

Adaptable Environmental Service Chains: The Challenges of Distributed Execution and Information Collection

George Athanasopoulos, Aphrodite Tsalgatidou, Pigi Kouki, Ioannis Pogkas,
Michael Pantazoglou

Dept. of Informatics and Telecommunications, National and Kapodistrian University of Athens,
Greece

{gathanas, atsalga, pigikouki, jpogkas, michaelp}@di.uoa.gr

Abstract. Considering the current transformation of Environmental Information Systems to environmental services accessible over the web, the provision of adaptable environmental services is becoming an emerging challenge. Within this context, solutions that support the adaptation and distributed execution of service chains seem promising. In this paper we present a platform catering for the provision of data-driven adaptable environmental service chains using contextual information from external sources. Two core features of this platform, presented in this paper, are the collection of contextual information and the distributed execution of service chains. The provision of this platform is one of the goals of the EU-funded project Envision

Keywords: Environmental Service Chains, Data-Driven Process Adaptation, Distributed Process Execution

1 Introduction

The provision of adaptable service oriented processes (called service chains hereafter) is a vision pursued by several researcher communities since the onset of the Service Oriented Computing (SOC) paradigm. Despite the existing controversy on the definition of service adaptation [1][2], it can be conceived as the ability of a service chain to “adapt to changes of the process environment and/or to the modification of end-user needs” [3]. Adaptation is a higher-level goal required in numerous application domains, e.g. crisis management, e-Commerce, etc. In this paper, we examine adaptation from the point of view of environmental applications and systems.

Environmental Information Systems (EIS) are currently shifting towards the SOC paradigm and transforming into environmental services, e.g. spatial data services. Several initiatives and directives are pushing this transformation, e.g. INSPIRE (DIRECTIVE 2007/2/EC), GMES (Global Monitoring for Environment and Security) or SEIS (Shared Environmental Information System). In this framework, the need for

adaptable environmental services is being pushed forward by the continuous emergence of more and more environmental service-oriented systems.

Environmental systems normally rely upon the use of several information elements that correspond to the parameters of the associated environmental models. The required information elements may stem from numerous information sources available in the web or privately managed networks, e.g. weather forecasts provided by open sites or privately managed sensor data and historical information data sources. Additional sources are also expected when considering the transformation of the web into the “Internet of Things” vision [4]. Therefore, the collection of information related to a system should be performed in an open manner, i.e. a manner where the set of sources is not rigidly specified and controlled. Moreover, environmental systems consist of tasks that necessitate either the use of large volumes of computational resources or the manipulation and exchange of large sets of information elements. Distributed execution (and parallelization whenever feasible) has previously been identified as an appropriate means to accommodate this need. Service orientation supports the distribution of comprising tasks, but the use of a centralized model for managing the execution of service compositions is a bottleneck for the exchange of large volumes of information. Therefore, the control of service compositions should be performed in a decentralized manner catering for the exchange of large volumes of information, i.e. in a manner where several nodes, rather than a single server coordinate the execution of a service composition.

Existing approaches towards the provision of adaptable service processes are either focusing on the adaptation and centralized execution of service chains consisting of single types of services, i.e. web services, or on the use of pre-specified and well adaptation of service chains through process reconfiguration or via the use of additional services. Most of them are able to support neither, the integration and adaptation of service chains, which comprise distinct types of service and information sources, nor their distributed execution.

All these needs, i.e. large volume of exchanged information, decentralized control of service compositions, integration of distinct types of service and information sources, call for novel approaches that are able to facilitate the provision of adaptable service chains via the use of available information and services. In this paper we outline an approach supporting the adaptation and distributed execution of environmental service chains based on information collected from several sources. This approach is part of the Envision project [6], which provides an environmental services infrastructure with ontologies that aims to support non ICT-skilled users in the process of semantic discovery and adaptive chaining and execution of environmental services. In the following, we briefly present the approach ensued in Envision and then we elaborate on the mechanisms used for the collection of information from several sources, and the distributed execution of environmental service chains. We conclude this paper with a comparison of our approach and mechanisms against similar approaches and a summary of our work and future work plans.

