Mixed Initiative Use Cases for Semi-Automated Service Composition: A Survey

Jan Schaffner Harald Meyer

Hasso-Plattner-Institute for Software Systems Engineering,
Prof.-Dr.-Helmert-Str. 2-3,
14482 Potsdam, Germany
{jan.schaffner, harald.meyer}@hpi.uni-potsdam.de

ABSTRACT

Semi-automated service composition with mixed initiative interactions, where both user and machine jointly contribute to the creation of composed services, is currently subject to intensive research. In this paper, we give an overview over recent research approaches by presenting three different semi-automated service composition tools. As the main contribution of this paper, we introduce three mixed initiative use cases characteristic for semi-automated composition, which we have extracted and generalized from the presented approaches and then extended. Based on these use cases and additional distinctive properties, we give a qualitative evaluation of the presented approaches.

Categories and Subject Descriptors

I.2.0 [**Computing Methodologies**]: Artificial Intelligence

General Terms

Human Factors

Keywords

semi-automated, service composition, mixed initiative, web service, business process management

1. INTRODUCTION

While automated planners are able to reduce complexity, inflexibility and error-proneness akin to the creation of composed services, several drawbacks can be identified: Automated planning relies on the availability of complete formal representations of the domain knowledge and the "state of the world".

The task of formally specifying a domain in sufficient fidelity so that it can be used for automated planning presents a tremendous challenge. If the domain knowledge is incomplete, an automated planner might not be able to produce a plan. In contrast, a human planner can draw upon his experience with a specific domain when he or she creates a composed service.

In a planning problem, the state of the world is represented by the initial state, which serves as a starting point for automated planners to find a path to a supplied goal. It is impossible to encode the complete world state in the initial state of a planning problem. Therefore, matchmakers will always plan with limited information about the world state, which can result in failing to deliver a plan.

The incorporation of the matchmaking technologies as used by automated planners into a semi-automated modeling tool for creating service compositions has several advantages: On the one hand, the problems of complexity, inflexibility and error-proneness of the created service compositions can be reduced or eliminated by building new mixed initiative use cases on top of the Semantic Web technologies used by automated planners. On the other hand, the modeler can rely on his or her experience with composing services. The modeler also has the opportunity to incrementally learn from the scenario and to refine his or her goals while developing the plan. This opposes the problem of planning with incomplete information faced by fully automated planning environments.

Moreover, the fact that fully automated service composition methods do not require a human in the loop poses an organizational and juridical impediment: It may be desirable that a concrete person is responsible for a particular business process. As this lowers the industry acceptance of automated planning techniques, their transition from research to industry is progressing slowly.

For the above reasons, semi-automated service composition is a strongly researched topic at present. However, a general definition of the term "semi-automated composition" is missing. There is also no overview of existing approaches available, neither do we have common understanding of the functionality that is characteristic for semi-automated composition. The goal of this paper is to fill this gap and to address these shortcomings.

The remainder of this work is organized as follows: Section 2 will give an overview over current research efforts in the field of semi-automated service composition. In section 3 we will introduce three mixed initiative use cases that enable semi-automated modeling tools to overcome the problems of error-proneness, complexity and inflexibility described above. Section 4 presents an evaluation of the presented existing approaches based on the mixed initiative use cases they support as well as additional criteria. Section 5 concludes the paper.

IW-SOSE'06, May 27–28, 2006, Shanghai, China.

2. EXISTING APPROACHES

The purpose of this section is to give an overview of current research efforts regarding semi-automated service composition. Three approaches will be presented according to their main characteristics.

2.1 Web Service Composer

Sirin, Parsia and Hendler [9] present a prototypical implementation of a composer for Web services. Their tool allows creating compositions of Web services that are semantically specified with OWL-S [5] and their execution.

The created service compositions can in turn be stored as OWL-S "process models". Process models are a part of OWL-S ontologies which is normally used to encode the choreography for a described service. Well-known control constructs from the area of Workflow Management can be used within OWL-S process models. It is therefore a suitable format for representing composed services.

The focus of their work is on filtering the list of available services at each composition step and thus helping the user to select the appropriate services.

