INTERNETINĖS TECHNOLOGIJOS

Possibilities of automatic and semi-automatic end-user driven web service composition

Dalė Dzemydienė

Mykolas Romeris University, Institute of Communication and Informatics, Professor, Dr.
Mykolo Romerio universiteto Komunikacijos ir informatikos instituto profesorė, daktarė
Ateities str. 20, LT-08303 Vilnius
E-mail: daledz@mruni.eu

Arūnas Miliauskas

Vilnius University, Institute of Mathematics and Informatics Doctoral student
Vilniaus universiteto Matematikos ir informatikos instituto doktorantas
Akademijos str. 4, LT-08663 Vilnius
E-mail: arunas.miliauskas@mii.vu.lt

Our research work relates to the main principles, means and current limitations of the end-user driven automatic and semi-automatic web service composition. It analyses automatic and semi-automatic composition approaches found in literature and classifies them as workflow-based, template-based and automatic methods. The aim of this research is to provide a proposal how to construct semi-automatic or automatic end-user driven web service composition. An approach is illustrated by the multi-complexity of service composition in travel domain. We analyze a conceptual solution that covers the whole composition process: from an end-user submitting composition requirements until the presentation of the a composition execution results. Some methods of an artificial intelligence (AI) planning research field were used in proposed web service composition approach.

Introduction

Nowadays methods and tools for web service composition are becoming the mainstream information technology. Web services can provide a more helpful way for accessing functionality over the network. When implementing new software solution there is often a good chance that some functions are already implemented and can be used by accessing them via web services. But rarely a single service can provide all needed functionality. Therefore the composition of several web services is needed for realization of multi-complex end-user needs.

We would like to use the definition of web services composition and execution as *the process of realization of the requirements of new services using the capability specification of the existing component services.* This definition is proposed by (Agarwal et al., 2008) and helps us to explain which web service composition method is more useful in implementation.

The web service composition is not a trivial task. Some issues should be taken into account when dealing with service composition:

- Number of web services increases and service repositories expected to grow large (Rao and Su, 2005). Therefore, a discovery of suitable services for a particular task can be difficult.
- Web services can be updated and existing compositions may become obsolete. For composition that uses many web services it may require a lot of effort to keep it up to date.

- Services are developed by different organizations, based on different conceptual models. Even for a software developer it can be a difficult task that often leads to many errors or takes a long time to understand the concrete meaning of data in different services.
- Current web services description languages like Web Services Description Language (WSDL) describe only syntax (not semantics) (Akkiraju et al., 2005). Therefore, a manual work is required for a discovery, invocation, and composition of these services.

All the mentioned issues show that web service composition is based on current technologies, architectures, languages, protocols targeted to software developer and is time-consuming, expensive, error prone and leads to very constrained solutions.

One of potential solutions tackling these problems is an automatic or semi-automatic service composition. A level of automation here may vary depending on level of involvement of the software engineers or end users in this process. Here by a semi-automatic service composition we mean that an end-user has to select abstract process and most relevant and suitable services are used from the proposed list of compatible services. Those services are discovered automatically, and a user is not concerned about data mapping or integration. In full automation scenarios composite process is synthesized and services selected based on user goal and service descriptions, without any user intervention.

The aim of this research is analyze possibilities and provide brief description of current trends, activities, issues and problems in automatic, semi-automatic web service composition. The review is presented by comparing main characteristics of different web service composition approaches. We provide classification of these approaches, identifying and selecting some methods which are most suitable and promising for semi-automatic end-user driven web service composition. An approach is illustrated by an example useful in travel domain and shows the end-user driven automatic web service composition possibilities. As a result the conceptual solution of system architecture for semi-automatic web service composition is proposed.

Possibilities of automatic and semi-automatic web service composition

The idea of automatic or semi-automatic web service composition is very promising. It was envisioned that Semantic Web movement could solve this problem because its goal is *"a web of data that can be processed directly and indirectly by machines"* (Berners-Lee, 2001). However, this idea was not realized to its full potential yet. This is shown by the statement created by one of the co-authors of this idea: *"This simple idea... remains largely unrealized."* (Shadbolt et al., 2006). Therefore a lot of research activities are carrying on and many research papers have been published for developing the automatic web service integration and composition possibilities.

The classification of automatic web service composition approaches is presented by (Kim et al., 2009; Rao and Su, 2005; Diagiampetri et al., 2007). Following some approaches, we can identify some commonalities and differences in the process of production of a service composite workflow:

- In workflow-based approaches the composite process is viewed as a workflow and presented as a directed acyclic graph. The process requires domain knowledge and developer involvement.
- In template-based approaches templates are predefined, created and an end-user can select one of them to create a workflow. User empowerment depends on template extensibility.
- In approaches based on AI planning it is a way to generate process automatically from problem specification. In most cases of AI planning implementations user domain knowledge and developer intervention are not required.