2 Data-Driven Service Chain Adaptation

The prime assumption of our work is based on the observation that a service chain, comprising heterogeneous services, should be able to utilize the information available within its environment and adapt its execution accordingly. To facilitate the provision of such adaptable processes we propose an approach that leverages information contained within a specific ‘space’ in order to adapt a service process using appropriate adaptation algorithms. Information within the context of this paper refers to structured data that have associated meta-information elements; the latter attribute meaning to structure data as well as spatial and temporal features e.g. validity time, which is dictated by the information providers. A space is considered to be the process environment, which is open to other processes and systems, e.g. Agent based systems and Sensor networks, for information exchange.

Fig. 1 graphically illustrates the proposed platform with the comprising components which are:

- A **Semantic Context Space Engine (SCS Engine)** responsible for the collection and handling of contextual information.
- A **Service Orchestration Engine (SO Engine)** responsible for the distributed execution of service processes.

The above two components constitute the **Execution Infrastructure** and are described in the following section.

The third important component of the proposed platform is:

- A **Process Optimizer (PO)** responsible for the adaptation of service processes based on the collected information. The Process Optimizer component implements an AI planner that facilitates the discovery of process plans that control the execution and adaptation of service processes. Additional details on the theoretical basis of the process adaptation mechanism are provided in [7].

The proposed approach and the illustrated mechanism (see Fig. 1) are rather generic and applicable to several application domains. In the case of EIS additional requirements related to information collection and exchange need to be addressed. Specifically, further to semantic properties, information manipulated in an EIS is inherently associated to spatial and temporal characteristics. Another important aspect is that of the volume information. EISs usually comprise activities that handle information, which may range from simple text messages measurable into few Bytes to vector or raster images measurable into several GBytes. In this context, the presented approach has two main advantages. The data-driven adaptation aims to reduce the number of tasks (thus also the number of message exchanges) performed in a service chain, provided that the information related to these tasks is available during execution. Similarly, the distributed execution of service chains can alleviate the delays caused by the transfer of large volumes of information from their original place to the execution engine, and can reduce the processing time of heavy tasks by exploiting untapped resources residing at underutilized nodes.

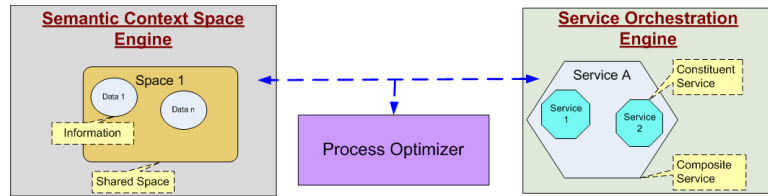


Fig. 1. High level architecture

In the following we present additional details on the two prime components catering for the collection of information elements and the distributed execution of service chains.

3 Execution Infrastructure

As we have stated before, the SCS Engine and the SO Engine constitute the execution infrastructure of our approach and are responsible for addressing the challenges of information collection and distributed execution respectively.

Issues related to providing an open platform for the collection of information from several sources and accommodating spatial and temporal characteristics are highly important for the accurate selection of information relevant to a specific environmental service chain. Along the same lines, supporting the distributed execution of service chains is important for overcoming potential bottleneck problems emerging from the exchange of large volumes of information and for enhancing the throughput of the execution infrastructure. Therefore, by addressing the challenges of information collection and distributed execution of service chains, we are also facilitating the performance and precision characteristics of the provided mechanism.

The prime design decisions used for addressing these requirements include the use of the TupleSpace paradigm[8] [9] for building the SCS Engine and the use of a Peer-to-Peer (P2P) architecture for the SO Engine. In the following we provide additional details on the properties of these two components.

