

Designing Criteria-Driven Scheduling as Integrated Service for IEEE-FIPA Compliant Multi-Agent Infrastructures

Christoph Hermann, Bernhard Jung, and Paolo Petta

Austrian Research Institute for Artificial Intelligence (OFAI)*
of the
Austrian Society for Cybernetic Studies (ÖSGK)
Freyung 6/6
A-1010 Vienna, Austria (EU)
c.d.hermann@gmail.com
bernhard.jung@ofai.at
paolo.petta@ofai.at

Abstract: The Foundation for Intelligent Physical Agents (FIPA) provides a rich set of standards for implementing industrial scale multi-agent infrastructures. Despite its manifold possibilities for achieving coordinated action execution based on interaction protocols, it still lacks direct support of multi-agent planning and scheduling for goal directed action execution. In this paper, we discuss a design strategy to integrate Design-to-Criteria (DTC) scheduling using the Framework for Task Analysis, Environment Modeling and Simulation (TÆMS) for explicit partial modelling of coordination issues into FIPA infrastructures, as represented by the Java Agent DEvelopment Framework (JADE). Following the notion of “coordination as a service”, we exploit the infrastructural facilities of the FIPA multi-agent platform, and re-use FIPA interaction protocols for exchange of partial TÆMS structures, as well as for committing to action execution.

1 Introduction

The *Foundation for Intelligent Physical Agents*¹ (FIPA) provides a rich set of standards for implementing multi-agent infrastructures. The FIPA reference architecture is only one out of many multi-agent platform architectures. *Reusable Environment for Task-Structured Intelligent Networked Agents*² (RETSINA) [Sy03] or *Java Agent Framework*³ (JAF) [Ho98] are further examples. Each multi-agent infrastructure has its focus on some specific features. JAF for example emanates from a multi-agent planning and scheduling viewpoint. FIPA on the other hand is more communication centric and stresses aspects including formal semantics of languages and interaction protocols.

*The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry for Science and Research and the Austrian Federal Ministry for Transport, Innovation and Technology.

¹<http://www.fipa.org/>, last visited Oct. 10, 2007

²http://www.cs.cmu.edu/~softagents/retsina_agent_arch.html/, last visited Oct. 10, 2007

³<http://dis.cs.umass.edu/research/jaf/>, last visited Oct. 10, 2007

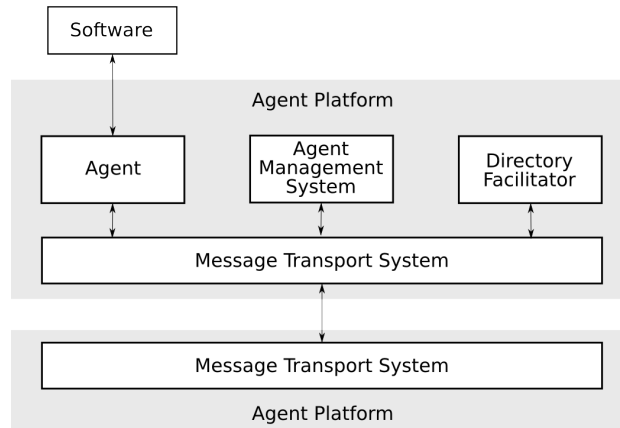


Figure 1: FIPA agent management reference model ([Fo04], Fig. 1, p.5)

Following the notion of *coordination as service* proposed by Viroli and Omicini in [VO06], we present a way of adding planning and scheduling features to a communication centric multi-agent infrastructure such as defined by FIPA: As for any other service provided by the multi-agent infrastructure, whenever an agent requests to use our scheduled method execution as coordination service, it is bound to comply with the specific associated rules and policies. The interaction protocols we provide then guarantee a flawless coordination process, even if scheduled method execution should fail. Note that while in the remainder of this paper we will explicitly address scheduling only, DTC in fact subsumes both planning and scheduling activities, avoiding excessive backtracking in favour of approximate solutions.

The remainder of this paper is organised as follows: In section 2 we briefly survey basic technologies underlying our work (FIPA, Jade, TÆMS and DTC-Scheduling). In section 3 we further motivate our goal of integrating DTC-Scheduling within a FIPA compliant platform. In section 4 we discuss our design in more detail. In section 5 we cover related work. Section 6 concludes with some early evaluation results and an outline of next steps to be undertaken. For further information, we refer the interested reader to the detailed documentation of results available in [He07].

