Automated Derivation of Executable Business Processes from Choreographies in Virtual Organizations

Ingo Weber[1], Jochen Haller[1], Jutta A. Mülle[2]
[1]SAP Research
[2]IPD, Universität Karlsruhe (TH)
Karlsruhe (Germany)
{ingo.weber, jochen.haller}@sap.com, muelle@ipd.uka.de

Abstract: In this paper, we address the challenge of deriving both, executable WS-BPEL processes and their respective WSDL interface specifications from choreographies written in WS-CDL for business processes in Virtual Organizations (VOs). The major issues hereby are the differences in the vocabulary of WS-CDL and WSBPEL as well as the information gap between a choreography and an executable orchestration. The information gap results from the requirement imposed by the VO environment to establish a process-based collaboration in a top-down fashion. High-level choreography descriptions are hereby the basis for the derivation of detailed executable processes. The first issue is addressed with a detailed translation table; the second, more severe one requires the use of a role specific knowledge base. This knowledge base delivers process subsets modeling detailed role internal activities while avoiding their exposure to collaborating roles. The combined solution is a CDL2BPEL algorithm.

1 Introduction

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 exposing 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 OASIS[1] and W3C[2], 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 partnering 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 and not the details of the local activities.

---

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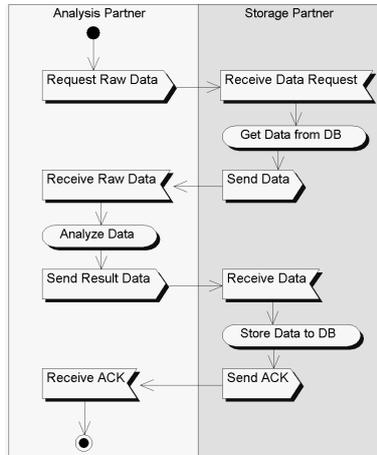[2]http://www.w3.org

Figure 1: The Analysis-Storage choreography part

A simple example from Collaborative Engineering is shown in Figure 1. The graph, modeled as an UML activity diagram[3], 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 referenced uniquely, and *AP* has 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 paper.

At the choreography level, the partners specify the global view of their co-operation 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 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 research work was conducted in the context of *Virtual Organizations (VOs)* within the European Union funded IST project TrustCoM, 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 de-

---

[3]Unified Modeling Language, see http://www.uml.org/

sired, following a top-down approach which is aligned with the VO formation and partner selection processes [RKH05].

Business application logic is hereby encapsulated in a 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 guarantee a defined execution order. In contrast, the global view encompasses the collaborative business process, which orders the interactions between the involved roles.

Given a choreography, the presented approach 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 [SO02], where private, executable processes maintain their confidentiality by only exposing views in terms of public processes to collaborating roles. The required local knowledge for role specific processes is introduced via a *Knowledge Base (KB)*. Although the approach focusses on the specific needs of virtual organizations, it is applicable in most 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.

The remainder of this paper is structured as follows: The problem statement, i.e., choreography support for VOs along with basic definitions form Section 2. Section 3 describes our solution comprising the local knowledge bases and the CDL2BPEL algorithm, which is based on the transformation rules available in the full version of this work in [Web05]. Related work is presented in Section 4. Section 5 outlines future work and concludes.

2 Choreography support for Virtual Organisations

Definition 1: A **Virtual Organization** is a combination of various parties (persons and/or organisations) 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 Organisation enjoy equal status and are dependent upon electronic connections (ICT infrastructure) for the co-ordination of their activities. (From [BvW98]). The conducted research was motivated by the VO environment introduced within the EU-funded project *TrustCoM*. TrustCoM focuses on VOs tackling collaborative projects in swift reaction to an emerging business opportunity. The life cycle of a VO is typically divided into four phases (from [SLS98]):

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

- The **VO Formation Phase** comprises of 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 specialised 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 [RKH05]. 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. (from [W3C05]).