3.1 Semantic Context Space (SCS) Engine

The main goal of the SCS Engine is the provision of an open platform for data acquisition, supporting the collection and sharing of information elements; the latter refer to semantically annotated structured data. From a functional point of view, the SCS Engine leverages clients to write, retrieve and to logically group information. The JavaSpace service of the Jini Framework [10] is serving as the underlying implementation basis. Extensions to JavaSpace are needed to satisfy requirements related to annotation with meta-information. In addition, extensions are also needed to support the logical grouping of information into “information scopes” of similar elements as well as for specifying affiliations among related information scopes.

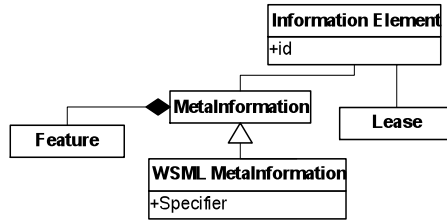


Fig. 2. Core Elements of the Information Model of the SCS Engine

As we already mentioned, the role of the SCS Engine is to collect semantically annotated information from external sources. Representative external sources are data providing services, databases, and/or user’s input. For the semantic description of the inserted data we use WSM-based annotations [11]. GML[12] primitives such as spatial and temporal properties are used for appending spatial meta-information to the collected information. Fig. 2 graphically illustrates the form of a typical information entity stored in the SCS Engine. The main attributes are: the unique identifier of each information element; a class named *Lease* which adds temporal properties, as it represents a fixed period of time in which the information element is considered to be valid; and a class called *MetaInformation* that is responsible for keeping all the semantic information related to the information element. The *MetaInformation* is annotated with semantic attributes, e.g. WSM attributes, and comprises *Features*. These *Features* are irrelevant to the feature elements of GML, and provide a generic container for holding implementation related properties.

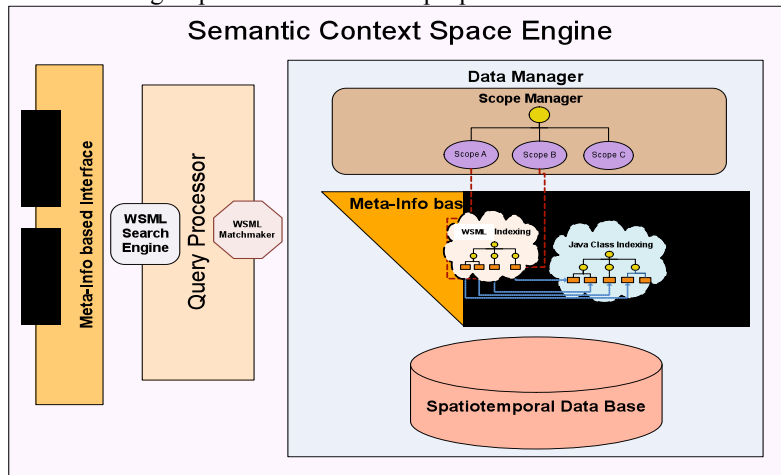


Fig. 3. SCS Engine architecture

Fig. 3 depicts the plug-in based architecture of SCS Engine. The comprising components include:

The **Data Manager** consists of the Scope Manager, the Meta-Info Based Index Tree, and the Type-based Index Tree. This element is responsible for the storing,

maintaining, and indexing of the provided information. As it can be seen in Fig. 3 a spatiotemporal database is used to manipulate spatial and/or temporal extensions.

The **Query Processor** is responsible for the discovery of information within a space. It does so by relying on a WSML-based search engine and an appropriate WSML-based matchmaker that will be developed.

The **Meta-Information based Interface** implements all interaction points of the SCS Engine with its environment. The accommodated API is accessible through a Java and a Web based interface and provides for executing two types of operations: content related and scope management operations.

3.2 Service Orchestration (SO) Engine

The purpose of the SO Engine is to adapt (if needed) and to execute service chains. To do so, it accepts as input extended BPEL process specifications provided by the PO. These specifications may incorporate a set of adaptation steps, which define how the engine can adapt the execution of a service chain based on information available at the SCS Engine. Internally, the SO Engine consists of: a) extensions to the ODE (www.apache.org) runtime engine and b) a P2P overlay, with peers responsible for the BPEL activities execution. The ODE engine's compiler is used in order to parse the BPEL process specification and to extract an execution plan, consisting of tasks that correspond to BPEL activities. In the sequel, these tasks are assigned into fine-grained agents, which are responsible for the particular BPEL activities. Our implementation uses JXTA (www.jxta.org) P2P technologies as the implementation basis. Each node of the p2p infrastructure corresponds to an agent for the execution of BPEL activities.

