Multi-Agent Systems in Practice When Research Meets Reality

Marco Lützenberger, Tobias Küster, Nils Masuch, and Johannes Fähndrich
DAI-Labor, Technische Universität Berlin
Ernst-Reuter-Platz 7
10587 Berlin, Germany
{firstname.lastname}@dai-labor.de

ABSTRACT

The applicability and usefulness of agent technology for real world problems is still a matter of discussion—even within the AAMAS community. While theoretical models have significantly matured and led to an exciting variety of results, there were only few attempts to validate these models in reality. In this paper we aim to report on challenges that occurred when agent theory was used to approach real world problems. In doing so, we focus on the concept of planning, since planning currently appears to be one of the most relevant concepts for distributed real world applications. We examine four agent-based applications and emphasise problems that occurred when agent theory was practically applied. We also show how these problems were countered and propose more general solutions based on these tailored and context-specific approaches. The aim of this paper is to bring the two diverging branches of agent theory and practice back together in order to better adapt agent technology to the requirements of professional software.

Keywords

Automated planning; Agent theory & practice; Challenges; Industrial adoption; Service oriented architecture

1. INTRODUCTION

Research in agents and multi-agent systems has significantly matured. It seems that we, as a community, have shifted from initial experiments in building individual agents or multi-agent systems to rigorous theoretical foundations based on increasingly sophisticated computational models. Yet, despite these advances, there were only few attempts to validate these models in real world applications. The reasons for this are not entirely clear, though, the problem was recognised by the AAMAS community. At AAMAS 2015, for instance, a separate panel was being held with the objective to identify problems that hamper the 'real world validation' of agent-technology and to determine steps to counter these problems. In this panel, the community aimed to clarify the questions on whether 'we as a field are doing enough for this validation' [23, p. 30] and 'what should we do to encourage such validation in real applications' [23, p. 30].

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, C. Jonker, S. Marsella (eds.), May 9–13, 2016, Singapore.*

It was not possible to find satisfying answers to these questions, though, the community agreed that it is necessary to foster the relationship between those doing foundational scientific research and those making autonomous agents and multi-agent systems a commercial or public policy reality. In more particular, practitioners were encouraged to share their experiences in applying agent-theory to real world applications and to report on problems in order to encourage improvements of theoretical foundations and to trigger further research.

In this paper we reply to this call and report on problems and limitations that were experienced during the development of real world agent applications. In doing so we focus on applications that were implemented by agent frameworks with support for the *service oriented architecture* (*SOA*) [11] design principle. When analysing existing agent applications, we recognised that a lot of works were implemented through agent frameworks with support for SOA. We explain this phenomenon with the fact that a programming framework, which merges SOA and agents innately provides many features that are required for the development of (professional) real world software, e.g. transparently connecting to and integrating with third party services and support for common protocol standards. In fact it has been argued [5, 33, 35, 40, 47] that connections to established programming paradigms is a significant factor for industrial technology adoption, thus, we consider our focus to be justified.

We start this paper by surveying concepts that have their foundation in (theoretical) agent research and that currently play a role in professional software engineering (see Section 2). For this paper we focus on one particular concept that repeatedly caused problems when applied to real world applications, namely *planning*. Thus, in Section 2.2 we elaborate on the concept of planning with particular emphasis on planning in SOA and agent environments. Based on a fundamental understanding for the latest advances in planning, we look into professional agent-based applications, where the concept of planning plays a fundamental role and discuss difficulties that occurred when planning theory was used for the implementation of these applications (see Section 3). In Section 4, we structure these problems into more general categories and summarise how planning concepts had to be adjusted in the previously presented applications in order to fit their purpose. Based on these 'tailored solutions', we identify *where* and outline *how* the theoretical foundations of agent planning have to be improved in order to cope with real world requirements. We conclude this work in Section 5.

2. SERVICE ORIENTATION

In this work we put a focus on applications that were implemented through agent frameworks with support for SOA. SOA is an established design principle, which made its way into professional software engineering and applications. Any merger of SOA and agent-based techniques thus provides support for developing autonomous and highly distributed entities, which can connect (in a standardised fashion) to a broad spectrum of available real world functionality (or services). In this section we provide an overview over principles of and current trends in merging SOA and agents. Due to its importance for real world applications we put particular emphasis on the concept of planning. In the next section we use this fundamental understanding in order to analyse existing agent applications.

2.1 Semantic Services

Currently there are four SOA trends, which are relevant for the agent world. Each trend provides remarkable solutions, yet, it has to be mentioned that some of these solutions are far away from practical applicability. Following Papazoglou et al. [38], these trends can be classified as:

- Service Foundations,
- Service Composition,
- Service Management,
- Service Engineering.

The category *Service Foundations* comprises principal concepts, e.g. a service-oriented middleware that connects service providers and offers service deployment and discovery mechanisms. The *Service Engineering* category, in turn, comprises the specification of services and of their internal processes.

Service Composition is based on concepts that can be classified as Service Foundations and describes the aggregation of multiple services to a value-added service, which, itself, may be a part of more complex compositions. In the beginning, orchestration was done manually, e.g. by means of service description standards like *WSDL* and orchestration languages like *WS-BPEL* [37]. These mechanisms were able to foster interoperability between services (and thus between agents that offer services), yet, the workflow was predefined, static and not dynamically adaptable unless the designer clearly specified alternatives or abstract adaptation patterns [48].

A more flexible approach to compose services is referred to as *automated composition of services*. Automated service composition is a concept, which aims to automatically find and combine services in order to achieve a certain goal. Automated service composition should be flexible and robust enough to compensate for unforeseen changes in the environment, e.g. service failures or the sudden emergence of services with a higher quality value. All of these aspects are discussed under the umbrella of *Service Management*. Furthermore, Service Management also comprises concepts like Self-* properties such as self-healing or self-optimising (the latter more commonly known as 'self-organisation', when it comes to the field of agent research).