In TrustCoM, the view based collaborative business process model is used, as presented by Schulz et al in [SO02], [SO01], which, in turn, is based on the specifications of the WfMC[4]. 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) in order to get a coherent and stable public workflow. The TrustCoM 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

---

[4]Workflow Management Coalition, among others see [WfM99] and [WfM02]

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.

2.1   The Information Gap

The information gap is our term of different levels of detail within the described collaborative business process model. 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.

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, specifying 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 glueing activities, which monitor e.g. service endpoint behaviour.

While the points 1 and 2 are addressed by the work presented in the following section, 3 is subject to future work. Point 4 should in our opinion be addressed by best practices for choreography design, and 5 tackles runtime behaviour which is considered out of scope for the presented work.

2.2 Language Differences

From the extensive set of available languages[5], we made the following three choices addressing the requirements and constraints as presented above in this section:

- The **Web Service Choreography Description Language (WS-CDL)** [W3C05] is used to specify choreographies. Although it is still in the process of standardization and severe structural critique was expressed [BDO05], it represents the most promising and expressive approach currently available.

- For executable (private) processes, the **Web Service Business Process Execution Language (WSBPEL)** 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.

- The public views are expressed in the **Web Service Description Language (WSDL)**. 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 Group standardizing WSBPEL.

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

Although the WS-CDL specification intends to be compliant with executable process languages as WSBPEL, the language elements differ in many respects. Where BPEL offers a set of atomic activities which can be combined elegantly to the desired behavior, WS-CDL knows certain elements which are far from being atomic: The cdl:interaction activity combines actual interactions with transactionality, timeouts, and assignments. cdl:WorkUnits are the sum of loops, conditional execution, variable value evaluation, and potential partial ordering of parallel activities. These and other differences make the actual derivation of executable BPEL processes hard, since the complex semantics of very expressive WS-CDL activities are not matched in the goal language. In certain cases, workarounds are possible. But in a subset of these cases, the workarounds cannot guarantee that the behavior of the generated processes is fully equivalent to the intended behavior from the choreography. The resulting semantic differences are outlined in the evaluation section.

In this section, the introduced terms were defined and the problems addressed by this paper were stated. Namely, there is 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

---

[5]As 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, XPDL, XLANG, WSFL, and BPDM for executable private processes.

differences resulting from the chosen set of languages. The next section is concerned with the solutions to these problems.

3   Technical Solution

In order to achieve an automated derivation of executable processes, the elements of a WS-CDL document are translated in a depth-first search through the XML tree. For each role in the choreography, a process is 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 2 shows a graphical representation of the outcomes of the BPEL derivation from the choreography in Figure 1. Clearly, the orchestrations are far more detailed, and the local substitutes for the high-level, private activities from the choreography are in place (the local service calls, which depend on the environment).

Also, the executable processes are concerned with initialization of the processes, which always means incoming messages in BPEL[6], as well as gluing activities. The latter include variable and message initialization, and apparent assignment statements. These runtime requirements are not to hard to meet, and the details of how this can be done are omitted here.

The derivation of BPEL and WSDL from WS-CDL is achieved in a 5-step algorithm, which is outlined in section 3.2. 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.

3.1   The Knowledge Base

The KB contains CDL patterns and their respective replacements in terms of BPEL activities as well as optional WSDL elements and even 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 goal documents.

