Collaborative, Dynamic & Complex Systems: Modeling, Provision & Execution

Vasilios Andrikopoulos, Santiago Gómez Sáez, Dimka Karastoyanova, Andreas Weiß
Institute of Architecture of Application Systems, University of Stuttgart, Stuttgart, Germany
{*vasilios.andrikopoulos, santiago.gomez-saez, dimka.karastoyanova, andreas.weiss*}*@iaas.uni-stuttgart.de*

Keywords: collaborative, dynamic & complex systems; service orchestration & choreography; pervasive computing; service networks; context-awareness

Abstract: Service orientation has significantly facilitated the development of complex distributed systems spanning multiple organizations. However, different application areas approach such systems in domain-specific ways, focusing only on particular aspects relevant for their application types. As a result, we observe a very fragmented landscape of service-oriented systems, which does not enable collaboration across organizations. To address this concern, in this work we introduce the notion of Collaborative, Dynamic and Complex (CDC) systems and position them with respect to existing technologies. In addition, we present how CDC systems are modeled and the steps to provision and execute them. Furthermore, we contribute an architecture and prototypical implementation, which we evaluate by means of a case study in a Cloud-enabled context-aware pervasive application.

1 Introduction

Complex software systems involving multiple, independent partners/software components collaborating in order to achieve one or more goals find predominant application in the current IT landscape. Cases of such systems from different domains are for instance business applications targeting enactment of complex business transactions and service networks, scientific workflows providing one approach for scientific experimenting in eScience, and pervasive systems representing one flavor of ubiquitous computing. Based on our research work towards building support systems for the development and execution of such applications in these domains, we conclude that while all the above-mentioned application areas concentrate on creating complex systems with very specific features critical for the corresponding domain, there are requirements valid across all domains. Our experience also shows that synergies between these domains can be exploited and potential benefits realized through reuse of research results and available software systems.

In this respect, in this work we investigate the requirements towards software systems in the above mentioned application areas with the purpose of identifying overlaps and differences. As we are going to show, the overlaps are significant and the differences are mainly due to the special focus on critical aspects in each domain, and not because the solutions are not relevant in the other domains. Based on these findings we introduce the innovative notion of *Collaborative, Dynamic and Complex (CDC)* systems aiming to cover all identified requirements and allowing to apply already existing technologies and software systems. CDC systems exhibit the aspects of modeling, provision and execution.

The contributions of this work target enabling the modeling, provision and execution of CDC systems and can be summarized by:

- The synthesis of existing technologies and approaches from the service-oriented computing paradigm and beyond, into a new, unified type of Collaborative, Dynamic and Complex (CDC) systems.
- The specification of the architecture for a framework that supports the various aspects (modeling, provision and execution) of CDC systems.
- The realization of this framework into a prototype called CoDyCo, and its use for the evaluation of our approach based on a case study.

The remaining paper is structured as follows. Section 2 looks into different application areas that deal with relevant for this work systems in order to highlight their similarities and establish the minimum set of requirements for our work. Section 3 presents our

proposal for CDC systems and positions them with respect to existing approaches. Section 4 introduces the architecture of a CDC-supporting framework required for implementing CDC systems. Section 5 reports on the overview and current state of CoDyCo. Section 6 summarizes a case study we performed using CoDyCo for purposes of evaluation. Finally, Section 7 presents related works, and Section 8 concludes the paper.

2 Motivation

In the following, we look into the areas of pervasive systems, service networks and scientific workflows. Our experience in research projects[1] shows, that despite the differences, the available approaches from these areas have many commonalities.

Pervasive Systems: Pervasive systems strive towards enabling the paradigm of ubiquitous computing and have been a subject of interdisciplinary research. Advances in pervasive systems have focused on the aspect of context-awareness, i.e. taking into account the context of physical and virtual entities, which is in fact a view of the physical environment, and the influence of the context on the applications the entities are using or participating in (Baldauf et al., 2007). A major requirement in these systems is the ability to adapt their behavior with respect to the context. Another major challenge is the optimization of the distribution of applications based on context data and resource consumption. The distribution of pervasive applications across multiple software system and hardware devices require their integration and coordination towards enabling a collaboration among participating devices and systems. Due to the dynamic characteristics of the environment of pervasive applications, with participants and devices appearing and disappearing constantly, supporting context sharing, adaptation, and scalability are particularly challenging. Since recently, Cloud Platforms in the scope of the Internet of Things and Smart Systems initiatives have been investigated from the point of view of enabling scalability, multitenancy and adaptability (Distefano et al., 2012). Later in this paper, we present a detailed description of a pervasive context-aware application for booking a taxi nearest to the location of a user based on the information provided by mobile devices available to the taxi drivers. We also use the taxi application for purposes of evaluation as a case study.

[1]For example SimTech `http://www.simtech.uni-stuttgart.de/`, S-Cube `http://www.s-cube-network.eu/`, 4CaaSt `http://www.4caast.eu/`, ALLOW Ensembles `http://www.allow-ensembles.eu/`.

