Data Driven Adaptation of Heterogeneous Service Oriented Processes

Georgios Athanasopoulos[*]

National and Kapodistrian University of Athens
Department of Informatics and Telecommuncations
gathanas@di.uoa.gr

Abstract. Within the currently forming pervasive computing environment services and information sources thrive. Instantiations of the service oriented computing paradigm e.g. Web, Peer-to-Peer (P2P) and Grid services are continuously emerging, whilst information can be collected from several information sources e.g. materialisations of the Web 2.0 and Web 3.0 trends, Social Networking apps and Sensor Networks. Within this context the development of adaptable service oriented processes utilising heterogeneous services, in addition to available information is an emerging trend. This paper presents an approach and an enabling architecture that leverage the provision of data-driven, adaptable, heterogeneous service processes. Core within the proposed architecture is a set of interacting components that accommodate the acquisition of information, the execution of service chains and their adaptation based on collected information.

1 Dissertation Summary

Our era has been marked by a shift in the way of thinking and acting across many domains such as business, science and community. Cornerstone in this new way of thinking is the notion of service. In spite of the lack of a commonly accepted definition a service can be conceived as a function that is offered by someone and may be used by anyone else, or as Douglas Barry sets it more formally: *a function that is well-defined, self-contained, and does not depend on the context or state of other services* [1].

This shift in thinking and acting has an effect on software systems and the way they are developed. Service Oriented Computing (SOC) focuses on the use of services as system constituent parts. It is regarded as an evolution to the Component-Based development and distributed object oriented computing, and has been widely accepted as the current and future trend in distributed system development. Among its goals is to promote the loose coupling and flexible integration of the system parts in a far better way than component and object oriented technologies do.

[*] Dissertation Advisor: Aphrodite Tsalgatidou, Associate Professor

1.1 Service Oriented Process Adaptation

Service Oriented Processes (SOPs) constitute an indicative materialization of the loose coupling notion of SOC, as they rely on the lenient integration of comprising services. They are normally defined in higher-level languages, e.g. BPMN [2], WS-BPEL [3], and provide descriptions of coordinated flows of constituent (atomic or compound) services. One of their prime characteristics that has contributed to their proliferated usage is their easy execution by contemporary orchestration engines, e.g. Apache ODE [1].

Even though, SOP development is regarded as an evolution of Enterprise Application Integration approaches [4], it is also a new paradigm that is referred as Mega-Programming [5]. An inherently assigned characteristic, stemming from this consideration, is that processes are expected to be long-running and stateful as well as they may involve the interaction with stateless or stateful services. In the frame of our approach, SOPs are regarded as systems, which operate within a specific environment, and perform pre-specified activities, via the use of external services, in an orderly manner producing and/or consuming related information during their execution.

Nonetheless, in the currently forming Pervasive Computing environment, resources such as services and information sources emerge with an increasing pace. The Service-Oriented Computing (SOC) model has been instantiated by several distinct paradigms, e.g. Web, P2P, OGC services, and a plethora of service instances are emerging every day. In addition, the emerging Sensor Web [6], and the materializations of the Web 2.0 [6] and Web 3.0 [7] paradigms, e.g. Social Networking applications, provide new types of information sources. In this context, rendering SOPs able to tap onto these resources is becoming a necessity rather than an option. These resources could be used for the adaptation of SOPs with the goal to optimize their execution. Hence, processes should be flexible enough in order to facilitate the use of services that have not been identified at design time, irrespectively of their type, and the use of information that may stem from emerging sources.

This notion of process adaptation differs from contemporary definitions, e.g. Cassati et al [8], in that process adaptation should also support process optimization. Hence, the benefits acquired by process adaptation should be measurable in terms of specific indicators (criteria). The list of criteria that can be used for the optimization of SOPs may comprise cost, execution time, throughput, etc.

In the scope of our research we argue that process adaptation should be used for the optimization of the process execution time. An approach to achieve this optimization is via the reduction of unnecessary process activities. Considering the properties of SOPs, i.e. long-running activities, it is plausible to expect that the reduction of unnecessary process activities can lead to smaller execution times and as a result to higher throughput.