The P2P overlay of the SO Engine is organized into peer groups where each group is responsible for a BPEL activity. The group members must have efficient computation and network capacity in order to execute the assigned activities, but there is no need to have high uptime. In our implementation, every resource is described with an appropriate JXTA advertisement and the peers communicate by exchanging messages using JXTA services. Also, every group has a group organizer that handles the load balancing, network congestion, and peer availability problems by selecting the peer member that will execute a particular BPEL activity. A group organizer must have certain qualities such as high availability and uptime, efficient network capacity and computational resources. If a group member is not responding or if it declares lack of ability to handle a request, the group organizer starts a re-deployment phase and selects another member of the group to handle that request. In case of a group organizer fault, another node is picked up to become the next group organizer, using a leader selection algorithm [13].

A typical execution example of the specified orchestration engine is presented next.

3.2.1 Execution Case Study

Our case study implements an oil spill scenario that is concerned about the impact of oil spills on cod populations. It consists of two models [14]: a) the Predict Oil Drift model, which models the oil distribution in a three-dimensional water body that predicts the trajectory of the oil cover to assess when the oil reaches the coastline and protected areas; and, b) the Predict Cod Effects model, which estimates the toxicity on cod populations. The Predict Oil Drift model uses input from several services, such as a weathering model to assess the impact of current weather and sea conditions on the oil's chemical composition, and accordingly its behavior in seawater. Additional information related to the coastline model and the sea level, are provided by corresponding services. The results of the oil drift prediction serve as input for Fig. 4 depicts the BPEL process for this scenario and available services (Weather and Sea Observations, Coastline, Bathymetry, etc); as it can be seen, the process consists of a receive activity, a reply activity, and a sequence activity with one assign and two invoke sub-activities.

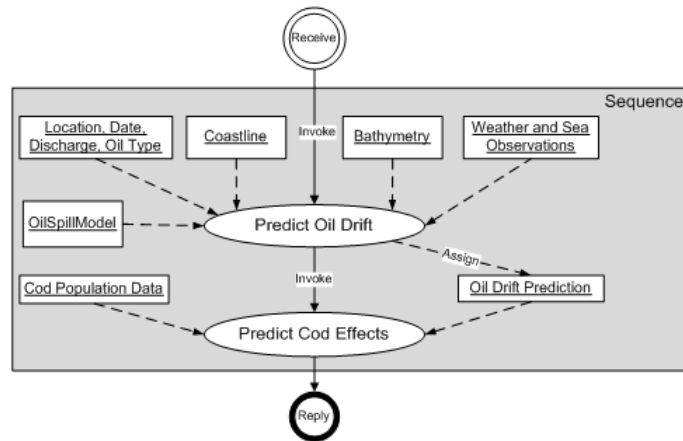


Fig. 4. BPEL process

Fig. 5. As we see in this figure, the execution plan is sent to peer R1 that is capable of handling the receive activity. R1 handles the incoming message and writes the output to the variable *OilSpillModel* via its resource handler using an advertisement. Afterwards, it propagates the execution plan with the remaining operations to node S, which is responsible for the sequence activity. Node S creates a number of subtasks. The first subtask is the invocation of *PredictOilDrift* activity. For this reason, node S propagates the execution plan to the node I1, which invokes the above service, writes the service output to variable *out1*, and propagates the execution plan further to node A1. Node A1 finds the advertisement for the variable *out1*, reads the value of variable *out1*, and assigns its value to variable *OilDriftPrediction* by publishing an updated advertisement. A similar process continues until the execution plan returns back to node S. At the final step, the execution plan is propagated one more time from node S to node R2, which is responsible for the reply activity. Node R2, using its resource

handler, reads *out2* variable, constructs a reply message, and returns the result to the ODE runtime.