Service Networks: Service Networks (SNs) (Caswell et al., 2008) are considered a specialized view on business processes, which focus on assisting business experts to evaluate the value of participating in a collaborative business activity. SNs are modeled as a network of business services exchanging offerings; basically, the composite perceived value of the exchanged offerings with the other services determines the value of participating in the network to one participant. Typical examples of SNs are supply chains; the taxi application described later can also be viewed as an SN. There is a significant gap between the meta-models used by business experts when designing the SNs, and the technological realization that needs to be bridged by means of software engineering techniques like model-driven development and code generation, whereas both top-down and bottom-up approaches are required. In addition, service networks are inherently collaborative activities and therefore imply efforts towards integration of applications across organizations.

A SOA-based realization of service networks, as well as a meta-model and graphical notation are presented in (Danylevych et al., 2010) and (Bitsaki et al., 2008). The interoperability of service implementations in an SN have been addressed by means of Web services and the high-level meta-model has been mapped on choreographies of composite service (i.e. organization-specific business processes). Additionally, choreographies take over the role of coordinating the services in a network, which addresses another important requirement. Changes in the perceived value of a network to a participant may initiate changes in the individual partners or in the network as a whole, which have to be propagated to their technological realization. We therefore identify the need for adaptation of service networks; some preliminary attempts to support only some types of service networks adaptation (Wagner et al., 2012) are already available. Monitoring the value of an SN for a participant is not directly measurable, but can only be derived based on monitoring data provided by the execution environment for choreographies, orchestrations and services. Approaches based on business activity monitoring, like (Guinea et al., 2011) and (Wetzstein et al., 2012) are only first steps towards the necessary technological support.

Scientific Workflows: Scientific workflows enable the modeling and execution of scientific experiments and are part of the technology landscape in eScience (Sonntag and Karastoyanova, 2010). A major requirement in this field is first and foremost the user friendliness of the approach, so that scientists do not face a high learning burden when using the experiment mod-

eling tools. The division between the way scientists model an experiment and the meta-models used in the supporting IT systems is significant and there are different approaches towards eliminating it (Sonntag and Karastoyanova, 2010). Both top-down and bottom-up approaches are required to enable the use of existing software and the development of experiments from scratch. The distributed nature of complex scientific experiments requires integration and composition of scientific computing software, which presents an additional challenge due to the lack of clearly defined software engineering principles for building scientific applications, including scientific workflows. Reusability is hampered by the heterogeneous landscape of applications and integration is of high complexity, because of the large number of available techniques for composition. Since scientific discovery is based on exploring physical phenomena, huge amounts of data are collected via numerous types of mobile devices and sensors (e.g. simulations of the distribution of CO_2 in the soil, weather forecasts, biological system simulations, simulations of manufacturing systems, etc.), and need to be processed. Computations in scientific workflows are time-consuming and also distributed, and most often they do not exhibit characteristics of pervasive applications. Adaptation during the modeling and execution of scientific workflows is a must, as evidenced by existing work (Sonntag and Karastoyanova, 2010), (Sonntag and Karastoyanova, 2012).

Despite the different focus of the application systems described above, they all exhibit overlapping characteristics that can be leveraged in a unified manner across the various areas. The following section presents our proposal toward this goal.

3 CDC Systems

We define *Collaborative, Dynamic and Complex* (CDC) systems as distributed systems enabling collaboration among participants across different organizations. Participants of CDC systems are services, representing software systems of different granularity, virtual and physical devices, and individuals. CDC participants join and leave the system at will in order to fulfill their individual goals. CDC systems are capable of adapting with respect to different triggers in the system and/or in their environment. CDC systems consist potentially of a large amount of participants dealing with large amounts of data as part of multiple interactions between them, following one or more coordination protocols. CDC systems have three fundamental aspects: *Modeling*, *Provision* and *Execution*.

With respect to *modeling*, we use choreographies to define the high-level, domain-specific models of CDC systems. *Choreographies* describe the interaction protocol of the involved participants and the participant roles' definitions. In SOA environments, individual participant roles are implemented by service *orchestrations* exposed as services, whereas their service interfaces are compliant with the participant role definitions modeled in the choreography. The *services* composed by the orchestrations are either available in the software landscape of the participating organizations, or are discoverable in global service registries. Utilizing these SOA-based approaches provides a flexible way of composing applications in complex systems and facilitates application integration. To enable *context-awareness*, choreographies and orchestrations, as well as involved services, have to incorporate in their models context information and define its use and reaction to potential changes. Since context information may be part of correlation data of orchestrations belonging to an enacted choreography, a mapping between context and *correlation* mechanisms has to be in place.

Performance indicators, like KPIs, utility, value, etc. are an inseparable part of the CDC system models. On the one hand, they are used to define the indicators according to which users will measure and evaluate whether they achieve their goals in a collaboration. On the other hand, this is the information needed to derive the data to be monitored during the execution of the CDC system. Therefore choreographies, orchestrations and services models have to contain elements defining the necessary *monitoring* information. In order to enable the *dynamic features* of CDC systems, constructs accommodating *adaptation* mechanisms in the choreographies and orchestrations have to be incorporated. Available approaches from the fields of workflow adaptation, flexible scientific workflows and pervasive dynamic flows, e.g. (Wetzstein et al., 2012) or (Sonntag and Karastoyanova, 2010) can be applied individually or in combination. Change propagation across all levels of the CDC systems and thus adaptation of choreographies can be identified as a major research challenge.

