An Integrated Development and Runtime Environment for the Future Internet

Amira Ben Hamida[1], Fabio Kon[5], Gustavo Ansaldi Oliva[5], Carlos Eduardo Moreira Dos Santos[5], Jean-Pierre Lorré[1], Marco Autili[4], Guglielmo De Angelis[6], Apostolos Zarras[3], Nikolaos Georgantas[2], Valérie Issarny[2], and Antonia Bertolino[6]

[1] Petals Link, France
[2] Inria, France
[3] University of Ioanina, Greece
[4] Università degli Studi dell'Aquila, Italy
[5] University of São Paulo, Brazil
[6] Institute of the National Research Council of Italy

Abstract. The Future Internet environments raise challenging issues for the Service-Oriented Architectures. Due to the scalability and heterogeneity issues new approaches are thought in order to leverage the SOA to support a wider range of services and users. The CHOReOS project is part of the European Community Initiative to sketch technological solutions for the future ultra large systems. In particular, CHOReOS explores the choreography of services paradigm. Within this project, a conceptual architecture combining both the development and runtime environments is realized. This chapter introduces the CHOReOS Integrated Development and Runtime Environment, aka IDRE.

Keywords: SOA, Service, Choreography, MDA, Cloud & Grid, IDRE, Governance, Middleware, Discovery, Access, TDD, V&V.

1 Context

Raising the Future Internet Challenges. The Future Internet (FI) context draws a global environment populated with a plethora of services. Such services are related to two - commonly identified by many FI initiatives - key FI dimensions, the Internet of (traditional) Services and the Internet of Things. The latter dimension is expected to considerably change the way we perceive the Internet today, by incorporating in it vast populations of physical objects or, from another viewpoint, sensors and actuators linking to the physical world. We take this SOA view of the FI one step forward by advocating choreographies of services i.e., compositions of peer interacting services as the primary architectural solution for leveraging and sustaining the richness and complexity of the FI. In this context, three key challenges, namely, scalability, heterogeneity, and awareness are raised. As already pointed out, the large scale of today's Internet becomes ultra large scale (ULS) in the FI, in terms of numbers of devices, services, things, users, requirements, and their infinite combinations within choreographies. Then, extreme heterogeneity is unavoidable in terms of the previous, and, additionally, in terms of interaction protocols at different levels, data, semantics, and related technologies. Third, awareness has to do with taking into account user requirements as well as context in all its dimensions, physical, system, and user context, as well as its volatility in the open, dynamic and mobile FI. The CHOReOS project is part of the European Community Initiative to sketch technological solutions for the future ultra large systems. In particular, CHOReOS explores the choreography of services paradigm.

Addressing the Future Internet Challenges. In this paper, we provide a comprehensive solution to the above particularly challenging issues. We realize the CHOReOS Integrated Development and Runtime Environment, aka IDRE. We exploit sophisticated research domains from the Service-Oriented Architecture (SOA) realm, including Service Discovery, Access and Composition as well as SOA Governance, together with the Model Driven Engineering (MDE) paradigm and the Cloud & Grid paradigms [9]. Building on MDE principles, the CHOReOS development process enables going from very high-level user requirements for service choreographies down to highly heterogeneous realizations of the final choreographies, where incompatibilities of the participating services are compensated for. It is worth noting that in the CHOReOS terminology (traditional) Services and the Internet of Things, become, respectively, the Internet of Business Services (IoBS) and the Internet of Thing-based Services (IoTS). To deal with environments where IoBS and IoTS coexist in a transparent way, CHOReOS IDRE relies on the integration, interoperability and large scale distribution capabilities provided by the Enterprise Service Bus middleware paradigm, which we extend and enhance to cope with the very heterogeneous deployment and interaction semantics and platforms of both types of services. Additionally, we develop sophisticated service discovery mechanisms in order to offer registration, classification, query and retrieval mechanisms adapted to the ULS populations of Business Services and Thing-based Services. Scalability issues are also considered at the levels of service access and provisioning, choreography deployment and need for computation, as well as management of vast populations of services and their data, where we exploit Cloud and Grid capabilities for offering a powerful and elastic platform of resources. Finally, we rely on the fundamentals of the Governance and Verification & Validation (V&V) domains for ensuring the quality of services and choreographies at both design and run time. Both functional and non-functional properties of services and choreographies are assessed, augmenting in this way our awareness of the composed choreographies. In this chapter, we introduce the IDRE conceptual view, detailing its subsystems and their respective functionalities. The remainder of the chapter is as follows. We provide an overview of the CHOReOS IDRE in Section 2. Section 2 is dedicated to the choreography synthesis. In Section 4, we detail the CHOReOS middleware. Section 5 introduces the Governance and V&V framework. Finally, we conclude in Section 6.