[1] `http://ode.apache.org/`

1.2 Background

Nonetheless, when looking into contemporary approaches for the provision of SOPs (dynamic or not) we see that they fail to accommodate all the aforementioned constraints, i.e. reuse of several types of services and information sources. For example process specification standards, e.g. WS-BPEL [3], or similar approaches, e.g. JOpera [9], are incapable of supporting the provision of adaptable processes. Even though they provide some form of flexibility, e.g. late-binding, or few of them the ability to accommodate heterogeneous types of services JOpera [9], they are capable neither to incorporate services, whose interface is not prescribed at the process design time, nor to use information from undefined sources.

More advanced approaches such as AI-based techniques, have been extensively applied in the provision of dynamically composed service oriented orchestrations [10]. Most of them focus on the provision of automatically constructed orchestrations (or similarly called task plans) that comprise Web services solely and neglect the information that may exist in the process environment, unless this is accessible through services. Along the same lines, the Context Aware Computing (CAC) research community has invested considerable efforts towards the use of contextual information for the adaptation of web service compositions, e.g. [11]. However, the majority of these approaches fail to address the interoperability concerns raised by the multiple instantiations of the service oriented computing paradigm [12].

Modern approaches, such as the ones based on Aspect Oriented Programming, e.g. [12], can provide the means for the adaptation of SOPs at runtime through the weavering of appropriate aspects. Nevertheless, the specification of hooks where these aspects will be integrated as well as the specification of the actual aspects themselves is an arduous task that should be performed by the process developer. In addition, when considering the vast amount of alternative services and/or information sources that could be incorporated in a process, the provision of the required aspects is becoming a daunting task.

Within this frame, the development of adaptable service processes, comprising heterogeneous services and information stemming from existing sources, is an emerging trend that calls for novel approaches.

2 Data Driven Adaptation

Our proposal to the specified requirements and research challenges is a set of middleware and tools catering for the provision of Data-Driven Adaptable Service Oriented Processes (DDA-SOP) that comprise heterogeneous types of services. The DDA-SOP approach is based on the combined use of a) the Tuplespace paradigm [13], for the collection and exchange of information with unanticipated information sources, and b) AI planning techniques [14], for the discovery of alternative execution paths that can exploit the collected information for the adaptation of a given SOP.

The prime assumption of our approach is based on the observation that *a SOP, comprising heterogeneous services, should be able to use the information available within its environment and adapt its execution accordingly*. In this frame, a process state is not solely depending on the values of its internal parameters, on the resulting outcomes from the invocation of its constituent services and/or on its internal operations but, also on information pertaining to the process environment, i.e. the process space; therefore, the latter should be taken into account during process development and execution.

The proposed approach (see Fig. 2) is implemented by a set of tools and middleware that accommodate the following three basic functional needs: Collection of contextual information; Execution of heterogeneous service processes; Process adaptation driven by collected information. More specifically these components are as follows:

- The *Semantic Context Space engine* (SCS engine) provides an open space where external information sources may place relevant information; in our context relevant information refers to semantically related information elements, which are structured and affiliated to concepts of a domain ontology.
- The *Process Optimizer* implements an AI planner that facilitates the discovery of process plans, which control the execution and adaptation of service processes at runtime.
- The *Service Orchestration engine* provides a BPEL-based engine that facilitates the execution of heterogeneous service orchestrations, whilst in parallel accommodates the monitoring and reconfiguration of processes according to the suggestions of the Process Optimizer.

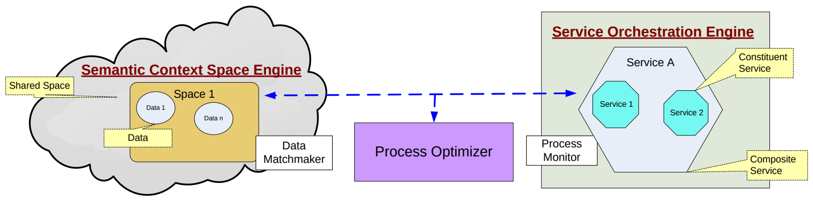

Fig. 1. Figure - High-level architecture

The contributions of this work can be briefly summarized into the following:

- An approach for mapping the problem of Data-Driven Adaptation of Service Oriented Processes to an extended, non-deterministic planning problem. To accommodate this, we use observations as the means for checking both the result of services invoked throughout the process and of the process environment, i.e. the space associated to a SOP. Details of the observed properties are used for defining queries that are executed on the associated space.