We already mentioned that this work is focused on agent frameworks with support for SOA, thus, we can assume that an agent's capabilities are offered as services. It is obvious that advances in automated service composition directly further the progress of agent-based computing. Despite the importance of automated service composition, real world applications are sparse. The reasons for this are not entirely clear, though, the lack of (more) comprehensive solutions might be a factor.

We deem automated service composition to be one of the most promising software-engineering concepts and thus proceed by discussing automated service composition in the light of agent-functionalities that are offered in compliance with SOA. We conclude this section with a detailed explanation of automated planning procedures—the ultimate driving-force for service composition.

Semantics.

In order to make autonomic behaviour in the process of service composition possible, agents have to be able to interpret the purpose and effects of other agents' services. This can be achieved with the help of semantics. Semantics usually rely on a formal specification and enable the definition of ontologies. Ontologies represent the domain of concern and define interrelationships between entities. Contrary to common programming languages, which can also be used for domain modelling, semantic languages come up with more complex constructs, such as restrictions, axioms and rules. These constructs make the automatic categorisation of information considerably more accurate. Established approaches in this area are the *Knowledge Interchange Format* (*KIF*) [15], the *Planning Domain Language* (*PDDL*) [17], the *Web Services Modelling Language* (*WSML*) [8] and the *Web Ontology Language* (*OWL*) [34].

Semantic Services.

Despite its capabilities, the ontological representation of the domain does not enable the automated interpretation of services. What is actually required is a structure that specifies how semantics are related to service properties and to an agent's functionalities, respectively. These structures exist in the form of semantic service description languages, e.g. *Semantic Web Services Framework* (*SWSF*),[1] *Web Service Modeling Ontology* (*WSMO*),[2] *Semantic Annotations for WSDL* (*SAWSDL*) [26], *Web Service Modeling Language* (*WSML*) [14], *DIANE Service Description Language* (*SDS*) [28] and the *Web Ontology Language for Services* (*OWL-S*) [31], to name but a few. Some of these languages are 'lightweight approaches', providing information about inputs and outputs only, others are more complex and allow for the definition of preconditions, effects or even Quality-of-Service parameters. The bottom-line is: semantic service description languages are able to make services machine-readable and interpretable.

Semantic Service Matchmaking.

Assuming an appropriate semantic description, it is possible to equip agents with mechanisms that are able to automatically discover 'promising' services. Such concept is commonly known as service matchmaking. Currently, there are several service matchmaking approaches available. One particular category is focusing on functional parameters such

[1]SWSF website: `www.w3.org/Submission/SWSF/`
[2]WSMO website: `www.w3.org/Submission/WSMO/`

as inputs, outputs, preconditions and effects (also referred to as IOPE) and is using taxonomical analyses, lexicographic methods, and rule reasoning for the comparison of service request and advertisement [24, 29, 32]. Other approaches put emphasis on Quality-of-Service properties (non-functional) in order to find promising service options. There are also approaches [4, 6], which combine both mechanisms. A service matchmaker can be understood as a one-step planner, thus, a service matchmaker can be always be used as a basic module for a more comprehensive planning approach.

2.2 Planning with Services

Most available AI planning approaches are based on a search through a solution space. Solutions that are based on service composition and the execution of composed services are scarce [18]. In this work, we focus on software agents that are compatible with the SOA paradigm, thus, we understand a service as a semantically described action of an agent and also as a basic building block of a construct, which we refer to as 'plan'. Developing plans is extremely challenging [36] and usually starts with the abstract creation of a plan and ends with its execution. Creating plans at runtime aggravates the problem by adding practical challenges [2].

AI planning is well explored, thus, most of the automated service composition approaches transform their services into PDDL descriptions and then use standard AI planning algorithms for solving the planning problem. Examples for such approaches are *WSPlan* [39], *OWLS-XPlan* [25], or were presented by Sirin et al. [45], Akkiraju et al. [1], or Hatzi et al. [20]. All these works are based on a similar mechanism, that is: i) transforming service descriptions to PDDL or another logic language, ii) using a planning algorithm to solve the search problem, iii) translating the results into a sequence of service calls. Due to this mechanism, the capability of most automated service composition approaches (even the latest developments, cf. Sabatucci and Cossentino [42]) is determined by the capability of the underlying logical language. Furthermore, depending on the particular approach, different theoretic properties like the *Open World Assumption* or the *Frame Problem* are 'flexibly interpreted' in order to meet the requirements of the modelling space of the applied language. The effects of these design decisions are commonly neglected.

Only a few solutions approach the problem of service composition directly in the service domain and by means of AI planning. One promising approach was presented by Sathya and Hemalatha [43]. The authors propose a high-level architecture including a *Universal Description, Discovery and Integration* repository and a mechanism that composes registered services to an invocation order. Fähndrich et al. [13] argue that state definition planning can be executed directly in OWL and propose to use OWL-S for planning based on service description. The authors suggest to describe either the start or the goal state and to use SWRL to describe the effects of actions. A similar approach to avoid transformations to languages other than OWL-S, was presented by Redavid et al. [41]. In order to avoid the dependency to an underlying language, the authors propose to create SWRL-DL rules from the input and output parameters of the service if the OWL-S service description provides no information about preconditions or effects. These rules can be stored in a domain ontology were initial knowledge and goals are constantly updated. After creating and storing SWRL rules in the ontology, a backward search is used in order to find a path that is able to achieve the pursued goal. This path can be considered as a composition of service calls. A similar service-based approach was presented by Cruz et al. [7]. The approach is based on services, which are described in OWL-S and which support SWRL rules. The output parameters of the services can be used for analysis or for input parameters of other services. To this end, the approach focusses on output parameters of services and not on 'world-altering' effects like in other solutions. Tong et al. [46] integrate agents with service composition techniques. The authors present a distributed planning algorithm, namely *Distributed Planning Algorithm for Web Service Composition* (*DPAWSC*), which transforms the service composition problem into a graph search problem according to the dependencies among service agents.

