Agents do it for Money - Accounting features in Agents

Jan Keiser, Benjamin Hirsch, Sahin Albayrak
{Jan.Keiser,Benjamin.Hirsch,Sahin.Albayrak}@dai-labor.de

DAI-Labor, Technische Universität Berlin, Germany

Abstract. This paper presents a novel way of incorporating accounting features into agents. Network management techniques and methods are investigated and adopted to the agent case. We then use an example to show how sophisticated accounting technologies can be used. The example has been implemented using the JIAC platform.

1 Introduction

In [1], Jennings et al. famously coined the phrase *Objects do it for free, agents do it for money*. While the authors aimed to highlight that agents are autonomous and therefore can choose to refuse the provision of services, it can nowadays also be taken quite literally. In fact, according to the AgentLink Roadmap [2], agent based technology will become a mainstream technology in the next 10 – 15 years. This also includes the commercial use of agents. Currently, agents are used in commercial settings, but mainly within a closed system, where all the participating agents belong to the same owner. This however will almost necessarily change, as agents become more mainstream, and more and more pervasive network access does not pose roadblocks to using services over the internet. However, the commercial use of agents within an open architecture can be difficult because of the distributed nature of agents, as the provided service, the amount of work being done, as well as the necessary interaction is difficult to turn into an appropriate cost.

The aim of this work is to enable agents to function within commercial and open settings while still staying in the agent concepts. In our view, this requires agent systems to deal with (complex) management issues in general, and accounting in particular. While "standard" approaches concerning accounting and even communication (via e.g. webservices) are certainly an option, this would lead to the solutions living outside the agent oriented concepts.

In this paper, we present a method to incorporate accounting features into agents while staying within the agent paradigm, allowing us to not only measure usage of agents but also create complex tariff schemes. The accounting features are based on a general management layer within the agent framework that is adapted from network management technologies. In network management, proved and tested techniques are used to control and manage network nodes.

While we implemented the techniques in the agent framework JIAC (Java Intelligent Agent Componentware), the proposed management structure as well as the accounting features can be applied to other agent frameworks as well.

The rest of this paper is structured as follows. In the next section (Section 2), we give an overview of management technologies in general and accounting methods and technologies in particular. Section 3 describes a high-level model of how agents and management techniques can be combined. We then describe in detail our generic accounting infrastructure which is embedded in this model in Section 4. Before we go into implementation details, we give an overview over the agent framework which we used to implement the concepts described in this paper. We conclude and give an outlook in Section 5.

2 Enterprise Management

Most of today's technologies for management concern the network layer, and are based on OSI Management [3]. OSI stands for Open Systems Interconnection, and has been the basis of standardisation efforts for connecting open systems [4].

OSI Management

OSI management supports the user as well as the provider of communication services during planning, supervision, and control of system activity. This is achieved by providing technologies to coordinate, distribute, and assign resources within the communication network. We can distinguish between three areas of OSI management, which are systems management, n-layer management, and n-layer operations (also known as protocol management).

- Systems management includes vertical management activities that concerns all seven OSI layers, and happens on the application level.
- N-layer-management details functions, services, and protocols for a given layer N.
- N-layer-operations or protocol management concern techniques for monitoring connections with the given layer.

The ISO working group solely defines management standards for the first category, systems management. Here, different standardisation efforts exist, including the specification of procedures that offer specific management aspects.

There are also a number of actual implementations of management technologies. Here, we want to introduce two of them, the second of which is fairly known even outside enterprise network management.

TMN for telecommunication networks [5] is based on OSI management. It distinguishes three layers, namely an information architecture, a functional architecture, and a physical architecture. Each of the architectures is described using functional groups, functional components, reference points, and interfaces. TMN generally supposes a distinct network for the transmission of management

relevant data, though it does allow to use the managed network to transmit data as well.

SNMP [6] is a well-known protocol for network management. While initially designed as an interim protocol, it has established itself and is currently available in its third incarnation. It is based on four elements: a management station, a management agent, a management information base, and a network management protocol. The first provides the user with management functionality, while the agent is a purely passive element located in the different managed devices. Manager and agent share a management information base which contains information about resource administration. SNMP provides three functions to interact, a namely *get*, *set*, and *trap*. The first two are for (manager initiated) reading and manipulation of variables defined in the management information base, while the last allows the managed entity to pro-actively send data to the managing component. Communication is done via UDP, and is generally routed over the managed network.