2 CHOReOS IDRE Overview

The CHOReOS IDRE relies on a modular SOA where a number of coarse-grained subsystems are integrated to support the overall development, from design to implementation, together with deployment and execution, of services choreographies in the FI. CHOReOS embeds the following subsystems: the CHOReOS Development Environment, the eXecutable Service Composition (XSC), the eXtensible Service Discovery (XSD), the eXtensible Service Access (XSA), Cloud and Grid Middleware, Governance and V&V Framework and finally the Monitoring (See Figure 1).

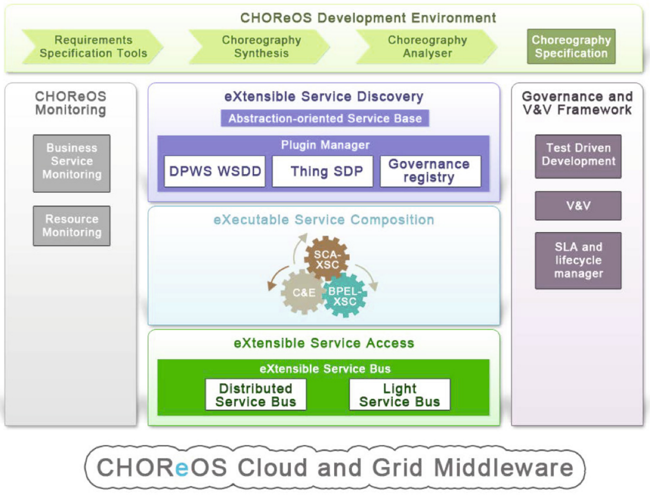
CHOReOS Development Environment
Requirements Specification Tools
Choreography Synthesis
Choreography Analyser
Choreography Specification
CHOReOS Monitoring
Business Service Monitoring
Resource Monitoring
eXtensible Service Discovery
Abstraction-oriented Service Base
Plugin Manager
DPWS WSDD
Thing SDP
Governance registry
eXecutable Service Composition
SCA-XSC
C&E
BPEL-XSC
eXtensible Service Access
eXtensible Service Bus
Distributed Service Bus
Light Service Bus
Governance and V&V Framework
Test Driven Development
V&V
SLA and lifecycle manager
CHOReOS Cloud and Grid Middleware

Fig. 1. CHOReOS IDRE Overview

3 CHOReOS Development Environment

ULS choreographies of services need to be created and analyzed with the aim of coping with the FI environments. For this purpose, the CHOReOS project provides a dedicated development environment (See Figure 2). Therefore, a model-driven development process is realized. First, thanks to dedicated *Requirements Specification Tools* the user requirements specification is captured. The final output of the requirements specification activity is a choreography specification (in the BPMN2.0 language), which serves as input to the next phases of the overall process. Second, the *Synthesis Processor* operates an automated synthesis of specific software entities, namely *Coordination Delegates*, that coordinate the collaboration among the services so as to enact the choreography in a fully distributed way. These are executed on top of the CHOReOS Middleware (See Section 4). Third, the development process ends with the scalability analysis performed by the *Choreography Analyzer*.

