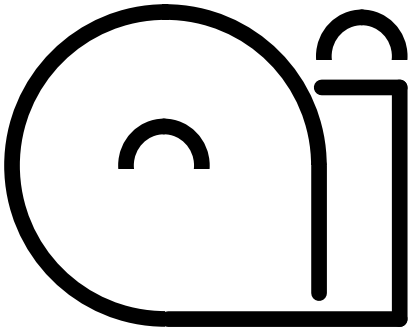


**Österreichisches Forschungsinstitut für /
Austrian Research Institute for /
Artificial Intelligence**

TR–2003–31

Bernhard Jung, Paolo Petta

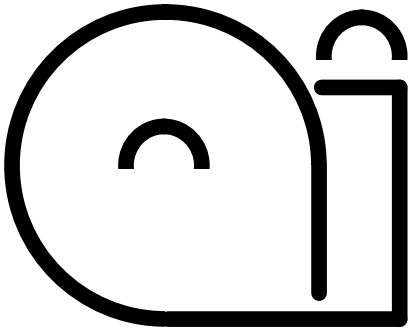
An assessment of the TÆMS/DTC framework in the context of coordinated scheduling and directions for improvements

● Freyung 6/6 ● A-1010 Vienna ● Austria ●

● Phone: +43-1-5336112 ●

● mailto:sec@oefai.at ●

● http://www.oefai.at/oefai/ ●



Österreichisches Forschungsinstitut für / Austrian Research Institute for / Artificial Intelligence

TR–2003–31

Bernhard Jung, Paolo Petta

An assessment of the TÆMS/DTC framework in the context of coordinated scheduling and directions for improvements

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

Citation: Jung B., Petta P.: An assessment of the TÆMS/DTC framework in the context of coordinated scheduling and directions for improvements. Technical Report, Österreichisches Forschungsinstitut für Artificial Intelligence, Wien, TR-2003-31, 2003

An assessment of the TÆMS/DTC framework in the context of coordinated scheduling and directions for improvements

Bernhard Jung bernhard@jung.name
Paolo Petta paolo@oefai.at

Austrian Research Institute for Artificial Intelligence (ÖFAI)*
of the
Austrian Society for Cybernetic Studies (ÖSGK)
Freyung 6/6, A-1010 Vienna, Austria
Phone: (+43-1) 5336112-12,
Fax: (+43-1) 5336112-77

October 16, 2003

Abstract

TÆMS, DTC and GPGP constitute an evolved framework for coordination in multi-agent systems. In this paper we focus on inconsistencies and semantic interpretation problems encountered during an implementation of a DTC scheduler. We try to disambiguate and simplify concepts and propose extensions and new features for TÆMS and DTC to better understand, use, and integrate the framework in an agent architecture.

We indicate how TÆMS/DTC can be modularised to form a kind of construction kit for local agent coordination and control and how they can be extended to support even domain-dependent context. With this, we aim to simplify application and integration of the TÆMS/DTC framework while preserving its core ideas.

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

1 Introduction

In the field of multi-agent systems and specifically in the context of distributed control, autonomous behaviours, multiple goals, and limited resources, the following are among the main aspects of the coordination problem:

Coordination between multiple tasks and goals within a single agent instance Agents pursuing multiple tasks in order to achieve one or more goals have to *sequence their actions* with respect to constraints ordering, task interdependencies, time limits, and goal priorities.

Coordination between multiple agents Multiple agents interacting in a common environment have interests to coordinate their actions to achieve shared goals, minimise redundant efforts, or avoid obstructive actions.

Coordination of resource usage When agents interact with their environment, modeling of resources and resource usage becomes important and thus coordination of resource usage, either between agents or within a single multi-tasking agent.

The TÆMS (Task Analysis, Environment Modeling and Simulation [TÆMS]) framework, taken jointly with GPGP (Generalized Partial Global Planning [GPGP]) and DTC (Design-to-Criteria Scheduling [DTC]), lies between design-time approaches (i.e., organisational restrictions or problem specific coordination protocols) that do not (fully) address problems and opportunities arising dynamically, and run-time approaches (i.e., introduction of significant communication overhead, while still allowing for short-term planning only) that struggle with the huge degrees of freedom of interaction space.

