Fast Composition Planning of OWL-S Services and Application[1]

Matthias Klusch and Andreas Gerber
German Research Center for Artificial Intelligence
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
{klusch@dfki.de, andreas.gerber@x-aitment.net}

Abstract

In this paper, we present the implementation, evaluation, and application of our OWL-S service composition planner OWLS-XPlan. Medical services described in OWL-S 1.1 and ontologies are converted to initial state and goal descriptions in PDDL 2.1, which are then used by the fast heuristic FF planner XPlan for generating an execution complete composition plan. Results of experimental evaluation of XPlan shows its top performance compared with other selected AI planners. OWLS-XPlan is used in an agent based mobile e-Health system for emergency medical assistance planning tasks.

1. Introduction

. Though the composition of complex Web services attracted much interest in different fields related to service oriented computing, there are only a few implemented composition planning tools publicly available for the semantic Web such as the HTN composition planner SHOP2 [11, 12] for OWL-S services. One problem with pure HTN planners is that they require task specific decomposition rules and methods developed at design time, hence are not guaranteed to solve arbitrary planning problems. That, in particular, motivated the development of our hybrid off-line composition planner for OWL-S 1.1 services, called OWLS-XPlan [13], which is guaranteed to find a solution if it exists, though the corresponding planning problem remains to be NP-complete.

OWLS-XPlan is integral part of the prototypically implemented agent based mobile eHealth system, called Health-SCALLOPS, for secure emergency medical assistance planning tasks, such as patient repatriation and relocation to selected hospital. An extended version of OWLS-XPlan, called OWLS-XPlan+, that allows for heuristic quasi-online re-planning of composite OWL-S services has also been implemented and is currently in use in a different e-health application scenario within the European project CASCOM.

The remainder of this paper is structured as follows. Section 2 briefly introduces OWLS-XPlan, followed by the results of the performance evaluation of its core planning module XPlan, and its implementation in sections 3 and 4, respectively. The use of OWLS-XPlan in the Health-SCALLOPS application is described in section 5. We briefly refer to related work and conclude in sections 6 and 7, respectively.

2. OWLS-XPlan Overview

The semantic web service composition planner OWLS-XPlan consists of several modules for pre-processing and planning of composite OWL-S services (cf. figure 1). It takes a set of available OWL-S 1.1 services, related OWL ontologies, and a planning request (goal) as input, and returns a planning sequence of relevant OWL-S services that satisfies the goal.

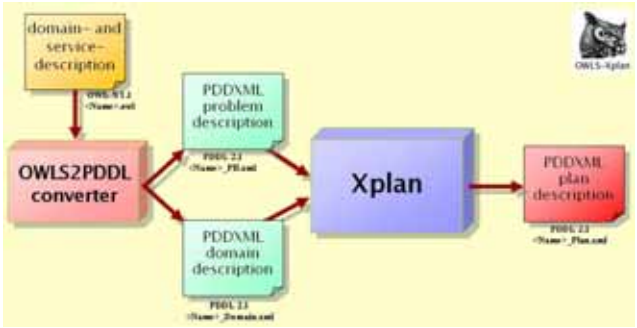


Figure 1. OWLS-XPlan overview

For this purpose, it first converts a given domain ontology and service descriptions in OWL and OWL-S 1.1, respectively, to equivalent PDDL 2.1 problem and domain descriptions using an integrated OWLS2PDDL converter. The domain description contains the definition of all types, predicates and actions, whereas the problem description includes all objects, the initial state, and the goal state. Both descriptions are then used by the AI planner XPlan to create a plan in PDDL that solves the

[1] This work has been supported by the German Ministry of Education and Research (BMBF 01-IW-D02-SCALLOPS), and the European Commission (FP6 IST-511632-CASCOM).

given problem in the actual domain. An operator of the planning domain corresponds to a service profile in OWL-S, while a method is a special type of operator for fixed complex services that OWLS-XPlan may use during its planning process. For reasons of convenience, we developed a XML dialect of PDDL, called PDDXML, that simplifies parsing, reading, and communicating PDDL descriptions using SOAP.