As identified in Section 2, two types of approach in modeling are required: top-down and bottom-up. *Top-down* CDC system modeling entails starting the development of the system with a choreography representing a realization of a high-level (domain-specific model), like a SN, scientific workflow, or pervasive application. Techniques required to map the choreography into orchestrations and services, like code generation and transformations, are available from software engineering and various existing SOA-enabling systems. The *bottom-up* approach involves deriving a meaningful choreography model based on existing or-

chestrations and/or services. In this case, deriving fault handling, monitoring and adaptation information is based on the corresponding capabilities of the involved services and correctness of the derived choreography.

The *provisioning* aspect of CDC systems entails the provision of the choreography, which also requires the deployment of orchestrations onto execution engines, their provisioning as services, populating the system with the corresponding context and correlation data, and configuring the monitoring infrastructure with the requirements from the CDC model. Mechanisms in service composition systems and scientific workflows for the provisioning of orchestrations and services are already available. Solutions for mapping monitoring requirements to monitoring probes are available in pervasive systems and service-based applications. The provision of a choreography results into adaptive and context-aware orchestrations available as a service. The choreography can be initiated multiple times for multiple interactions and can be started by any of the participating orchestrations or services, if they are allowed to do so by the choreography definition. Any underlying infrastructure should therefore enable sharing of resources across different CDC systems while correlating interactions to tenants and their users.

Running a choreography is therefore realized as a distributed *execution* of the collaboration among participating orchestrations and services. Since context-awareness is inherent to the CDC system model, the execution environment has to be able to support this property. Adaptation mechanisms, predefined in the system model (like abstract activities, binding strategies for services, reactions to context change, etc.) and such that are orthogonal to the model (like manual adaptation, forced termination, substituting a service endpoint, etc.) need to be realized by the execution environment. Furthermore, the execution environment of CDC systems must scale with their number of participants and their interactions, as well as the volumes of data exchanged. Monitoring information is necessary in order to enable such scaling.

4 CDC Framework Architecture

Figure 1 provides an overview of our proposal for a framework supporting the modeling, provision and execution of CDC systems. Starting from the modeling aspect, a *Choreography Editor* is required to create, visualize and manage the choreography models of the CDC systems. A *Transformer* component can then either generate orchestration templates that the CDC participants are meant to implement (in the top-down approach in the previous section), or derive possible

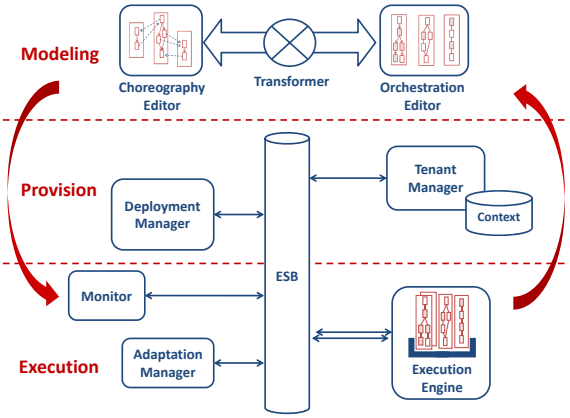

Figure 1: CDC-supporting framework — Architectural view

choreographies from existing orchestrations (in the bottom-up approach). In either case, an *Orchestration Editor* (not necessarily but preferably in the same environment as the Choreography Editor) should be available for orchestration visualization and manipulation. The transformer components also requires as input the service descriptions of the used orchestrations in the bottom-up approach or generates (abstract) service descriptions for derived orchestrations. Moving to the provisioning aspect, the *Deployment Manager* allows the assignment of the necessary for operation computational resources to the orchestrations involved in the modeled choreographies. Beyond physically deploying the necessary artifacts on an *Execution Engine*, this additionally entails the creation of all service endpoints necessary for accessing the orchestration logic by the system participants. The Deployment Manager also handles the information needed for late and dynamic binding to concrete service endpoints and provides it to the ESB during the execution of orchestrations.

In principle, multiple organizational domains may be using the same instantiation of this framework for different CDC systems. It is therefore necessary to offer *multi-tenancy capabilities* out of the box for all components *in the provision and execution aspect* of the framework. A *Tenant Manager* is responsible for this role, and implements administration and management capabilities for existing and new *tenants* (organizational domains) and their *users* (individuals or sub-systems in the same domain). The Tenant Manager is also meant to implement access control to both choreography and orchestration models, and to the computational resources corresponding to them during the execution of CDC systems, as assigned to them by the Deployment Manager. Only authorized parties should be allowed, for example, to participate in a given choreography. Furthermore, any collected *contextual information* relevant for tenants and users in terms of a representation of their environment, e.g.

their physical location or the quality of observed data, is stored and accessed through the Tenant Manager.

While the Deployment and Tenant Managers play prominent roles in the provision aspect of CDC systems, they are also heavily involved during CDC system execution, since both of them need to interact with the actual *Execution Engine* that runs the orchestrations defined during modeling. Furthermore, the Execution Engine has to provide fault handling capabilities, both for pre-defined fault and compensation handlers in the orchestration models, and for failures during execution like service failures and unavailability of other components in the framework (e.g. access to the Deployment Manager).