In order to create a composed service, the user follows a backward chaining approach. He or she begins with selecting a Web service that has an output producing the desired end result of the composition from a list of all available services. Next, the user interface presents additional lists connected to each OWL input type of the service producing the end result. In contrast to the first composition step, these lists do not contain all available services: They contain only those services that generate an output compliant to the particular input type they are connected to. An output of a service A is compliant to an input of a service B, if their types are exactly the same or if the output of A subsumes the input of B (i.e., the input of B is a specialization of the output of A). If a service is selected from the list of compliant services, this service's inputs must again be produced by selecting services producing compliant outputs. This is repeated until the user decides at one point to provide the inputs that are not connected to a compliant service by entering them as input values (or connecting them to compliant services that have no input parameters).

Creating the composed service by forward chaining (i.e., starting with the first activity in the process instead of the last one) is planned but not implemented in their prototype.

In addition to filtering on the compliance of the services in terms of their inputs and outputs, the user can apply further filtering based on the nonfunctional properties of the services. This only works for services that adhere to a specific OWL-S "service profile" (i.e., they implement the service profile). Once the user has selected a service profile, the system renders an UI element which allows him or her to provide values for the nonfunctional properties that are specified for the selected service profile. The user can then apply the filter, thereby further restricting the set of services that are presented for the current composition step.

2.2 CAT

Kim, Spraragen and Gil introduce CAT (Composition Analysis Tool) [3], a tool which illustrates their approach to interactive workflow composition.

The focus of their work is to assist the user in the creation of computational workflows. The authors' work is not directly related to service composition. However, we can conceive a computational workflow as a service composition. The activities of the workflow are represented by services that realize data transformations.

The authors have developed their own knowledge base format, which they use to semantically describe the components that can be used in a workflow and their input and output parameters: "Component ontologies" describe hierarchies of components, from abstract-level components to executable components. An abstract component represents a common set of features that applies to all components of that type. "Domain ontologies" semantically specify the data types which can serve as inputs and outputs of the components described in the component ontologies.

In CAT, the user can add components to the composition at any time. There is no need for the user to follow a strict backward or forward chaining composition. The "end result" of the composition can be specified by declaring outputs produced by components as the end result (or as a part of it). Control flow in CAT is described by explicitly linking inputs and outputs of different services together. Values of input parameters can also be default values from the respective ontologies or values entered by the user.

Instead of filtering the set of services that can be included in a composition, CAT provides a list of suggestions about what to do next. These suggestions resolve errors and warnings, which are also displayed. The idea is that consequently applying suggestions will produce a "well-formed" workflow as a result. The authors therefore introduce a set of properties that must be satisfied by the composition in order to be well-formed. These properties ensure that

- the composition has an end result,
- all components' inputs are satisfied
- all components have been specialized to executable components
- all components produce outputs relevant for producing the end result
- for all links between components there is a "subsumes"-relation between the output of one component and the input of the other component
- the composition does not contain redundant links or components.

Depending on whether these properties are satisfied or not, the ErrorScan algorithm (which is also provided in [3]) determines which suggestions are presented to the user.

CAT uses heuristics to determine the ordering of the suggestions, so that more recent and more severe errors are displayed before warnings that do not necessarily have to be resolved in order that the workflow is well-formed. It is noteworthy that the suggestions in CAT have the property of being corrective or additive: Applying a suggestion never causes more errors than it resolves.

2.3 PASSAT

Myers et al. present PASSAT (Plan-Authoring System based on Sketches, Advice, and Templates) [6], an interactive tool for constructing plans. PASSAT is not directly concerned with the creation of composed services, but its concepts can be mapped into the context of service composition.

PASSAT is based on hierarchical task networks (HTN) [10], while the model has been extended to realize some concepts that are outlined below. In HTN planning, a task network is a set of tasks (or service calls) that have to be carried out as well as constraints on the ordering of these tasks. Moreover, it consists of a set of constraints that must be valid before the execution of the tasks and information about how the tasks instantiate variables. Because the variables (partly) describe the state of the world before and after the execution of a specific task, the constraints on these variables can be used to express preconditions and effects.

The HTN based approach naturally imposes top-down plan refinement as the planning strategy the user must adhere to: The user can start by adding tasks to a plan and refine them by applying matching HTN templates. A template consists of a set of subtasks that replace the task being refined, as well as the preconditions and effects of applying individual tasks and the entire template. It is noteworthy that the user has the possibility to override unmatched constraints when applying a template. This is especially desirable when comprehensive domain knowledge (i.e., a collection of templates) cannot be provided. Task refinement is repeated until the plan contains no activities that can be further expanded.