The application of web service languages motivates further theoretic and practical solutions. In more particular, it is necessary to discuss problems that occur when planning theory is being executed. The first challenge here is to differentiate two types of services:

World-altering services are services, which cause changes in the environment when being executed. During planning these effects are only simulated on a virtual state, however, during the execution of a plan, services have effects. These have to be handled if the plan fails during execution.

Information services are services, which provide information, without having any other effect. The effect (the abstract knowledge of an agent) of such a service cannot be described in PDDL [19].

During the planning process, the effects of executed information services have to be considered. To account for this, the planning process has to instantiate individuals that represent the abstract knowledge in the current state, respectively. Summing up, we do not only have to deal with the commonly known frame problem [16], but with this new problem of remembering state-knowledge as well.

Additionally the semantic service research is done in compliance with the *Open World Assumption*, which means that facts can be either 'true', 'false', or 'unknown' (compared to the *Closed World Assumption*, where facts can only be 'true' or 'false') [39]. The creation of a locally closed world eases these two challenges, yet, the underlying theoretic problem on the other hand remains unsolved [9].

Practically there is no reasoner available, which exceeds the capabilities of *SWRL:B*[3] and *SWRL:X*[4], e.g. modelling of lists, sets, maps, and instance or class creation. Thus, contrary to PDDL, semantic services maintain the expressiveness of describing more than 'add lists' and 'delete lists' in an effect, however, the usage is limited due to the lack of theoretic reasoning capabilities of current reasoners.

3. SCENARIOS

There are many scenarios how those advances in service technology can be used in current trends such as the *Internet of Things*, human-computer-interaction, smart cities,

[3]SWRL:B website: `www.daml.org/rules/proposal/builtins.html`

[4]SWRL:X website: `swrl.stanford.edu/ontologies/built-ins/3.3/swrlx.owl`

etc. In this section, we present a number of current research projects: The 'smart home' of *Connected Living*, the charging optimisation of *MSG EUREF*, the intermodal route planning of *IMA*, and the service orchestration of EMD. Each project uses some of the aforementioned service paradigms; in particular, each employs some form of planning. In the following, we look at each of those in more detail.

3.1 Connected Living

The *Connected Living* (CL) project is about creating a 'smart home', featuring different devices and services. Each of those devices is represented by a software agent, providing services for the different sensors and actuators of the device, e.g. for reading a humidity sensor or for turning off the heater or for switching on a lamp. Those services can be controlled via 'assistants', i.e. another kind of agent providing different web UIs and/or rules for controlling the device agents. Those agents can be added to and removed from the platform dynamically at runtime.

The CL assistants and rules have been implemented in a demonstration platform that has been evaluated in a small number of selected households. Parts of the CL system eventually passed into the *IO-LITE* platform[5].

With each of the services being semantically annotated with input and output, precondition and effect, as well as some quality-of-service metrics, those device agents' services can be planned with, and orchestrated to more complex services according to the user's goals.

Consider the following scenario: Instead of manually operating the light, the blinds, or the air conditioning, the user simply states the goal 'I want my room to be warm and bright'. First, the system would determine what 'my room' refers to, using context information provided by the user, and filtering the services that relate to the same room. Then, it matches services that have an effect on brightness, such as turning on different lamps in the room, or operating the blinds, as well as services regulating the temperature, like heaters and air conditioning. Depending on the current conditions, some of those services are applicable, and others are not—opening the blinds will not make it brighter if it is dark outside, and turning on the heater does not do any good if it is already too warm in the room. The remaining services are then assessed w.r.t. the quality-of-service attributes that are relevant to the user in the current situation, and one or more of the services are executed.

The planning problem encountered in this project is not too difficult. While the number of devices can be rather high, and each device can have multiple services, those services usually have discrete parameters, making the search-space as a whole finite and not too large. Also, as few of the devices' effects depend on effects from other devices, service chains are rather short; in fact, a single service execution is often enough to achieve a goal.

3.2 MSG EUREF

The project *MSG EUREF* is part of a larger group of projects in the *Schaufenster Elektromobilität* that are concerned with finding optimal charging schedules for electric vehicles in a micro-smart grid (MSG) [30]. The different projects are based on the same abstract domain model and the same optimisation algorithm, with a few specific adaptations for each of the projects.

[5]IO-LITE website: `iolite.de/`

The MSG EUREF project is currently in the final stage of its field test at the EUREF campus. In this field test, the application creates charging schedules for two local buffer storages, six charging stations for electric vehicles, and a combined heat and power plant.

The core of each of those projects is a domain model, describing the different components of the micro smart grids, such as different charging stations, storages, and vehicles, as well as abstract concepts such as prognoses, bookings, and charging schedules. Those models are used in a multi-stage optimisation for finding an optimal schedule for charging the vehicles and local storages for meeting a certain goal (usually a set of bookings) while making best use of locally produced energy and low energy prices [21].

The optimisation is subdivided into several stages:

1. Using machine learning to predict energy available from local production.
2. Assignment of vehicles to bookings, swappable storages to vehicles, and vehicles to charging stations.
3. Optimisation of the charging schedules using stochastic optimisation.
4. Load-smoothening using local storages and/or combined heat and power plants.

Each of those stages is represented as a service provided by a different agent. For some stages, different implementations are available (reflecting the requirements of the different sub-projects), to be orchestrated to the overall optimisation process [22].

The schedule is created using a $(\mu/\rho + \lambda)$ evolution strategy, a form of genetic algorithm [10]. Starting with an empty schedule, the algorithm repeatedly recombines the 'parent' schedules, creating and mutating 'offspring' schedules and assessing them by simulating the charging processes in those schedules. A numeric quality measure is determined by calculating the weighted sum of some key metrics, such as ratio of bookings fulfilment, and total energy cost. The schedules that have the highest quality are carried over to the next generation and the process is repeated until the quality converges.