Based on the classification proposed by (Rao and Su, 2005) we would like to classify service composition approaches by using slightly different groups of definitions and provide arguments for that.

In (Kim et al., 2009; Rao and Su, 2005) an attempt was made to define the process of automatic service composition. We select one defined in (Kim et al., 2009). It consists of the following phases:

- Specification phase: provide an easy way for a user to specify task goals, requirements and constraints without extensive domain knowledge;
- Planning phase: provide an automatic way to compose an abstract workflow based on the specification;
- Validation phase: provide techniques to ensure that the composite process realized via an abstract workflow satisfies the user's stated task goals;
- Discovery phase: provide a way to discover services that satisfy task specifications in the workflow;
- Execution with monitoring phase: provide a framework for monitoring executing service and provide automatic fault-handling mechanisms.

Automatic or semi-automatic composition can be classified according different dimensions. Here we analyze 3 groups of approaches based on methods how an end-user (without any programming background) is involved in the creation of a service composition workflow:

- Workflow-based: an end-user is responsible for creating a composition workflow.
- Template-based: an end-user doesn't create a workflow himself / herself, the most relevant template can be chosen to simplify composition.
- Automatic: a user provides high-level goals and based on that composition is synthesized.

In workflow-based approaches composition is seen as workflow (Casati et al., 2001). Therefore web services are composed by creating control and data flow among them. Workflow-based approaches split into two groups (Rao and Su, 2005): static or dynamic workflow generation.

The difference between them lies on level of user involvement. In static approach the user creates an abstract process model and only web service selection and binding are done automatically. On the other hand, in the dynamic approach a process model and service selection are made automatically. Combined approaches are also mentioned.

One of the approaches discussed is template-based service composition. Using this method, a workflow template is selected and it is bound to specific web services. Most papers promote this approach as the most practical one (Kim et al., 2009; Mehandjiev et al., 2010). One of the advantages of this approach is a compromise between flexibility, user empowerment and encapsulation of complexity that is crucial for a user without any programming background. Template-based approaches differ in level of abstraction of templates and languages used for template specification. Most common languages are OWL-S (Kim et al., 2009; Sirin et al., 2005), BPEL with WSDL (Geebelen et al., 2008; Jie et al., 2008). In most cases languages are extended with special constructs to be suitable for template specification.

SOA4All project (2008-2011) aimed at integrating SOA with the Web 2.0 and Semantic Web (Domingue et al., 2009) and one of the addressed challenges was automated web service composition. The project research results are related to the topic of the template based composition (Mehandjiev et al., 2010; Lécué et al., 2010). Mehandjiev et al. (2010) proposes an approach of assisted service composition for end users. Successful compositions are saved into reusable templates where technical details are hidden from the user. Semantic technologies are used to hide service composition complexity such as control or data dependencies. The user is qualified to select a template from a list and choose a preferred service. Still we couldn't find information what language was used to specify a composition template. Another paper (Lécué et al., 2010) describes an approach where composition templates are generated from log files.

A composition is synthesized from generated and already existing templates using a parametric design procedure with a blackboard based multi agent system and later is optimized by changing services to the most relevant ones. Again, no information regarding template format is provided.

Other approaches (Geebelen et al., 2008; Jie et al., 2008) modify BPEL with a special annotation to define templates. Geebelen et al. (2008) proposes a method where BPEL is rendered using a similar approach as dynamic web page content is generated and send to a browser. A model–view-controller (MVC) pattern with a ruby-on-rails framework is used. The template in this approach represents a view in this pattern and is BPEL with special dynamic annotations, and is created at a runtime the same way as HTML web page is created for a user. In another research (Jie et al., 2008) BPEL is extended with special tags for a template definition. In the template abstract services are marked with partner tags which during template instantiation are replaced with real web services definitions. In all BPEL based approaches template instantiation produces an executable BPEL workflow.

Another workflow template based approach is presented in a paper (Sirin et al., 2005). It adheres to an earlier work of (Sirin et al., 2004) where a method of translation OWL-S process model to SHOP2 domain was presented. But in addition to the previous work, the following one (Sirin et al., 2005) explored possibilities of creating workflow templates and using them to create flexible compositions. OWL-S is seen as a specification language for these templates. It provides enough control flow elements like *Perform, Sequence, Choice,* etc. for defining *Composite_Process* from lower grain components - atomic processes. But this is not enough for creating abstract workflows. Therefore OWL-S was extended with the new type of process – the so called *Abstract_Process*. This *Abstract_Process* is not bound to concrete web services and is used for an abstract template specification. This template is mapped to the Hierarchical Task Network (HTN) domain, where a non-primitive task represents an abstract activity. In the HTN planning (Ghallab et al., 2004) the tasks are decomposed by methods to subtasks (and these further) until all tasks are broken down to primitive tasks. HTN-DL is an extended HTN formalism and is used to compose a process from template definition.