A core feature of PASSAT is its automated planning mode, which allows the user to have the system expand all remaining tasks, applying the templates that are currently available to the system.

PASSAT also features an "advice" mechanism that allows the user to specify high-level policies for the overall plan being created. These policies are global constraints that restrict the set of actions that the user can undertake when developing a plan. However, they can be relaxed and overridden and need not to be necessarily satisfied to reach the overall goal. The automated planning mode also takes these policies into account when it selects the templates for refining the open tasks.

Opposing the strict top-down refinement approach implied by the use of HTN networks, PASSAT provides a "plan sketch facility": This allows the user to freely arrange tasks that need not to be necessarily fully specified and that can reside on different layers of abstraction (regarding the template hierarchy). After the user has outlined a plan sketch, the system tries to find possible expansions by applying matching templates. The user can then choose one of these expansions to be included in the plan and return to the normal planning mode.

PASSAT also informs the user about open tasks and outstanding information requirements in order for the plan to be completed. Therefore, it presents the user with an agenda of actions such as "expand task", "instantiate variable" and "resolve constraint".

The system helps the user to choose from the applicable templates at a given composition step by keeping track of past user experience: A statistic about how often a template has been applied in plan refinement is encoded in the templates.

3. MIXED INITIATIVE USE CASES

In this section, we will describe three use cases characteristic for semi-automated service composition approaches that enable them to overcome the problems of complexity, inflexibility and error-proneness akin to conventional service compositions, as described in section 1. We extracted these use cases from the approaches presented in section 2. We generalized and extended them according to our own research findings.

The use cases "Filter Inappropriate Services", "Suggest Partial Plans" and "Check Validity" will be developed throughout the remainder of this section. Before investigating these use cases in detail, we will refresh our understanding of how to semantically describe the capabilities of a service.

3.1 Prerequisites

Services can be *information-providing*, *world-altering* or both. The execution of information-providing services results in a change of the information space at a given point in time. In OWL-S ontologies [5], the *inputs* and *outputs* of a service describe the data transformation that is accomplished by a service. If a service has world-altering capabilities, the preconditions and effects describe a part of the state of the world before and after its execution. Abbreviated, this information is called the IOPEs of a service. We will use this terminology throughout the remainder of this paper.

While service compositions are usually seen as a set of activities among which an ordering relation exists, we can also conceive them as a set of states and transitions. In doing so, the transitions denote the individual activities (i.e., the services or service calls).

The execution of a service results in a change of the state. The information space and the state of the world in a given state in a service composition depend on the IOPEs of the services that precede this state. A transition (i.e., a service call) from a state A to another state B is allowed only if the inputs and preconditions of the service describing the transition are satisfied in state A. The other way round, this transition implies that the outputs and effects are valid in state B. The outputs and effects that become valid at this point are added to the outputs and effects of the preceding states. These are all the outputs and effects that have either been produced by upstream service executions or that have been provided by the "environment" (e.g., data which is available to all services in a composition per definition). This is also depicted in figure 1. The circles represent states, the rectangles represent transitions (i.e., service calls) and the dotted lines represent the outputs and effects that make up the information and world state. Please note that an effect has the power to negate other effects that may hold prior to the execution of the service that produces the effect.

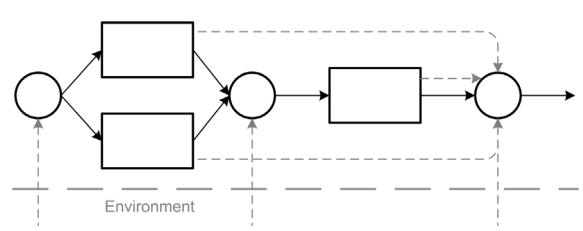

Figure 1: A state depends on the outputs and effects of all precedent services

A large number of service compositions can be modeled without preconditions and effects. Every composition which has the goal to realize a data transformation can be described only using inputs and outputs. However, if we want

to take preconditions and effects into account when creating composed services, we have to ensure that they can be evaluated later at runtime.

3.2 Filter Inappropriate Services

A major problem with creating composed services is that the number of activities that can be selected might be extremely high, depending on the domain. For instance, SAP's Enterprise Service Repository today contains more than 500 services, and growing [7]. As users of a tool for creating service compositions cannot oversee such a vast amount of available options, it is desirable to filter the set of available services. Such filtering can be done based on semantic service descriptions.