The *Adaptation Manager* is responsible for triggering and managing the adaptivity features of CDC systems by providing mechanisms for different types of adaptations across the levels of the systems. It implements and/or coordinates the actions necessary to enable the adaptation constructs from the CDC system model and the ones implemented only on the level of the execution environment. The Adaptation Manager collaborates also with the Deployment Manager when necessary, e.g. for re-binding service endpoints, and with the Execution Engine, e.g. for injecting a new activity and control connectors into an existing orchestration or deploying a new orchestration in case a choreography has been changed. The Adaptation Manager acts on information provided by the *Monitor* component which monitors and analyses the behavior and performance of the executed orchestrations, of the enacted choreographies, and also of the execution components in the framework. The Monitor must be configurable based on the monitoring information required for the CDC system and is responsible for providing to the users of choreographies and orchestrations personalized views of the relevant monitoring information on their devices.

Leveraging the SOA paradigm, all components in the framework relevant to execution should be provided as services and communicate through an *Enterprise Service Bus (ESB)* solution to facilitate their integration. Furthermore, each component should be designed and implemented allowing for both types of scalability: *horizontal* (modify number of available instances as required) and *vertical* (adjustment of available computational resources for each component) (Vaquero et al., 2011).

5 Implementation

In this section we present *CoDyCo*, a realization approach of the CDC-supporting framework presented in the previous section. As shown in Fig. 2, two separate environments are distinguished in the CoDyCo architecture: a *Modeling and Monitoring Toolset*, and a *Runtime Environment*.

Starting from the Modeling and Monitoring Toolset, CDC choreographies are specified in the BPEL4Chor language using our *BPEL4Chor Designer*. BPEL4Chor was first introduced by (Decker et al., 2008) as an extension of the WS-BPEL language (OASIS, 2007) and stemming from the business transactions field. However, BPEL4Chor choreographies are by definition not executable and therefore the transformation of BPEL4Chor definitions to WS-BPEL process (orchestration) skeletons (Reimann, 2007) is supported in CoDyCo by a *BPEL4Chor Transformer* (Fig. 2). Based on the choreography topology, the participants' grounding definitions, i.e. their WSDL interfaces, and their message links, this transformation generates the executable BPEL orchestrations for each participant in the choreography. The skeletons only model the interactions between partners so that together they can enact the choreography. Manual refinements can be performed on the created orchestrations, using our Mayflower BPEL Designer (Sonntag et al., 2012) developed in the context of the SimTech project[2] as an Eclipse-based BPEL designer. These refinements allow defining specific process logic for each participant, for example by reusing predefined process fragments from a process fragment library, as demonstrated in (Sonntag et al., 2012) and (Schumm et al., 2010). Both choreography definition and transformation functionalities are wrapped as an Eclipse Graphical Editor, and provide a palette with the graphical elements of the choreography language (Weiß et al., 2013). Only the top-down modeling approach (see Section 3) is currently supported by the Modeling and Monitoring Toolset; supporting the bottom-up approach is part of our future work. For more information on the status of the tools the interested reader is referred to (Weiß et al., 2013).

The deployment and instantiation of the BPEL processes generated by the Modeling and Monitoring Toolset is done on our Mayflower BPEL Engine. This engine is an extended version of the open-source Apache ODE Engine. It provides an interface for event publishing and configurable filtering and a BPEL event model[3] (Khalaf et al., 2007), which have been specialized for the purposes of monitoring and triggering dynamic adaptation of process instances(Sonntag et al., 2012). Functionalities provided by the Mayflower

[2]The SimTech project: `http://www.simtech.uni-stuttgart.de`

[3]ODE PGF: `http://www.iaas.uni-stuttgart.de/forschung/projects/ODE-PGF/`

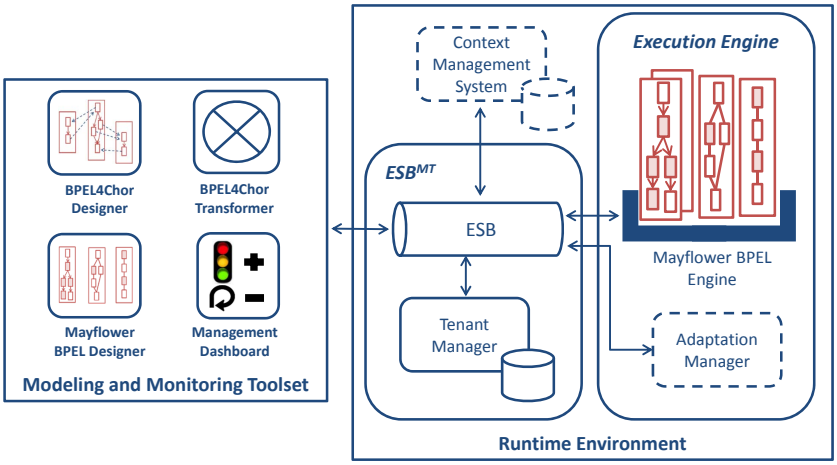

Figure 2: The CoDyCo System: Overview and Current State

BPEL Engine address the requirements of the CDC execution environment in terms of orchestration execution, storing of historical information, failure and compensation handling, and in combination with the Adaptation Manager also enable some dynamic adaptation patterns. For instance, dynamic adaptation of process instances triggered by humans are supported in CoDyCo through the *Modeling and Monitoring Toolset*. More specifically, the Mayflower BPEL Designer interacts with the Mayflower BPEL Engine and the Adaptation Manager and so allow the users to view the status of process instances and trigger their adaptation. Based on this users can adapt the process instance by changing its graphical representation.