FCAPS

Management procedures are collected under the name FCAPS, which is an abbreviation for the five areas of management it covers.
Fault Management concerns all activities relating to the discovery, reaction, and solution of faults. This includes the definition of classes of failures, monitoring system behaviour, and the generation and transmission of failure-information.
Configuration Management details the identification, collection, control, provision, and execution of data which are important for the initialisation, execution, and adaptation of the system.
Accounting Management deals with the evaluation of services as well as the definition of rules governing accounting of (a combination of) those services. Furthermore, processes and data required to do actual accounting, like customer information and electronic bills are described. It should be noted that accounting is a necessary requirement for any billing system. It provides information and administration of communication, computing resources, storage space etc.
Performance Management or quality management concerns two areas. On the one hand, we have the monitoring of performance parameters such as the number of messages received or sent, the amount of data transmitted, and reaction times. Furthermore, it includes the possibility to configure monitored systems in order to improve performance.
Security Management includes security technologies such as authentication, authorisation, access control, and encryption, as well as the provision of security-relevant information.

Accounting Management

Most research and development activities of accounting management exist under the name AAA that is an abbreviation of authentication, authorisation and

accounting. Accounting management contains the collection of resource consumption data for the purposes of capacity and trend analysis, cost allocation, auditing and billing [7]. This includes the measurement, rating and assignment of the consumption data as well as the communication of these data between appropriate parties.

Most of the proposals, specifications and solutions in the field of accounting management consider only the communication layer of applications (e.g. data transfer and network access) or do not cover further aspects of accounting such as charging. Simple accounting management architectures and protocols are specified by ITU-T [8], IETF [7], OMG [9] and M3I [10, 11]. Examples for accounting attributes and record formats are ADIF [12] and IPDR [13].

3 Agents and Management

In the context of agents, accounting has to our knowledge not yet been implemented in a general and re-usable fashion. As argued in the introduction however it appears that, as agents move more towards open systems, metering and accounting gains importance.

Rather than focussing on accounting alone and developing an accounting solution from scratch, we base our work on the large body of work that has been done within the network management community as detailed in the previous section. In particular, we adopted the FCAPS management structure as a basis for our accounting implementation. We chose FCAPS for several reasons. It is a thoroughly tested and established approach which is used widely. Also, it allows us to extend our framework to provide not only accounting features but potentially any of the other FCAPS areas such as failure or performance management.

In the remainder of this section detail how we integrate management functionality in agents. While we have implemented it within a particular agent framework, we aimed at making the integration general enough to be applied to most agent frameworks.

Figure 1 shows three layers which occur in a managed agent system. We focus here on the functional aspects, and make no claim about the physical distribution of those functionalities. However, most basic mechanisms are generally located within the single agent. The layers describe on an abstract level the internal structure of the agents (with respect to manageability), the basic management techniques, and the abstract FCAPS management areas.

The lowest level details an agent architecture (e.g. a BDI architecture) which is to be managed. This can be described in terms of communication, knowledge management, execution, and events. Communication is of course an important element of a multi-agent system, but is also important from the view of a single agent. The low level technologies employed, as well as higher levels such as ACL's (e.g. FIPA ACL [14]) and protocols fall under that header. Knowledge management details the internal state of the agent. Actual computation, manipulation of knowledge, and the behaviour of the agent are based on an execution

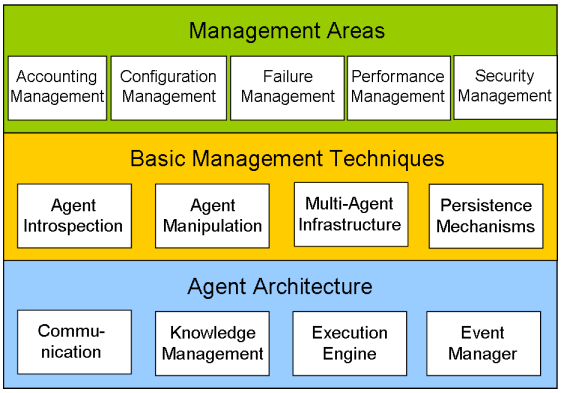

Fig. 1. Three layer management architecture

engine. Lastly, agents have to have means to communicate changes in their state, such as the sending and receiving of messages, invoked services, changes in the knowledge base, plans, and goals etc. The event manager provides an interface for management components to access those architecture specific events.