A web service composition using Situation calculus was presented by McIlraith and Son (2002). The situation calculus extends first order logic with actions and situations. Golog is a language built on top of situation calculus. It provides constructs (like sequences, nondeterministic choices, conditionals, loops) which enable defining a complex action and describing how they are comprised of primitive actions. Since originally Golog was created without information gathering – sensing actions, these types of actions are also added to this approach. User requirements are specified as preferences, defining which action is desired in a particular situation. A plan is synthesized using A^* planner originated from the Simple Breadth First Planner (Reiter, 2001).

Automatic web service composition

Under the automatic web service category fall approaches in which no pre-specified control flow exists, and a workflow is synthesized from user's goals, inputs outputs, pre-conditions and effects (IOPE) of individual services. Many papers (Kim et al., 2009, Rao and Su 2005) call these methods "AI planning" because in the majority of works AI planning methods are used to create compositions. But we see this definition ambiguous because some of template based composition approaches also use AI planning methods, for example, HTN (Sirin et al., 2005).

Here we accept AI planning definition provided by (Ghallab et al., 2004) where AI planning is defined as computational study of a deliberation process (planning process) that, aiming to achieve some pre-stated objectives, chooses and organizes actions by anticipating their outcomes. There is no statement that templates or any other predefined organization of actions cannot be used in AI planning. Therefore we do not refer to AI planning in our classification.

Based on a paper (Digiampietri et al., 2007) we define automatic web service composition as a process where based on user provided goals the system automatically creates a composition that satisfies these goals.

Sirin et al. (2004) suggests using HTN planning for an automatic web service composition. The paper shows how OWL-S service descriptions can be transformed to SHOP2 domain. But for this approach OWL-S composite service definition must exists. Also, this method cannot handle OWL-S composite processes where Split, Split – Join constructs are used. This is due to the fact that SHOP2 planner cannot handle concurrency. Another constraint is that atomic processes in OWL-S must be either world altering or information providing service, but not both. The reason is that the method uses interleaved planning, where, during planning process, information-providing services are executed and world-altering services are simulated (based on input, output, preconditions and effects).

Another approach was presented by Pistore et al. (2005). Here composite service is synthesized from abstract BPEL4WS processes. These abstract BPEL4WS processes define an interaction protocol with component web services. Each of them is translated to a state transition system (STS). Then based on composition requirements new STS (for composite process) is generated which is automatically translated to a composite executable BPEL4WS program. Synthesis approach is based on symbol model checking and uses system MBP that is built on top of symbol model checker named NuSMV. Composition requirements are expressed in the EAGLE language. This method shows that it is possible to create composition having only composition requirements and a definition of abstract processes.

The automatic composition (when no additional information is used, apart from composition requirements, goals and IOPE is required from end-user) has the following benefits and advantages:

- Extreme flexibility is mentioned by (Svatek and Vacura, 2005).
- The only approach when predefined templates do not exist and end-user lacks domain knowledge (Kim et al., 2009).

Despite that, the following disadvantages of automatic web service composition have been reported:

- Unpredictable results if all conditions are not perfectly specified (Svatek and Vacura, 2005).
- Composition effort is estimated bigger than in template based approach due to higher search space (Agarwal et al., 2008). Therefore it is difficult to synthesize a correct composition and this approach is not trusted in real-world scenarios yet (Kim et al., 2009).
- Lack of user empowerment and interaction (Kim et al., 2009).

Here we argue that if knowledge about relations between tasks, goal and services exists then composition can be efficient as in a template-based scenario. Such functions can be provided by domain ontology. Ontology is specified by domain experts and by using it we can avoid software developer involvement and give more empowerment to an end-user.

An example of end-user driven composition in travel domain

In order to illustrate the main features of user driven web service composition possibilities we provide an example. We choose a travel domain in which we would like to demonstrate multi complexity of web service composition requirements and then provide composition walkthrough. For instance, John, an end user, lives in Lithuania, and decides to have a leisure weekend in Paris. In order to arrange this travel himself he needs to:

- Select appropriate means of transport. That may include various types of transport, like:
  - finding plane tickets;
  - renting a car;
  - finding a bus to get from the airport.
- Choose potential points of interest (libraries, monuments, restaurants etc.).

- Find accommodation (a hotel, a hostel, a camping site, etc.). Accommodation type depends on the traveller's preferences and type of travel (group of students travelling across Europe will have different requirements than a business person going to arranged meeting).

All required activities can be accomplished by the user via internet, most likely all transport, accommodation service providers have web sites with all important information and all needed services can be booked and purchased online.