2 Underlying Technologies

2.1 FIPA and JADE

The *Foundation for Intelligent Physical Agents (FIPA)* as an *IEEE Computer Society Standards Organisation* evolved from a precursor association formed by several companies and organisations with special interest in promoting agent-based technologies as industrial standard of practical relevance. FIPA defines domain and implementation

independent requirements for interoperable agent-based systems. The current set of specifications that are ready for implementation are grouped under the term FIPA 2000.

The core FIPA specifications identify essential support services: The *Agent Management System* (AMS) maintains a white pages service, i.e., a directory of agent references. The *Directory Facilitator* (DF) provides a yellow pages service. Agents playing the DF role provide a registration service for agent capabilities as well as an agent capability discovery service. An agent is defined as fundamental actor aggregating one or several service capabilities to form a unified and integrated execution model [BR01]. Agents communicate with one another via the *Message Transport System* (MTS) using the *Agent Communication Language* (ACL) with formally defined semantics. Agent communication is message based on *speech act theory* by John Austin [Au62] and extensions by John Searle [Se70]. Message content is expressed by statements in a so-called *Content Language* (CL). While FIPA does not enforce a specific CL, it provides the definition of its own *Semantic Language* (SL). The MTS is responsible for (i) passing ACL-statements between agents, and (ii) inter-connecting multiple agent platforms. Figure 1 illustrates the FIPA 2000 agent management reference model.

The *Java Agent DEvelopment Framework*⁴ (JADE) is a FIPA 2000 compliant software framework [Be01]. Its development is supervised by the *JADE board* and released under the terms of LGPL version 2. In addition to being FIPA compliant, JADE eases agent-based software development by being designed as a fully distributed platform; providing ready to use implementations of agent interaction protocols; hiding complex intra- and inter-platform communication behind a simple API; and providing a framework for agent construction. Using the well established Java programming language, JADE maps nearly all FIPA concepts to the object-oriented paradigm. JADE provides a plug-in mechanism called *kernel-level services* which deals with platform specific features, such as the automatic translation of ACL encodings. As a representative for FIPA compliant multi-agent infrastructures, JADE was picked as the multi-agent platform of choice for our reference implementation described in section 4.

2.2 TÆMS and DTC-Scheduling

The *Task Analysis, Environment Modeling and Simulation* (TÆMS) framework is a formal, domain-independent modelling language to represent hierarchical task structures. It can be understood as a model of an agent's partial view of a distributed goal tree [Le04]. TÆMS provides language constructs for specification of different ways to achieve a specific goal. Its main abstractions are *tasks* and *methods*. Tasks represent goals which can be further decomposed into sub-goals, which can be again tasks or methods. Methods are atomic actions and therefore always leaf nodes of the distributed goal tree. Additionally, TÆMS has support for explicit *interrelationships* (IRs) between tasks – such as *enables*; *disables*; *hinders*; and *facilitates* – which do not follow a strict hierarchical goal decomposition principle. TÆMS structures thus generally are not trees.

⁴<http://jade.tilab.com/>, last visited Oct. 10, 2007

Figure 2 illustrates such a goal decomposition containing IRs. *Resources* (such as the Vacuum-Cleaner in Figure 2) are used to model interactions with the environment.

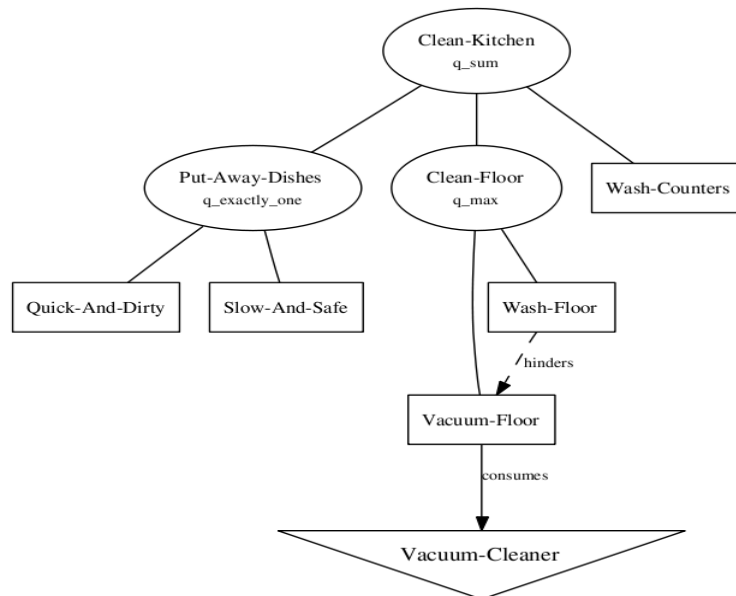


Figure 2: An example TÆMS structure, based on the small cleaning kitchen example in [Ho99]