The next layer provides basic management mechanisms which build on the lower level. It works as a wrapper between the management interfaces of the used agent architecture and the higher-level management functions. Agent introspection provides standardised information about the state of the agent in terms of the four elements detailed above. Furthermore, the state of the agent should not only be monitored, but techniques for the manipulation of the internal state have to be provided. This includes changing the knowledge base, triggering events, adapting, adding or deleting goals, and changing the execution state of the agent. Also, abilities of the agent that refer to the infrastructure, such as migration, termination of the agent, and the (de-) registering of services (see FIPA Management [15]) have to be manageable. Lastly, any mechanisms pertaining to persistence of the agent need to be available.

Layer three uses the basic management mechanisms offered in layer two in order to provide value-added management functions such as fault, configuration, accounting, performance, and security management as independent from the actual agent framework as possible. In terms of agent based systems, this would provide

- detection of faulty or abnormally behaving agents, and providing means to evaluate and recover from errors occurring in the system;
- configuration mechanisms to adapt the system to new requirements, for example by including new features, or replacing old versions of agents with newer ones in an organised fashion by consideration of dependencies between modules;
- metering of resources and mapping to services offered by agents, allowing for a centralised account management and billing mechanisms [7];

– quality control, reliability, load balancing mechanisms (possibly based on information outside the actual agent system) and detection as well as elimination of bottlenecks;
– secure communication, public key infrastructures, access control for agent services, and trust relationships between agents and communities (see e.g. [16–18]).

There are two ways of realising above requirements and provide management technologies to agent frameworks. The first would be to extend the agent framework with one of the management technologies described above. For example, agents could be extended by adding an SNMP component which provides and manages data. This would also require the extension of management technologies to include agent relevant information. Alternatively, one can use the technology provided by agents themselves to manage the system instead of extending network management mechanisms. Here, existing techniques (such as FIPA communication) can be adapted to reflect management mechanisms.

We have chosen the latter approach, and base management extensions on FIPA standards.[1] This has the advantage of keeping a reasonably homogeneous system, where the available technologies are used as far as possible. Also, the nature of agents allows to for example provide management agents which monitor a set of agents, and which themselves can take advantage of agent technologies such as flexibility, robustness and intelligent behaviour. Last but not least the managed agents may keep their level of autonomy, e.g. by allowing them to refuse management requests which are in conflict to the own goals.

In order to really get a management framework which can be applied to different agent frameworks and which allows the interoperability between different implementations of this abstract model, several additional requirements should be met. Layer two mechanisms provide clearly defined services, use FIPA-ACL as communication language, and support most of the popular content languages (such as FIPA-SL, KIF[19], and RDF[20]). To this end, FIPA protocols are used. The Monitoring services use the FIPA Subscribe Interaction Protocol in order to provide information about events that occurred within the agent (push-strategy), and the FIPA Query Interaction Protocol to get the current state of (an element of) the specified agent (pull strategy). Agent control is provided using the FIPA Request Interaction Protocol, allowing to request specified actions from the agent.

In order to further detach the actual agent framework from the FCAPS management layer, ontologies describing the state, events, and agent elements should be standardised to allow inter-operability of different agents with the management framework. It should be noted that the nature of different systems precludes a set of events and actions that map to all possible events and actions. However, there is a common subset of events that *can* be specified. For example, the FIPA Abstract Architecture is used as basis for standardised events. Furthermore, many agent frameworks employ some sort of BDI-like architecture, which again should be abstracted to a set of events including the creation,

[1] The specifications mentioned in this paper can be found at `http://www.fipa.org`.

modification, and deletion of beliefs, goals, and plans. Last but not least, events concerning the life cycle of agents need to be provided. This includes the actual change of an agent's life-cycle state, such as creation, deletion, and migration.

Having said that, it should also be noted that using agents to manage systems can have serious implications on several levels. Most obviously perhaps, management mechanisms are also needed to control the system when the autonomy of the agents leads to unpredictable and emergent behaviour that is in conflict to the intrinsic intention. It seems counter-intuitive to employ agents (which can behave unexpectedly) to control agents (though, by centralising and providing means of control, this argument can be countered to a certain extend). Also, management agents would be highly critical systems, and steps should be taken to ensure that they can work as reliably and securely as possible.