However, arranging this requires a lot of manual work: searching numbers of web sites for relevant information, purchasing the appropriate tickets at different web stores.

Some travel, accommodation, transport service providers created web services and the tasks currently performed by end user could potentially be accomplished by a software agent. However, these compositions are currently created by a software developer and are restricted to very simple scenarios (e. g., business travel).

Ability for the end user based on preferences, context to create a web service composition, execute it without any assistance of a software developer, is still a promising and yet unresolved issue. We provide our vision of the solution in the scenario walkthrough.

Based on arguments above, we select an automatic composition as the most relevant in provided scenario. We make an assumption that travel and accommodation service providers have created semantically annotated services. There are tools for this purpose like web service modeling ontology (WSMO) or ontology in web ontology language (OWL) for semantic web service specification (OWL-S). These specifications are included into domain ontology by domain experts.

Domain ontology is one of the main prerequisites for this approach. The planning and travel domain should be specified by domain experts.

Based on the arguments by Sirin et al. (2004) we select an HTN planning method and planner SHOP2 for web service composition.

Capturing the end-user travel requirements

In order to create a composition an end-user must be able to submit composition requirements in a convenient way. Many papers support an idea providing requirements as goals, but few of them emphasize distinction between planning goals and composition high-level goals (Portchelvi et al., 2012). We agree that it would be impossible for an end-user without any software engineering knowledge and define a planning problem in an AI planner (e. g., SHOP2) understandable format.

In order to translate high end-user requirements to the planning problem we adhere to a Goal-Based Service Framework (GSF) proposed approach (da Silva Santos et al., 2009). There should exist a travel domain specification (created by domain experts) based on Goal-Based Service Ontology (GSO). Domain specification supports an end-user goal, task decomposition and mapping tasks with services. Apart from domain concepts, the ontology may formally specify their relationships, dependencies, and example cases (Dzemydienė et al., 2011).

In our travel domain example we envision that an end-user does not provide all travel parameters, options, but rather provides a goal – this is based on a predefined travel template. Thus in our travel scenario John decides to have a leisure type weekend trip. He provides a goal – to have a leisure weekend trip, the type of trip that is achieved by fulfilling *Leisure_weekend_trip_task* – the structure of which is defined by *Leisure_weekend_trip_pattern*. That means the trip will have same structure (stop points, transport types, accommodation etc.) as defined in the leisure weekend trip pattern.

Houda et al. (2010) provides public transportation ontology for the travel domain and defines journey patterns, like *Leisure_journey_pattern*. Using a similar approach we argue that there should exist similar patterns for a trip, like *Leisure_trip_pattern, Leisure_weekend_trip_pattern* (the one that is chosen by John in our example) or *Business_trip_pattern*.

Here we do not use goal decomposition to sub-goals but use a high-level complex task that is further decomposed into lower level subtasks.

Travel pattern translation to planning domain

In order to use traveling patterns in terms of goals in GSO, we must map the GSF approach (da Silva Santos et al., 2009) with travel domain definitions (Houda et al., 2010). We provide an example for this in a diagram below (Fig. 1).

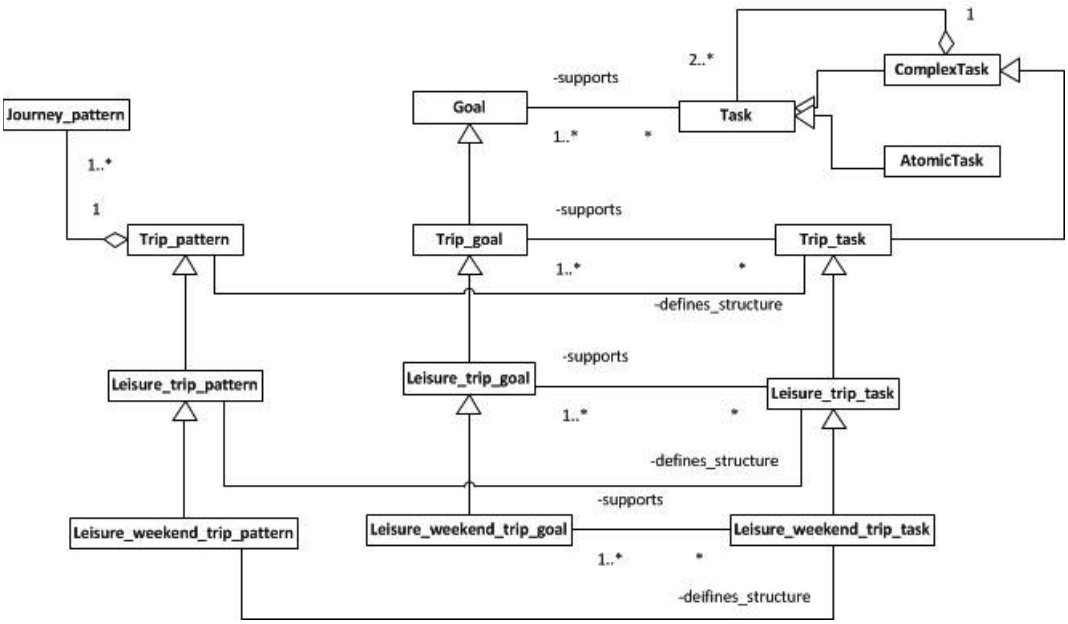
We envision that there exist different types of travel patterns. For a public transportation Houda et al. (2010) provide travel patterns like *Service journey pattern, Interesting journey pattern*, *Shopping journey pattern* taxonomy. We distinguish *journey* and *trip* concepts. *Trip* is an act of going to another place and returning back. *Journey* is one piece of travel, going from one place to another. So, the *leisure_weekend_trip* pattern is a composition of several *journeys.*