- Appropriate algorithms for extending the definition of a non-deterministic planning problem description with the inclusion of related observations and activities. The proposed approach relies on the use of ontologies and similarity measurements for the extrapolation of an original set of observations on a given SOP. In essence, our approach introduces additional related sensors for monitoring extra properties of a given SOP.
- A Generic Service Model capable of supporting the specification of common as well as distinct characteristics of various types of services. This service model enables us to use services irrespectively of their type in a seamless way. Distinct features, e.g. spatial and temporal characteristics of OGC services, provide us with required properties for customizing our approach, e.g. spatial and temporal properties are used in querying for related information as well as for discovering services.
- An environment catering for the provision of Data-Driven Adaptable Service Oriented Processes comprising heterogeneous types of services. The provided set of tools accommodates all functional needs of our approach, i.e. starting from the optimization of a given SOP model with the inclusion of appropriate adaptation steps, to the collection and exchange of related information with a SOP running on the service orchestration engine.

In addition to this list of outcomes our approach also contributed to the:

- Design and implementation of an engine facilitating the collection and exchange of information annotated with several types of meta-information. This engine is an open source, extendable implementation of a Tuplespace service that includes extensions for the collection and exchange of information, which is annotated with semantics, e.g. RDF-based, or WSML-based, as well as spatial and/or temporal meta-information.
- Specification of a Peer-to-Peer Service Definition Language, namely PSDL, with groundings for the description of JXTA services. PSDL is a WSDL-based [**?**] language with extensions for the description of Peer-to-Peer services.
- Design and implementation of a middleware catering for the dynamic invocation of Peer-to-Peer services described in PSDL. This middleware is an extension to the Web Service Invocation Framework [**?**] that uses PSDL service descriptions for the static and dynamic invocation of Peer-to-Peer services.

2.1 Process Optimization

Both the Semantic Context Space engine and the Service Orchestration engine can facilitate the provision of service-oriented processes, which are capable of exchanging information with external sources/actors. Nevertheless, the provision of processes that are able to incorporate additional behaviors, i.e. in terms of services, and information that is related to a given process, is facilitated via the Process Optimizer.

Building on the representation of the automated process composition problem as a non-deterministic, partially observable planning problem (see Def. 1) we can reuse existing planners for the discovery of policies, i.e. conditional plans, that guide the execution of a process for achieving a set of defined goals [15]. Conditions in such plans, expressed as if-then-else structures, assert the values of specific observations and decide on the execution of specific sets of actions.

Definition 1. *A non-deterministic, partially observable system (Σ) can be defined as a tuple $\Sigma = \langle S, A, R, O, X \rangle$, where:*

- *S is the finite set of states of the associated state transition system,*
- *is the finite set of actions $A = \{a_i | i \leq n \wedge i, n \in \mathbb{N}\}$*
- *$R \subset S \times A \times S$ is the transition relation,*
- *O is a finite set of observation variables $O = \{o_i | i \leq n \wedge i, n \in \mathbb{N}\}$*
- *$X_O : S \times \{\top, \bot\}$ is the relation for the evaluation of observation variables $o \in O$ on each state $s \in S$. Within our context the value of an observation variable is independent of the action that may have preceded*

In the frame of such planning problems the transition relation R maps the execution of an action $a \in A$, on a state $s \in S$ (assuming that a is applicable on s) to possibly more than one successor states i.e. $S' \subseteq S, \quad |S'| \geq 1$. An action a is applicable on a state $s \in S$, if and only if, there exists a state $s' \in S$ such that $R(s, a, s')$ stands. The set O contains the finite set of observation variables o_i whose values are evaluated at runtime. The value of each observation variable at each state is defined by the observation relation X. A simplification normally introduced to avoid the unnecessary complexities and to render the planning problem finite, is to consider observation variables as boolean variables whose values could be either true or false (i.e. $\{\top, \bot\}$). Therefore if $X_o(s, \top)$ holds at a state $s \in S$ then the value of variable o at state s is *True*. The dual holds in cases where variable o is *False*. In cases where both $X_o(s, \top)$ and $X_o(s, \bot)$ hold, variable o has an undefined value.