2 Background

TÆMS provides a framework to model hierarchical task-structures for multiple agents. Beyond simply structured decomposed tasks and goals, it also supports descriptions of interdependencies between subtasks (within one or between different agents) and of interrelationships between atomic actions and resources. TÆMS uses a quantitative approach to describe tasks and methods in the three

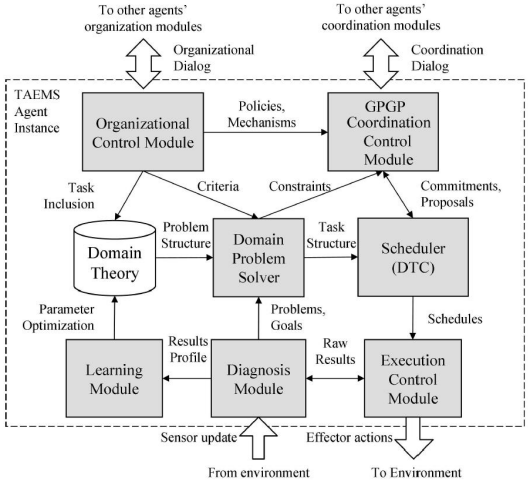


Figure 1: TÆMS-based Agent Architecture (from [Wagner et al., 2003a])

dimensions of quality, cost and duration. Uncertainties are taken into account by using discrete probability distributions.

GPGP uses TÆMS in multi-agent planning to identify and generalise types of coordination relationships. DTC exploits TÆMS to find particular sequences of actions to achieve agents' goals that respect shared actions, time and resource restrictions. Simulation, a third area of TÆMS applications, is considered only partially in this paper, in the context of scheduling.

TÆMS, DTC and GPGP are typically deployed together with a domain planner, an execution and monitoring component, and some additional modules (see figure 1).

TÆMS has been successfully applied in several implementations and domains, including emergency response control [Phelps et al., 2003]; supply chain management [Wagner et al., 2002a, Wagner et al., 2002b, Wagner et al., 2003b]; information gathering [Lesser et al., 2000]; domotics [Lesser et al., 1999]; and distributed sensing [Horling et al., 2001].

The origins of TÆMS date back to the early 1990s.

The existence of several semantic ambiguities and inconsistencies of use of TÆMS concepts across TÆMS itself, GPGP and DTC — introduced partly as a consequence of extensions — are to some extent well known and explicitly stated (e.g. [Horling et al., 2003, Wagner, 2003]), but some further issues arose during

our work on the implementation of a DTC scheduler. The following sections describe TÆMS and DTC briefly; afterwards we address some of these problems encountered.

3 A short introduction to TÆMS

The TÆMS modelling language uses a plain-text LISP-like format to describe task-structures. Files contain descriptions of objects, classified as nodes, resources, interrelationships, and commitments[1].

The node classes of task-group (roots), task (intermediate nodes) and method (leaves) are used to describe hierarchical task-structures modeled as acyclic directed graphs. While the distinction between task-groups and task is only a formal one (regarding the location in the graph), methods differ substantially. Methods are an agent's atomic actions whose outcomes are described in a quantitative probabilistic way for reasoning purposes. The worth or quality of a task depends on the sequence of subtasks and methods and their individual qualities. The exact way to calculate a task's quality is specified by a so called quality accumulation function (QAF). Currently specified QAFs take into account sequencing constraints on the subtasks and functions to calculate the quality (maximum or minimum quality, sum of all individual qualities, quality of last executed subtask). *Non-local* methods cannot be executed by the agent itself but require commitments of other agents (see below).

Resources model dependencies between task-structures and environment. TÆMS supports consumable and non-consumable resources, which share attributes such as current level, and upper and lower bounds. For consumable resources, level adjustment by production and consumption is permanent, for non-consumables levels are reset after method execution.