These patterns need to be defined and mapped with GSO before planning. The structure of the selected pattern can drive task decomposition to lower grained tasks which at the end will be mapped to web services.

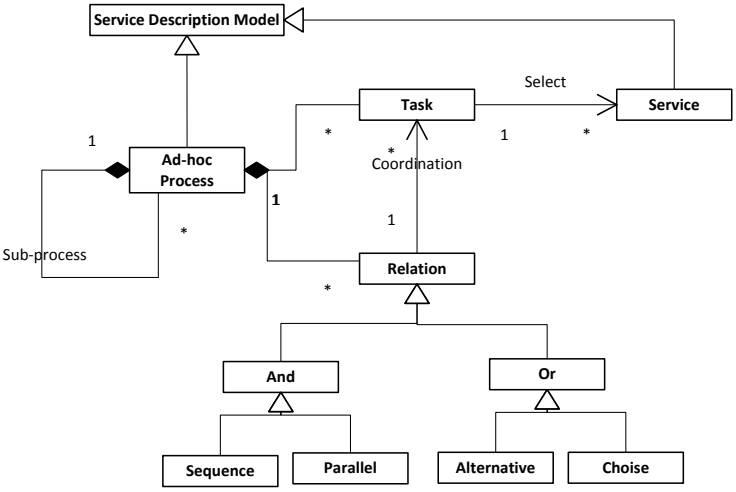
The entity *Journey_pattern* is from public transportation ontology (Houda et al., 2010). Entities *Goal, Task, CompexTask, AtomicTask* are from GSO. Others are our proposed entities (see Fig. 1).

Here we specify different types of goals (specializations of the goal in GSO) for each kind of pattern. Also, we suggest creating specializations for GSO Complex Task in order to achieve different types of goals. Using this approach a high-level goal (like "Have leisure weekend trip") can be translated to an AI planning problem definition.

Since we propose the usage of SHOP2 planner, the selected pattern description in OWL must be translated to a SHOP2 domain. SHOP2, like all HTN planners, uses a notion of an HTN method to capture how tasks are decomposed to subtasks. So an OWL pattern should be translated to the HTN method. Xiao et al. (2011) shows how ad-hoc processes could be created from ontology descriptions. An ad-hoc process is defined as a set of tasks that have to be per-



F i g. 1. ***Description of a travel goal domain based on GSO***



F i g. 2. ***An ad-hoc process model***

formed without strict order. The ad-hoc process model shows that it consists of simple flow control structure elements.

Sirin et al. (2004) shows how OWL-S process models can be encoded as SHOP2 domains. An OWL-S composite process uses the following control structures: Sequence, Unordered, Choice, If-Then-Else, Iterate, Repeat-Until, Repeat-While, Split and Split+Join. Therefore we conclude that any ad-hoc process can be transformed to an OWL-S composite process which, based on Sirin et al. (2004), can be translated to a SHOP2 planning domain. It is worth mentioning that SHOP2 cannot handle concurrency, therefore a solution how concurrent action must be handled should be presented.

Here we provide an idea how travel patterns can be defined and translated to an HTN planning domain. In order to prove this a more detailed research on this topic is required and is planned in the future.