While this project is also concerned with creating a plan, or a schedule, the approach to planning is entirely different than in the one before. Here, the number of different actions is very limited (charging or discharging any of the storages), but the actions have multiple continuous parameters (start of charging, end of charging, and charging power). The effect of the actions is well-understood, to the point where it can be accurately simulated, and small changes in input parameters yield small changes in the action's result, making the application of stochastic and local search algorithms possible.

3.3 IMA

The *Intermodal Assistance for Megacities*, or *IMA*[6] project aims to tackle the problem of integrating ever diversifying mobility services: Next to public transportation and private cars, car and ride sharing are becoming more prevalent in large cities, and often a combination of different means of transportation is the best option.

[6]IMA website: `ima.dai-labor.de`

In IMA, mobility providers can register and offer their services by implementing interfaces for corresponding routing requests. The approach allows for the development of high-level functionality, which autonomously accesses, assesses and utilises available services. Those are integrated into an assistance system, which calculates and proposes intermodal routes that are tailored to the individual preferences and requirements of users. To make this work, services are specified through semantic service descriptions, using OWL-S, containing preconditions, effects, and descriptions of the service's attributes, to be matched against the user's preferences.

Based on a semantically enriched service landscape, intermodal route calculation becomes a typical planning problem. The algorithm accesses the distributed platform and uses a semantic service matchmaking component to determine services that fit to the user's preferences and characteristics (e.g. if the user has no driver's license, the planner must not include car-sharing services). After the matching-procedure, all locations or stations of possible mobility services are integrated as nodes into a graph, which are in turn assembled to clusters indicating potential changing locations between modes of transportation. Next, the costs are estimated by an objective function considering the user's preferences, such as time, monetary costs, ecological footprint, and others. In order to be able to set the preferences into relation, each of them is normalised according to the worst estimation for the respective route. Based on this heuristic, an optimal intermodal routing solution can be searched for on the graph using the A-star algorithm.

The system was implemented as a distributed multi-agent system, encapsulating bits of functionality as individual software agents. In doing so, inter-agent communication works 'out-of-the-box', and new agents can be deployed to the system without need for further configuration, even at runtime. As a consequence, mobility service providers simply have to implement an agent providing the respective service, interfacing with the backend service running on their own hardware, and deploy this agent to the IMA system. This approach facilitates the providers' data sovereignty, increases trust and eases the access to the system. The agent-based approach enables the development of more sophisticated route-finding approaches, e.g. approaches in which software agents represent mobility providers as bidders in auctions where a customer's mobility requests are negotiated.

The IMA system is deployed in Berlin, Germany. The application includes detailed information about the local public transportation network, three car-sharing and two bike-sharing companies as well as one taxi service provider. The resulting graph comprises approximately 12,000 nodes, which are connected by more than 4 million edges. The overall graph size is significantly determined by the user's particular settings, thus, the systems' response time varies. An average response time of roughly 5 seconds was determined during a field test study. This time lag was required to compute the first set of results, yet, the philosophy of IMA is not only to rely on the best results, but to search all kinds of alternative options (in particular intermodal route options). To make this work, IMA uses a threshold of 20 seconds to search for further routing options. Results are ranked and presented to the user, who can select from the proposed options. The applied planning concepts were able to tailor route-finding in IMA to the individual preferences of the user. In more than 90 % of the cases, the user selected the same option that the IMA system assessed with the highest quality. This was also shown during the above-mentioned field test.

In IMA, planning is used in two ways: First, the low-level route finding algorithm is an example for a domain-specific, highly optimised planning algorithm. Secondly, a planning algorithm is used in order to determine the potential combinations of available routing actions, e.g. via car or via train. While those actions have very similar effects—getting the user from A to B—they can have very specific preconditions as well as quality-of-service parameters, such as the duration, cost, or eco-friendliness.

3.4 EMD

The project *EMD* (*Extendable and Adaptive E-Mobility Services*) is concerned with developing semantic service descriptions for real-world services and with using those descriptions for searching, matching, and planning with those services, so that they can be reused and orchestrated to more complex processes.

Services—both agent actions and web services—are semantically annotated using OWL-S with preconditions and effects being described in SWRL. The domain ontologies are modelled with OWL, or OWL ontologies are derived from existing Java or EMF models. Those services can then be searched for or matched against a service template, both, at development time and at runtime, using the Semantic Service Matcher $SeMa^2$ [32].

The matched services (or the service templates for matching at runtime) are then imported into BPMN process diagrams and thus orchestrated to more complex services, which can again be semantically annotated and added to the list of available services. Those processes are then either transformed to executable agent components or are invoked directly using a process interpreter agent [27].

Besides matching individual services against some service template and the manual assembly of those services to larger processes, the project is also concerned with the automated planning of complex service orchestrations [13], both at development and at runtime. Other than in many related approaches (see Section 2.2), EMD is aiming at planning directly at the OWL-S level instead of using PDDL as an abstraction, making this approach more complex, but also more applicable in practice: Services are not required to be known a-priori, but can be discovered at runtime, plus, service parameters can be any semantically described complex data type. This allows interoperability on a semantic layer and enables fuzzy service discovery by preserving grounding and feasibility. In the project EMD, 13 agents have been implemented to provide 42 example services.

Besides using the services that were defined especially for EMD, the project also aims at being integrated with the other projects (or rather, the services defined in those projects), particularly with MSG EUREF and IMA. This, however, gives rise to further challenges: As the projects were not originally designed to work together, heterogeneous technologies are used and similar concepts are described in different domain models. These and other challenges, which can also be found in many real-life problems, will be the topic of the next section.