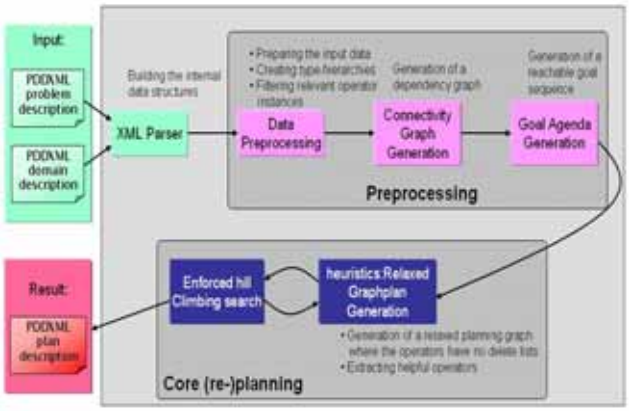
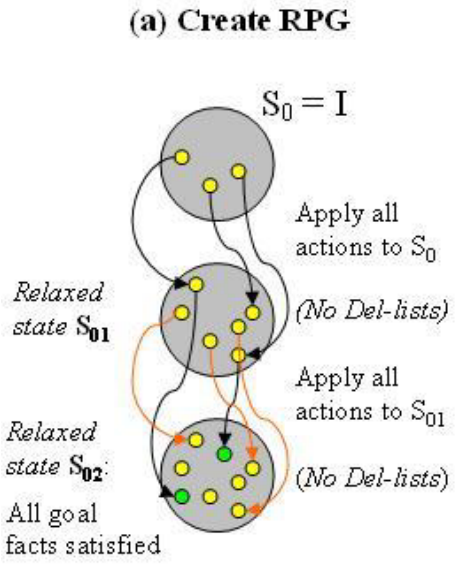
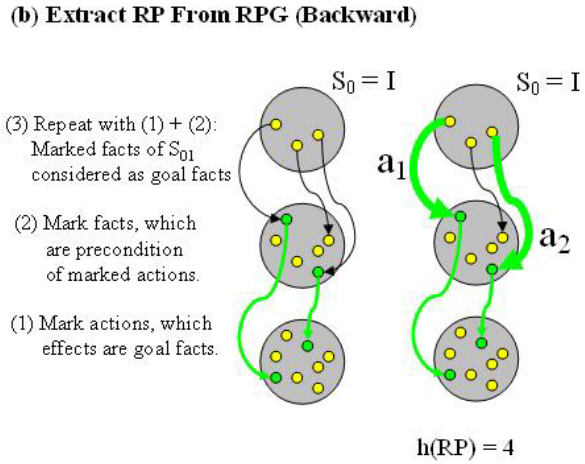
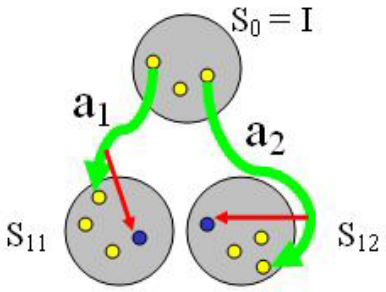


Figure 2. XPlan planning module

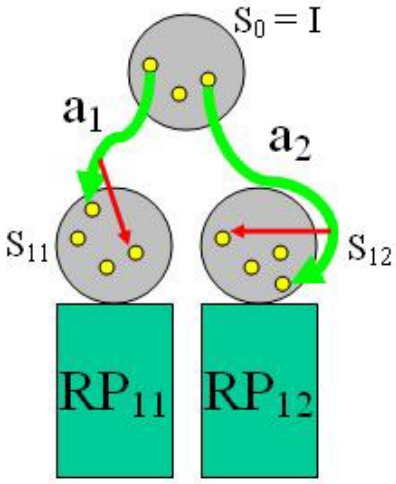
The planning module XPlan (cf. figure 2) is a heuristic hybrid FF planner based on the FF planner developed by Hoffmann and Nebel [4, 5, 6]. It combines guided local search with relaxed graph planning, and a simple form of hierarchical task networks (HTN) to produce a plan sequence of actions that solves a given problem. If equipped with HTN methods (composed services), XPlan uses only those parts of decomposed methods that are required to reach the goal state







(1) Apply all helpful actions to S_0 *including previously ignored Del-lists* (negative effect facts ●) which yields complete states.