When creating composed services, users select services and add them to the composition. In a given state, it is possible to filter the selection according to semantic descriptions: It is desirable that services requiring inputs and preconditions that are not satisfied in a given state will be filtered, effectively reducing the number of choices presented to the user.

While filtering based on the services' IOPEs restricts the set of presented services to those which are compatible with the current state, the set can be further restricted by filtering based on the nonfunctional properties of the services.

Nonfunctional properties do not only offer a possibility to record juridical relevant information like a publisher's name and address, but also quality indicators for services. Such indicators can be measures that address the performance (in terms of response time), error rate or robustness of a service, as well as issues like scalability, reliability, geographical coverage, invocation cost and many more.

When creating service compositions, the user may find himself in a situation where more than one available service offers the functionality that is needed to go to the next state. At this point, the editor should allow the user to assign values to the nonfunctional properties of the presented services. These values are then evaluated by the matchmaker and only those services that both provide the desired functionality and comply with the user-specified nonfunctional properties are presented for selection.

A technical issue that has to be resolved is the fact that it is unlikely that all the semantic service specifications for the services providing equivalent functionality contain the same set of nonfunctional properties. Possible strategies depend on the language concepts of the used semantic specification framework (e.g., OWL-S [5], WSMO [8] or WSDL-S [1]), and on whether or not the editor incorporates the concept of abstract services.

An abstract service represents a set of service capabilities (i.e., the functionality a service provides). Abstract services in composed services can be used to realize a late binding of the concrete services at runtime: The engine that executes the process can discover all currently registered services that implement the functionality specified in the abstract service.

As specifications for abstract services are generalizations of concrete services, it would make sense to annotate them with a set of nonfunctional properties that is common to all concrete services. That way, the editor could offer a selection of abstract services for which the values of their nonfunctional properties can then be assigned by the user.

The use case of showing the applicable services for moving to the next state can be further enhanced by ordering the list of possible services according to the "degree of match" that the matchmaker returns for a service: The services that require exactly the inputs and preconditions that hold in the given state (i.e., an "exact match") should be presented first. Further ordering of the list can be based on the minimal distance between the respective concepts in the taxonomy. This distance can be translated into a classification of the "goodness" of the match, according to Li and Horrocks [4].

Another (and probably more accurate) possibility to order the list of applicable services would be to consult ratings of how often the user has selected the particular services. As this does not directly involve service semantics, this will not be further detailed here. However, it would be conceivable to incorporate a rating facility in the semantic service descriptions. The editor could update the descriptions in order to maintain a nonfunctional property such as "user rating".

3.3 Suggest Partial Plans

Automated planners will always plan according to an algorithmic planning strategy, such as for example forward- or backward chaining of services, or both. Human planners will in contrast not always behave according to this schema when they model composed services. Users might have a clear idea about some specific activities that they want to have in the process, without a global understanding how the whole will fit together as a process.

A possible user behavior is to start modeling the composed service by adding some activities and chaining them together, and then continue with an activity with unsatisfied inputs and preconditions representing some state later in the composition. In such and similar cases, it might be desirable for the user to let the editor generate valid service chains that connect two unrelated activities. This is depicted in figure 2.

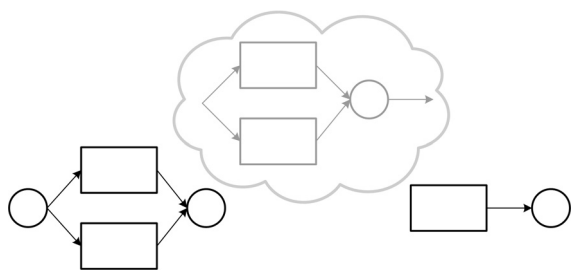

Figure 2: A partial plan capable of connecting two unrelated services

Connecting two unrelated activities in a composition constitutes a standard planning problem, given the state after the execution of activity A (the initial state), the state before the execution of activity B (the goal state), the set of activities, as well as the set of all possible states and the set of all possible state transitions. The two latter sets can be derived from the semantic specifications of the activities (i.e, the services). The problem is therefore handed over to an automated planning engine.

A special case of the "Suggest Partial Plans" use case is when activity B (i.e., the activity to connect to) is the last activity in the composition. When this function is executed, the editor finishes the service composition.