Nevertheless, even though this representation (Def. 1) is capable of supporting the non-deterministic and partially observable nature of services, it needs additional extensions so as to accommodate the requirements of data-driven adaptable processes [16]. Briefly the limitations of this representation are that a) it cannot consider information, which comes from sources that are not prespecified, i.e. information that is neither produced by the interacting services, nor the process at hand, and b) it fails to consider information that is related, but not exactly what is expected by the process. To accommodate these concerns we provide appropriate extensions, which consist of a) additions to the set of observations (O) and the set of actions (A) so as to leverage the consideration of related information, and b) a mechanism facilitating the valuation of observations (i.e. observation variables) based on queries executed over an open set of information elements that can be collected from external sources.

These two extensions are implemented by the Process Optimizer and the Semantic Context Space engine respectively. Although both extensions are of equal importance for the provision of data-driven adaptable processes, in the following we elaborate on the extensions of the observation and action sets.

2.2 Process Expansion

A crucial step in the provision of Data-Driven Adaptable Service-Oriented process is the introduction of extensions to the observation and action sets of the planning problem (Def. 1). This step can be regarded as *the incorporation of additional sensors for monitoring the process, along with appropriate actions for handling the accruing believes.* Nevertheless, this expansion processes should be guided so as to avoid the introduction of an overwhelming set of observations and actions that will make the planning problem practically intractable.

Starting with a WS-BPEL specification, called D hereafter, the first step is to identify the initial set of observations and actions associated to the process at hand. With regards to the initial set of observations, this should maximize the coverage of process states, whilst the initial set of actions should include the services associated to the process along with the process's internal activities, e.g. assign activities.

Similar to other existing approaches, e.g. [17], the most appropriate candidate for monitoring, i.e. assigning observations, is the set of interacting parties. Interactions with external service providers can be modeled via the exchange of messages over a set of predefined channels [17]; read and write operations performed over these channels are the means for controlling the state of the process. In this context, the initial set of observations includes observation properties, which monitor the outcomes of the services interacting with the process at hand. It is important here to note that in the frame of our work we do not consider observations for service interactions performed inside loop control structures. This is because such a consideration would entail an (explicit or implicit) ordering on the set of observed properties so as to avoid consistency errors, e.g. using observed information that is for a consequent execution step and not for the current.

Given the set of original observations (O) and actions (A) along with the related states (S), transition relations (R) and observation valuations (X), extracted from the provided process model D, the steps of the model expansion process include a) the semantic-based extension of the observation set, b) the extension of the action set with actions capable of supporting the exploitation of the introduced observations and c) the consolidation of the extended action and observation sets. In the following we provide additional details on each of the comprising steps of the expansion process.

2.3 Observation Set Expansion

The underpinning assumption which drives the observation expansion is that *instead of considering partially matching results to the performed observations we may as well look for exact matches to partially matching (i.e. similar) observations.* Hence, the set of observations $_D$ linked to D is expanded with the introduction of similar candidate observations.

Definition 2. *An observation property* $o \in O$ *can be defined as a tuple* $o = \langle name_o, vc_o, to \rangle$ *, where:*

– *$name_o$ is the name of the observation,*
– *vc_o is the semantic concept associated to the given observation and*
– *t_o is the syntactic type, i.e. specifies the related domain of values for the observation o.*

To facilitate the expansion we introduce two additional features, i.e. an expansion function (exp_o) and an expansion ratio property (r_{exp}). The r_{exp} property dictates the minimum similarity distance (i.e. the cut-off value) between the concepts of an ontology so as to consider them part of the same set (i.e. expansion set). Thus, given an ontology (V), a concept (vc) and an expansion ration (r_{exp} the exp_o function (see 1) returns all concepts of the provided ontology with similarity to vc equal to or greater than r_{exp}.

$$exp_o : \mathbb{R} \times V \rightarrow 2^V \tag{1}$$

For the calculation of the expansion function (1) several similarity distance measurements can be used. Such a measure providing an indication of similarity between two concepts, i.e. $vc_a, vc_b \in V$ based on the IS-A relation hierarchy, is Dices coefficient. We need to state here that our approach is capable of supporting additional measures through the inclusion of appropriate plug-ins.