4 Agents and Accounting

Now that we have detailed how agents and management technologies can be combined, we describe how this theoretical approach can be applied in the context of accounting. We do this first by providing a scenario which incorporates accounting features. Using this, we will then describe the accounting architecture and show how we implemented it in the agent framework JIAC.

4.1 Scenario

In order to make the issue a bit more touchable, we present the following scenario:

A small company provides the online game "crazy eight". Several users may play against each other on virtual tables. Each table is represented by a game master agent and each player is represented by a human player agent, which also provides the user interface for the game and the online charge control. To pay for playing games every registered user has chosen one of the provided tariffs and decided to get information about the current account. A linear tariff based on the duration of the game (e.g. one cent per 10 seconds) and a flat-rate (e.g. 20 cents for a game) are available. The first tariff uses a charging function parameterised by rate and period and a tariff scheme based on time events.

Additionally to these tariffs the company now wants to provide a new tariff based on the played cards of the user without changing the implementation of the game. To do so also the definition of a new tariff scheme is needed that is based on the associated events.

At this point, we want to take a step back. Firstly, we observe that agents need to have some sort of introspection, i.e. the ability to know about internals such as service provisioning, resources consumed, time, and more. This data has to be made available to other agents, or managing entities. Also, the other direction of information flow is needed, i.e. the ability of outside agents (or managing entities) to directly influence the behaviour of the agent, for example the accounting agent telling the service provisioning agent to cancel the (executing) service if the credit limit of the user is reached.

4.2 Accounting Architecture

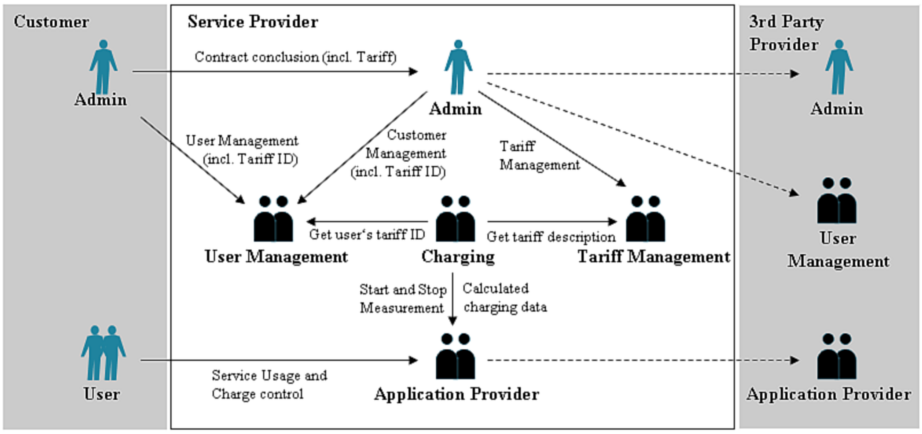

Fig. 2. Roles and interactions during accounting management

Figure 2 shows a generic accounting infrastructure while Figure 3 details the interaction sequence that occurs in the example of Section 4.1. The user agent (e.g. human player agent) requests application services (a card game service), which are provided by the managed application provider (the game master agent). The charging agents use the introspection services described in layer two of the management model to be informed about start and stop of application services. In this case, during the initialization or negotiation phase after a charging agent has received the information about the requested service, he uses a service of the user management to get the profile of the application service and user. The service profile contains a list of all supported tariffs or an empty list for non-commercial services. The user profile contains the valid tariff (as contracted between customer and provider) for the given service and user. Afterwards a service of the tariff management gives the description of the tariff, which is related to a charging function to calculate the price.

Accordingly, also by using the introspection services provided by the application provider the charging agent activates only the metering of events relevant to the charging function of the tariff (e.g. playing a card realised by sending a data speechact). If one of these events occurs during service provision (i.e. some state on which the function depends changes) the current price will be calculated again. As a result of a changed price the accounting rules of the charging agent may trigger actions, e.g. to stop the service or to inform the user about the current price by using the manipulation services of the application provider. If the application service stops, the metering will be deactivated and the final charge will be calculated. The service provider also plays the role of a customer

if its application provider agents uses services of 3rd-party providers (see dashed lines of Figure 2).