If there are multiple possibilities to get from activity A to activity B, the editor should present the alternatives. In

advanced scenarios it might also be useful if the editor presented a rating of the alternatives based on the aggregation of a nonfunctional property such as cost, for example.

In any case, the generated service chain should be editable by the user.

3.4 Check Validity

Because the human planner has full control over the modeling of the business process in a semi-automated environment, it is natural that errors are likely to be introduced into the composition. It is therefore necessary to provide the possibility to check the overall process validity.

Like the use cases that have been defined earlier in this section, the validity check can be realized using Semantic Web technology. A simple validity check would be to verify that the inputs and preconditions are satisfied for each activity in the service composition.

Such validity checks could be executed by the user at the end of the modeling process. However, in order to better support the user in creating composed services, validation should be interleaved with the actual modeling of the composed service: The user should be informed about unresolved issues in an unobtrusive way.

Unresolved issues arise from activities in the composition which violate one ore more aspect of a set of desirable properties for well-formed workflows, which has been introduced by Kim, Spraragen and Gil [3]. A workflow is well-formed, if

- one or more activities are contributing to the composition's "end result",
- the inputs and preconditions of every activity are satisfied,
- every activity is either an "end result" or produces at least one output or effect that is required by another activity,
- it does not contain redundant activities.

These properties should be checked after each user action so that the list of unresolved issues can be updated. To further assist the user, the editor could even suggest appropriate actions to resolve the open issues. Such actions include adding or removing activities to or from the composition as well as adding or removing control flow between two activities.

A problem that arises when suggesting actions is that the suggested action can introduce new errors to the composition. In doing so, a suggested action could produce more errors than it resolves.

4. EVALUATION

In the following, we will present an evaluation of the semi-automated composition approaches presented in section 2 based on the supported mixed initiative use cases as well as additional criteria.

4.1 Supported Mixed initiative Use Cases

In the following, we will give an overview of the mixed initiative use cases that are supported by the semi-automated composition approaches presented in section 2.

Web Service Composer filters the list of services that can be included in the composition at each composition step. This realizes the use case "Filter Inappropriate Services" that was presented in section 3 of this paper. However, the realization of this use case in Web Service Composer is restricted in two ways:

First, the tool only considers inputs and outputs, i.e., the mere data transformation that services realize. The preconditions that must be satisfied before the execution of the services and the effects that the executions of the services have on the state of the world are not taken into account.

Second, the selection of appropriate services is done per input of a downstream service that must be satisfied, which is due to the strict backward chaining approach imposed by the tool. This means in consequence that the plans constructed with the tool are not always optimal. For example, when one service operation delivers two outputs each of which satisfies a different input of a downstream service, this services operation has to occur twice in the composed service.

Web Service Composer supports two extensions of the use case "Filter Inappropriate Services" that have also been identified in section 3: First, the tool can further restrict the set of filtered services according to user-specified values of nonfunctional properties that are common to that set. Second, the list of filtered services which is presented to the user is ordered according to the goodness of match: Services that exactly produce a necessary input for a downstream service (i.e., an exact match) are ranked higher than services that produce outputs that subsume the necessary inputs.

PASSAT is the only tool of those included in this survey that partially supports the use case "Suggest Partial Plans". PASSAT is a tool for interactive plan authoring based on HTN networks. The user can invoke an automated planning mode to expand open tasks in the plan. This can be seen as a specialization of the use case "Suggest Partial Plans" in the sense that partial plans can only be generated from the current state to a state in which the composition is finished, i.e. all tasks can be executed. However, this realization of the use case is restricted in the way that the user must have completed the plan on a high level of modeling - otherwise the task network cannot be expanded.

In section 3 we have described a possible extension of the "Suggest Partial Plans" use case: If there is more than one alternative for a partial plan, a ranking of user-specified nonfunctional properties should determine the order in which the alternatives are presented to the user. In PASSAT, the user can specify high-level policies (e.g., "maintain an overall cost total of less than $ 100") which are also taken into account when automated template expansion is performed. This can be seen as a realization of that extension, as the alternative for a template expansion that conforms best to the specified policies will be presented to the user.

PASSAT also supports the use case "Check Validity", as it interleaves a checking mechanism with the actual planning process: After each user action the system updates an agenda showing open information requirements that must be satisfied in order to have an executable plan. As an extension to this mechanism, PASSAT orders the agenda according to user-specified criteria.