The adaptation operations that can be performed on a process instance in our current implementation are, e.g. re-execution or forced iteration of activities (Sonntag and Karastoyanova, 2012), insertion and deletion of process elements, or their substitution. The changes made on the viewed process instance are then propagated to the Adaptation Manager Component, which is responsible for performing the actual adaptation on the concrete process instance in the engine. This is also made possible by auxiliary functions in the Mayflower BPEL Designer, mirrored in the Mayflower BPEL Engine, like enabling user subscriptions to monitoring events published by the engine, and hence retrieval of real-time information about a concrete process instance, and built-in actions per process instance like stop, suspend, resume, etc. (Sonntag et al., 2012). Automatic adaptation of running process instances, like injection of process fragments is currently not supported by the Adaptation Manager in CoDyCo [4].

Communication between participants collaborating in the scope of a specific choreography instance, and between the different components comprising the runtime environment is established through the multi-tenant open source ESB solution *ESB^{MT}* (Strauch et al., 2012a), (Strauch et al., 2012b). ESB^{MT} enhances an existing ESB Solution, Apache ServiceMix [5] with multi-tenant awareness both at the administration and management, and messaging levels. Management and administration functionalities implemented by a *Tenant Manager* enable the dynamic deployment and configuration of service endpoints with tenant- and user-specific information, while tenant-aware messaging capabilities isolate tenants' messages routed to the service endpoints (Strauch et al., 2012b). As tenants represent organizational units, e.g. Taxi Company A, which has N taxi drivers, and M potential taxi customers, their communication in the scope of a choreography is supported in the ESB^{MT} through tenant-aware service endpoints. At the moment we are working on the integration of our execution engine with the ESB^{MT} as a JBI Service Engine, to enable multi-tenancy support for orchestrations and choreographies, too.

The Management Dashboard is the component in the Modeling and Monitoring Toolset responsible for providing analyzed and personalized monitoring information to users and tenants about the execution state of choreographies and orchestrations. Our prototype visualizes the execution status of orchestrations, the data they produce and consume, and the adaptations that have been performed (Sonntag et al., 2012). This is possible due to the interaction of the Management Dashboard with the Mayflower BPEL Engine via its event publishing interface, as described above. Our approaches for monitoring choreographies (Wagner

[4]Note that this is possible only manually, as described above.

[5]Apache ServiceMix: `http://servicemix.apache.org/`

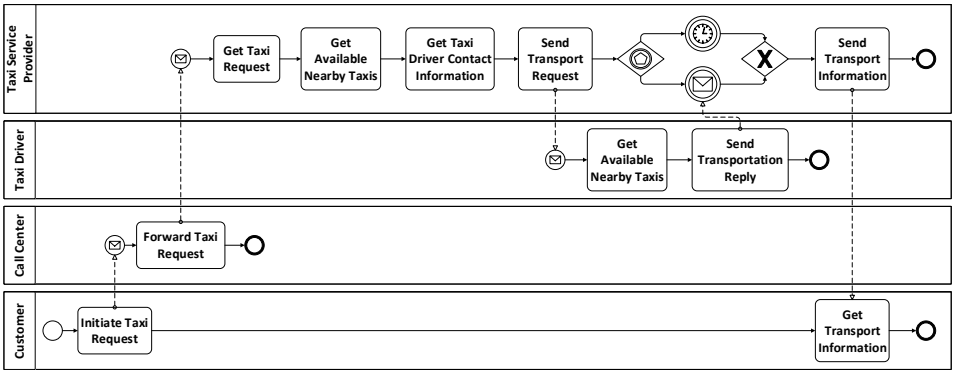

Figure 3: The BPMN process diagram of the Taxi Application.

et al., 2012) and KPIs (Wetzstein et al., 2012) are not yet integrated into CoDyCo.

The *Context Management System* stores the system context (i.e. a set of context properties of all active tenants) and constantly synchronizes its current configuration with the tenant specific applications by retrieving context data from each of the tenant users created in the system, e.g. from a location provider embedded system in a specific taxi cab. Context information is used in the execution of choreographies by the corresponding orchestrations through *Context Integration Processes* (CIPs) (Wieland et al., 2007) and may trigger their adaptation. Context-aware adaptations will be handled by a collaboration of the *Adaptation Manager*, *the Mayflower Engine*, the *Context Management System (CMS)*, and the *Tenant Manager* and is part of our future work on the implementation. The modeling tool is currently also missing features supporting modeling of context in the choreographies.

6 Case Study

In the scope of project 4CaaSt[6], the *Taxi Scenario* use case has been defined, where a service provider offers a taxi management software as a service to different taxi companies, i.e. tenants. Taxi company customers, who are the *users* of the tenant, submit their taxi transportation requests to the company they are registered with. The taxi company uses the taxi management software to contact nearby taxi drivers. Once one of the contacted taxi drivers has confirmed the transportation request, the taxi management software sends a transport notification containing the estimated arrival time to the customer. As discussed in Section 2, the Taxi Scenario constitutes a pervasive context-aware application and therefore, in the scope of this work, an ideal candidate for the evaluation of our proposal.