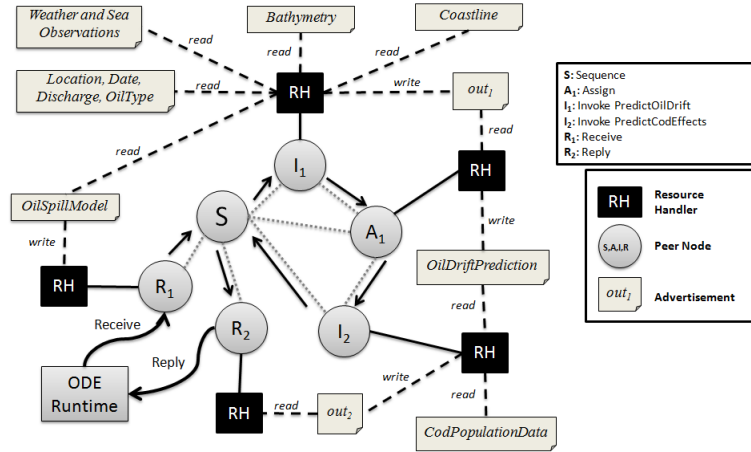


Fig. 5. Execution example

4 Related Work

The research fields that are inherently associated to the work presented in this paper are those related to service process orchestration and semantic-based tuplespace computing.

The most common approach for process execution is the deployment of a single centralized orchestration engine, while scalability is achieved by replicating the engine and using load balancing algorithms. This approach is currently followed by most engines like Apache ODE (www.apache.org), JBOSS jBPM (www.jboss.org), IBM WebSphere (www.ibm.com/software/websphere) and Microsoft BizTalk server (www.microsoft.com/biztalk). In this paper we present a different approach where individual activities within a process are distributed among the available computing resources, thus allowing the activities to be placed near the data they operate. An approach similar to ours is the one in NINOS [15]. The differences between our approach and the latter one have as follows: a) NINOS requires the use of additional brokers to be deployed to handle the publish subscribe operations while we use the mechanism provided by the JXTA, b) our orchestration engine is able to adapt process execution, while NINOS doesn't and c) our engine follows a more fine-grained approach for distributing resources to agents, by using advertisements for all the shared variables and resources. To the best of our knowledge there exists no other similar approach catering for the execution and adaptation of service processes in such a distributed manner.

With regards to existing approaches catering for the provision of semantically enhanced Tuplespaces, most of them (e.g. sTuples, Conceptual Spaces, and Triple Space Computing [16]) can be considered as knowledge bases that utilize RDF for the

representation of semantic information. Contrary to those, the SCS Engine provides an extendable Meta-Information model that can support various meta-information schemes for the annotation of Information Entities. Further to that, the SCS Engine accommodates a mechanism for the logical grouping of information, i.e. scopes, that differs from the scoping mechanism specified by Merrick [17] or the one used by the Semantic Web Spaces [18], in the use of a flexible affiliation mechanism that can support the expression of various types of relationships among scopes.

5 Conclusions

The provision of adaptable environmental service chains is emerging as an important challenge in the field of environmental services and applications. This field poses significant additional concerns to the respective approaches applied in the SOC domain (e.g. to adaptable service processes). Two aspects of significant importance are the collection of information that has semantic, spatial and temporal annotations, from several external sources, and the distributed execution of environmental service chains, which comprise the exchange of large volumes of information.

These two aspects have been taken into consideration for the design of the service chain execution infrastructure being developed in the Envision project that has been presented in this paper. Despite the constraints of our approach in terms of semantic annotations needed for the description of information elements and the required capabilities of the peers of the orchestration engine's p2p overlay, we can claim that it can satisfy the requirements imposed by the environmental service application domain, i.e. collection of information that is semantically, spatially and temporally annotated from several sources and the exchange of large volumes of information. Moreover, the provided mechanisms and the employed models have been designed in an open and extendable manner so as to support their extension with additional features. For example, the SCS Engine can be enhanced with a higher level logic-based reasoner catering for the extraction of knowledge out of collected information.