Table 1: Connection of research projects, planning problems and challenges.

	Element	CL	EUREF	IMA	EMD
Planning	Informed Search			x	
	Local Search/Optim.		x		
	Semantic Services	x		x	x
	Service Matching	x		x	x
	Semantic Planning				x
Challenges	Size of Search Space		x	x	x
	Communication	x		x	x
	Gen.-Purpose Plan.			x	x
	Interoperability	x			x
	Dev. Tools	x	x		x
	Beyond PDDL	x			x

4. CHALLENGES

While specialised planning algorithms are commonly used throughout industry, there are still many open problems that hinder the application of automated general-purpose planning in real-world applications.

While each of the research projects presented in the previous section uses some form of planning, their approaches to planning are very different, ranging from informed search algorithms and optimisation for solving specific problems to semantic service matching and planning w.r.t. different available services and their preconditions and effects. Due to this heterogeneity, it is difficult to compare those projects and their respective solutions. Nevertheless we will show their most important characteristics as well as the challenges that arose from those characteristics in Table 1.

In the following, we discuss some of those challenges, and briefly outline how they could be tackled.

Communication and Interaction.

Examined applications—IMA in particular—showed that effective planning processes are frequently slowed down by the distribution of relevant planning entities and the communicational overhead that is required to exchange information between these entities. Because of this, IMA refrains from distributed planning and applies a centralised approach, which caches information that is required for the planning. The approach fits its purpose, yet, on the expense of precision and flexibility. One minor side effect of this 'adaptation' is a wrongly calculated arrival time (e.g. due to outdated traffic information), however, it is also possible that the planning algorithm proposes to use a car sharing vehicle, which is not available any more (e.g. because of outdated booking information). What we generally suggest is to find ways to decrease the information exchange in distributed planning approaches. IMA showed that caching is a way to do that, however, in this case it is necessary to include the concept of 'ageing data' in order to account for information that is not up-to-date.

Domain-specific vs. general-purpose planning.

The presented projects use very different planning algorithms: On the one hand, domain-specific, highly optimised planning algorithms are used for the routing problem and for finding charging schedules. Those algorithms do not need semantic annotations, as the semantics of the actions is often encoded in the algorithms themselves.

On the other hand, different semantically annotated services are selected and orchestrated using general-purpose planners, operating on those services' preconditions and effects. General purpose planners will never catch up, and thus never replace specialised planners in their respective domains, but there are things they could adopt. For one, some planning algorithms are incorporating heuristics, such as the minimum number of steps to reach the goal, or the number of fulfilled sub-goals in a given state. Integrating this kind of heuristics into planning algorithms like *Graph-Plan* or partial order planning, as attempted, e.g. in [3], holds great promise for the future.

Interoperability among system borders.

Looking at the aforementioned scenarios, it becomes clear that the interconnection between systems of different domains will be one of the key advantages of service-oriented multi-agent systems. Especially upcoming trends like the Internet of Things, or *IoT* metaphor with presumably millions of nodes, will push the development to open and adaptive architectures. However, the need for interoperability raises a significant challenge, that is: Having a common domain model in order to exchange information. IMA and EMD are closely related to each other, therefore the developers decided to use the same model for both projects in order to ensure compatibility. The developers, however, reported that such approach involved a significant number of synchronicity problems, e.g. version control, different interpretation of attributes, or update cycles, to name but a few—let alone that the concept of common ontologies will not cope with future application requirements, especially with the requirements of the IoT.

A more general, holistic approach is required. First experiments showed that *ontology matching* [12] might work here. The basic idea of ontology matching is to (automatically) identify relations between entities of (possibly) different ontologies. Available approaches usually apply one or more matching strategies. Common strategies are name-based, string-based (*Jaro-Winkler measure* or *Cosine-similarity*), or structure-based and semantic-based strategies. The concept of ontology matching appears to be promising, especially when dealing with large numbers of (different) applications with heterogeneous models. Yet, in order to work in real world applications, several challenges have to be tackled [44]—most importantly (especially in the IoT context) 'large-scale matching'. Also, most matching solutions are design-time approaches, neglecting adaptability at runtime. Finally, an appropriate weighting of the different matching techniques within the aggregation process is missing. Tests have shown that, depending on the ontologies, specific techniques seem to be more suitable than others. An automatic approach to identify the proper technique for the respective situation is still missing.

Development and Debugging Tools.

Programming with autonomous agents in general, and with planning agents in particular, presents the developer with different challenges than in conventional programming. Besides obvious points, such as accurately annotating ac-

tions with their preconditions and effects, it can also be difficult to determine why a software agent is behaving in a certain way. The action could have been triggered by a rule, or by a plan, and that plan, again, has been selected in accordance to some goal, under the very specific circumstances that held when the goal was evaluated. It is not as easy to link a triggering event to a resulting action, and vice versa. On the one hand, this can make programming easier, as the developer does not have to care about *how* something is achieved, but on the other hand, in case something fails, it is also much harder to determine the cause of the failure.

For example, a classical exception stack trace shows the call hierarchy of a function and thus shows why and under what circumstances that function was called. But in case of a planning agent, it will just show that the action was invoked by the plan execution engine, but not why that action was selected. This calls for new programming and debugging tools, keeping track of the reasoning performed by the agent.

Also, while this is not in particular a challenge of planning, the distributed nature of agents can complicate debugging, as well. Here, a number of approaches exist, both for modelling interactions and for remotely monitoring and controlling agents and multi-agent systems.

Planning 'beyond PDDL'.