[6]The 4CaaSt project: `http://www.4caast.eu`

Figure 3 shows the processes the Taxi Scenario application realizes. The simplified BPMN diagram has three lanes depicting the three participants of the application choreography: Customer, Taxi Company, and Taxi Service Provider. If a Customer wishes to book a Taxi, he sends an initial request to the Taxi Company call center (usually through a Web GUI), which forwards it to the Taxi Service Provider. The Taxi Service Provider process determines the nearby available taxis and the contact information of the taxi drivers using CIPs (Wieland et al., 2007) (not shown here for brevity). Subsequently, the transport request is sent to each available taxi driver, and their responses are collected for a specified duration. The gathered transport information is sent back to the Customer. Implementing the Taxi Scenario required the manual design of all involved processes as orchestrations for each participant and their interactions, as well as the implementation of most services involved in them (except from those that already existed, e.g. a context provisioning framework exposed as a service (Knappmeyer et al., 2010)).

For the evaluation of our approach we started with the BPMN diagram in Fig. 3 which we translated into BPEL4Chor. Figure 4 shows the processes depicted in Fig. 3 as participants in our BPEL4Chor Designer. The rectangular shapes in the editor view in Fig. 4 stand for the choreography participants, whereas the message links and their directions are depicted by labeled arrows. The set of taxi drivers are represented by the Taxi Transmitters participant, standing for the devices carried by the drivers. Inside each participant, the control flow regarding its communication behavior such as receive or send activities is visible. The resulting choreography model was then transformed into a series of BPEL4Chor artifacts by the BPEL4Chor Designer: a participant topology specifying the involved participants in the choreography, the participant types, and the message links between participants.

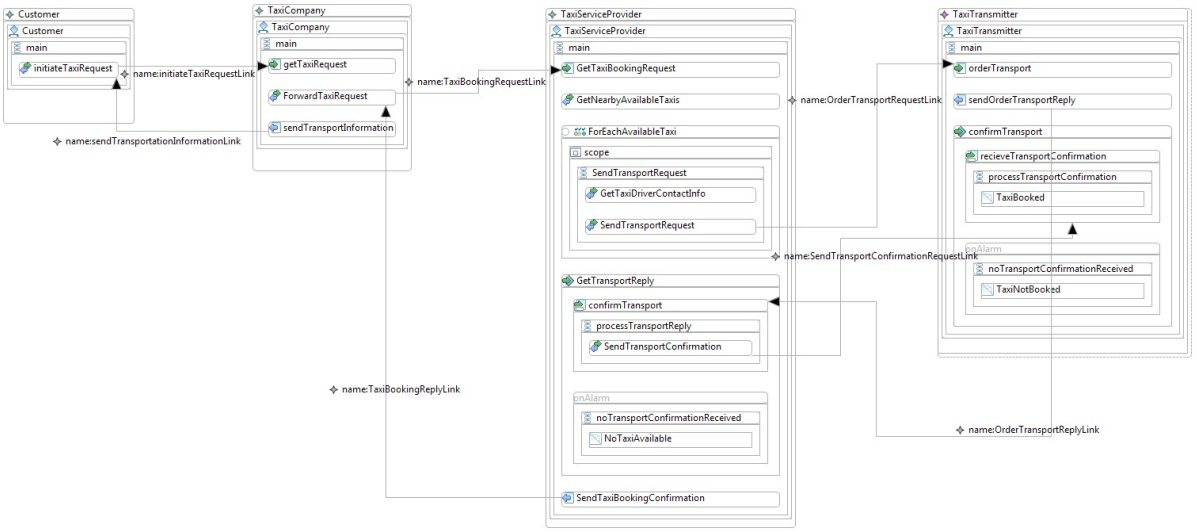

Figure 4: Excerpt of the Taxi Application choreography modeled with the Chor Designer.

The BPEL4Chor artifacts are used by the BPEL4Chor Transformer to generate Abstract BPEL processes and WSDL files. These WSDL files contain the technical information about the interfaces between the participants, i.e. the port types, operations, messages, and partner links. Each previously modeled participant was transformed into exactly one Abstract BPEL process. Basic executable completion of the Abstract BPEL processes, i.e. their transformation into executable ones, is supported by the BPEL4Chor Designer, as well as the manual refinement of the process logic in each participant that is not part of the choreography. The resulting (executable) processes were then deployed and executed successfully in our Mayflower BPEL Engine. Through this process we have also identified a number of technical issues with the current implementation of the BPEL4Chor Designer and Transformer (Weiß et al., 2013). For example, not all BPEL activities are currently supported, and the generated WSDL files need manual definition of the involved message types. We are already working on addressing these deficiencies.

7 Related Work

As discussed in (Barker et al., 2009), the interaction between participants in a choreography can be modeled following the interaction, or interconnection modeling approaches. The former approach models atomic interactions between participants through *interaction activities*, while the latter interconnects the communication activities of each participant of the choreography. The WS-CDL[7] language standard supports the interaction approach. Using the WS-CDL language as the basis, the Savara[8] project aims to provide tooling support for a top-down choreography modeling approach. Interconnection modeling approaches are supported in the CHOReOS Integrated Development and Runtime Environment[9], in the Open Knowledge European project[10], and in BPEL4Chor (Decker et al., 2007). The CHOReOS environment supports the choreography specification using BPMN 2.0 collaborations (CHOReOS Consortium, 2011), and encompasses choreography adaptation based on service availability and QoS assurance.