Another, more thorough realization of the use case "check validity" can be found in CAT. Here, the tool checks at each composition step if the composition complies with a set of properties that describe the "well-formedness" of the composition. In case these properties are violated, the system

consequently presents a list of warnings and errors. As an extension of this use case, the authors present an algorithm that presents the user with suitable suggestions for next steps based on the evaluation of the well-formedness criteria. The applicability of CAT has been shown in the domain of seismic hazard analysis; however, it remains unclear why the authors opted for developing their own correctness criteria for computational workflows rather than building upon more established approaches to verify workflow correctness, such as the soundness criteria introduced by van der Aalst [11]. Also, the authors do not describe how their notion of well-formedness relates to the soundness criteria for workflows.

4.2 Additional Evaluation Criteria

The mixed initiative use cases of a semi-automated composition approach are its most important characteristic. However, there are more criteria that allow further distinction among such approaches. In the remainder of this section, these criteria will be presented and applied to the semi-automated composition approaches presented in section 2. Table 1 gives an overview of how we evaluate the presented approaches according to these criteria.

An important criterion for the user who created composed services is the way in which composed services can be modeled with the system he or she utilizes. As human planners are likely to feel constrained when they are forced to adhere to an algorithmic planning strategy, the tools should give the users maximum freedom in modeling their compositions.

Web Service Composer imposes a strict backward chaining planning strategy to the user. The user has to start with the last activity in the composition, i.e., the activity producing the desired end result. The inputs of this activity are then recursively satisfied until the first activity in the composition (e.g., a user input) is reached. Due to the strict backward chaining approach only the last activities of compositions created with Web Service Composer can determine the end results, which is also problematic.

Another criterion that is highly important to the user of semi-automated composition tools is whether a graphical user interface is provided. Web Service Composer provides the user with a graphical user interface, while CAT and PASSAT come with mere textual modeling environments. Especially for complex compositions, the user can hardly oversee the causal relations between the activities.

Semi-automated service composition approaches reason over domain knowledge that is specified in ontologies. In order to support the maintainability of the composed services that are created using semi-automated composition tools, standardized formats should be used for the ontologies. Because the formal specification of domain knowledge presents a tremendous challenge, organizations have to rely on available ontologies that have been created by other parties as building blocks for assembling their domain knowledge. Web Service Composer builds upon OWL-S, which is an open format. CAT and PASSAT in contrast are building upon proprietary formats for encoding domain knowledge.

When service capabilities and functionalities are specified in ontologies, the data transformation realized by a service's execution can be specified as well as the change in the state of the world that the execution of a service implies. Two semi-automated service composition tools out of the three presented in this survey only reason on the inputs and outputs of the services that can be included in the compositions (i.e., the data transformation that the services effect). However, a large number of possible applications (i.e., the set of computational workflows) can be described only using inputs and outputs. PASSAT is the only approach among those presented here that explicitly supports constraints on the state of the world.

When modeling composed services, we naturally expect the possibility to model control flow between the individual activities. None of the presented tools features control constructs. This is probably related to the fact that most tools reason only on the inputs and outputs of the services that can be included in a composition: If preconditions and effects are not considered, the control flow of a composition derives implicitly from the data flow. Being the only approach of those investigated in this paper that supports preconditions and effects, PASSAT lacks a notion of explicit control flow.

A semi-automated service composition tool should somehow ensure that the composed services being created with it can be executed. This can be done either by directly proving the user with an execution environment or by exporting the compositions into an executable format.

Web Service Composer allows to directly execute composed services by calling the individual services via the WSDL interface that is provided in the groundings of the OWL-S ontologies used. As the tool acts as the Web service client for all calls, it does not support complex choreographies.

In addition to its Web service execution functionality, Web Service Composer is able to store the composed services as OWL-S process models. Together with an OWL-S grounding, the service compositions are executable on platforms other than Web Service Composer. CAT and PASSAT are unable to store composed services in an open format. The de-facto standard format for executable service compositions is WS-BPEL [2]. It is striking that none of the tools offers WS-BPEL export functionality.

5. CONCLUSION

In this paper, we have given an overview of recent research approaches towards semi-automated composition by presenting three different semi-automated service composition tools.

As the main contribution of this work, we have developed three mixed initiative use cases for semi-automated composition, which we have extracted and generalized from the presented approaches and then extended.

Based on these use cases, we have given a qualitative evaluation of the approaches presented. Additionally, we have developed a number of distinctive properties that are characteristic for semi-automated approaches and applied them to the existing ones.