(2) Create RPG for each successor state of helpful action (s. (a))

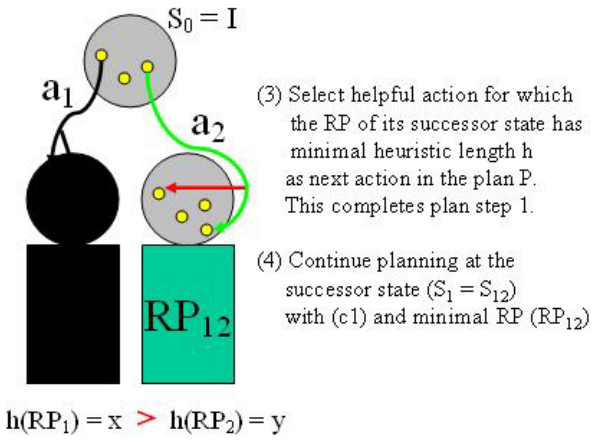


Figure 3. XPlan planning step example: (a) Create relaxed planning graph RPG, (b) extract relaxed plan RP from RPG, (c) Select heuristically optimal helpful action (bold green) as action in the plan sequence; h(RP) = Number of actions in the relaxed plan heuristically equals the length of RP and P.

For each sub-goal g of the determined goal agenda, at each planning step i, XPlan quickly builds a relaxed planning graph RPG(i) in a fast goal reachability test heuristically ignoring negative effects of actions, and the corresponding relaxed plan RP(i) in a backward pass from g to S_i.

The relaxed plan contains all paths of applicable actions that lead from g to S_i, of which only those in its first action-layer 0 are called helpful. In the following, XPlan focuses on the helpful actions of RP(i) only, hence reduces the search space. Please note that the relaxed plan is not necessarily correct.

In order to decide which helpful action to select as the next action in a valid plan sequence, it applies each of them to S_i and adds the previously ignored Del-list facts yielding the complete state S_{ij}, where j in {1,.., l}, denotes the j-th helpful action applied to state S_i .

For each of these states the relaxed plan RPG(i,j) is built to heuristically search for the relaxed plan RP(i,j) with heuristically minimal length h(RP(i,j)). In this context, the "plan length" h(RP(i,j)) just denotes the sum of all actions in all action-layers of the RP. Finally, XPlan retains the action A_{ij} with heuristically minimal goal-distance and starts the next planning step i+1 with S_{ij}. If there are multiple RPs of equal length, it repeats the same decision process starting at state S_{i1} (like a breadth first search restricted on helpful actions), and then S_{i2}, ..., S_{il} until a minimum is found.

Eventually, all created plans for sub-goals g of the goal agenda are respectively concatenated which yields the final plan sequence P. The plan then gets executed, and if it fails, XPlan allows re-planning from the most recent valid state produced by action execution, to avoid a total re-planning, if possible. For more details on OWLS-XPlan in general, and XPlan in particular, together with examples of service translation from OWL-S to PDDXML we refer the reader to [13]. The software package OWLS-XPlan 1.0 is available at [15].

3. Evaluation of XPlan

We evaluated the performance of XPlan, using the publicly available benchmark of the international planning competition IPC3 [2], and compared the results with that of the four top performing IPC3 participants, i.e. FF planner, Sim-Planner, and the HTN planners TLPlan, and Shop2. XPlan was tested without task specific methods. Planning performance was measured in terms of

(a) the planning completeness, i.e. the total percentage of solved problems (cf. figure 4),
(b) the average plan length (cf. figure 5), and
(c) the average plan quality, i.e. the average distance of individual plans from the optimal plan length (cf. figure 6)

in relation to the complexity of the given problems. The complexity of a planning problem is defined as the number of objects of the type definitions specified in the given planning problem domain description. We grouped all test cases of the IPC3 test scenarios leading to 122 problems in total into complexity classes with an increasing number of objects.