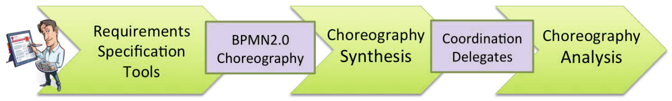

Fig. 2. CHOReOS Development Process

3.1 Requirements Specification Tools

The Requirements Specification Tools are mainly responsible for enabling domain experts to specify functional and quality requirements on services and service-based applications, and in turn, to enable the domain expert to produce a first draft choreography specification. First, the *Specification Expressing Tool and DataBase* provide the domain expert with service consumer requirements and associated attributes. The service consumer specifies requirements using a structured approach facilitated by mobile tools – such as the iPhone app (application). There can be many service consumers with many user needs. The expressed requirements are recorded in a DataBase along with attributes for quality, priority and situation. Associated with the requirements is a quality model, which relates the user requirements on service-based applications to QoS on services aggregated in these applications. Second, the *Requirements Management and Analysis Tool* provides the domain expert with requirements management and analysis functions. These functions are provided to help the domain expert to pull out individual requirements in order to form a set of requirements for choreography. Third, the *Requirements Engine* executes a matching and grouping algorithm to cluster the service consumer and domain expert expressed requirements. A 'calculate similarity' algorithm, enables the requirement comparison for similarity using natural language processing techniques. The output from this component is grouped requirements for choreographies. Finally, the *Matching Tool and User Task Model Database* are responsible for matching the requirements on the choreography specification to user task models using a matching tool. A set of CTT (Concur Task Trees) task models, describing structured activities that are often executed during the interaction with a system are defined and stored in a database. Finally, the prioritized quality-based requirements and user task models are then associated with choreography strategies, which are expressed in the form of patterns by the choreography designer. The final output of this process is a first draft choreography specification and a set of associated requirements to inform the discovery of abstract services.

3.2 Synthesis Processing

Advancing the foundational background on software coordination via automated coordinator synthesis [3,8,11], the Synthesis Processor subsystem is mainly responsible of synthesizing the coordination delegates that are in charge of suitably coordinating, in a distributed way, the services participating to the choreography. The approach starts from the BPMN2 choreography model and from the set of discovered services. The first input comes from the refinement of the CTT models and choreography patterns (and hence, the first draft choreography specification discussed in Section 3.1). The latter comes from the exploitation of the service base management mechanisms described in Section 4.1. Thus, the synthesis process assumes that the services into the registry/base have been discovered so that they satisfy the local (to the service) functional and non-functional requirements that have been specified for the choreography and, hence, can be considered as potential candidates to participate in the global choreography process. Finally, the choreography synthesis produces the coordination delegates that will be then managed by the service composition engine for choreography realization purposes presented in Section 4.2, hence accessing the participant services through the service access subsystem presented in Section 4.3.

3.3 Choreography Analyzer

Given the ultra large scale of FI choreographies, automated analysis mechanisms become necessary to support choreography evolvability. The Choreography Analysis component is mainly responsible for analyzing either a serialized BPMN2 choreography specification or the set of coordination delegates issued by the synthesis process (Section 3.2). Two kinds of analysis are currently supported and implemented in the form of subcomponents, namely choreography *Scalability prediction* and choreography *Stability and Interdependencies Analysis*. In the following, we describe each of these subcomponents. The *Scalability Prediction* relies on two mechanisms: the QoS Prediction and the Scalability Analysis. The QoS Prediction aims at estimating the behavior of the choreography (written in BPMN2.0) regarding QoS parameters such as service response time, capacity, reliability, availability of a composition, etc. The prediction takes into account the choreography execution context (the number of user requests, the number of concurrent choreographies, the available resources), but captures it in a single state. In turn, the Scalability Analysis considers various possible states of the choreography execution. It uses for this issue the QoS Prediction mechanism for single state prediction and a mathematical model describing the dynamics of changes in the choreography execution. The *Stability and Interdependency Analyzer* is primarily responsible for performing change impact analysis based on the existing dependencies between choreography participants. In addition, the component also applies the analysis to a set of concrete services and coordination delegates that realizes the choreography. The analyzer component relies on model-to-model (M2M) transformations to obtain the dependency graph from either a choreography BPMN2.0 specification or a set of coordination delegates. Finally, the analyzer relies on graph analysis techniques to calculate a variety of dependency-centric measures, including graph centralities [10] and stability.

4 CHOReOS Service-Oriented Middleware

The CHOReOS middleware targets two different but interrelated domains of services: Business services and Thing-based services. Based on this inherent characteristic, the high-level architecture of the CHOReOS middleware comprises corresponding domain-specific mechanisms that support the discovery of services, the access to services, and the execution of service compositions. The specificities of the functionality offered by the domain-specific mechanisms are hidden by corresponding unified "eXtensible" middleware mechanisms that unify the access to the domain-specific middleware mechanisms. In addition, computationally- and storage-intensive tasks of both the middleware and the choreographies are supported by the CHOReOS Cloud and Grid services. In the following, we describe the CHOReOS middleware (See Figure 3).