The Open Knowledge framework employs a multiagent protocol to control the interactions between participants in the choreography. Therefore, participants must be specified and deployed prior to the choreography enactment, and adaptation based on context modifications is not considered. As discussed in the previous sections, BPEL4Chor wraps the choreography specification in a layer atop of WS-BPEL which contains the choreography control flow, its participants description and message links between them, and the mapping support to their concrete communication descriptions (WSDL). BPEL4Chor does not support the explicit specification of rules for context-aware adaptation purposes, but decouples the choreography specification from communication specific details, enabling extensibility for dynamic context-aware choreography adaptation.

[7] WS-CDL:`http://www.w3.org/TR/ws-cdl-10/`

[8] `http://www.jboss.org/savara`

[9] EU Project CHOReOS: `http://www.choreos.eu/`

[10] Open Knowledge: `http://www.openk.org/`

Context-aware systems have been widely studied in the scope of Ubiquitous Computing. In (Baldauf et al., 2007) a set of context-aware systems are presented, and a comparison focusing on the architectural principles of context-aware middleware and framework to ease the development of context-aware applications is provided. The CoWSAMI middleware infrastructure utilizes Web services for managing location context in open ambient intelligence environments (Athanasopoulos et al., 2008). The utilization of an ESB as the central piece for communication support in context-aware systems is discussed in (Chanda et al., 2011), where a Context-aware ESB (CA-ESB) is proposed to discover and orchestrate services based on the users' location and available services in specific regions.

Concerning different context views in pervasive environments, in (Abdulrazak et al., 2010) micro and macro context-awareness modeling approaches are presented. The former describes the users' surroundings and aims to provide access to local context data, while the latter aggregates local context data to provide a global perspective of different spaces. Self-configuration operations in micro context-awareness models involve coordination of peers in a decentralized manner, making choreographies suitable for modeling the coordination between peers. Furthermore, in (Roy et al., 2008) high system availability is achieved by decentralizing the coordination of entities collaborating in context construction and decision making activities in open intelligence spaces ensures.

Context-aware workflows as an approach for easing the development of context-aware applications are presented in (Wieland et al., 2007). Thus, they propose Context4BPEL, a WS-BPEL[11] extension for explicitly modeling the influence of context on workflows. However, WS-BPEL supports orchestration of services within a business process, while choreography modeling approaches demand a further semantic support for specifying process interactions from a global view. Further research on workflow flexibility has been conducted by integrating support of human interactions during the execution of scientific workflows in (Karastoyanova et al., 2012). This approach triggers human interactions for non-automated activities via a framework supporting a multi-protocol communication between a scientific workflow management system and pluggable communication devices. All these approaches are focusing on only one particular aspect of CDC systems.

[11]WS-BPEL 2.0: `http://docs.oasis-open.org/wsbpel/2.0/`

8 Conclusions and Future Work

Our investigation into different application areas like pervasive systems, service networks and scientific workflow systems that have been influenced by service-orientation illustrated a series of overlapping characteristics that have not been leveraged so far. Toward this purpose, in this work we introduced the notion of Collaborative, Dynamic and Complex (CDC) Systems as dynamic distributed systems that allow participants from different organizations to collaborate to fulfill their goals. We discussed three fundamental aspects of CDC systems: modeling, provision and execution, and presented the architecture of a framework that supports these aspects. We then showed the current status of the prototypical implementation of CoDyCo, a system that realizes this framework. A case study on a context-aware pervasive application was then presented for purposes of evaluating our proposal.

Currently, we are working on improving the state of the CoDyCo prototype by addressing the deficiencies identified during the case study. Future work is aimed at finalizing the different aspects of our proposal. Concerning modeling and provision CDC systems, the bottom-up modeling approach has to be realized, as well as enabling context-awareness in choreographies and orchestrations for both type of approaches. This also entails the realization of the context management system. In addition, multi-tenancy awareness has to be enabled for choreographies and orchestrations, and reflected in the Execution Engine. The Management Dashboard has to be integrated with approaches to monitoring KPIs and business transactions which is a step towards enabling the monitoring of choreographies. In terms of adaptation, available approaches context-aware adaptation and automatic adaptation of orchestration has to be integrated in CoDyCo. Finally, the scalability features of the CoDyCo components have to be investigated further in the scope of our Cloud computing research.

9 Acknowledgments

This work is funded by the projects FP7 EU-FET 600792 ALLOW Ensembles and the German DFG within the Cluster of Excellence (EXC310) in Simulation Technology.

REFERENCES

Abdulrazak, B., Roy, P., Gouin-Vallerand, C., Giroux, S., and Belala, Y. (2010). Macro and micro context-awareness

for autonomic pervasive computing. In *Proceedings of iiWAS'2010*, pages 427–434. ACM.

Athanasopoulos, D., Zarras, A. V., Issarny, V., Pitoura, E., and Vassiliadis, P. (2008). Cowsami: Interface-aware context gathering in ambient intelligence environments. *Pervasive and Mobile Computing*, 4(3):360–389.

Baldauf, M., Dustdar, S., and Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277.

Barker, A., Walton, C. D., and Robertson, D. (2009). Choreographing web services. *IEEE Transactions on Services Computing*, 2(2):152–166.

Bitsaki, M., Danylevych, O., van den Heuvel, W.-J., Koutras, G., Leymann, F., Mancioppi, M., Nikolaou, C., and Papazoglou, M. (2008). An Architecture for Managing the Lifecycle of Business Goals for Partners in a Service Network. In *Proceedings of ServiceWave'08*, Lecture Notes in Computer Science, pages 196–207. Springer Berlin Heidelberg.