---

[6]Note, that here only the AnalysisProvider's process needs to be invoked from outside. It then calls the SP process synchronously, so that one can be sure, all processes are available before starting the work.
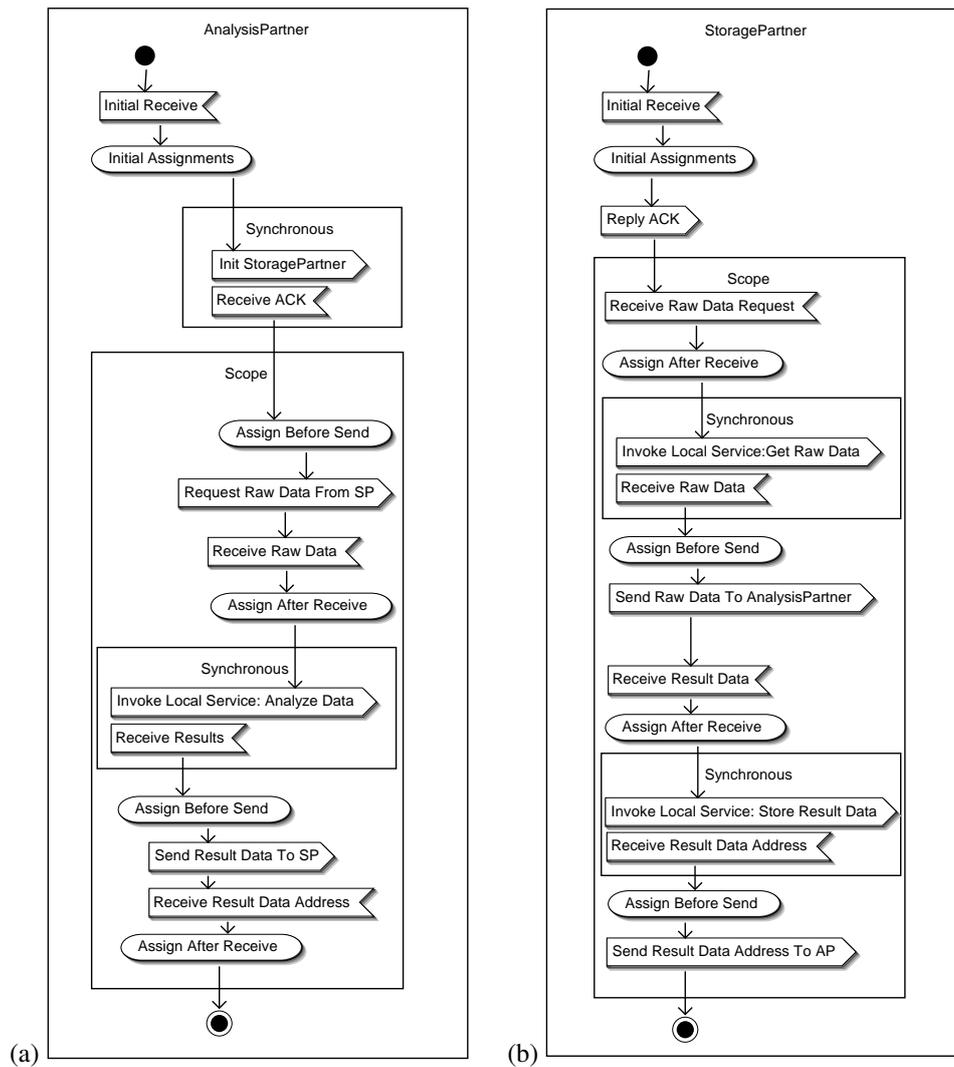
Figure 2: The resulting BPEL processes, derived from the Analysis-Storage choreography part in Figure 1. **(a)** The process for the AnalysisPartner role. **(b)** The process for the StorageProvider role.

Using this technique encourages the reuse of both choreographies and patterns and introduces a dynamic element into the derivation. The KB can be deployed and accessed solely locally at each member of a VO, 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.