In Section 2.2, we elaborated on the complexity of planning processes, which operate on semantic description languages and which are executed in the web service domain. Below, we transfer these problems to service-agent environments and emphasise factors that aggravate the successful use of semantic descriptions for real world agent planning and service composition:

Continuous parameters vs. finite predicates: In many toy domains, object properties are modelled as predicates, representing finite state, which narrow the search space in contrast to real applications. In reality, parameters are continuous elements. These are hard to guess during a search process. As an example consider the following two descriptions: *IsOpen(Door)* and *Open(Door, 42cm)*

Expressiveness: Depending on the description logic of the language used to describe states, preconditions, effects, domain restrictions, and pursued goals, the reasoning on a service's effects can be indeterministic. Such phenomenon was demonstrated by means of the description logic *OWL-DL SROIQ(D)*.

Concretisation: In many cases, precondition and effects of services are abstract descriptions. These have to be concretised during the plan execution, since effect descriptions cannot provide all the required details. Such an effect can be demonstrated through the instantiation of an entity. The instantiation process has to set all mandatory parameters (either by inheritance or by inference).

Knowledge representation: At runtime, the agent has access to the representation of the currently executed plan. Additionally acquired knowledge as well as the interpretation of this knowledge cannot be represented. Furthermore, stronger reasoning at runtime on preconditions and effects is needed.

Blending planning and execution: Execution of information services during plan time and monitoring of service execution at runtime are additional concepts, which should be reflected in the description of services. Also, it is not possible to formally represent failures that occur during the execution of a plan. This is not limited to, but includes Quality-of-Service parameters.

The (automatic) composition of functionality that is described by semantics, has great potential—if not the highest potential of all planning-related techniques. Yet, available approaches require much improvement and further research in order to cope with real world problems.

5. CONCLUSION

In this paper we showed that, despite the fact that the agent-community deems several approaches as 'adequately matured', there still are significant problems when these theories are turned into reality. We used the concept of planning to emphasise the practical implications of these problems. In order to do that, we analysed existing agent-based applications with an adequate level of sophistication and maturity and that were developed to serve a real world purpose. The aim of this analysis was to collect planning-related problems that occurred during the implementation of these applications and to show how and to what extend these problems were countered.

In total we identified five categories of problems. To start with, software agents that serve a real world purpose frequently have to communicate to entities in different computer networks via Ethernet or mobile communication protocols. In terms of transmission rates, these protocols are highly optimised, thus, we do not see much potential for improvements here. Rather, more efficient 'real-world-ready' planning approaches are required, or, from a scientific point of view it is necessary to answer questions like: *Is it possible to implement planning algorithms that feature the same level of quality as established approaches, but at the same time require a decreased number of interactions between the involved entities?* Established planning approaches (see Section 2.2) are—in theory—able to include large numbers of agents, yet, in reality it was shown that these approaches reach their limits with 30 to 40 entities that are involved in the planning process. In the IMA project this limitation was countered through a centralised planning approach, which grabs and caches information from all available agents before planning is done. The approach fits purposes, though, on the expense of the planning's quality and flexibility. Due to the fact that information is being cached, planning is not based on the latest data, e.g. up-to-the-minute traffic information or newly deployed or adjusted mobility and transportation services. Thus, for a more comprehensive application of agent technology in a similar context, the communication overhead between planning-relevant entities has to be decreased. The IMA project showed that such problem can be approached by information caching mechanisms.

Secondly, in order to account for the connection of many different applications, some form of ontology matching is required. Examined applications (IMA and EMD in particular) avoided such approach by defining a common domain model, yet, the analysis of both applications showed that a common foundation quickly reaches its limit. Especially when it comes to the maintenance tasks, version control,

or update cycles, common approaches are clearly inferior to the more flexible ontology matching. Ontology matching (at least current approaches), however, is not ready for real world applications. There are several problems that have to be tackled—most importantly support for 'large-scale matching'. This feature becomes increasingly important in the context of the Internet of Things. Other problems are the missing support for adaptability at runtime and an appropriate weighting of the different matching techniques within the aggregation process.

Thirdly, we emphasised that two different types of planning were applied, namely specialised and general purpose planning. The advantage of general purpose planners is the possibility to comprehensively reuse existing software (mostly with minor additions). Yet, general purpose planners are usually slow and thus not applicable to problems where results are required quickly. The analysis of existing agent-based applications substantiated this thesis since specialised planning solutions were preferred. However, when looking into specialised solutions we were able to identify one feature that can be adopted by general approaches, that is, the incorporation of heuristics. Some approaches include heuristics like the minimum number of steps to reach the goal or the number of fulfilled sub-goals in a given state. We learned that these heuristics are promising and suggest adding them to general purpose planners as well. Following Bercher et al. [3], such integration is by all means possible.

We deem automatic composition of semantically described functionality to have the highest potential of all planning-related techniques. Yet, first tests clearly showed that available approaches require much improvement and further research (e.g. in terms of continuity, expressiveness, concretisation, and knowledge representation) in order to cope with real world problems.

Finally, we identified the need for new programming and debugging tools. Programming for itself is complex—and programming in distributed environments significantly more challenging. Classical exception stack traces show call hierarchies of functions and under what circumstances these functions were called, yet, in the case of a planning agent, it is difficult to get more information than the name of an action that (might have) caused problems. More detailed (and highly important) information, e.g. why the action was selected in the first place, are not visible to the programmer. In order to facilitate any professional adoption of agent-based frameworks we propose the development of new programming and debugging tools that help developers to keep track of the reasoning performed by the agent. After all, comprehensive tools support is arguably [5, 33, 35, 40, 47] a significant driver for industrial adoption of technology.

Wrapping up we can say that agent-technology is able to meet the requirements of real world applications through the use of service paradigms. The problem, however, is that agent-based programming frameworks and runtimes do not work 'out of the box' and frequently have to be adapted (e.g. with respect to their performance). This might hamper any broader application or validation of agent-based models, since programmers prefer to focus on the development of the application—not on dealing with shortcomings of the framework. This constellation is exceptionally unfortunate, especially in the light of the ever-new evolving software development trends that may significantly profit from the agent-based approach. As an example consider the Internet of Things, where agent-technology can significantly contribute to an easy integration of intelligent software, sensors, mobile devices and human beings.