Additional issues that have been considered include security and deployment considerations. With respect to security, issues related to accessing of information spaces can be handled through the assignment of appropriate roles and credentials. Nevertheless, additional issues such as the collection of erroneous information from malevolent sources deserve further investigation. Similarly, regarding the deployment of the whole infrastructure, all components have been designed in an independent manner catering for their distribution over distinct nodes accessible through the net. Their interaction is supported over the use of appropriate web based protocols and standards, e.g. web services and http based interactions, so as to avoid network hindrances, e.g. firewalls and NATs.

We need to state here again that the implementation of the presented approach and of the comprising components is still not completed. Therefore, our future work plans include the finalization of the component implementations and the introduction of additional features and properties to them, e.g. the use of other languages and mechanisms for the description of semantic, temporal and spatial annotations. Last

but not least, one of our future objectives is to provide measurable evaluations for the infrastructure as a whole as well as for each of the comprising components.

6 References

1. Brogi, A., Popescu, R.: Service adaptation through trace inspection. *Int. J. Business Process Integration and Management* 2(1), 9-16 (2007)
2. Moser, O., Rosenberg, F., Dustdar, S.: Non-intrusive monitoring and service adaptation for WS-BPEL. In : *Proceeding of the 17th international Conference on World Wide Web*, New York, pp.815-824 (2008)
3. Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., Shan, M.-C.: Adaptive and Dynamic Service Composition in eFlow. Technical Report HPL-2000-39, Software Technology Laboratory HP Laboratories Palo Alto, Palo Alto (2000)
4. Commission of the European Communities: Internet of Things — An action plan for Europe., EU (2009)
5. ENVISION: Environmental Services Infrastructure with Ontologies. (Accessed Jan 2010) Available at: <http://www.envision-project.eu/>
6. Athanasopoulos, G., Tsalgatidou, A.: An Approach to Data-Driven Adaptable Service Processes. In : *Proceedings of the 5th International Conference on Software and Data Technologies*, Athens
7. Rossi, D., Cabri, G., Denti, E.: Tuple-based technologies for coordination. In : *Coordination of Internet agents: models, technologies, and applications*. Springer-Verlag, London, UK (2001) 83--109
8. Gelernter, D.: Generative communication in Linda. *ACM Trans. Program. Lang. Syst.* 7(1), 80-112 (1985)
9. Waldo, J., Team, J.: *The Jini™ Specification*, 2nd Edition. Addison-Wesley Professional (2000)
10. ESSI WSMML working group In: *Web Service Modeling Language*. Available at: <http://www.wsmo.org/wsmml/index.html>
11. Portele, C.: *OpenGIS® Geography Markup Language (GML) Encoding Standard*. OpenGIS standard, Open Geospatial Consortium Inc. (2007)
12. Xia, S.: Optimal Leader Election Scheme for Peer-to-Peer Applications. In : *International Conference on Networking Sixth International Conference on Networking (ICN'07)*, Sainte-Luce, Martinique, France, pp.29-29 (2007)
13. Reed, M.: Quantitative analysis of alternate oil spill response strategies using OSCAR. *Spill Science & Technology Bulletin* 2(1), 67–74 (1995)
14. Jacobsen, G.: A distributed service-oriented architecture for business process execution. *ACM Trans. Web* 4(1), 1--33 (2010)
15. Nixon, L., Simperl, E., Krummenacher, R., Martin-recuerda, F.: Tuplespace-based computing for the semantic web: A survey of the state-of-the-art. *Knowl. Eng. Rev.* 23(2), 181--212 (2008)
16. Merrick, I., Wood, A.: Coordination with scopes. In : *ACM Symposium on Applied Computing*, pp.210-217 (2000)
17. Tolksdorf, R., Paslaru Bontas, E., Nixon, L.: Towards a tuplespace-based middleware for the SemanticWeb. In : *IEEE/WIC/ACM International Conference on Web Intelligence WI2005*, pp.338-344 (2005)

