues for Cross-Organisational B2B Interactions, 2001. 11, 35
- [39] Karsten A. Schulz and Maria E. Orlowska. Towards a Cross-Organizational Workflow Model. In *PRO-VE '02: Proceedings of the IFIP TC5/WG5.5 Third Working Conference on Infrastructures for Virtual Enterprises*, page 652. Kluwer, B.V., 2002. 11, 16, 35
- [40] Minxin Shen and Duen-Ren Liu. Coordinating Interorganizational Workflows Based on Process-Views. *Lecture Notes in Computer Science*, 2113:274–283, 2001. 36
- [41] Troy J. Strader, Fu-Ren Lin, and Michael J. Shaw. Information Infrastructure for Electronic Virtual Organization Management. *Decis. Support Syst.*, 23(1):75–94, 1998. 9
- [42] Satish Thatte. XLANG: Web Services for Business Process Design. http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm, 2001. Copyright 2001 Microsoft Corporation. 24
- [43] W. M. P. van der Aalst. Loosely Coupled Interorganizational Workflows: Modeling and Analyzing Workflows Crossing Organizational Boundaries. *Information and Management*, 37:67–75, 2000. 35
- [44] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow Patterns. *Distrib. Parallel Databases*, 14(1):5–51, 2003. 24

- [45] W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998. 23
- [46] W.M.P. van der Aalst. Pi calculus versus Petri nets: Let us eat “humble pie” rather than further inflate the “Pi hype”, 2003. 22
- [47] W.M.P. van der Aalst and M. Weske. *The P2P Approach to Interorganizational Workflows*, pages 140–156. Springer, 2001. 35
- [48] W3C. *SOAP Version 1.2*. W3C Recommendation 24 June 2003. 20
- [49] W3C. *Web Service Choreography Interface (WSCI) 1.0*. W3C Note 8 August 2002. 11, 35
- [50] W3C. *Web Services Architecture*. W3C Working Group Note 11 February 2004, see <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/>. 11, 21
- [51] W3C. *Web Services Description Language (WSDL) 1.1*. W3C Note 15 March 2001. 13, 20
- [52] W3C. *XML Path Language (XPath) Version 1.0*. W3C Recommendation 16 November 1999. 46
- [53] W3C. *XSL Transformations (XSLT) Version 1.0*, 1999. W3C Recommendation 16 November 1999. 46
- [54] W3C. *Web Services Choreography Description Language*, 2005. W3C Latest Working Draft from October 8th, 2005, work in progress. 11, 13, 24, 53
- [55] WfMC. *Interface 1: Process Definition Interchange Process Model*. Document number wfmc-tc-1016-p version 1.1 final, Workflow Management Coalition, 1999. 11, 35
- [56] WfMC. *Workflow Process Definition Interface - XML Process Definition Language*. v1.0 Final Draft. Document number wfmc-tc-1025, Workflow Management Coalition, October 2002. 11, 35

Index

π -calculus, [23](#)

Abstract BPEL, [30](#)

Business Process, [22](#)

Business Process Management, [22](#)

Business Process Modeling, [22](#)

Choreography, [10](#)

Dead-path elimination, [24](#)

Executable BPEL, [30](#)

Information Gap, [12](#)

Knowledge Base, [41](#)

Language Differences, [14](#)

Orchestration, [11](#)

Petri-Nets, [23](#)

Service-oriented architecture (SOA), [20](#)

TrustCoM, [9](#)

UML Activity Diagram, [32](#)

Virtual Organization, [9](#)

Web Services Architecture (WSA), [21](#)

Web Services Stack, [21](#)

Workflows, [22](#)

WS-CDL, [24](#)

WSBPEL, [29](#)

WSFL, [24](#)

XLANG, [24](#)