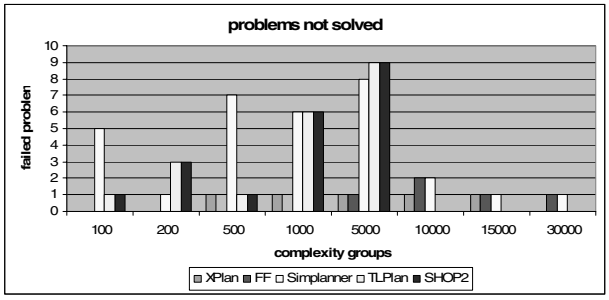


Figure 4. Completeness

First, we tested the completeness of planning (cf. figure 4). It turned out that XPlan fails to find a solution for problems of mid range complexity only, whereas the FF planner failed to solve the most complex problems. There were no results reported for TLPlan and SHOP2 for the last six test cases, they failed a lot in solving problems of low and mid range complexity, but performed very well in solving more complex problems. Main reason is that the HTN planners turned out to be equipped with methods that better enabled them to solve highly complex problems in most domains.

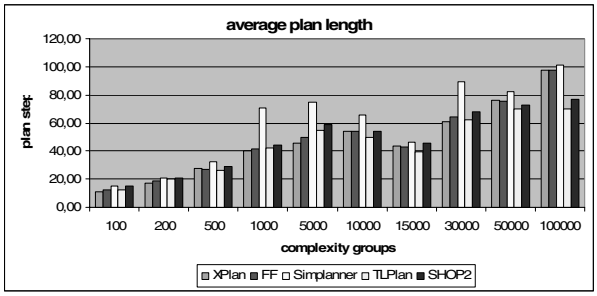


Figure 5. Average plan length

Figure 5 summarizes the results of testing the average plan length in relation to the complexity of the problem definition. The HTN planners produced shorter plans than their competitors with increasing complexity of the problem, whereas XPlan outperformed all other planners for given problems of low and mid range complexity.

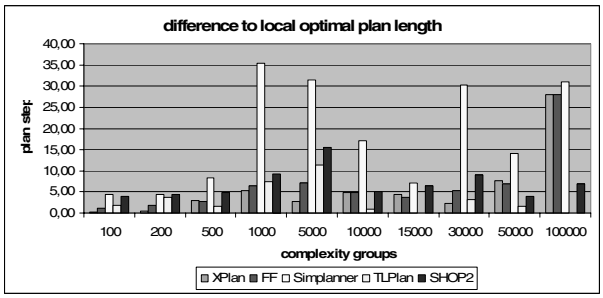
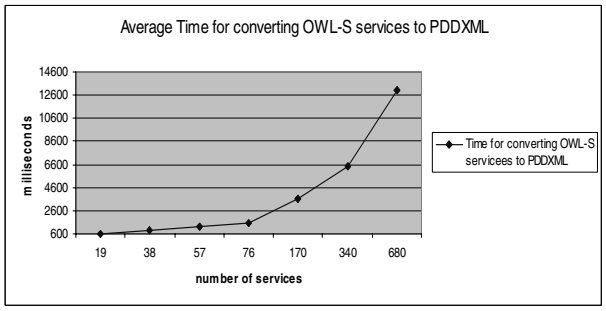


Figure 6. Average plan quality

Finally, we measured the average plan quality in terms of the average distance of individual plans from the optimal solution of a given problem (cf. figure 6). That is, we counted the number of additional plan steps of a solution generated by an individual planner compared to that of the shortest plan created for the given problem, averaged over all test cases per complexity class. In this respect, except for the most complex problems, XPlan outperformed the other planners



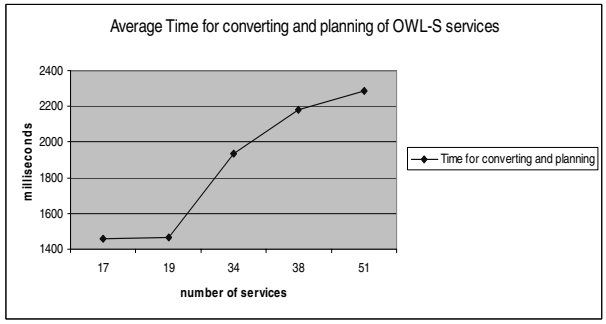


Figure 7. Average run time for conversion and planning by OWLS-XPlan

.