Generating a plan for travel web service composition

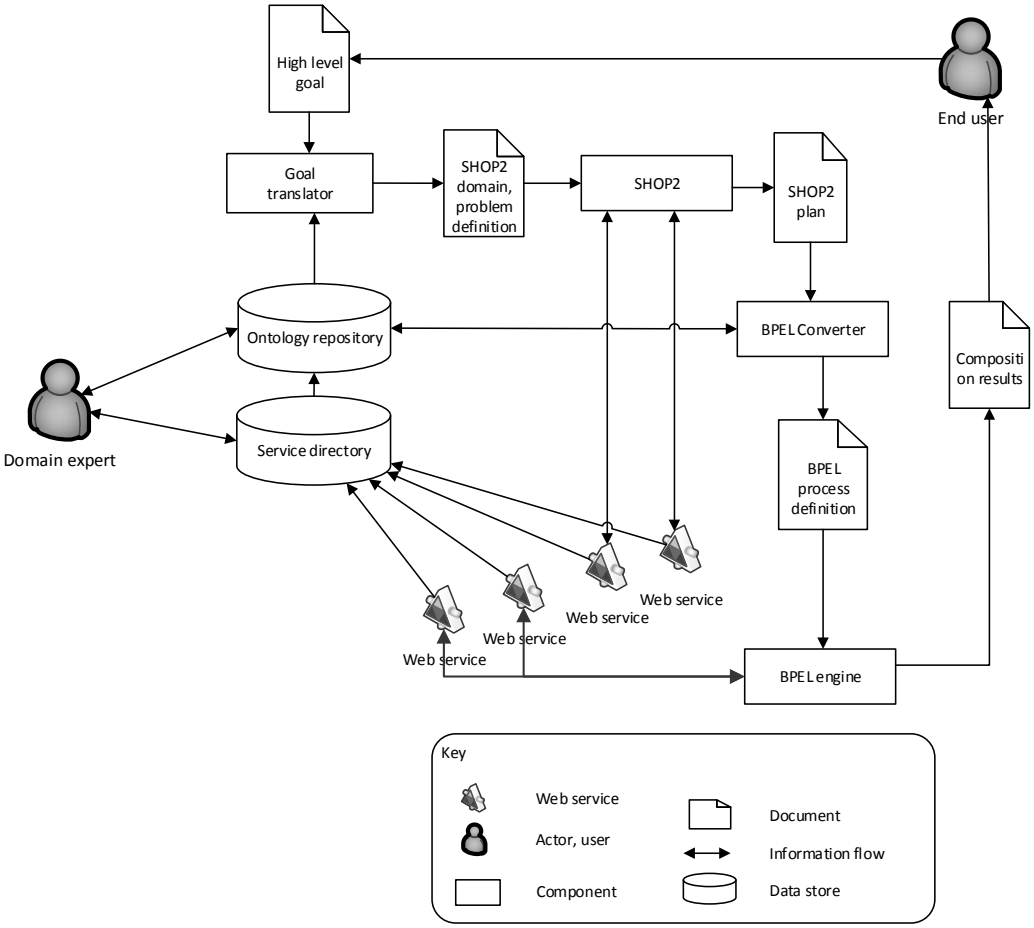
Having a planning domain and a problem provided a SHOP2 planner can search for a plan. Sirin et al. (2004) demonstrated that during the planning process information gathering actions could be executed.

In the example, during the planning travel information services must be requested online. The received information regarding flight schedules, available accommodation, points of interest must be used in travel planning. Based on that, a travel plan can be synthesized.

The plan contains world altering primitive actions – this represents calls to web services which state changes, for example, booking flight tickets.

We must transform the plan to executable workflow. The generated abstract plan must be converted to an executable description. Business process execution language (BPEL) is a good candidate for the composite, executable workflow as today there are many BPEL engines in the industry. Sirin et al. (2004) provides an example how an abstract plan (generated by SHOP2) can be transformed to the OWL-S process definition. Due to the similarities between OWL-S and BPEL we envision that a similar approach could be used for creating a BPEL composite workflow. GSO and a domain definition will be a primary source of knowledge for this process. We plan to test this approach in practice in the future.

We propose architecture as a structure of components where each component is either an already existing software component (SHOP2, BPEL Engine) or an integration component (Goal translator, BPEL Converter). One of the main reasons for this is decomposition of automatic web service composition problem to smaller problems that potentially already have solutions. Each sub-problem is solved by a specific component and the interchange between components is made using messages (documents). Data-flow between components and main actions is provided in Fig. 3.



F i g. 3. ***Conceptual solution of end-user driven service composition represented by data flow processes***

Goal translator is a component that by using domain ontology creates planning domain and problem definitions in a SHOP2 format from high-level end-user goals.

SHOP2 is an existing AI planner that can be extended with functions for an information sensing action execution (Nau et al., 2003). Information queries to external web services are made during planning. SHOP2 produces an abstract plan.

BPEL converter is a component that transforms an abstract AI plan to a web service composition in BPEL.

BPEL engine is a component that executes the process defined in BPEL. Currently there are plenty of BPEL engines in industry. Composition execution results are presented to an end-user.

Domain expert is an important role because the person is responsible for creating a domain definition in GSO that will be the core of the whole service composition process.