3.2   The CDL2BPEL Algorithm

We did not attempt using XSLT[7] for the transformations due to its limitations. XSLT is designed for generating XML result documents out of XML source documents in a straight-forward manner. Here, we need more sophisticated mechanisms e.g. for validation: WS-CDL channel variables with a usage set to 'once' may actually be used more than once, and unrolling all the possible contingent execution paths that a choreography can take is a virtually impossible quest for XSLT. However, the implementation needs to be able to detect such errors in the choreography.
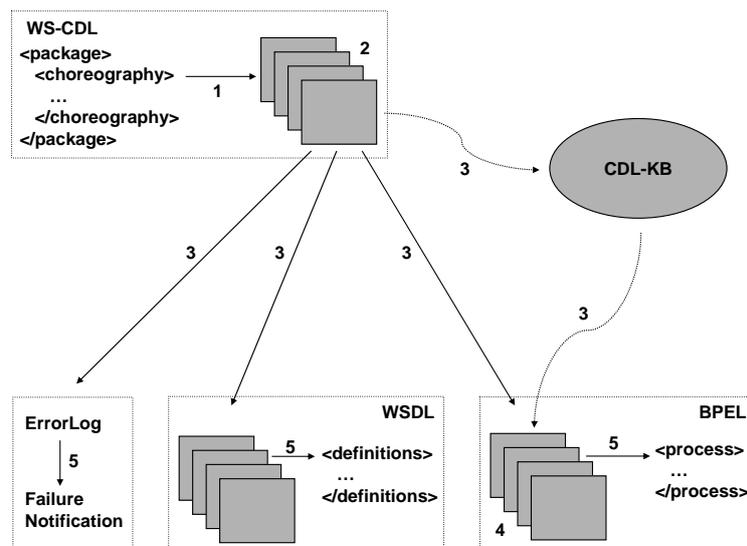

Figure 3: The five steps of the CDL2BPEL algorithm

Figure 3 shows the various steps of the algorithm graphically. In detail, the five steps are:

1. Read the choreography, create and initialize the corresponding Java objects from the WS-CDL elements.

---

[7]Extensible Stylesheet Language Transformations, see XSLT 1.0 [W3C99] and XLST 2.0 (http://www.w3.org/TR/xslt20/ , work in progress)

2. Validate the choreography (Correct variable and channel usage, the correct number of child elements with appropriate attributes and conditions such as guards)

3. Translate from CDL to BPEL and WSDL

   - Traverse the XML tree from the root choreography, 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)

   - 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:partner link types, bpel:properties)

4. Validate the generated BPEL processes from the holistic perspective, focussing on the partner links, operations, port types and exchanged variables. This static check ensures that the set of BPEL processes can be executed together based on their sequence of exchanges.

   - Add necessary gluing activities where obvious, e.g. assignments.

   - Remove superfluously structuring activities (e.g. a sequence with one child) which are leftovers from step 3).

5. Generate the BPEL and WSDL code from the objects OR return failure note

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 with each maintaining an own BPEL engine. Thus, this decision can be seen as enabling more flexibility, but shifting the enforcement of the participant-role constraints to another spot.

For further reading, the details of the translation are listed in the translation table in [Web05]. For most of the WS-CDL elements, a match in BPEL was found. In particular, the cdl:silentAction and cdl:choice with non-observable conditions are cases for which, by design of WS-CDL, no match can be found in BPEL. One other construct has to be mentioned here as well: A cdl:choice with both, blocking and non-blocking cdl:workUnits as children can only be translated to a workaround - a set of activities in BPEL which can differ in their execution from the intended behavior in the choreography. This is only the case if certain complex interdependent sets of guard conditions fall together with events

taking place at particular points in time. It is the most challenging case of language differences between WS-CDL and BPEL, and only for that we did not find a perfectly satisfying solution.

3.3 Prototypical Implementation and Evaluation

For both, the CDL2BPEL algorithm and the KB, a proof-of-concept prototype has been implemented as Java Web Services based on Axis (1.2RC3) ([Fou05]). The Knowledge Base builds on a relational database. The business processes in Figure 2 demonstrate graphically the result of applying the implementation to the CDL document belonging to the choreography from Figure 1.

[Web05] gives a conceptual mapping for all WS-CDL elements to BPEL and WSDL. The CDL2BPEL algorithm uses this mapping and the 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. The outputs are processes which are indeed executable.

However, 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 time-out to happen later than intended or even missing an event completely.

A solution to these issues requires basically changes or extensions in BPEL, thus being postponed until the next version[8] of BPEL becomes available.