Apart from the implicit interrelationships (IRs) between tasks, methods and resources according to hierarchical task-structures, TÆMS supports explicit hard and soft IRs. Hard IRs have an on/off style effect on methods or tasks, enabling them to be executed successfully or preventing accrual of quality. Soft IRs increase (*facilitate*) or decrease (*hinder*) qualities. These IRs originate from tasks or methods and are activated when tasks gain quality or methods are executed successfully. The *limits* IR originates in resources and is activated upon out of bounds conditions. The resource-based interrelationships *produce* and *consume*

[1]Further types described in [Horling et al., 2003, Wagner, 2003] are not relevant for issues to be discussed.

must originate in methods and are activated during the method's execution-time. The kind of influence depends on resource type and IR's mode, which can be duration independent — resources are changed only once (if consumable) of twice (if non-consumable) — or per time unit (resources are adjusted at each discrete time tick during the method's runtime).

Commitments fall into local and non-local, depending on their influence on the local agent's view. Commitments are a agents' promises to others to execute or omit the execution of specific tasks or methods within certain time-frames. Non-local commitments refer to non-local methods, whereas local commitments restrict the agents themselves. Agents may be forced to provide other agents with results from own tasks or forbidden to execute methods to not interfere with other agents. *Hard commitments* must not be broken, *normal commitments* should be kept as long as the agent itself is not thereby handicapped heavily, *what-if commitments* are broken when major local scheduling problems arise.

4 Another short introduction: DTC

TÆMS task structures can be regarded as descriptions of coordination, planning and scheduling problems. These can be solved by the design-to-criteria scheduler, if only partially, as it does not perform multi-agent planning, but just considers the commitments coordination restrictions when generating schedules.

DTC emerged out of Design-to-Time (DTT, [Garvey and Lesser, 1995]) and is covered in several publications (e.g., [Wagner et al., 1996, Wagner et al., 1997a, Wagner et al., 1997b, Wagner et al., 1998, Wagner and Lesser, 1999, Wagner, 2000]. Throughout DTC, criteria are used, specifying relative preferences for solutions according to quality, cost, and duration, and the uncertainties along these dimensions. Additional reward can be assigned upon respecting cost or duration limits or achieving some minimum quality. The slider metaphor is usually used to describe these criteria (Figure 2).

The basic DTC algorithm consists mainly of three parts: alternatives generation, method ordering, and application of critics. The first step forms the planning part. Alternatives are unordered lists of methods able to achieve quality for the overall task-groups. As tasks can be achieved in different ways according to their QAF, this problem is not trivial, exposing a combinatorial explosion. Therefore the process is driven by criteria based evaluation that prunes less promising alternatives. The whole process is done bottom-up, creating simple alternatives for low-level tasks first, and then combining these to alternatives for intermediate

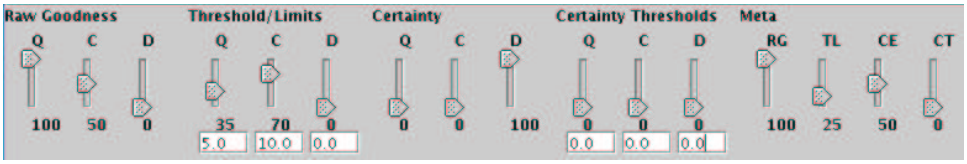


Figure 2: The slider metaphor for describing criteria. (Taken with modifications from [Wagner et al., 1998].)

and top-level tasks. For each alternative, two criteria-based ratings are calculated. *Baseline rating* uses qualities, durations, and costs of methods as described in TÆMS, whereas *potential rating* considers positive effects (from facilitating IRs) to quality, cost, and duration. These ratings are then used to prune subtask alternatives. After generation of all top-level alternatives, the same rating is used to select promising candidate alternatives for the next step.