Given the set of original observations O_D that are semantically associated to the concepts of an ontology V, applying exp_o generates a set of extra concepts. If O_V is the set of concepts assigned to the observations in O_D, i.e. $O_V = \{vc_i | vc_i = vc_{o_i}, \forall o_i \in O_D\}$, then O'_V contains the additional concepts introduced through the expansion process (2).

$$exp_o\left(r_{exp}, O_V\right) = O'_V \tag{2}$$

This set of extended concepts (2) can be regarded as a set of candidate observations to be considered by the process D. However, this set of candidate observations is partially defined, as O'_V contains solely the semantic concepts (vc_i) of the extended observations. Candidate observations should be refined so as to specify their syntactic types (t_i) and names as well. Moreover, the set of candidate observations should be pruned from observations (i.e. semantic concepts) that cannot be handled by available actions. These refinements are presented next.

2.4 Action Set Expansion

The set of actions (A_D) originally specified for a process (D) point to service operations that are already defined in the process specification. This set of actions should be extended with the inclusion of additional actions, i.e. service operations, that will be able to use the extended set of observations. To facilitate this extension we introduced a service query engine and appropriate heuristics. The prime goal of this engine is to discover alternative service chains (Sc) comprising one or multiple actions (i.e. service operations) that can use as input the extended observations and lead to the achievement of the process goals or sub-goals.

Definition 3. *Sub-goals S_{sg} in the context of this paper refer to intermediate states of the specified process D, i.e. $S_{sg} = S_D - (S_0 \cup G)$.*

The service query engine uses as input the merged set of observation concepts, i.e. $O_V \cup O'_V$, the set of process states (S_D), and the set of process goals (G). The outcome of the discovery process is a service chain Sc, which satisfies the specified search objectives; a service chain Sc (see 3) is typically defined as a finite, ordered set of actions (i.e. service operations).

$$Sc = \{a_0 \prec a_1 \prec \cdots \prec a_n,\ and\ n \in \mathbb{N}\} \tag{3}$$

In the frame of our approach we consider candidate service chains corresponding to acyclic finite automata. This is in order to comply with the rule for shorter execution paths; by introducing cyclic service chains we risk ending-up with longer paths, depending on the iterations that may incur at execution time. Moreover, we do not assign additional observations for monitoring the outcomes of the discovered service chains. This is because introducing additional observations would increase the complexity of the accruing planning problem, without modifying its nature, e.g. render it fully observable, or providing additional benefits.

To accommodate the reuse of heterogeneous types of services, i.e. not just Web services, we reuse and built upon a generic service model that facilitates the modeling of the commonly shared and distinct characteristics of several types of services [**?**]. According to this model a service is a collection of operations, which are accessed by service clients via the exchange of appropriate messages over the web at specific endpoints.

Given the set of features specified in the service model candidate service chains should have input messages with concepts defined in the merged set of observation concepts, i.e. $Oc \cup Oc'$, and their outcomes should lead to either the goal states (G) or sub-goal states (S_{sg}) of process D.

Nonetheless, the formulation of the respective service queries is primarily controlled by the discovery strategy to be used. The two most appealing ones are the forward and backward search strategies. In the backward search strategy, the goal is to define service queries, which set constraints on the expected output messages (Os) and post-conditions (C_{post}) of a candidate action, in addition to checking whether input messages contain ontology concepts related to the merged concept set $Oc \cup Oc'$. Contrary to that, in a forward search strategy the goal is to define queries that constraint the input messages (Is), i.e. starting with the ontology concepts of merged concept set $Oc \cup Oc'$, and checking if a specific process goal (G) or sub-goal (S_{sg}) is achieved. In the provided implementation we employed a forward search strategy. The description of the employed strategy is skipped for reasons of brevity.