Our results show that no semi-automated modeling tool known to us is complete in terms of the functionality it provides: None of the presented existing approaches covers all the mixed initiative use cases that have been developed in section 3 of this paper.

While the use cases "Filter Inappropriate Services" and "Check Validity" are both realized in two semi-automated composition approaches, the use case "Suggest Partial Plans" is only addressed in one of them: PASSAT [6] partly implements this use case with its feature of automated plan expansion. Plans in PASSAT are represented as Hierarchical

	Web Service Composer	CAT	PASSAT
Imposed planning strategy	Backward chaining	None	Top-down refinement
Modeling environment	Graphical	Textual	Textual
Knowledge base	OWL-S	Non-standard	Non-standard
Reasoning	Output-input subsumption	Output-input subsumption	Takes complete IOPEs into account
Control constructs	Not provided	Not provided	Not provided
Compositions are executable	Tool acts as Web service client	No	No
Output format	OWL-S process model	Non-standard	Non-standard

Table 1: Evaluation according to additional criteria

Task Networks (HTN). The user can invoke an automated planning mode that expands any open tasks within a plan.

The presented approaches could hardly be extended in order to provide the missing functionality in terms of the presented mixed initiative use cases. Two out of the three tools presented are only capable of reasoning over input and output types, which only covers the class of information-providing services. In contrast, the described mixed initiative use cases also consider the class of world-altering services. Another problem with extending the functionality of the existing approaches is that all of them use different ontology formats to represent domain knowledge, making it difficult to integrate them.

The ontology format underlying a semi-automated composition approach is also important when we consider the fact that the task of formally specifying domain knowledge is very delicate. It will be necessary for organizations utilizing this kind of technology to use ontologies that have been created by other parties. It is therefore necessary to build upon standards. Two of the three presented tools implement a proprietary ontology format, which hinders the exchangeability of domain knowledge between organizations.

Another problem common to all presented existing approaches is their weak usability. Two out of three approaches provide mere textual interface. The tools with graphical user interfaces are prototypical proof-of-concept implementations. None of the presented approaches provides a user interface that would enable domain experts to create composed services without training. Thus, the industrial applicability of semi-automated service composition technology is limited today.

6. REFERENCES

[1] R. Akkiraju et al. Web Service Semantics - WSDL-S. Technical report, Apr 2005. `http://lsdis.cs.uga.edu/projects/METEOR-S/WSDL-S`.

[2] A. Arkin et al. Web Services Business Process Execution Language Version 2.0. Technical report, Oct 2005. `http://www.oasis-open.org/apps/org/workgroup/wsbpel/`.

[3] J. Kim, M. Spraragen, and Y. Gil. An intelligent assistant for interactive workflow composition. In *IUI '04: Proceedings of the 9th international conference on Intelligent user interface*, pages 125–131, New York, NY, USA, 2004. ACM Press.

[4] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the Twelfth International World Wide Web Conference (WWW2003), 20-24 May, 2003, Budapest, Hungary*. ACM, 2003.

[5] D. Martin et al. OWL-S: Semantic Markup for Web Services. Technical report, Nov 2003. `http://www.daml.org/services/`.

[6] K. L. Myers et al. PASSAT: A User-centric Planning Framework. In *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*, Houston, TX, USA, 2002. AAAI.

[7] B. Richardson. The View from SAPPHIRE: No Desktop Left Behind, 2005. `http://www2.cio.com/analyst/report3603.html`.

[8] D. Roman, H. Lausen, and U. Keller. D2v1.2. Web Service Modeling Ontology (WSMO). Technical report, Apr 2005. `http://www.wsmo.org/TR/d2/v1.2/20050413/`.

[9] E. Sirin, B. Parsia, and J. Hendler. Filtering and Selecting Semantic Web Services with Interactive Composition Techniques. *IEEE Intelligent Systems*, 19:42–49, 2004.

[10] A. Tate. Generating Project Networks. In *Proceedings of the Fifth Joint Conference on Artificial Intelligence, Cambridge, MA, USA*, pages 888–893. Morgan Kaufmann Publishers, 1977.

[11] W. M. P. van der Aalst. Verification of workflow nets. In *ICATPN '97: Proceedings of the 18th International Conference on Application and Theory of Petri Nets*, pages 407–426, London, UK, 1997. Springer-Verlag.