Method ordering is solved in an approximative way by method rating heuristics that select individual methods to be added to an initially empty schedule. These heuristics account for timing and ordering restrictions from sequencing QAFs, resource limits, interrelationships and commitments. The algorithm starts with the set of unordered methods from the previous step and an empty schedule. Execution of each candidate method is simulated, and the heuristics are applied to judge the situation and rate the method. The best rated method is added to the schedule, and the procedure repeats for the remaining set of candidate methods.

As there is no backtracking in the sequencing step, optimal solutions may not be found. To overcome this, problem critics are introduced that use heuristics to improve schedules. They do not fix schedules directly, but pass new alternatives to the method sequencing component.

When a certain number of schedules has been found or a given deadline for the scheduling process approaches, the schedules are rated again and the best rated schedule is returned for execution. Method execution and monitoring, and potential rescheduling is outside the scope of DTC.

5 Inconsistencies and semantic interpretation problems

Inconsistencies and semantic interpretation problems concerning scheduling exist in TÆMS that are partially well known and documented ([Wagner, 2003, Wagner and Horling, 2001, Wagner et al., 2003a]). But even those that are documented sometimes lack a detailed description and discussion. In this section the problems we had to cope with during the implementation of a DTC scheduler are discussed.

5.1 QAFs: Quality accumulation and subtask ordering

QAFs play a central role in the capability of TÆMS to describe alternative ways to achieve a task. As stated in [Wagner and Horling, 2001], QAFs evolved from simple functions for quality accumulation to more complex functions restricting subtasks ordering. It is not necessarily a bad idea and a simplification in modelling to have such subtask restrictions being imposed by the task itself rather than by numerous *enabling* interrelationships between subtasks. Nevertheless, splitting up the functions of quality accumulation and ordering constraints might help simplify the concept of QAFs and solve the semantic interpretations problems described next.

For *sequencing* QAFs, the quality of a task is accumulated after all subtasks have been executed: only then it can be determined whether ordering restrictions have been violated and whether the task gets any quality at all. Other QAFs, e.g. *sum*, do not put such restrictions on the order of the subtasks: task qualities can be calculated at any time, i.e. before all subtasks have been executed. These cases exemplify two different modes of quality accumulation, with different times of quality accumulation. These times play a central role in interrelationship activation. We pursue this aspect with two examples.

An information retrieval task can be modelled as consisting of many subtasks that gather information. The more subtasks are executed, the better the (likely) quality of the overall task. But after gathering, information has to be consolidated. TÆMS can model this consolidation only as a method, and not implicitly with the task. The consolidation method and the gathering task can be grouped together with a *sum all* QAF that requires both to be executed successfully and that depends on the quality of the gathering task. Additionally, an *enables* IR is established between the gathering task and the consolidate method to enforce correct

ordering. A question now arises: What happens in this model when after the consolidation even more gathering methods are executed? The total quality would increase even if this "new information" had not been consolidated. This problem can be handled by explicitly establishing *disables* IRs from the consolidation method to all gathering methods. In this example we might need a definition for quality accumulation that stops accumulation as soon as the task is regarded as finished.

In the previous example the quality for the task could only grow because of the *sum* QAF. In the case of a another QAF such as the *min* QAF, the quality of the task may be reduced. Any subtask might achieve quality for the task and by that activate an interrelationship. Later on another subtask might fail and thus pull the quality of the task down. In what state does the previously activated interrelationship remain? What happens to methods and tasks that have already been influenced in the meantime? There are different ways of handling this.

In the context of scheduling with DTC, these problems do not seem to occur because for each alternative it is exactly known which subtasks have to be executed, as for QAFs that allow a subset of the subtasks to be executed, all available subsets have been generated (although they might have been pruned). The subtasks that have to be executed and thus the time of quality accumulation is therefore well defined. But what about IRs that could be activated in the meantime by the task? In simulation and for scheduling algorithms that expand alternatives of QAFs differently, the semantics are not clear.

The following proposal might simplify these questions and improve consistency of interpretations. It has to be noted that this idea is not yet fully elaborated, and thus the impact on TÆMS, DTC and GPGP cannot be foreseen fully.