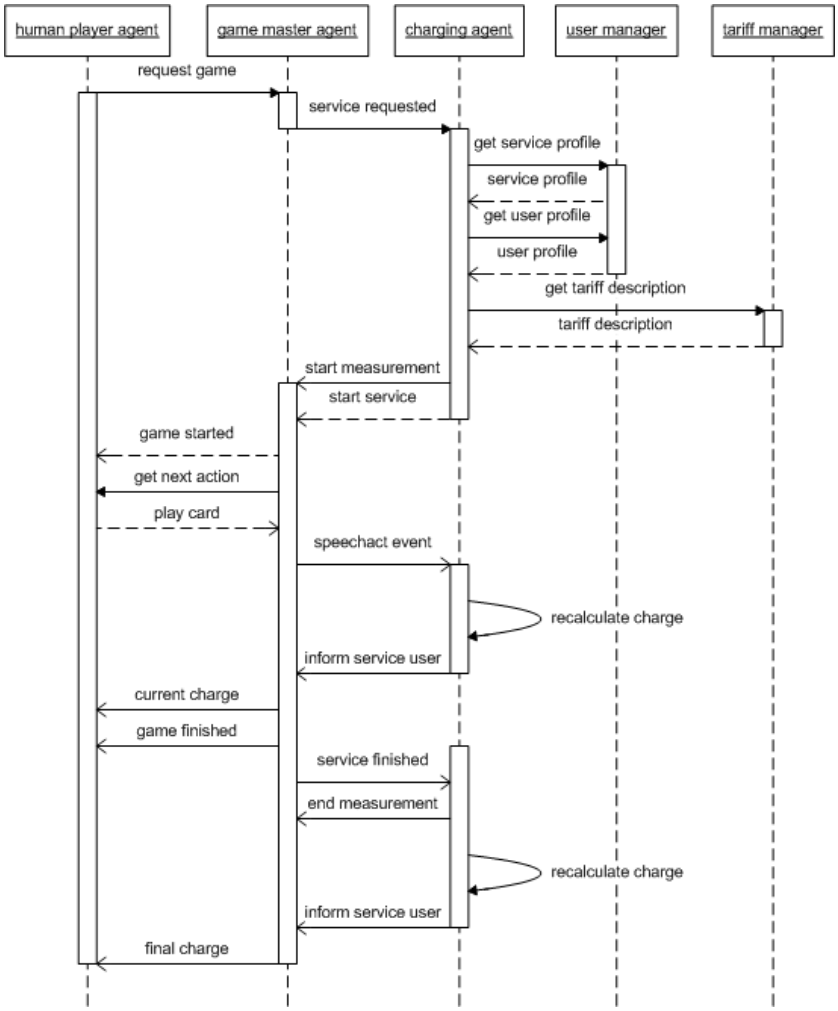

Fig. 3. Sequence chart of the card game scenario where a card-based tariff is chosen and one card was played

By using the introspection and manipulation mechanism introduced in Section 3, the provided services need not be re-implemented for accounting purposes. Furthermore, the possible tariffs are not restricted to just a few bits of information like service duration or number of access, which have to be decided upon before bringing the service online. Instead, all events measurable by the underlying agent framework may be used. Also the charging information about

used subservices of 3rd-party provider may be considered by the tariffs. Some frameworks (including *JIAC*) allow even to add new (measurement) components to agents during run-time. It should be noted that in the case of service chaining with a revenue sharing model a proper prediction of final charges is very difficult if the number of involved services becomes large and the used tariff schemes are more complex.

4.3 Accounting in *JIAC*

We have implemented our accounting approach in and with the FIPA-compliant agent framework *JIAC* [21, 22]. In the following we will describe the characteristics of this agent framework, before detailing the implementation of accounting. Note that the implemented management features are partially used in numerous projects [23, 24].

JIAC consists of a run-time environment, a methodology, tools that support the creation of agents, as well as numerous extensions, such as web-service-connectivity, an owl-to-*Jadl* translator and more. An agent consists of a set of components.