The most noteworthy aspects of TÆMS are not so much its task decomposition model, but rather its coverage of aspects of relevance for the modelling of soft real-time requirements; handling of uncertainty via probability distributions [Ho99]; and support for the modelling of worth-oriented domains (i.e., where goals can be reached to a certain *degree*, rather than only in an all-or-nothing fashion [Le98, ZR96]). Each method comes with a triple of discrete probability distributions describing quality; cost; and duration of method execution. This approach offers the means to find paths representing ways to achieve a goal to a certain degree. The aggregation of probabilities in sub-task relationships is defined by so-called *quality accumulation functions* (QAFs).

Design-to-Criteria (DTC) scheduling [Wa97] offers a flexible and domain-independent approach to task scheduling. This scheduling algorithm constructs an ordering of TÆMS methods that suffices

- the restrictions of task decomposition;
- temporal constraints;
- commitments of agents;

- resource usage; and
- relative preferences for valid solutions, according to quality; cost; and duration.

Consideration of such relative preferences for solutions sets DTC apart from other scheduling strategies. VIE-CDS [Ju03] is a Java implementation of the DTC scheduling algorithm and the TÆMS framework. It is an integral part of our reference implementation presented in section 4.

3 Integration Design Considerations

Our aim was to design the integration of a scheduling component in as generic a way as possible, so as to ideally enable all FIPA compliant agent frameworks to adopt our approach and thereby enable even inter-platform scheduling. Before we outline our reference implementation, we discuss some ideas and related work which led to our approach.

We first considered exploiting the notion of kernel-level services defined in JADE, the FIPA-compliant platform used as reference for implementation. This low-level feature of JADE would have appeared to allow for a tight integration with the platform; however, it lacks coverage by the FIPA standard. We therefore looked one infrastructural level higher, at the agent level, for a different integration possibility. In FIPA, an agent and everything it can do in its environment are well defined. Agents can register their capabilities as services with the DF, and other agents may use them. So why not follow the “coordination as a service” concept, and register a coordination mechanism with the DF? The notion of coordination mechanism, however, is somewhat abstract, making it a research challenge in its own right to define exactly *what* to register with the DF.

Agent services are not self-explanatory; and taken by themselves, service registrations are just strings. Consequently, we had to provide a related ontology, offering agents the vocabulary necessary to deal with scheduled method execution. Along with the vocabulary, we also needed to define the basic conditions for proper use: This is the point where we re-used FIPA interaction protocols to define under which conditions agents may talk to one another about scheduling issues.

Several architectural design strategies for multi-agent systems exist. There could be a single superior instance coordinating all scheduled action execution. This superior instance would be responsible for collecting all information about action dependencies; building the schedule; and monitoring schedule execution. In case the superior scheduling instance should already have all information necessary for scheduling and execution monitoring, this system design would resemble a mainstream client/server architecture, where the server knows about the clients' capabilities and orchestrates them. Even though such a *centralised* layout introduces a single point of failure, there are scenarios where opting for centralised control can be defended.

To avoid the single point of failure feature/disadvantage, the system designer could allow for several of such scheduling instances. In such a *multi-centric* approach, a fixed scheduling agent could be assigned to each working agent. This would result in a limited advantage over the centralised strategy, because a failing scheduling agent would still cause several dependent agents to stop working (but again, as for the centralised approach, this layout could make sense under certain circumstances). Yet another option, then, would be to relax the fixed hierarchical design and allow for run-time discovery of scheduling services. The advantages of run-time service discovery, however, do not come for free: The multi-agent infrastructure has to provide elaborate communication features for:

- querying the yellow pages agent for a scheduling service;
- contacting the scheduling agent identified; and
- negotiating scheduling conditions with the scheduling agent.

A multi-agent infrastructure providing appropriate communication features as well as a yellow pages service, still shares the same kinds of vulnerability: The single point of failure has merely moved from the scheduling agent to the yellow pages agent. But from the designer's point of view, and assuming that the yellow pages service is provided by the multi-agent infrastructure, the responsibilities for yellow pages service-replication and accessibility have now been delegated.