We did not have specific information about the underlying computing hardware used in the IPC3 competition for run time measurement. Figure 7 shows the reasonably fast run time of converting and planning by OWLS-XPlan on a Siemens-Fujitsu Amelio 1425 notebook with 1.8 Ghz Intel Centrino, and 1 GB RAM.

4. Implementation

OWLS-XPlan has been implemented in Java and C++, and provides an integrated graphical user interface (cf. figure 8 and 9).

Figure 8. OWLS-XPlan graphical user interface (1)

The planning component XPlan is implemented in C++, uses the Microsoft MSXML parser for PDDXML definitions and generating plans in XML format. Besides, XPlan also provides an interface for direct interchange of

planning data without having to use PDDXML as interchange format. In addition, OWLS-XPlan provides an integrated PDDXML editor that allows the experienced user to directly change the planning goal, and edit the initial state ontology for a given planning problem.

Figure 9. OWLS-XPlan graphical user interface (2)

This initial state ontology is assumed to be provided to the system for planning; we acknowledge that this assumption might be a major hurdle for inexperienced users to actually use the OWLS-XPlan software, so we are currently working on a novice user query interface for OWLS-XPlan version 2. The resulting plan is being displayed (cf. fig. 17) and can be further optimized with respect to given QoS parameters by means of ILP based optimization with newly available equivalent services.
OWLS-XPlan v1 has been made publicly available to the semantic web society at the portal semwebcentral.org on December 16, 2005 [15].

5. Application Health-SCALLOPS

The service composition planner OWLS-XPlan is used in an agent based mobile eHealth system for emergency medical assistance (EMA) planning tasks, called Health-SCALLOPS. OWLS-XPlan runs on the server of a national EMA centre to support the planning of patient relocation to selected hospitals, or patient repatriation. Medical transport services are offered on line by a variety of medical transport companies in the internet.

Use case scenario. For the use case of Health-SCALLOPS as sketched in figure 10, we developed 33 appropriate business application services in OWL-S 1.0 with imported OWL ontologies. We distinguish between the following five roles Health-SCALLOPS users can take

1. Patient, or someone acting on his/her behalf,
2. EMA centre for medical mission planning,
3. Emergency physician with an ambulance car,
4. Hospital physician for local treatment and triggering of relocation to a selected hospital, and
5. Health insurance of the patient.

Figure 10. Health-SCALLOPS use case (overview)

Suppose, for example, that Mika meets with an accident. The emergency call to the emergency medical assistance centre (EMA) is given by Mika using his personal Health-SCALLOPS software agents on his PDA. In response to this call, EMA is planning an appropriate medical mission by use of OWLS-XPlan, and alerts the emergency medical doctor in an ambulance car requesting him/her to accomplish this mission. The mission data contain not only the exact GPS position of and summary of the situation given by the patient but the route to the scene and the nearest selected hospital. After providing first aid, the emergency doctor sends the mission data and pre-diagnosis to the hospital doctor to allow for preparation of emergency treatment. Upon arrival at the hospital, the doctor recognizes an additional fissure of Mika's eye that requires further special treatment in an eye-clinic abroad. On behalf of Mika, the doctor asks the EMA centre to plan for this mission of medical patient transport planning. Given that a variety of different medical and non-medical transport companies provide their services on the semantic web, EMA is composing an appropriate transport plan for Mika which a distinguished EMA assistant is then executing.