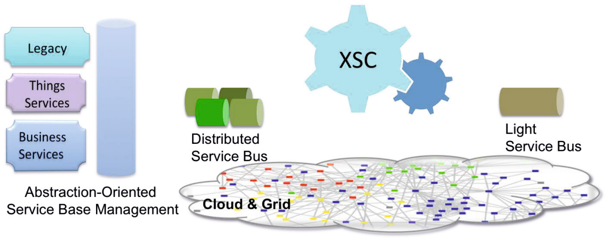

Fig. 3. CHOReOS Middleware Architecture

4.1 eXtensible Service Discovery

The CHOReOS IDRE provides a multi-protocol service discovery service. Actually, it relies on an Abstraction-oriented Service Base Management (AoSBM) [2], [1] that stores and classifies in a suitable way an important amount of services data. This base is populated by an extensible plugin-based mechanism. The latter is responsible for extending the service discovery to both business services and FI things, by plugging domain-specific discovery protocols like, e.g., the Governance Registry for Business services and the Things Discovery Protocol (TDP). The registry is populated either in a passive or active way thanks to the Plugins. Moreover, the XSD relies on Semantic Knowledge Base ontologies to enable the devices (Things) discovery. Regardless of their heterogeneity, the CHOReOS XSD provides a unique abstraction referencing services, which empowers the ability of realizing choreographies of services.

4.2 eXecutable Service Composition

Service choreographies in CHOReOS are supported by different execution platforms so as to cope with the diversity of service technologies found in a ULS environment. The enactment of a choreography is taken into account by the XSC mechanism. The latter takes as input the choreography synthesized previously by the Choreography Development Process and discussed in Section 2. A set of Coordination Delegates specifies the choreography and are then dedicated to the right XSC. Specifically, *BPEL-based XSC* enables the implementation of coordination delegates using BPEL, while *SCA-based* XSC supports the implementation of coordination delegates using SCA. In a complementary way, the *Thing-based Composition & Estimation* component deals with the composition of Thing-based services to handle requests for interacting with the physical world. While enacting a choreography, some services may not respect the initially contracted agreements and choreography reconfigurations need then to be operated. For that end, the CHOReOS XSC relies also on a reconfiguration and substitution mechanism.

4.3 eXtensible Service Access

In ULS environments, millions of services either Business or Things oriented are deployed in a distributed manner allover the ambient context. Besides the fact that they are coming from heterogeneous sources and are dedicated to different aims, they are also implemented using distinct standards and technologies. Consequently, in order to make these services collaborate within choreographies, it is essential to provide a middleware technology that unifies their access. Within CHOReOS we exploit the Enterprise Service Bus paradigm, which provides a glue technology supporting connectivity and communication techniques. Further enhancements are realized with the aim of making the CHOReOS XSA support both Business and Things services. Indeed, the XSA is based on an enhanced service bus paradigm to overcome the heterogeneity of the FI. This paradigm is represented by the *eXtensible Service Bus (XSB)*. The latter enables multi-protocol access to both Business Services of the IoBS domain and Thing-based Services of the IoTS domain, as well as cross-domain access. In particular, it enables interoperability among heterogeneous interaction paradigms of both domains, while conserving as much as possible their semantics. The XSB is an abstract bus that prescribes only the high-level semantics of the common bus protocol. This semantics follows a *Generic Application-GA-* abstraction paradigm. Moreover, the XSB relies on the *Distributed Service Bus* (DSB) [4] that provides support for accessing business services. We rely on the Petals DSB to ensure this functionality. Additionally to the native bus capabilities the DSB supports distribution and offers the core runtime middleware. The DSB is leveraged to the FI features through the adaptation to the cloud middleware. It benefits from the provided hardware resource infrastructure, in fact. Further enhancements intend to take advantage from the cloud elasticity discussed in Section 4.4. Furthermore, in order to target IoTS domain, we provide the *Light Service Bus* (LSB), which is a lightweight concrete bus realization of XSB and its GA semantics, dedicated to IoTS, hence, accounting for its dynamics and resource constraints while enabling access to heterogeneous Things. In particular, the GA semantics is conveyed on top of a substrate protocol (DPWS) that is suitable for the IoTS domain.