As already mentioned above, the implementation was tested with TrustCoM choreographies, stemming from collaborative engineering examples. The choreographies we used departed from simple examples with two roles and eleven activities up to more complex examples with five roles and 40 activities, not counting interactions. In principle, the approach of the CDL2BPEL algorithm is valid beyond the TrustCoM related samples. The translation table is independent of any application scenario as is the Knowledge Base concept. To apply a particular implementation instance in an application scenario, the only application specific dependency relates to the Knowledge Base content. An occurring cdl:silentAction to "Analyze data" for instance will be resolved by a Knowledge Base query and therefore, its content has to deliver the fitting process part. "Analyze data" may be relevant for an aerospace choreography as wells as an automotive one, but the process parts will differ depending on the example.

---

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

The evaluation has shown, that in most practically relevant cases the algorithm yields executable processes. Exceptions, besides above cdl:choice example, may arise with the use of the *align* attribute in cdl:interactions which express required transactional behaviour. Translated to BPEL, this involves the usage of compensation handlers and rollback variables storing the used variable's initial state. While BPEL alone can cope with simple transactions, we believe that in complex collaborations involving for instance long running or dynamic transactions among multiple roles would require the use of other WS standards such as WS-Transaction in conjunction with WS-Coordination. Another exception is the channel concept in WS-CDL which has no direct counterpart in BPEL. Furthermore, tokens can be declared in WS-CDL which relate to correlation sets in BPEL. In contrast to CDL, an activity which initialises a correlation set has to be explicitly defined [Web05].

4   Related Work

The main issue of this paper is the automated derivation of executable business processes specified in WSBPEL and WSDL from WS-CDL modelled business choreographies in an interorganizational environment.

There are several publications on issues related to interorganizational workflows ([ACea04, CCJL04, vdA00, vdAW01]). The overlapping and differing aspects of orchestrations and choreographies were already identified in [Pel03], but focussing on WSCI [W3C] and not on WS-CDL.

The tranformation of choreographies to WSBPEL is part of a comprehensive design methodology. Several approaches exist that propose such architectures for Business Process Modeling. As one of these, [Hav05] 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"[9] which could be used for automated BPMN generation, based on a given WS-CDL definition.

The ebXML BPSS[10] offers a public view on business processes during design time which has some 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 CPA[11]. Instead of striving for automation

---

[9][Hav05], p. 41

[10]ebXML: Electronic Business using eXtensible Markup Language (http://www.ebxml.org/), BPSS: Business Process Specification Schema. See [Ira01], [Mar03], and [NDE+01]

[11]Collaboration Protocol Agreement, part of ebXML

in setting up a collaboration, ebXML rather expects people to agree upon a CPA, taking existing executable processes into account.

In [MNN04], 15 different XML-based specifications for business process modeling are compared. The authors state, that WSBPEL is one of the most complete language in terms of available features, that supports our choice of BPEL as target language. The already obsolete BPML[12] co-exists, but receives far less support from the industry, possibly because it is not directly related to Web Service orchestration.

A conceptual model for the mapping from WSBPEL to WS-CDL is presented in [MH05]. 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 [MH05]. 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.

5 Conclusion and Future Work

The paper describes the task of deriving executable public and private processes from a high-level choreography, and presents a solution in form of the CDL2BPEL algorithm. In conclusion, the main challenges initially pointed out, namely overcoming in the first place the information gap when departing from high-level choreography descriptions could be solved with the introduction of the knowledge base. The latter fills the gaps between the global and the local collaboration perspective with partner-internal local knowledge. Both, knowledge base and the Web service oriented implementation of the CDL2BPEL algorithm achieve the remaining goal of an automated derivation approach within highly dynamic VO environments.