Mobile Health-SCALLOPS. The mobile graphical user interface of Health-SCALLOPS provides role-based

functionality to the individual user, and has been implemented in Java for running under WinMobile 2003 on HP's iPAQ PocketPC series 5500 (cf. figure 11).

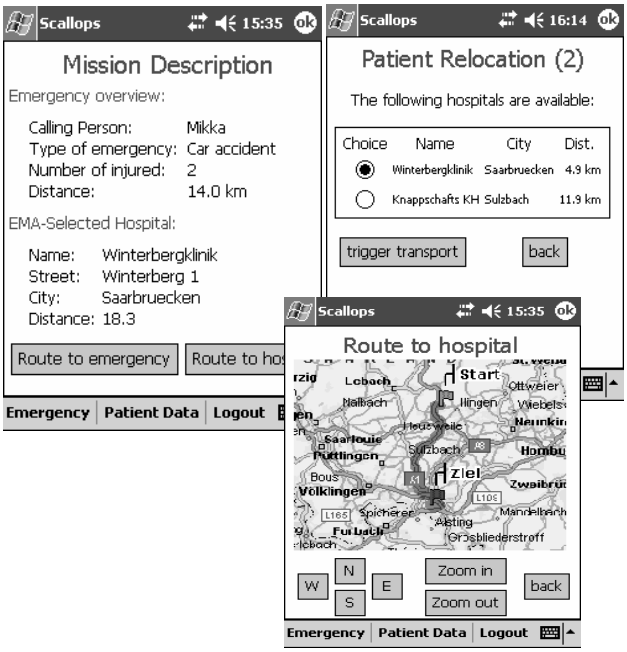


Figure 11. Mobile Health-SCALLOPS GUI (part of)

It allows for the initialization (by the patient, the eyewitness of an accident, or hospital doctor), the planning (by EMA centre), and the execution of EMA missions (by mobile emergency physician in ambulance car), and also offers GPS-based route planning.

Secure and executable composition plans. In Health-SCALLOPS, we use a variant of OWLS-XPlan that generates privacy preserving and executable medical mission plans. For ensuring the executability of planned sequences of mission related services on the WSDL-grounding level (operation modes, message types) we adopted the approach presented in [16]. In particular, the planning oriented matchmaker module, OWLS-MXP, a variant of OWLS-MX [17], is responsible for continuously checking all pairs of available WSDL service groundings of advertised OWL-S services for their I/O composability at data type level. In particular, it checks whether all pairs of XSLT data types of output-input parameter bindings as stated in the OWL-S process models are coherently matching (either type equivalence or type subsumption). This is to ensure a valid data flow between consecutive services of the semantic service composition plan. OWLS-MXP provides OWLS-XPlan with the set of currently available OWL-S services each which annotated with a list of grounding composable services. This enables OWLS-XPlan to check at the end of each planning step whether the actual composition plan extended by one helpful action (cf section 2) is also executable.

For verifying whether the generated plan preserves the privacy of patient data before actually executing it, the secure composition planning agent (SCPA) of the EMA centre in the Health-SCALLOPS application scenario performs as follows [18].

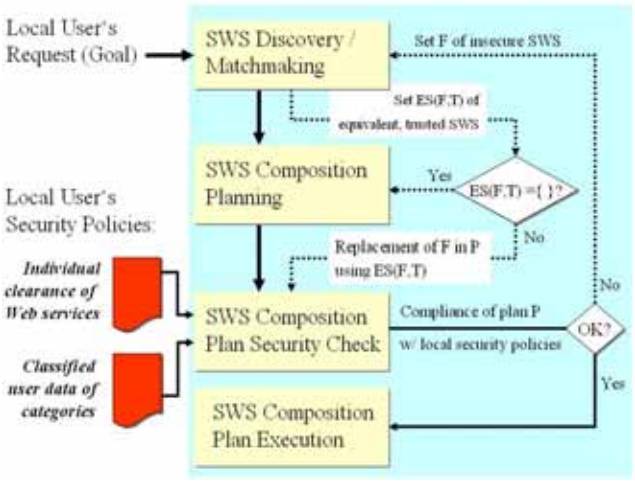


Figure 12: Secure Health-SCALLOPS service composition planning (overview)

The SCPA gets the request for some desired service in OWL-S from the user, as well as her local security policies in terms of the security classification of personal data and clearance of known OWL-S services as input. It then attempts to discover services that are semantically relevant to the request using the integrated OWLS-MX matchmaker.