The idea is to introduce two different task types. The first type accumulates quality at a certain time. For sequenced subtasks this is usually the time after the last subtask has been executed; for other QAFs, like *sum*, it might be at any time. After the quality accumulation process, interrelationships get activated and further execution of not yet executed subtasks does not change the task's quality anymore — it freezes quality. The task is considered finished, it has achieved its quality and new subtasks can no longer contribute to it. They are allowed to be executed, may even fail, but the task remains unchanged (even for QAFs like *last*, that usually take the quality of the last executed subtask). The time of accumulation might be given explicitly (e.g., during scheduling, when the subtasks to be executed are known) or implicitly, when an interrelationship originating from the task is used.

The second type performs continuous quality accumulation. After the execution or changes in the quality of a subtask, the quality of the task itself is adjusted.

Interrelationships get activated, but can also be deactivated again after quality drops to zero due to a failing subtask. The quality of such tasks describes more a "world state" than the quality of a task with a well-defined end.

The function of the QAFs of a task might be summarised as and split up into:

- putting sequencing constraints on the subtasks or allowing any ordering,
- describing the way quality from the subtasks is accumulated, and
- defining when the quality is accumulated (once or continuously).

Making the semantics more explicit by focusing on these three aspects separately might improve understanding of QAFs for simulation and scheduling. It could further simplify extensions with new complex accumulation functions, as for example a weighted sum that would resemble the concept of *weights* proposed in [Horling et al., 2003].

5.2 Source factor scaling for interrelationships

TÆMS supports the concept of source factor scaling with interrelationships. The idea behind is that the better a method performs, the stronger the influence of the IR. If the method achieves its maximum quality, the IR has full influence. The problem is that maximum quality is not well-defined. It could be the maximum of the quality distribution for the method, but if this method was facilitated by another, the achievable maximum would be increased.

Apart from the question of the maximum, this concept imposes additional complexity on the calculation of the quality of an influenced method due to uncertainty.

Sample scheduling with the DTC scheduler of the Java Agent Framework ([JAF]) seems to show that source factor scaling is only partially supported: it works for simple methods that are not affected by soft interrelationships, but not for affected methods and whole tasks.

5.3 Interrelationships originating in and pointing to tasks

Another issue that arises together with quality accumulation is the activation of interrelationships and their effects. The activation of IRs coming from methods is well-defined: it becomes active after the method has been executed successfully. For interrelationships originating in tasks, the outcome depends on the time

of quality accumulation. With the ideas outlined before for different task types, activation status of IRs would always be defined and thus allow quality to change for continuous tasks.

The second part to consider with IRs is the way they affect the methods and tasks they point to. For method targets, this is quite clear: they are allowed to be successfully executed (*enable* IR), successful execution may be prevented (*disable* IR), or the quality may be increased or decreased (soft IRs). When pointing to a task, more than one interpretation is conceivable:

- the interrelationship affects each method of the whole subtree below the task independently, as if separate IRs existed for each method;
- the accumulated quality of the task itself is affected after having been calculated by the QAF.

The first interpretation is the one usually used with TÆMS and DTC. The second interpretation matches the notion of an explicit quality accumulation time. In this second case, the IR does not have to be active at the time when all subtasks are executed, because it does not influence their qualities but the quality of the task at the time of accumulation. This interpretation can also be used with tasks that have continuous accumulation. If the semantics of the first interpretation is wanted, direct interrelationships to the subtasks have to be established or a special IR type could declare such a meaning.

5.4 Simulating resources, resource usage and limiting interrelationships

There are two resource models in TÆMS: *per time unit* and *duration independent*. Although they can be mixed in producing, consuming, and limiting interrelationships concerning the same resource, it is helpful to discuss them first separately.