4.4 Cloud and Grid Middleware

The Cloud and Grid Middleware services provide basic services that support computational- and storage- intensive tasks performed either by the CHOReOS middleware services, or by the choreographies that are built on top of the CHOReOS middleware. The Cloud service can allocate and deallocate resources dynamically according to service demand. Tasks such as encoding large amounts of video in a citizen journalism application can take advantage of the Grid service. The allocation of Cloud machines for execution of choreographies is performed by the CHOReOS middleware in a way that is transparent to choreography users, designers, and developers. The CHOReOS middleware uses the *ServiceDeployer* component to allocate new nodes from the *NodePoolManager* and then deploy and run new services on them. In these nodes, CHOReOS will execute major choreography components (e.g., proxies, adapters, coordination delegates) for service access at runtime. To this end, the *EnactmentEngine* will use the NodePoolManager and the ServiceDeployer to set up the choreography environment, allocating the required nodes, deploying the required software and enabling the execution of the choreography. To achieve scalability and portability, the NodePoolManager is able to allocate new nodes in multiple underlying execution platforms. A CHOReOS node may be part of a Cloud Infrastructure as a Service (IaaS) platform; these can be provided by a public Cloud such as Amazon EC2 or Rackspace, or a private Cloud, for example, executing the OpenNebula or OpenStack open source Cloud software. The CHOReOS monitoring service will provide data to runtime QoS and V&V enforcers. If a QoS violation is detected, for example, the Cloud service can be used to allocate new nodes in an attempt to improve QoS.

5 Governance and V&V Framework

ULS choreographies bring into play a very large number of services, users and resources employing the system for different purposes. Therefore, methodologies and approaches that will permit the smooth integration of independently developed pieces of software need to be implemented. In IT Systems, the Governance approach enables supervising such large systems. Indeed, a set of processes, rules, policies, mechanisms of control, enforcement policies, and best practices are put in place throughout the life-cycle of services and choreographies, in order to ensure the successful achievement of the SOA implementation. Activities such as policy definition, auditing & monitoring, and finally evaluation & validation are recommended. Within CHOReOS, we implement a Governance and V&V Framework (See figure 4) that underly the services, and choreographies lifecycle. Precisely, the Service Level Agreement-SLA and lifecycle management deals with the lifecycle of relevant resources such as services, service level agreements, and choreographies. Further, the V&V Components perform the testing of services before their involvement in choreographies. Online testing of services and choreographies at runtime is also operated. Finally, the Test Driven Development Framework (TDD) operates a series of complementary tests.

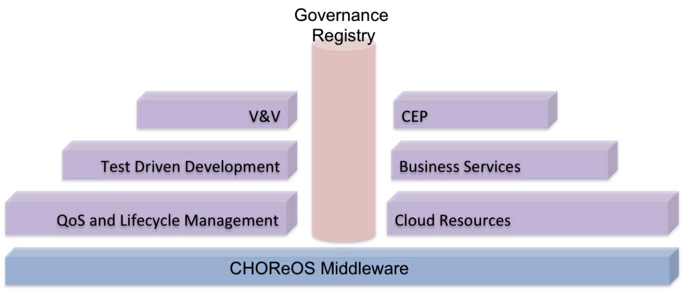

Fig. 4. CHOReOS Governance and V&V Framework

5.1 SLA and Lifecycle Management

The SLA and Lifecycle management activities [12] are responsible for offering the capabilities that ease the management of the resources, these can be services, choreographies, policies, and service level agreements life-cycles. Meanwhile, it also helps managing the roles and responsibilities of the users of the Governance Framework, by assigning credentials. Design time policies might define which, when, and where to use standards and insure compliance between them. The design time policies may also consider the fact of setting out corporate namespaces, common coding conventions, etc. When a choreography is enacted, in order to ensure its good behavior, it is ultimately necessary to enforce and manage the non functional contracts of involved services, according to defined policies. Within CHOReOS, a distributed monitoring system is envisaged in order to assess the ULS choreography properties. The SLA governance is realized by the Business Service Monitoring discussed in Section 5.4.