JIAC's component model allows to exchange, add, and remove components during run-time. The components interact among each other by agent internal messages. Standard components (which themselves can be exchanged as well) include a fact-base component, execution-component, rule-component, and more [25]. These components provide individual messages to manage the appropriate actions, e.g. the *MonitorGoalMessage* executed by the *GoalBean* allows to subscribe for changes on the goal stack.

Agents are programmed using the language *Jadl* [26]. It is based on three-valued predicate logic [27], thereby providing an open world semantics, and implements an BDI approach. It comprises four main elements: *plans elements*, *goals*, *rules*, *ontologies*, and *services*.

Communication is based on services. A service invocation consists of several steps, and is based on a meta-protocol, an extension of the FIPA request protocol. First, the agent contacts the Directory Facilitator (DF) in order to receive a list of possible services that could fulfil the current goal. After selecting a service and informing the DF, the agent receives a list of agents providing that service. Now an optional negotiation protocol can be executed with which the actual service provider is chosen. Only then is the actual service protocol executed. The meta protocol handles service-independent elements like security, communication failures, and other management-related issues.

Now we describe the *JIAC*-based implementation of the accounting architecture as part of a comprehensive management framework. Firstly, for layer two of the management model (see Figure 1) we have implemented management components providing an consistent interface (within the agent) to the bottom layer components of the agent architecture described above. This agent internal interface allows to register or deregister for events matching a specified pattern. These matched events may be delivered immediately or at regular intervals. The ontology shown in Figure 4 describes the structure of the managed *JIAC*-specific

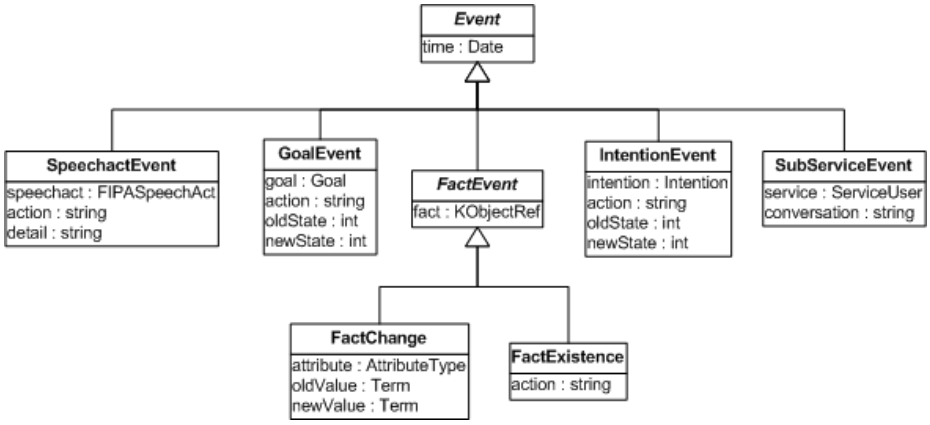

Fig. 4. Extensible ontology for the management of JIAC-based agents

events that need to be monitored for management functionality (e.g. accounting), such as speech acts, facts, goals, and intentions.

The introspection and manipulation services makes the internal interface available to other agents. We use priority mechanisms of *JIAC* to give the management related actions and messages a higher importance than the application services. At the moment, we do not consider access controls, or levels of au-

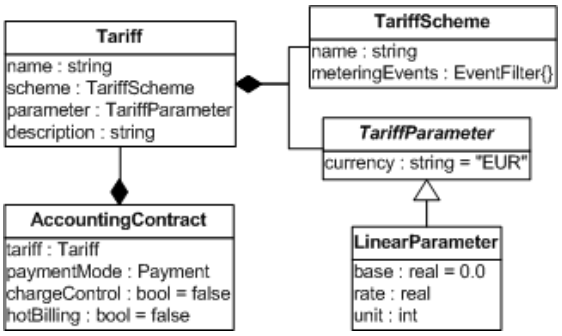

Fig. 5. Ontology for the description of tariffs

tonomy. The managed agents always provide all management capabilities if the managing agent is authorised. Restrictions of actions on specified agent elements and the ensuring of consistency of management actions with own goals are future work. Having said that, it should again be noted that the provision of those interfaces does not mean that agents necessarily loose all their autonomy. Instead, it is left to the programmer to decide which methods to employ, or how to react to management requests.