Based on our analysis we argue that (most of the time) performance problems do not result from 'insufficient implementations' of concepts that have their roots in agent theory but rather from limitations of agent theory that is applied to real world problems. Nevertheless, considering these theories to be inapplicable is not helpful as well, thus, we identified the particular bottlenecks and proposed ways to improve the fundamental concepts. We identified challenges in both, theory and practice:

Theory: Communication overhead, expressiveness of description languages, knowledge representations, ontology alignment.

Practice: Development tools i.e. for debugging, execution monitoring, planning in continuous parameter spaces, heuristics to guide the shift from domain specific to general purpose planning.

We hope that our work will help theorists to better understand the problems and requirements that practitioners have when applying agent-based programming to reality. We have to find a joint approach to further the adoption of agent-technology, after all we, as the agent community, are strongest when we are working hand in hand.

REFERENCES

[1] R. Akkiraju, B. Srivastava, A.-A. Ivan, R. Goodwin, and T. Syeda-Mahmood. SEMAPLAN: Combining planning with semantic matching to achieve web service composition. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06)*, 2006.

[2] T. Aubonnet, L. Henrio, S. Kessal, O. Kulankhina, F. Lemoine, E. Madelaine, C. Ruz, and N. Simoni. Management of service composition based on self-controlled components. *Journal of Internet Services and Applications*, 6(1):1–15, 2015.

[3] P. Bercher, T. Geier, and S. Biundo. Using state-based planning heuristics for partial-order causal-link planning. In I. J. Timm and M. Thimm, editors, *KI 2013: Advances in Artificial Intelligence*, volume 8077 of *Lecture Notes in Artificial Intelligence*, pages 1–12. Springer Berlin / Heidelberg, 2013.

[4] D. Bianchini, V. D. Antonellis, M. Melchiori, and D. Salvi. Semantic-enriched service discovery. In *Proceedings of the 22nd Conference on Data Engineering Workshops*, pages 38–38, 2006.

[5] B. Burmeister. Industrial application of agent systems: Lessons learned and future challenges. In L. Braubach, W. van der Hoek, P. Petta, and A. Pokahr, editors, *Multiagent System Technologies, 7th German Conference, MATES 2009, Hamburg, Germany, September 2009, Proceedings*, volume 5774 of *Lecture Notes in Artificial Intelligence*, pages 1–3. Springer Berlin / Heidelberg, 2009.

[6] A. Carenini, D. Cerizza, M. Comerio, E. D. Valle, F. D. Paoli, A. Maurino, M. Palmonari, and A. Turati. GLUE2: A web service discovery engine with non-functional properties. In *Proceedings of the IEEE*

6th European Conference on Web Services (ECOW'08), pages 21–30, 2008.

[7] S. A. B. Cruz, A. M. V. Monteiro, and R. Santos. Automated geospatial web services composition based on geodata quality requirements. *Computers & Geosciences*, 47(0):60–74, 2012.

[8] J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, and D. Fensel. The web service modeling language (WSML). Technical report, Digital Enterprise Research Institute (DERI), 2005. http://www.wsmo.org/TR/d16/d16.1/v0.21/.

[9] P. Doherty, W. Lukaszewicz, and A. Szalas. Efficient reasoning using the local closed-world assumption. In S. A. Cerri and D. Dochev, editors, *Artificial Intelligence: Methodology, Systems, and Applications*, volume 1904 of *Lecture Notes in Computer Science*, pages 49–58. Springer Berlin / Heidelberg, 2003.

[10] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Natural Computing. Springer, 2003.

[11] T. Erl. *SOA Principles of Service Design*. Prentice Hall, 2007.

[12] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer, 2007.

[13] J. Fähndrich, N. Masuch, H. Yildirim, and S. Albayrak. Towards automated service matchmaking and planning for multi-agent systems with OWL-S – approach and challenges. In *Service-Oriented Computing – ICSOC 2013 Workshops*, volume 8377 of *Lecture Notes in Computer Science*, pages 240–247. Springer International Publishing, 2014.

[14] D. Fensel, F. M. Facca, E. Simperl, and I. Toma. *Semantic Web Services*. Springer Science & Business Media, 2011.

[15] M. Genesereth. Knowledge interchange format. Proposed Draft NCITS.T2/98-004, American National Standard, 1998.

[16] A. Gerevini, A. Saetti, and I. Serina. Case-based planning for problems with real-valued fluents: Kernel functions for effective plan retrieval. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012)*, pages 348–353, 2012.

[17] M. Ghallab, A. Howe, C. Knoblock, D. Mcdermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL – the planning domain definition language, 1998.

[18] M. Ghallab, D. Nau, and P. Traverso. The actor's view of automated planning and acting: A position paper. *Artificial Intelligence*, 208:1–17, 2014.

[19] L. Guzmán, A. Jaime, C. Ovalle, and A. Demetrio. Web service composition: a semantic web and automated planning technique application. *Ingeniería e Investigación*, 28(3):145–149, 2008.

[20] O. Hatzi, D. Vrakas, M. Nikolaidou, N. Bassiliades, D. Anagnostopoulos, and I. Vlahavas. An integrated approach to automated semantic web service composition through planning. *IEEE Transactions on Services Computing*, 5(3):319–332, 2012.

[21] C.-E. Hrabia, T. Küster, M. Voß, F. D. P. Pardo, and S. Albayrak. Adaptive multi-stage optimisation for EV charging integration into smart grid control. In Q. Chen, P. Torroni, S. Villata, J. Hsu, and A. Omicini, editors, *PRIMA 2015: Principles and Practice of Multi-Agent Systems*, volume 9387 of *Lecture Notes in Artificial Intelligence*, pages 622–630. Springer International Publishing, 2015.