Future work is planned to build incrementally upon the presented derivation approach. As an immediate extension, we planned automatic process, private and public, deployment within a process execution environment. Technologywise, the choice of BPEL for private and WSDL for public processes introduces a large number of possible BPEL engines for this purpose. In the case at hand, the open source ActiveBPEL (http://www.activebpel.org) engine was chosen after careful testing due to its stable implementation and sensible architecture. 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 one of the authors [HRK05]. It will be implemented based on the presented work.

---

[12]Business Process Modeling Language, [Ark02]

Far-term future work could include basing the knowledge base on BP-focussed semantic descriptions of the tasks to be fulfilled. Therefore, the patterns could be generic, in that they would be compared with a functional description of a part of the choreography.

References

[ACea04]   G. Alonso, F. Casati, and et al. *Web Services - Concepts, Architectures and Applications*. Springer, 2004.

[Ark02]   Assaf Arkin. Business Process Modeling Language. San Mateo, CA: BPMI.org, 2002. Proposed Final Draft.

[BDO05]   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.

[BvW98]   René Bultje and Jacoliene van Wijk. Taxonomy of Virtual Organisations, based on definitions, characteristics and typology. VoNet: The Netwsletter @ http://www.virtual-organization.net/, 1998.

[CCJL04]   L.F. Cabrera, G. Copeland, J. Johnson, and D. Langworthy. Coordinating Web Services Activities with WS-Coordination, WS-AtomicTransaction, and WS-BusinessActivity, 2004.

[Fou05]   Apache Software Foundation. Web Services - Axis, November 2005.

[Hav05]   Mike Havey. Essential Business Process Modeling. Copyright 2005 O'Reilly Media, Inc, 2005.

[HRK05]   Jochen Haller, Philip Robinson, and Yuecel Karabulut. Security Controls in Collaborative Business Processes. In *6th IFIP Working Conference on VIRTUAL ENTERPRISES (PRO-VE'05)*, 2005.

[Ira01]   Romin Irani. Collaborative Electronic Business is here to stay: An Introduction to ebXML. http://www.webservicesarchitect.com/content/articles/irani02.asp, 2001.

[Mar03]   Benoit Marchal. An Introduction to the ebXML CPP. http://www.developer.com/xml/article.php/2247851, 2003.

[MH05]   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.

[MNN04]   Jan Mendling, Markus Nüttgens, and Gustaf Neumann. A Comparison of XML Interchange Formats for Business Process Modelling, 2004.

[NDE+01]   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.

[Pel03]   C. Peltz. Web Services Orchestration and Choreography. *Computer*, 36(10):46–52, 2003.

[RKH05]   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.

[SLS98]   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.

[SO01]   Karsten A. Schulz and Maria E. Orlowska. Architectural Issues for Cross-Organisational B2B Interactions, 2001.

[SO02]   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.

[vdA00]   W. M. P. van der Aalst. Loosely Coupled Interorganizational Workflows: Modeling and Analyzing Workflows Crossing Organizational Boundaries. *Information and Management*, 37:67–75, 2000.

[vdAW01]   W.M.P. van der Aalst and M. Weske. *The P2P Approach to Interorganizational Workflows*, pages 140–156. Springer, 2001.

[W3C]   W3C. *Web Service Choreography Interface (WSCI) 1.0*. W3C Note 8 August 2002.

[W3C99]   W3C. XSL Transformations (XSLT) Version 1.0, 1999. W3C Recommendation 16 November 1999.

[W3C05]   W3C. Web Services Choreography Description Language, 2005. W3C Latest Working Draft from October 8th, 2005, work in progress.

[Web05]   Ingo Weber. Automation in Collaborative Business Process Instantiation. Master thesis at the department of informatics, Universität Karlsruhe, Germany, November 2005.

[WfM99]   WfMC. Interface 1: Process Definition Interchange Process Model. Document number wfmc-tc-1016-p version 1.1 final, Workflow Management Coalition, 1999.

[WfM02]   WfMC. Workflow Process Definition Interface - XML Process Definition Language. v1.0 Final Draft. Document number wfmc-tc-1025, Workflow Management Coalition, October 2002.