The *duration independent* model is the older and simpler one. A method produces or consumes a certain amount of a resource. The time of consumption is the beginning of the method execution and the time of production is the end. It is not well defined what happens exactly to the level of a resource when it exceeds the upper or lower bound. It could be set to the boundary level or it may overrun the bounds. When the resource level reaches or overruns the bounds, all outgoing limiting IRs become active. These limiting IRs do not take into account the occurrence and extent of lower or upper violation.

The *per time unit model* produces or consumes a certain amount of a resource at each time tick during the method's runtime, and limiting IRs with this model influence methods only during the time units they are active. A simulation of this model even for scheduling purposes is difficult due to uncertainties in the method's duration, the amounts produced or consumed, and hence the resource level. This contrasts to the simulation mode for all other TÆMS objects, which can be simulated sufficiently on a per method basis. The required *per time unit* simulation, and the needs to maintain and combine probability distributions that are partially dependent on each other, increase the complexity significantly.

The discussion of a combination of these two modes in a self-affecting way can more plainly illustrate the problems that have to be handled. Consider a method that produces a resource in the per time unit model and is itself limited by the same resource in a duration independent way. In the middle of the method's execution, the resource level overruns and the limiting interrelationship is activated. Does the IR affect the method execution? And if it does, how exactly have quality, cost and duration to be changed? Is only the part after the activation of the interrelationship affected, or the whole method? This simple example points out the need for simpler models or clear semantics that help to understand and simulate the per time unit model.

5.5 Uncertainty and independence assumption

The introduction of uncertainty in TÆMS had strong impact on the handling of TÆMS structures in the context of scheduling [Garvey and Lesser, 1995, Wagner et al., 2001a]. The maintenance of probability distributions can become quite difficult and requires substantial memory space for combinations of even small single distributions. A compaction of distributions is used in DTC scheduling to cope with this problem, at the cost of losing precision [Wagner, 2000].

The quality, cost, and duration distributions are supposed to be independent. This simplifies working with distributions but might not be the best model for many methods. One can easily imagine an example where quality and cost are duration dependent. But even with initially independent distributions, subsequent events may affect this property. The example usually used is of a task with two subtasks whose qualities are combined by a *minimum* QAF and that are connected by an *enables* interrelationship (Section 4.2.3 in [Wagner, 2000]). If the task, respectively the QAF of the task, has no knowledge of the interrelationship between the two subtasks, the minimum calculated of both qualities does not take into account the dependency between the subtasks. To cope with this problem, the

maintenance of dependent distributions is required, which introduces overheads for combining distributions.

The example described above is not the only one. With resources that follow the "per time unit" model, resource levels are no longer time independent. The activation of a limiting interrelationship depends not only on the amount produced or consumed, but on the duration and timestamp, too. Therefore, the effects of such an IR on the next method cannot be calculated without taking the start time and the dependency between start time and resource level into account. Nearly the same problem arises when interrelationships are delayed, which means that it affects methods not immediately after activation but with a certain delay. To determine whether a method is influenced by a delayed IR, the indirect dependency between the start time of the method and the activation time of the IR must be accounted for. This indirect dependency exists due to the dependency that both have on the method that activated the IR.

The current work on DTC addresses the independence assumption problem only after finishing method ordering and right before the final ranking of schedules, by using a tree-based in-context analysis (Section 4.2.3 in [Wagner, 2000]). A few simple experiments indicate that the problems with delayed IRs are only partially addressed in the current DTC implementation of JAF. In cases with overlapping uncertain methods and delay times, the effect of the IR is not calculated correctly.

6 Proposals for TÆMS and DTC

In this section, some ideas are presented that may improve and simplify understanding and scheduling of TÆMS structures. The main goal is not to alter TÆMS and DTC in fundamental ways, but to simplify, disambiguate, and restructure existing concepts, and integrate a few new ideas.

6.1 A TÆMS Meta Language

Quality accumulation functions: As already stated by [Wagner et al., 2001b], TÆMS lacks a meta language to describe quality accumulation and sequencing restrictions in tasks. Such a meta language that concentrates on explicitly describing the semantics of the three aspects discussed earlier — quality accumulation, method sequencing, and time of quality accumulation — might help in understanding and support for complex QAFs and sequencing constraints.