5.2 Verification and Validation Support

The Governance framework provided by the CHOReOS project implements a comprehensive strategy for managing both choreographies and services. The project put a special emphasis on governance aspects related to choreography-oriented V&V activities by defining policies, and rules governing (e.g., enabling, regulating, etc.) them [6]. The idea of V&V governance was originally proposed in [7] to support an on-line testing [5] session when a service asks for registration within a registry. In this vision, only services passing the testing phase are logged in the registry. As a result, the registry is expected to include only "high-quality" services that passed the validation steps foreseen by a more general governance framework. In addition to the registration of a new service, the on-line validation process could be also extended to other events, like the release of a new service version. Note that when entering a new service registration in a registry, the service provider is naturally wishful to promote the service and therefore can be explicitly willing to submit it to on-line testing. On the other hand, the notification of a service upgrade could be notified only sporadically. The governance mechanisms oriented to V&V activities could mitigate this aspect by means of specific policies and obligations that the service providers should abide by, when binding their services to a choreography. During the life-cycle of a choreography, a service that was originally registered to play a given role in such choreography could be modified or become deprecated. In addition, it is also possible that a single service may play one or more roles defined by a choreography. Finally, the same service may be involved in several choreographies with differnt roles, as well. In all these scenarios, the V&V governance rules that the CHOReOS project is proposing aim at prescribing that any modification (i.e. activation, modification, cancellation) to either a registered service, or to a role defined by a choreography should activate a new appropriate online testing session. In this sense, the Governance Registry is an important component for SOA Governance. Indeed, as described above, the CHOReOS Governance Framework enhances the canonical functionalities provided by a Service Registry with a set features supporting online testing techniques. Specifically, each feature is implemented and managed by proxing the Service Registry with a set of dedicated handlers. Such handlers are conceived as mechanisms permitting to modify the registration procedure of a service with additional functionalities. In particular testing handlers activate testing sessions on services for which a registration request, or a modification of the associated entry, is received.

5.3 Test-Driven Development Framework

The main goal of Rehearsal, the CHOReOS testing framework, is to support Test-Driven Development (TDD) of web service choreographies. Using the framework, a choreography developer can perform multiple levels of tests to guide the choreography development. TDD is performed in a testing, or offline, environment where some of the concrete services may not be available. To achieve that, Rehearsal provides mechanisms for emulating real services or a part of the choreography by using mocks, which is a well-known TDD practice. In addition, the framework provides mechanisms for applying unit, integration, and scalability testing. At development-time, services may be created or adapted to implement the choreography roles properly. Unit testing aims at validating the correct behavior of atomic services. Integration testing aims at validating the messages exchanged by the services when they are composed to implement a role. Finally, compliance tests may also be applied to verify whether a service or a composition of services plays the role correctly. Rehearsal also supports the scalability testing of choreographies. Using this feature, the developer can assess the choreographies in different scales. Through this assessment, which is performed offline, the developer can estimate the needed infrastructure aspects (e.g., instances of virtual machines allocated to a service) to assure a performance metric (e.g., response time) in the online environment. As it is a framework, Rehearsal usage, itself, does not imply TDD application. It must be composed with other classes to create a concrete and executable application. For so, a methodology is proposed to guide developers in the application of TDD in choreography development using Rehearsal. This methodology is divided in four phases: (i) Creation and adaption of atomic services; (ii) Integration of services to compose choreography roles; (iii) Integration of roles to compose the choreography; (iv) Acceptance and scalability testing. The framework provides a tool to support each of these phases. All tests written using TDD serve both as an executable specification of the choreography behavior and as a means for V&V at design time. Later, at runtime, the same tests may be used with the online system to verify the proper behavior of the choreography in the production environment.