Conclusions

The main possibilities of the approaches for automatic and semi-automatic web service composition are analyzed paying more attention to end-user divining composition. The analysis shows that there is still a lack of methods and tools that focus on multi-complex possibilities of specifying end-user's requirements in a conve-

nient way. Most of approaches are related to the plan synthesis and the major part seeks to adopt an AI planning for the web service composition. However, there are approaches that try to solve a web service composition problem without AI planning methods.

Despite many papers published there still isn't a strict and clear classification of end-user driven composition approaches. We showed that defining categories of the template-based and AI planning-based is ambiguous.

Most of web service composition approaches propose template-based methods as the most feasible way of problem realization. We argue that an automatic composition when there is a domain knowledge specified by the ontology can be as practical as a template-based one.

The AI planning methods are suitable for end-user driven web service composition realization. Despite that a web context places additional constraints, requirements that classical AI planning cannot cope with these problems. Therefore, AI planning methods that allow relaxing of some of restrictive assumptions of a classical AI planning (example like partial state observability) are useful.

We provided an example that shows how this problem of the automatic service composition can be broken down to smaller ones, and how which can be handled separately. We provided a conceptual architecture of the system that is oriented to reuse existing assets in this research fields.

REFERENCES

AGARWAL, V., CHAFLE, G.; MITTAL, S.; SRIVASTAVA, B. (2008). Understanding approaches for web service composition and execution. In: *Proceedings of the 1st Bangalore Annual Computer Conference*. ACM, p. 1–8.

AKKIRAJU, R.; FARRELL, J.; MILLER, J.; NAGARAJAN, M.; SCHMIDT, M.; SHETH, A.; VERMA, K. (2005). Web service semantics – WSDL-S.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. (2001). The semantic web. *Scientific American*, 284, 5, p. 28–37.

CASATI, F.; SAYAL, M.; SHAN, M.C. (2001). Developing e-services for composing e-services. In: *Advanced Information Systems Engineering*. Springer Berlin Heidelberg, p. 171–186.

DA SILVA SANTOS, L. B.; GUIZZARDI, G.; GUIZZARDI, R. S. S.; DA SILVA, E. G.; PIRES, L. F.; van Sinderen, M. (2009). GSO: Designing a well-founded service ontology to support dynamic service discovery and composition. In: *Proceedings of the Enterprise Distributed Object Computing Conference Workshops, 2009. EDOCW 2009. 13th*. IEEE, p. 35–44.

DIGIAMPIETRI, L. A.; PÉREZ-ALCÁZAR, J. J.; MEDEIROS, C. B. (2007). AI Planning in Web Services Composition: a review of current approaches and a new solution. In: *VI ENIA-Proceedings of the XXVII Brazilian Computer Society Conference (CSBC2007)*.

DOMINGUE, J.; FENSEL, D.; DAVIES, J.; GONZÁLEZ-CABERO, R.; PEDRINACI, C. (2009). The Service Web: a Web of Billions of Services. *Towards the Future Internet*, p. 203.

DZEMYDIENĖ, D.; JASIŪNAS, E.; KRIAUČIUKAS, V.; MILIAUSKAS, A. (2011). Possibilities of managing of informational resources in the adaptive e-service providing system by using domain ontology. In: *Proceedings of the XV Computer Science Conference*. LIKS. Vilnius. Žara, p. 57–63.

GEEBELEN, K.; MICHIELS, S.; JOOSEN, W. (2008). Dynamic reconfiguration using template based web service composition. In: *Proceedings of the 3rd Workshop on Middleware for Service Oriented Computing*. ACM, p. 49–54.

GHALLAB, M.; NAU, D.; TRAVERSO, P. (2004). *Automated planning: theory and practice*. Morgan Kaufmann, ISBN 1-55860-856-7

HOUDA, M.; KHEMAJA, M.; OLIVEIRA, K.; ABED, M. (2010). A public transportation ontology to support user travel planning. In: *Proceedings of the Fourth International Conference on Research Challenges in Information Science (RCIS)*. IEEE, p. 127–136.

JIE, G.; BO, C., JUNLIANG, C.; LEI, Z. (2008). A template-based orchestration framework for hybrid services. In: *Proceedings of the Fourth Advanced International Conference on Telecommunications. (AICT'08)*. IEEE, p. 315–320.

KIM, A.; KANG, M.; MEADOWS, C.; IOUP, E.; SAMPLE, J. (2009). *A framework for automatic web service composition*. Naval Research Lab Washington DC Center For High Assurance Computing Systems (CHACS).

LÉCUÉ, F.; GORRONOGOITIA, Y.; GONZALEZ, R.; RADZIMSKI, M.; VILLA, M. (2010). SOA4All: an innovative integrated approach to services composition. In: *Proceedings of IEEE International Conference on Web Services (ICWS)*. IEEE, p. 58–67.