With each agent constituting a scheduling service, we have a *fully distributed* system planning and scheduling design. A well-known architecture where each agent has planning, scheduling, and execution monitoring capabilities is *Generalized Partial Global Planning*⁵ (GPGP) [DL92], which has its roots in *Partial Global Planning* (PGP) [DL91]. The main idea behind GPGP is to extend the original PGP approach by:

- using TÆMS as formal domain-independent framework to communicate more abstract and hierarchically organised information;
- using TÆMS to detect coordination relationships; and
- separating the process of coordination from local scheduling [DL92].

GPGP assumes a local scheduler and local TÆMS structures representing an agent's partial subjective view of system knowledge at each agent. Information for the scheduler can be represented as local and non-local commitments to entries in the current task structure, or by altering and extending the local task structure. The main field of application for GPGP and TÆMS are worth-oriented domains [Le04]. GPGP defines the following five coordination mechanisms [DL95], which inspired our approach:

- **Updating Non-Local Viewpoints** addresses the identifying of other agents with overlapping beliefs and the gathering of further information about them. Here, “beliefs” are equated to tasks in the local TÆMS structure.

⁵<http://dis.cs.umass.edu/research/gpgp/>, last visited Oct. 12, 2007

- **Communicating Results** enforces a policy that defines which agents participating in scheduled action execution communicate which result qualities to which agent. For example, participating agents could exchange result qualities of each finished task among them all. Another example would be to communicate only result qualities of the goal task.
- **Handling Simple Redundancy** determines which agent should execute a method when multiple agents have the same method execution capabilities.
- **Handling Hard Coordination Relationships** preserves a strict temporal order given by the IRs “enables” and “disables”, with low negotiability.
- **Handling Soft Coordination Relationships** preserves a temporal order given by the IRs “facilitates” and “hinders”, with high negotiability.

By adopting these ideas from GPGP, our approach should be powerful enough to meet requirements of centric; multi-centric; and fully distributed FIPA-based multi-agent scheduling architectures.

4 DTC-Scheduling as FIPA-Based Service

A major design decision taken is to handle planning, scheduling, and execution monitoring (PSEM) as a single big building block⁶. Consequently, these tasks cannot be distributed over different agents. This approach restricts the multi-agent system designer, but saves communication costs by avoiding repeated transmission of bulky representations of TÆMS structures. Each agent is required to provide a simple mechanism for manipulation of its local TÆMS structures. This mechanism has to take care of properly proclaiming local TÆMS knowledge, as well as implementing all scheduling related interaction protocols. Overall, we do not want the agent programmer to be confronted with internals of the scheduling and method execution process, but still provide the greatest degree of control possible.

As mentioned earlier, agents are required to share vocabularies defined by ontologies. Based on ideas taken from [CP99], we use the well-known UML formalism to visualise our ontology structure in Figure 3.

- **BasicOntology** is the root ontology we extend. We assume it to contain all primitive concept and aggregate definitions.
- **TAEMSOntology** mainly deals with wrapping TÆMS structures, defining the vocabulary for TÆMS structure exchange, and providing a predicate to refine a given TÆMS structure.

⁶In the design phase, numerous tradeoffs between flexibility and usability had to be taken. In particular, we do not aim to answer the question whether it is better to use dedicated planning, scheduling, and execution monitoring agents or to provide these abilities to each agent. Therefore, we rather speak about agent roles than concrete agents in the following. Interaction protocols are the glue between these roles.

- **DTCOntology** adds DTC-specific predicates and terms to TAEMSOntology. Its main purpose is to define the vocabulary necessary for requesting scheduled method execution, as well as providing VIE-CDS with scheduling criteria.
- **TAEMSMetaInfOntology** contains a single concept. This ontology is intended to be refined by application specific ontologies which provide meta-information about information encoded in a TÆMS structure. For example, this ontology could be used to provide information about how to resolve redundancies.