Irrespectively of the applied search strategy and in order to avoid searching through a very large set of candidate services, we exploit the given problem description for the introduction of appropriate search bounds and heuristics. Based on the objectives set beforehand, the introduced adaptations should reduce the set of actions used for achieving the goals (G) or sub-goals (S_{sg}) of process D.

Definition 4. *We, introduce a function* $lenghtTo\,(s_a, s_b)$*, which returns the distance, i.e. the cardinality of the set of transitions, between state* s_a *and the requested state* s_b *of a process or of a given service chain (4).*

$$lenghtTo : S \times S \to \mathbb{N} \tag{4}$$

Based on (4), for a process D with a set of starting states s_0 and an intermediate or goal state $s_i \in \{S_{sg} \cup G\}$, a candidate service chain Sc with a starting state s_0 and the same goal state $s_i \in \{S_{sg} \cup G\}$ (i.e. Sc leads to the same state as in D) a search bound can be expressed as follows:

$$lenghtTo\,(s_0, s_i) > lenghtTo\,(s'_0, s_i) \tag{5}$$

Using this search bound we can avoid falling into an endless and iterative search over a large set of candidate service chains. To further enhance the search process we can also introduce additional heuristics, by exploiting knowledge from the problem domain. In conclusion, the outcome of the action set expansion process is a finite set of candidate service chains (6) for the given set of candidate observations, when searching over a finite service registry:

$$SC = \{Sc_{vc} | Sc_{vc}, vc \in O_V \cup O'_V\} \tag{6}$$

2.5 Observation and Action Set Consolidation

The calculated candidate observation (O'_V) and service chain sets (SC) contain values that do not correspond to each other, hence before proceeding with the formulation of the extended planning problem, i.e. ED, these sets should be cleaned up. Thus, starting from the set of additional service chains $Sc \in SC$ each of the candidate observations, i.e. $o' \in O'_V$ that is not used from any of the identified service chains, i.e. $\nexists Sc_{vc} \in SC \wedge vc_{Sc} = vc_{o'}$, is removed from the candidate observations set. The pruned set of candidate observations O''_V contains all semantic concepts that are used by the identified service chains.

The next step, is the specification of the corresponding observation variables. According to the definition given for an observation variable, i.e. Def. 2, apart from the associated semantic concept, the syntactic type along with a name are also required. Based on our hypothesis that each candidate observation, i.e. $o' \in O'_V$ should be used by at least one service chain, i.e. $Sc \in SC$, we can safely assume that the syntactic type of each observation variable should correspond to the one expected by the candidate service chain, Sc. Nonetheless, as a candidate observation may be used by several service chains we should introduce different observation variables for each of these chains unless they use the same syntactic type for the candidate observation; in the latter case we consider that they refer to the same variable. Each of the newly introduced variables according to the definition given in Def. 1 is expected to stand prior to the execution of the related service chains, whilst in other cases it may be undefined. The resulting observation variable set for the extended planning problem is the union of the original observation and the pruned observations O'', i.e. $O_{ED} = O_D \cup O''$.

Following the formulation of the extended observation variable set, the consolidation process continues with the merge of the extended action set A_{SC} and the accruing state sets, i.e. S_{SC}, to the original action A_D and state S_D sets. Since additional service chains $Sc \in SC$ can be modeled as independent, finite automata (i.e. STS_{Sc_i}), the resulting action and state sets can be calculated from the composition [17] of the original process automaton with each of the additional service chain automata, i.e. $STS_{ED} = STS_D \otimes STS_{Sc_0} \otimes STS_{Sc_1} \cdots \otimes STS_{Sc_n}$.

The formulation of the extended planning problem finishes with the specification of the initial and goal states of the extended planning problem. These correspond to the constraints of the original process D. The resulting planning problem can be then fed to an external planner for the identification of possible solutions. The resulting solution corresponds to the automaton of the controlling process, i.e. the extended adaptable process, that can be transformed to a WS-BPEL process model. This transformation process is not described here for reasons of brevity.