In addition, it collects the corresponding security types published by the respective web service provider agents. If the matchmaker finds services detected as being equivalent to the requested one, it directly passes the top ranked one according to its QoS value to the security checking module for letting it verify whether its published security policy complies with both given local security policies and the web service's security type.

If no equivalent service is found, the OWLS-MX module passes the set of services to its composition planner, named OWLS-Xplan. In case a composition plan with more than one service is generated, the compliance of published security types of all web services involved in the plan is checked against the local security policies of the user.

In contrast to usual access control mechanisms, the security checking of the SCPA relies on type-based information flow analysis. Thereby, the approach also includes dynamically computed data of web services and

their security classification, and its proliferation to other services. In any case, the composition plan gets executed only if the security types of all web services meet the local security policies. So, the plan as a whole is formally verified as being secure. Otherwise the SCPA triggers a re-planning activity to be performed as follows. The security checker provides the matchmaker module with a set F(P) of services of plan P that caused P to not comply with the local security policies, in order to select one semantically equivalent service with a different published security policy for each or at least some of them. If successful, the composition planner simply modifies the original plan considered by replacing each service in F by its substitute, and returns the modified plan to the security checker for verification. If there exists no services in F(P) for which equivalent service can be found (and which are not yet tried), the composition planner generates a new plan by means of heuristic replanning [9]. In any case, if the modified plan is also provably insecure, the SCPA repeats the same procedure until a secure composition plan is generated, or it returns a failure otherwise.
More details on the security checking of OWL-S service plans generated by OWLS-XPlan are provided in [18].

6. Related work

There exist quite a few different approaches to service composition planning in the literature. They can roughly be classified into process oriented approaches, and data or signature oriented approaches. Members of the first presumes a goal that specifies the global behaviour of the desired service in terms of the set of possible desired conversations, or process flow to be accomplished by synthesizing the process models of available services that can either be modified during composition [2], or not [1]. Specification of the behaviour usually takes the form of FSMs, situation calculus [8], or linear temporal branching logic formulas. Any signature-oriented or data-driven composition approach does not take the process of a service into account but tries to instantiate a goal specification given by the signature of a desired service, i.e. its input/output behaviour together with constraints and user preferences only. Such an instance is a sequence of atomic or other composite services considered as black boxes that accomplishes the goal.

OWLS-XPlan falls into the latter category, and is tightly related to classical planning in AI. An overview of AI planning techniques and their application to the Web service composition planning problem is provided in [9]. An accessible approach to solutions of the problem of cyclic composition planning via model checking is in [14].

The service composition planner that is most relevant to OWLS-XPlan is Shop2 (Simple Hierarchical Ordered Planner 2) developed at the University of Maryland, USA [12]. Shop2 is a hierarchical task network (HTN) planner well-suited for working with the hierarchically structured OWL-S process model. Shop2, like HTN planner in general, replaces those elements of the provided methods (workflows) by special methods or atomic actions until the composition plan contains only atomic actions that correspond to available web services. During planning, web services are not executed, hence do not affect the world state.

Both XPlan and Shop2 base on the closed world assumption, use PDDL, allow external (call-back) functions, and generate totally ordered and instantiated plan sequences for a given initial state, goal and planning domain. However, among others, they differ in their way of planning. In essence, Shop2 plans are generated by use of pre-given decomposition rules (methods), hence a solution to the planning problem is not always guaranteed to be found [7]. In contrast, the hybrid XPlan as part of OWLS-XPlan first tries to plan by means of method decomposition, and if this is not successful, it exploits its relaxed graph plan algorithm to find a solution, if it exists. In addition, for decomposition of given methods it is using only relevant parts by discarding useless actions, thereby reducing the plan size in total.