LI, X.; WU, C. (2009). Research on OWL-S Service Automatic Composition Based on Planning. In: *Proceedings of International Conference on Information Engineering and Computer Science. (ICIECS),* IEEE, p. 1–4.

MCILRAITH, S.; SON, T. C. (2002). Adapting Golog for composition of semantic Web services. In: *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers; p. 482–496.

MEHANDJIEV, N.; LECUE, F.; WAJID, U.; NAMOUN, A. (2010). Assisted Service Composition for End Users. In: *Proceedings of the IEEE 8th European Conference on Web Services (ECOWS)*. IEEE, p. 131–138.

NAU, D. S.; AU, T. C.; ILGHAMI, O.; KUTER, U.; MURDOCK, J. W.; WU, D.; YAMAN, F. (2003). SHOP2: An HTN planning system. *Artif. Intell. Res. (JAIR)*, vol. 20, p. 379–404.

PISTORE, M.; TRAVERSO, P.; BERTOLI, P.; MARCONI, A. (2005). Automated synthesis of composite BPELWS web services. In: *Proceedings of the IEEE International Conference on Web Services, (ICWS)*. IEEE, p. 293–301.

PORTCHELVI, V.; VENKATESAN, V. PRASANNA; SHANMUGASUNDARAM, G. A. (2012). Study on composition goal in automated web services composition approaches. *International Journal of Computer Applications*, vol. 55, issue 11, p. 4–8.

RAO, J.; SU, X. (2005). A survey of automated web service composition methods. In: *Semantic Web Services and Web Process Composition*. Springer Berlin Heidelberg, p. 43–54.

REITER, R. (2001). *Knowledge in action: logical foundations for specifying and implementing dynamical systems*. Cambridge: MIT press, ISBN 0-26218218-1

SHADBOLT, N.; HALL, W.; BERNERS-LEE, T. (2006). The semantic web revisited. *Intelligent Systems, IEEE*, 21.3, p. 96–101.

SIRIN, E.; PARSIA, B.; WU, D.; HENDLER, J.; NAU, D. (2004). HTN planning for web service composition using SHOP2. *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 1, no 4, p. 377–396.

SIRIN, E.; PARSIA, B.; HENDLER, J. (2005). Template-based composition of semantic web services. In: *AAAI Fall Symposium on Agents and the Semantic Web,* p. 85–92.

SVÁTEK, V.; VACURA, M. (2005). Automatic Composition of Web Analysis Tools: Simulation on Classification Templates. In: *First International Workshop on Representation and Analysis of Web Space (RAWS-05)*. Online: http://CEUR-WS.org.

XIAO, H.; ZOU, Y.; TANG, R.; NG, J.; NIGUL, L. (2009). An automatic approach for ontology-driven service composition. In: *Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications (SOCA'2009)*. IEEE, p. 1–8.

VARTOTOJO VALDOMOS INTERNETINIŲ PASLAUGŲ AUTOMATINĖS IR PUSIAU AUTOMATINĖS KOMPOZICIJOS ATLIKIMO GALIMYBĖS

Dalė Dzemydienė, Arūnas Miliauskas

S a n t r a u k a

Šio mokslinio tyrimo tematika nagrinėja internetinių paslaugų kompozicijos atlikimo priemones ir būdus. Straipsnyje aprašomi vartotojo poreikiams pritaikytų internetinių paslaugų kompozicijos automatiniai ir pusiau automatiniai kūrimo būdai ir metodai, plačiau nagrinėjamos šių metodų galimybės ir apribojimai. Analizuojami moksliniuose straipsniuose pateikiami internetinių paslaugų kompozicijos atlikimo metodai ir išskiriami trys pagrindiniai šių paslaugų kompozicijos būdai: darbų srautų modeliais grindžiamas, paslaugų šablonais grindžiamas ir automatinis paslaugų kompozicijos metodas. Tyrimo tikslas – pateikti pasiūlymą, kuris leistų automatiniu ar pusiau automatiniu būdu kurti internetinių paslaugų kompozicijas pagal vartotojų poreikius. Pusiau automatinio internetinių paslaugų komponavimo uždavinio sprendimo būdą iliustruoja sudėtingas kelionės planavimo pavyzdys. Internetinių paslaugų kompozicijai atlikti siūloma taikyti dirbtinio intelekto planavimo metodus. Pateikiama tokio uždavinio sprendimo koncepcija, kuri grindžiama fragmentiniais kituose projektuose gautais paslaugų komponavimo rezultatais ir bando sujungti visą internetinių paslaugų kompozicijos procesą: nuo vartotojo keliamų kompozicijos reikalavimų įvedimo iki tinkamo paslaugų kompozicijos rezultatų pateikimo.