3 Conclusions

The provision of adaptable Service-Oriented Processes (SOPs) in the currently forming web is becoming a necessity as the number of untapped resources increases. Within this new Web era, services, of various types, and information are partially utilized by existing approaches tackling the provision of adaptable SOPs. To this end our approach, the Data-Driven Adaptation of Service Oriented Processes (DDA-SOP), can accommodate the provision of adaptable SOPs that exploit both existing services, irrespectively of their type, and information, i.e. semantically annotation structured data, for their adaptation. The main goal of our approach has been the increase of the overall process performance. To achieve this goal we re-design the given processes so that when related information emerges at the process environment along with the use of existing services we end up with smaller execution paths.

The evaluation of our prototype implementation has clearly shown that our approach can lead into significant performance improvements even from small adaptation rates. These improvements get even more evident in the case of long running processes, where improvements appear from very small adaptation rates, e.g. $< 2\%$. As a result we also observed an increase in the throughput of the executing process, i.e. the number of requests served per second, which is proportional to the achieved performance gains. We have to note that the cost paid for the calculation of DDA-SOPs may seem to counter the benefits accruing by our approach. A crucial factor in the calculation of these costs is the complexity and size, i.e. in terms of involved actions, of a given process model. Nevertheless, considering that these costs are paid once prior to the deployment and execution of the resulting DDA-SOP this is not a prohibiting cost.

The provided prototype implementation has unveiled a wide range of future directions that deserve further research. These directions include improvements and optimizations to our prototype implementation as well as emerging research fields such as: a) investigation of the implicit interaction pattern as a mechanism for the loose coupling of complex processes, b) use of different planning techniques such as stochastic models, e.g. Markov based planning techniques, and c) use of different similarity measurements for the discovery of related observations.

References

1. Barry, D.K.: Service-oriented architecture definition. Web Services and Service-Oriented Architectures (2010)
2. OMG: Business process model and notation (bpmn), v. 2.0, 2011. OMG recommendation
3. Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., Guízar, A., Kartha, N., Liu, C.K., Khalaf, R., König, D., Marin, M., Mehta, V., Thatte, S., van der Rijn, D., Yendluri, P., Yiu, A.: Web services business process execution language version 2.0. OASIS standard (April 2007)
4. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web services. Springer (2004)
5. Pautasso, C., Alonso, G.: From web service composition to megaprogramming. In: Technologies for E-Services. Springer (2005) 39–53
6. Botts, M., Percivall, G., Reed, C., Davidson, J.: Ogc sensor web enablement: Overview and high level architecture. In: GeoSensor networks. Springer (2008) 175–190
7. Spivack, N.: Web 3.0: The third generation web is coming (2006)
8. Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., Shan, M.C.: Adaptive and dynamic service composition in eflow. In: Advanced Information Systems Engineering, Springer (2000) 13–31
9. Pautasso, C., Alonso, G.: Jopera: a toolkit for efficient visual composition of web services. International Journal of Electronic Commerce (IJEC) **9** (Winter 2004/2005 2004) 107–141
10. Rao, J., Su, X.: A survey of automated web service composition methods. In: In Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, SWSWPC 2004. (2004) 43–54
11. Qiu, L., Shi, Z., Lin, F.: Context optimization of ai planning for services composition. In: (ICEBE'06) IEEE International Conference on e-Business Engineering, IEEE (2006) 610–617
12. Kongdenfha, W., Saint-Paul, R., Benatallah, B., Casati, F.: An aspect-oriented framework for service adaptation. In: Service-Oriented Computing–ICSOC 2006. Springer (2006) 15–26
13. Rossi, D., Cabri, G., Denti, E.: Tuple-based technologies for coordination. In: Coordination of Internet agents. Springer (2001) 83–109
14. Russell, S., Norvig, P., Intelligence, A.: A modern approach. Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs **25** (1995)
15. Nau, D., Ghallab, M., Traverso, P.: Automated Planning: Theory & Practice. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2004)
16. Athanasopoulos, G., Tsalgatidou, A.: An approach to data-driven adaptable service processes. In: ICSOFT (1). (2010) 139–145
17. Pistore, M., Barbon, F., Bertoli, P., Sharparau, D., Traverso, P.: Planning and monitoring web service composition. In: Artificial Intelligence: Methodology, Systems, and Applications, 11th International Conference (AIMSA 2004). (2004) 106–119