7. Conclusion

We presented the implementation, evaluation, and application of our OWL-S service composition planner OWLS-XPlan. Selected emergency medical assistance related services described in OWL-S 1.1 and corresponding OWL ontologies are converted to initial state and goal descriptions in PDDL 2.1. These are then used by the fast planner XPlan for generating an execution complete composition plan. Results of experimental evaluation of XPlan show its top performance compared with other selected AI planners. OWLS-XPlan is used in an agent based mobile e-Health system for emergency medical assistance planning tasks.

8. References

[1] D. Berardi, D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Mecella. Automatic service composition based on behavioral descriptions. *Journal of Cooperative Information Systems, 14(4)*, 2005.

[2] T. Bultan, X. Fu, R. Hull, and J. Su. Conversation specification: A new approach to the design and analysis of e-service composition. *Proceedings of World Wide Web Conference WWW, Budapest, Hungary*, 2003.

[3] I. P. Competition. IPC3. *Homepage: http://planning.cis.strath.ac.uk/competition/*, 2002.

[4] J. Hoffmann. A heuristic for domain indepndent planning and its use in an enforced hill-climbing

algorithm. *Proceedings of 12th Intl Symposium on Methodologies for Intelligent Systems, Springer Verlag*, 2000.
[5] J. Hoffmann. The Metric-FF planning system: Translating Ignoring Delete Lists to Numeric State Variables. *Artificial Intelligence Research (JAIR), vol 20*, 2003.
[6] J. Hoffmann and B. Nebel. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research (JAIR)*, (14):253–302, 2001.
[7] A. Lotem, D. Nau, and J. Hendler. Using planning graphs for solving HTN problems. *Proceedings of AAAI/IAAI conference, USA*, 1999.
[8] S. McIllraith and T. Son: Adapting Golog for composition of semantic Web services. *Proceedings of International Conference on Knowledge Representation and Reasoning KRR, Toulouse, France*, 2002.
[9] J. Peer. Web Service Composition as AI Planning: A Survey. *Technical Report, University of St. Gallen, Switzerland Available at http://elektra.mcm.unisg.ch/pbwsc/docs/pfwsc.pdf*, 2005.
[10] M. Schmidt. Ein effizientes Planungsmodul fuer die lokale Planungsebene eines InteRRaP Agenten. Master's thesis, Universitaet des Saarlandes, 2005.
[11] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau. HTN planning for web service composition using SHOP2. *Journal of Web Semantics, 1(4)*, pages 377–396, 2004.
[12] D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau. Automating DAML-S web services composition using SHOP2. *Proceedings of the 2nd International Semantic Web Conference (ISWC2003), pages 20-23, Sanibel Island, Florida, USA*, 2003.
[13] M. Klusch, A. Gerber, M. Schmidt: Semantic Web Service Composition Planning with OWLS-XPlan. *Proceedings of the AAAI Fall Symposium on Semantic Web and Agents, Arlington VA, USA, 2005.*
[14] A. Cimatti, M. Pistore, M. Roveri, P. Traverso: Weak, strong, and strong cyclic planning via symbolic model checking, *Artificial Intelligence,147(1/2), pp. 35 - 84, 2003.*
[15] OWLS-XPlan: http://projects.semwebcentral.org/projects/owls-xplan/
[16] B. Medjahed, A. Bouguettya, and A.K. Elmagarmid. Composing Web services on the semantic Web. *Very Large Data Bases (VLDB), 12(4),* 2003.
[17] M. Klusch, B. Fries, K. Sycara: Automated semantic web service discovery with OWLS-MX. Proceedings of International Conference on Autonomous Agents and Multiagent Sys6tems (AAMAS 2006), Hakodate, Japan, ACM Press, 2006
[18] D. Hutter, M. Klusch, M. Volkamer, A. Gerber: Provably secure execution of composed semantic web services. *Proceedings of the 1st International workshop on Privacy and Security of Agent-Based Collaborative Environments (PSACE), Hakodate, Japan,* 2006