5.4 CHOReOS Monitoring

Relevant data such as the functional as well as the non functional attributes of services are useful for the system supervision. The data from hardware resources helps the middleware to engage in reactive measurements to correct problems as they occur. However, monitoring ULS choreographies and systems raises challenging issues such as dealing with the scalability, the distribution and the heterogeneity. The CHOReOS IDRE addresses these requirements by relying on a distributed and event-based infrastructure for monitoring both Business Services and hardware resources. Finally, a Complex Event Processor-CEP-, ensures the respect of the dictated policies. Once services are deployed or exposed on the DSB, the CHOReOS Monitoring performs runtime assessment of Service Level Agreements and control of the communications taking place within a choreography, thanks to the *Business Service Monitoring*, which gathers data from the Distributed Service Bus. Communication Monitoring is achieved by subscribing to events triggered by the running services. While, *QoS Runtime Assessment* relies on the implementation of the WSDM standard. Then, the *Resource Monitoring* is a ganglia-like monitoring system that interacts with the Cloud and Grid Middleware. First, it actively supplies notifications to interested subsystems about relevant events, such as overloaded systems, out-of-memory conditions, or hardware failures. Second, it maintains an overview of the current and recent status of system resources to be able to respond to queries about them. Queries are useful to support creation or destruction of virtual machine instances according to load, services allocation, or services migration. We address the scalability and distribution issues by considering for each each CHOReOS node a local component collecting data (primarily memory, disk, and CPU usage). Then, data is aggregated between distributed nodes in a hierarchical manner.

6 Conclusion

The FI world challenges the SOA by raising scalability, distribution and heterogeneity issues. The CHOReOS project addresses these issues by providing responses at several levels. The CHOReOS Integrated and Runtime Environment gathers top-level technological SOA approaches including Model-driven Architectures, SOA Discovery, SOA Composition and SOA Governance. In this chapter, we have presented the CHOReOS platform as well as its main components. Ongoing works concern the realization of the IDRE in ULS choreography use cases.

Acknowledgments. This work is partially funded by the EU 7^{th} Framework Program, contract CHOReOS FP7-ICT-2009-5 - Objective 1.2 (grant 257178). The work presented in this paper only reflects the authors views. The Commision is not liable for any use that may be made of the information contained therein.

References

1. Athanasopoulos, D., Zarras, A., Issarny, V.: Towards the maintenance of service oriented software. In: Proc. of the 3rd CSMR Workshop on Software Quality and Maintenance, SQM (2009)
2. Athanasopoulos, D., Zarras, A., Vassiliadis, P., Issarny, V.: Mining service abstractions - nier. In: Proc. of the 33rd International Conference on Software Engineering (ICSE), pp. 944–947 (2011)
3. Autili, M., Mostarda, L., Navarra, A., Tivoli, M.: Synthesis of decentralized and concurrent adaptors for correctly assembling distributed component-based systems. Journal of Systems and Software (2008)
4. Baude, F., Filali, I., Huet, F., Legrand, V., Mathias, E., Merle, P., Ruz, C., Krummenacher, R., Simperl, E., Hamerling, C., Lorré, J.: Esb federation for large-scale soa. In: Proc. of the ACM Symposium on Applied Computing, SAC 2010, pp. 2459–2466 (2010)
5. Bertolino, A., De Angelis, G., Kellomäki, S., Polini, A.: Enhancing service federation trustworthiness through online testing. IEEE Computer 45(1), 66–72 (2012)
6. Bertolino, A., De Angelis, G., Polini, A.: Validation and verification policies for governance of service choreographies. In: Proc. of the 8th International Conference on Web Information Systems and Technologies, WEBIST (to appear, April 2012)
7. Bertolino, A., Polini, A.: Soa test governance: Enabling service integration testing across organization and technology borders. In: Proc. of Software Testing, Verification and Validation Workshops (ICSTW), pp. 277–286 (April 2009)
8. Calvanese, D., De Giacomo, G., Lenzerini, M., Mecella, M., Patrizi, F.: Automatic service composition and synthesis: the roman model. IEEE Data Eng. Bull. 31(3), 18–22 (2008)
9. Issarny, V., Georgantas, N., Hachem, S., Zarras, A., Vassiliadis, P., Autili, M., Gerosa, M., Ben Hamida, A.: Service-Oriented Middleware for the Future Internet: State of the Art and Research Directions. Journal of Internet Services and Applications 2(1), 23–45 (2011)
10. Newman, M.: Networks: An Introduction, 1st edn. Oxford University Press (2010)
11. Tivoli, M., Inverardi, P.: Failure-free coordinators synthesis for component-based architectures. Sci. Comput. Program. 71, 181–212 (2008)
12. Zribi, S., Bénaben, F., Ben Hamida, A.: Towards a service and choreography governance framework. In: Proc. of the I-ESA Conference, Valencia Spain. Springer, Heidelberg (to be published, 2012)