Caswell, N. S., Nikolaou, C., Sairamesh, J., Bitsaki, M., Koutras, G. D., and Iacovidis, G. (2008). Estimating value in service systems: A case study of a repair service system. *IBM Systems Journal*, 47(1):87–100.

Chanda, J., Sengupta, S., Kanjilal, A., and Bhattacharya, S. (2011). CA-ESB: Context Aware Enterprise Service Bus. *International Journal of Computer Applications*, 30(3):1–8.

CHOReOS Consortium (2011). CHOReOS Dynamic Development Model Definition (D2.1).

Danylevych, O., Karastoyanova, D., and Leymann, F. (2010). Service Networks Modelling: An SOA & BPM Standpoint. *J.UCS*, 16(13):1668–1693.

Decker, G., Kopp, O., Leymann, F., Pfitzner, K., and Weske, M. (2008). Modeling Service Choreographies Using BPMN and BPEL4Chor. In *Proceedings of CAiSE'08*, pages 79–93. Springer.

Decker, G., Kopp, O., Leymann, F., and Weske, M. (2007). BPEL4Chor: Extending BPEL for Modeling Choreographies. In *Proceedings of ICWS'07*, pages 296–303. IEEE Computer Society.

Distefano, S., Merlino, G., and Puliafito, A. (2012). Enabling the cloud of things. In *Proceedings of IMIS'12*, pages 858–863.

Guinea, S., Kecskemeti, G., Marconi, A., and Wetzstein, B. (2011). Multi-layered monitoring and adaptation. In *Proceedings of ICSOC'11*, pages 359–373. Springer.

Karastoyanova, D., Dentsas, D., Schumm, D., Sonntag, M., Sun, L., and Vukojevic, K. (2012). Service-based Integration of Human Users in Workflow-driven Scientific Experiments. In *Proceedings of eScience'12*, pages 1–8. IEEE Computer Society Press.

Khalaf, R., Karastoyanova, D., and Leymann, F. (2007). Pluggable framework for enabling the execution of extended bpel behavior. In *Proceedings of WESOA'07*, pages 376–387. Springer.

Knappmeyer, M., Kiani, S. L., Frà, C., Moltchanov, B., and Baker, N. (2010). ContextML: a light-weight context representation and context management schema. In *Proceedings of ISWPC'10*, pages 367–372. IEEE.

OASIS (2007). Web services business process execution language version 2.0.

Reimann, P. (2007). Generating BPEL Processes from a BPEL4Chor Description. Studienarbeit 2100, Universität Stuttgart, Institut für Architektur von Anwendungssystemen.

Roy, P., Abdulrazak, B., and Belala, Y. (2008). Approaching context-awareness for open intelligent space. In *Proceedings of MoMM'08*, pages 422–426. ACM.

Schumm, D., Karastoyanova, D., Leymann, F., and Strauch, S. (2010). Fragmento: Advanced Process Fragment Library. In *Proceedings of ISD'10*, pages 659–670. Springer.

Sonntag, M., Hahn, M., and Karastoyanova, D. (2012). Mayflower - Explorative Modeling of Scientific Workflows with BPEL. In *Proceedings of BPM'12*, pages 1–5. CEUR Workshop Proceedings.

Sonntag, M. and Karastoyanova, D. (2010). Next Generation Interactive Scientific Experimenting Based On The Workflow Technology. In *Proceedings of MS'10*. ACTA Press.

Sonntag, M. and Karastoyanova, D. (2012). Ad hoc Iteration and Re-execution of Activities in Workflows. *International Journal On Advances in Software*, 5(1 & 2):91–109.

Strauch, S., Andrikopoulos, V., Leymann, F., and Muhler, D. (2012a). ESB^{MT}: Enabling Multi-Tenancy in Enterprise Service Buses. In *Proceedings of CloudCom'12*, pages 456–463. IEEE Computer Society.

Strauch, S., Andrikopoulos, V., Sáez, S. G., Leymann, F., and Muhler, D. (2012b). Enabling Tenant-Aware Administration and Management for JBI Environments. In *Proceedings of SOCA'12*, pages 206–213. IEEE Computer Society.

Vaquero, L., Rodero-Merino, L., and Buyya, R. (2011). Dynamically scaling applications in the cloud. *ACM SIGCOMM Computer Communication Review*, 41(1):45–52.

Wagner, S., Fehling, C., Karastoyanova, D., and Schumm, D. (2012). State Propagation-based Monitoring of Business Transactions. In *Proceedings of SOCA'12*, pages 1–8. IEEE.

Weiß A., Andrikopoulos, V., Gómez Sáez, S., Karastoyanova, D., and Vukojevic-Haupt, K. (2013). Modeling Choreographies using the BPEL4Chor Designer: an Evaluation Based on Case Studies. Technical Report 2013/03, IAAS, University of Stuttgart.

Wetzstein, B., Zengin, A., Kazhamiakin, R., Marconi, A., Pistore, M., Karastoyanova, D., and Leymann, F. (2012). Preventing KPI Violations in Business Processes based on Decision Tree Learning and Proactive Runtime Adaptation. *Journal of Systems Integration*, 3(1):3–18.

Wieland, M., Kopp, O., Nicklas, D., and Leymann, F. (2007). Towards context-aware workflows. In *Proceedings of CAiSE07*, pages 11–15. Springer.

All links were last followed on February 5, 2014.