Up to now, standards for the introspection or manipulation of agents do not exist. But nevertheless to provide interoperability, we plan to implement the services using the referred FIPA protocols as alternative to the currently used *JIAC* meta-protocol. This includes conversion of the content between *Jadl* and the standardised languages SL, KIF or RDF, and translations between the different management ontologies (e.g. between the *JIAC*-management and a more general BDI-management). As proof of concept, we have implemented services for multi-agent infrastructure of the abstract model using FIPA protocols, conversion of the content to *SL* and the ontology *fipa-agent-management*.

Based on these implementations we have realised parts of the management areas described by the abstract model, such as the accounting infrastructure introduced before. Also, applicable tools for the run-time administration of agents, user management, and the management of accounting information, including the creation of tariffs and tariff-schemata, were developed.

```
(act getSpeechactLinearPrice
    (var ?usageId:string ?account:ServiceAccount ?tariff:Tariff)
    ...
    (script
        (var ?meteredEvents:class:java.util.HashMap ?events:Event[]
             ?amount:real ?money:Money ?parameter:LinearParameter
             ?base:real ?rate:real ?unit:int ?currency:string)
        (seq
           // get tariff parameters
           (eval (att parameter ?tariff ?parameter))
           (eval (att currency ?parameter ?currency))
           (eval (att base ?parameter ?base))
           (eval (att rate ?parameter ?rate))
           (eval (att unit ?parameter ?unit))

           // get measured events
           (getEvents (var ?usageId ?meteredEvents))
           (bind ?events (fun getEntry ?meteredEvents "SpeechactFilter"))

           // calculate current charge based on number of events
           (bind ?amount (fun Real.add ?base (fun Real.mul ?rate
                   (fun int2real (fun Int.div (fun getLength ?events)
                   ?unit)))))

           // update current charge
           (bind ?money (obj Money (amount ?amount) (currency ?currency)))
           (update (att price ?account ?money))
        )
    )
)
```

Fig. 6. Code example for a charging function based on speechact events and linear parameter

In *JIAC* tariffs are described using the ontology shown in Figure 5. A tariff contains a tariff scheme and parameters. To get highest possible flexibility to define new tariffs, tariff schemes are described by a list of filters for metering events (see ontology in Figure 4), and the corresponding charging functions are implemented as plan elements (see example in Figure 6). Because expertise is

needed for the flexible definition of tariff schemes, we have introduced tariff parameters which enables non-expert service providers to easily adapt the tariffs. In this example the attributes of the linear parameter are evaluated before reading out the list of metered events that match the speechact filter specified in the tariff scheme. Afterwards the charged amount is calculated based on the linear parameter and the length of the event list. Finally, the account will be updated with the new amount.

An example of a filter is shown in Figure 7. Here, the speechacts are filtered based on the performative "data" and the fact that they are received within a service protocol.

```
(obj ReceivedDataSpeechactFilter SpeechactFilter
    (EventFilter.name "SpeechactFilter")
    (EventFilter.role "de.dailab.management.accounting.role.ControlRole")
    (action "received")
    (detail "protocol")
    (speechacts (obj SpeechactPattern (performative "data")))
)
```

Fig. 7. An example of a Speechact-Filter

5 Conclusion

In this paper we have described how advanced accounting mechanisms can be incorporated into agent frameworks. Based on management technologies known from networking, we have provided a general framework which allows agents to measure, meter, and bill for services they provide. This grounding allows us to for example extend the agent frameworks towards other FCAPS areas such as performance or configuration management.

We have implemented the underlying framework and the accounting infrastructure within JIAC, and shown how accounting features can be used in agent frameworks. As through the use of webservices and the internet in general open systems are bound to be more and more pervasive, it becomes necessary for agent frameworks to provide methods to deal with the commercial implications of providing services to outside entities (as opposed to having an agent framework that in essence is a distributed application).

Future work includes restrictions of actions on specified agent elements, as well as ensuring consistency of management actions with the goals of the managed agents. The extension of the framework to include more FCAPS areas (such as security and failure) is another direction of future work. It should be noted that in the end, the different areas cannot be viewed separately as they are often intertwined. For example, the issue of trust management and its implications on accounting needs to be investigated in the context of security. Also service level agreements and their enforcement are important issues that need to be addressed.