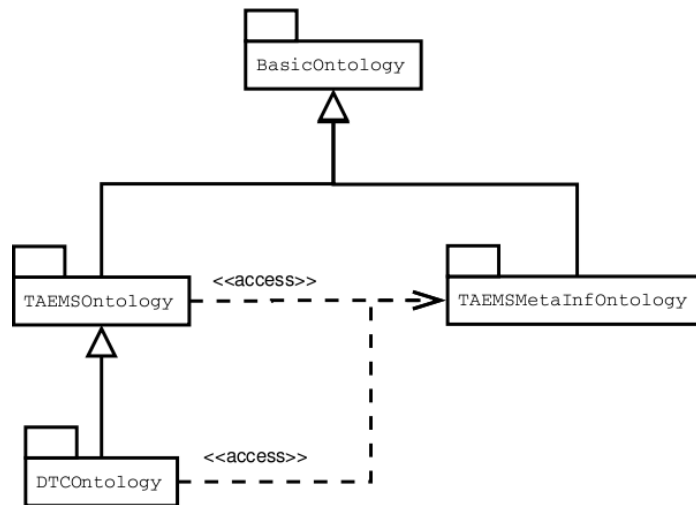


Figure 3: The structure of the scheduling ontology

Service publishing and discovery occurs via the DF. On the one hand, we make standard use of it to register the PSEM service, and on the other hand we exploit it to publish an agent's local TÆMS knowledge. To avoid the DF to be a single point of failure, we recommend to use techniques such as DF replication and federation.

Our scheduling integration approach does not require any component to be added to a FIPA compliant agent platform. We specify it as an orchestration of well defined FIPA interaction protocols. The whole scheduled method execution process consists of five phases, where each phase corresponds to the use of a specific FIPA interaction protocol:

1. **Request scheduled action execution:** The request for scheduled action execution triggers the whole planning, scheduling, and action execution monitoring process. The requester queries the DF for an agent providing a PSEM service. Then it requests a scheduled action execution implemented as FIPA Request Interaction Protocol. Note that this stage has no directly equivalent GPGP mechanism.
2. **Updating non-local viewpoints:** The agent updates its non-local viewpoints and builds a partial global TÆMS structure. This updating occurs as the PSEM

agent looks at its already built partial global plan, and for all tasks and methods in this TÆMS structure queries the DF for agents having TÆMS knowledge about these tasks or methods. These are requested via the FIPA Request Interaction Protocol to share TÆMS knowledge they consider important for the specific task or method. The agent then merges the received TÆMS structures into the existing stub. It continues until no tasks or methods arrive, which could be refined further. This stage subsumes updating non-local viewpoints and handling simple redundancy from GPGP.

3. **Scheduling:** The resulting partial global plan is passed to the scheduling part of the PSEM component. If the scheduling request message contains scheduling criteria, the DTC scheduler is configured with them. Since so far only the problem structure has been determined, it is now time to fetch proposals for concrete TÆMS method execution. Here we use the FIPA Query Interaction Protocol. Subsequently, the scheduler schedules this TÆMS structure and provides a set of possible schedules which is passed on to the next phase. This stage subsumes handling hard and soft coordination relationships from GPGP.
4. **Schedule selection:** The schedule selection phase is realised with the FIPA Request Interaction Protocol. If all commitment providers confirm their commitment proposal at the start and finish times selected by the DTC scheduler, a feasible schedule has been found. This stage subsumes communicating results from GPGP.
5. **Execution monitoring:** During this phase, the PSEM agents wait for success and actual method execution quality messages. If all methods of the selected schedule are executed at the right time, PSEM informs the requester agent about the successful schedule execution. If an action execution fails, some re-planning and re-scheduling have to be started. Such recovery strategies are the topic of future work.

Merging partial TÆMS structures is a delicate task. At the moment, we take a very optimistic approach, expecting sub-task decomposition of equally named tasks from different agents to be non-conflicting even if the task sub-nodes differ. Another maybe better approach would be to group these two partial TÆMS structures under a virtual task node. Figure 4 illustrates our current merging approach.

This scheduling protocol could be regarded as a three-layered hierarchical architecture, with the scheduled execution requesting layer on top; the PSEM layer in the middle; and the method execution layer at the bottom. Successful protocol flow across these layers is well defined via FIPA interaction protocols. It must also be possible to propagate failures through these layers. Errors in the top-down direction are propagated via the FIPA Cancel Meta-Protocol. If an error in the bottom-up direction has to be reported, the message type of standard protocol flow is changed from INFORM to FAILURE. Failures at the method execution level occur when the executing agent is not able to meet the requested execution requirements. Such an error could be recovered by the PSEM layer. For example, it could switch to another schedule or use slack times. If the