[22] C.-E. Hrabia, F. D. P. Pardo, T. Küster, and S. Albayrak. Multi-stage smart grid optimisation with a multiagent system (demonstration). In *Proceedings of 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015)*, pages 1937–1938, 2015.

[23] International Foundation for Autonomous Agents and Multiagent Systems. *AAMAS 2015 Conference Program*. IFAAMAS, 2015.

[24] P. Kapahnke and M. Klusch. Adaptive hybrid selection of semantic services: The iSeM matchmaker. In *Semantic Web Services: Advancement through Evaluation*. Springer Berlin / Heidelberg, 2012.

[25] M. Klusch, A. Gerber, and M. Schmidt. Semantic web service composition planning with OWLS-Xplan. In *Agents and the Semantic Web: Papers from the AAAI Fall Symposium. AAAI Fall Symposium, November 4-6, Arlington, VA, USA*, pages 55–62, 2005.

[26] J. Kopecký, T. Vitvar, C. Bournez, and J. Farrell. SAWSDL: Semantic annotations for WSDL and XML schema. *IEEE Internet Computing*, 11(6):60–67, 2007.

[27] T. Küster, M. Lützenberger, and S. Albayrak. A formal description of a mapping from business processes to agents. In M. Baldoni, L. Baresi, and M. Dastani, editors, *Engineering Multi-Agent Systems*, volume 9318 of *Lecture Notes in Artificial Intelligence*, pages 153–170. Springer International Publishing, 2015.

[28] U. Küster, B. König-Ries, M. Klein, and M. Stern. DIANE: A matchmaking-centered framework for automated service discovery, composition, binding, and invocation on the web. *International Journal of Electronic Commerce*, 12(2):41–68, 2007.

[29] J. Li. A fast semantic web services matchmaker for OWL-S services. *Journal of Networks*, 8(5):1104–1111, 2013.

[30] M. Lützenberger, N. Masuch, T. Küster, D. Freund, M. Voß, C.-E. Hrabia, D. Pozo, J. Fähndrich, F. Trollmann, J. Keiser, and S. Albayrak. A common approach to intelligent energy and mobility services in a smart city environment. *Journal of Ambient Intelligence and Humanized Computing*, 6(3):337–350, 2015.

[31] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. Sycara. Bringing semantics to web services: The OWL-S approach. In *Semantic Web Services and Web Process Composition*, pages 26–42. Springer Berlin / Heidelberg, 2004.

[32] N. Masuch, B. Hirsch, M. Burkhardt, A. Heßler, and S. Albayrak. SeMa^2 – a hybrid semantic service matching approach. In *Semantic Services: Advancement through Evaluation*, pages 35–48. Springer-Verlag, 2012.

[33] V. Mařík and D. McFarlane. Industrial adoption of agent-based technologies. *Intelligent Systems, IEEE*, 20(1):27–35, 2005.

[34] D. McGuinness and F. van Harmelen. OWL web

ontology language overview. Technical report, W3C, 2004. http://www.w3.org/TR/owl-features/.

[35] J. McKean, H. Shorter, M. Luck, P. McBurney, and S. Willmott. Technology diffusion: analysing the diffusion of agent technologies. *Autonomous Agents and Multi-Agent Systems*, 17:372–396, 2008.

[36] D. S. Nau, M. Ghallab, and P. Traverso. Blended planning and acting: Preliminary approach, research challenges. *AAAI*, pages 4047–4051, 2015.

[37] Organization for the Advancement of Structured Information Standards (OASIS). *Web Services Business Process Execution Language (WS-BPEL) Version 2.0*, Apr. 2007.

[38] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: a research roadmap. *International Journal of Cooperative Information Systems*, 17(2):223–255, 2008.

[39] J. Peer. A PDDL based tool for automatic web service composition. In H. J. Ohlbach and S. Schaffert, editors, *Principles and Practice of Semantic Web Reasoning*, volume 3208 of *Lecture Notes in Computer Science*, pages 149–163. Springer Berlin Heidelberg, 2004.

[40] M. Pěchoučyek and V. Mařík. Industrial deployment of multi-agent technologies: review and selected case studies. *Autonomous Agents and Multi-Agent Systems*, 17(3):397–431, 2008.

[41] D. Redavid, L. Iannone, T. Payne, and G. Semeraro. OWL-S atomic services composition with SWRL rules. In A. An, S. Matwin, Z. W. Raś, and D. Ślęzak, editors, *Foundations of Intelligent Systems*, volume 4994 of *Lecture Notes in Computer Science*, pages 605–611. Springer Berlin / Heidelberg, 2008.

[42] L. Sabatucci and M. Cossentino. From means-end analysis to proactive means-end reasoning. In *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2015)*, pages 2–12, 2015.

[43] C. Sathya and T. Hemalatha. An adaptive architecture for autonomic orchestration of web services. *International Journal Of Engineering and Computer Science*, 2(4):972–975, 2013.

[44] P. Shvaiko and J. Euzenat. Ontology matching: State of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering*, 25(1):158–176, 2013.

[45] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau. HTN planning for web service composition using SHOP2. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(4):377–396, 2004.

[46] H. Tong, J. Cao, S. Zhang, and M. Li. A distributed algorithm for web service composition based on service agent model. *IEEE Transactions on Parallel and Distributed Systems*, 22(12):2008–2021, 2011.

[47] D. Weyns, A. Helleboogh, and T. Holvoet. How to get multi-agent systems accepted in industry? *International Journal of Agent-Oriented Software Engineering (IJAOSE)*, 3(4):383–390, May 2009.

[48] F. Zambonelli, N. Bicocchi, G. Cabri, L. Leonardi, and M. Puviani. On self-adaptation, self-expression, and self-awareness in autonomic service component ensembles. In *Proceedings of the 5th IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW 2011)*, pages 108–113, 2011.