References

1. Jennings, N.R., Sycara, K., Wooldridge, M.: A roadmap of agent research and development. Autonomous Agents and Multi-Agent Systems **1** (1998) 275–306
2. Luck, M., McBurney, P., Shehory, O., Willmott, S.: Agent Technology Roadmap. (2005)
3. ITU-T: Information technology – Open Systems Interconnection – Systems management overview. ITU-T Recommendation X.701, ISO/IEC 10040 (1998)
4. ITU-T: Information Technology – Open Systems Interconnection – Basic Reference Model: The Basic Model. ITU-T Recommendation X.200, ISO/IEC 7498-1 (1994)
5. ITU-T: Principles for a Telecommunications Management Network. ITU-T Recommendation M.3010 (2000)
6. Case, J., Fedor, M., Schoffstall, M., Davin, J.: A Simple Network Management Protocol (SNMP). RFC 1157, IETF (1990)
7. Aboba, B., Arkko, J., Harrington, D.: Introduction to Accounting Management. RFC 2975, IETF (2000)
8. ITU-T: Information technology – open systems interconnection – systems management:usage metering function for accounting purposes. Technical report, ITU Telecommunication Standardization Sector (1995)
9. OMG: Federated charging and rating facility. Technical report, Fraunhofer FOKUS (2002)
10. M3I-Consortium: Charging and accounting system (cas) design. Technical report, Market Managed Multi-service Internet Consortium, ETH Zürich (2000)
11. M3I-Consortium: Cas implementation. Technical report, Market Managed Multi-service Internet Consortium, ETH Zürich (2001)
12. Aboba, B., Lidyard, D.: The accounting data interchange format (adif). Technical report, IETF (2000)
13. IPDR: Network data management - usage: For ip-based service. Technical report, IPDR Organisation (2001)
14. FIPA: FIPA ACL Message Structure Specification. FIPA Specification SC00061G (2002)
15. FIPA: FIPA Agent Management Specification. FIPA Specification SC00023K (2004)
16. Schmidt, T.: ASITA: Advanced Security Infrastructure for Multi-Agent-Applications in the Telematic Area. PhD thesis, Technische Universität Berlin (2002)
17. Bsufka, K.: Public Key Infrastrukturen in Agentenarchitekturen zur Realisierung dienstbasierter Anwendungen. PhD thesis, Technische Universität Berlin (2006)
18. Bsufka, K., Holst, S., Schmidt, T.: Realization of an Agent-Based Certificate Authority and Key Distribution Center. In Albayrak, S., ed.: Intelligent Agents for Telecommunication Applications. Number 1699 in Lecture Notes in Artifical Intelligence (1999) 113–123 Third International Workshop, IATA'99, Stockholm, Sweden, August 1999.
19. Genesereth, M.R., Fikes, R.E.: Knowledge interchange format version 3.0 reference manual. Technical Report Logic-92-1, Stanford University, Computer Science Department (1992)
20. Lassila, O., Swick, R.: Resource description framework (rdf) model and syntax specification. Technical Report WD-rdf-syntax-971002, W3C (1999)
21. Fricke, S., Bsufka, K., Keiser, J., Schmidt, T., Sesseler, R., Albayrak, S.: Agent-based Telematic Services and Telecom Applications. Communications of the ACM **44**(4) (2001) 43–48

22. Sesseler, R., Albayrak, S.: Service-ware framework for developing 3g mobile services. In: The Sixteenth International Symposium on Computer and Information Sciences. (2001)
23. Albayrak, S., Milosevic, D.: Generic intelligent personal information agent. In: International Conference on Advances in Internet, Processing, Systems, and Interdisciplinary Research. (2004)
24. Wohltorf, J., Cissée, R., Rieger, A.: BerlinTainment: An agent-based context-aware entertainment planning system. IEEE Communications Magazine **43**(6) (2005) 102–109
25. Sesseler, R.: Eine modulare Architektur für dienstbasierte Interaktion zwischen Agenten. PhD thesis, Technische Universität Berlin (2002)
26. Konnerth, T., Hirsch, B., Albayrak, S.: JADL — an agent description language for smart agents. In Baldoni, M., Endriss, U., eds.: Proceedings of the DALT'06 Workshop, Springer Verlag (2005) to appear.
27. Kleene, S.C.: Introduction to Metamathematics. Wolters-Noordhoff Publishing and North-Holland Publishing Company (1971) Written in 1953.