QAF would no longer be stated as single symbols, but as triplets covering aspects discussed. The semantics of symbols for quality accumulation function and ordering constraints could be defined in the meta language, thus allowing for easier extension of the range of available functions and constraints.

Method execution characteristics and Criteria: Another respect for which a meta language might prove helpful, is the description of method execution characteristics and quality. Using quality, cost, and duration, appears to be a domain-independent common denominator, but in some cases it might be useful to extend it to multi-dimensional costs or qualities. The dimensions used and their mapping to criteria could be generalised and externalised in a meta language. The core of DTC would remain domain-independent, but could provide better support for complex domains.

The definition of quality, cost, and duration distributions would be replaced by distributions in all dimensions specified in the meta language. The grouping of dimensions to "slider banks" would also be defined in the meta language, whereby the concepts of limits and threshold, certainty, and certainty thresholds would remain unchanged for all dimensions.

Domain-dependent contextual decisions: A meta language with complex sequencing constraints and accumulation functions might also be the starting point to provide support for domain-dependent contextual decisions. The only context that TÆMS can access from within a given TÆMS structure is the set of already executed methods and the implicit context that is inherent in the task-structure itself. Accessing external context to decide on the exact behaviour of certain QAFs or interrelationships could make TÆMS task structures more general and take responsibility from a domain planner.

The idea would be to declare extension points in the meta language for domain-dependent user-defined QAFs and sequencing constraint functions that would not be more specific than the standard due to the requirements for DTC scheduling, but would allow to choose a concrete behaviour depending on an external context.

Modelling support: To exploit a domain dependent extension for complex QAFs and sequencing constraint functions, modeling support can be useful. This support could address the definition of complex QAFs and sequencing constraints functions, and their dependencies on external context. This might be

done by describing context dependent practices for tasks. Contextual graphs [Brezillon, 2003] could be a starting point for such a modelling, as they seem to easily integrate in user modelled projects.

Semantic disambiguation: The meta language can also be used to explicitly define the semantic interpretations for some of the issues discussed in the previous section. This can help in a common understanding of TÆMS task-structures both for humans and schedulers to decide on the intended simulation, heuristics and decisions.

Support for choosing different interpretations and complexities in DTC can help to evaluate the impact of different interpretations and models on a DTC scheduled MAS system.

6.2 Embedding DTC — Need for a construction kit?

Design-to-Criteria Scheduling has been embedded in several different ways for different problems. Real-Time Agent Control [Wagner and Lesser, 1999] fully exploits the scheduling capabilities of DTC, whereas the Soft Real-Time Agent Control Architecture (SRTA, [Horling et al., 2002]) focuses on the planning aspects and uses an additional partial order scheduler.

Because different integration attempts rely on different features of DTC, it could be a good idea to modularise DTC to allow a dynamic assembly of DTC parts for a certain problem or even a certain situation. The design of the existing DTC implementation for JAF allows for inexpensive extension by new rating heuristics, whereas other main parts are not yet replaceable: e.g. in [Wagner and Lesser, 1999] it is mentioned that the pruning method does not fit perfectly for some kinds of TÆMS structures. More sophisticated pruning, only for such special situations, could improve overall performance without getting too general and thus too slow.

Another issue with a more modular and tunable DTC is pointed out in [Raja and Lesser, 2001]. Certain situations require or allow for special reasoning. In a very uncertain environment the exact simulation of resources for example is counterproductive, because the schedule is likely to fail very soon and scheduling takes time that could already have been spent on executing. In another situation a very sophisticated resource modelling could help to generate a very certain long-sighted schedule.

6.3 Negotiation and execution support - Interfaces to domain-dependent components

The initial idea for negotiation support was already laid out in [Garvey et al., 1994] and should be mentioned here on all accounts. A negotiation interface between the scheduler and a domain planner seems to be necessary for good integration, nevertheless it seems to be a hard task to achieve. There is support for explaining scheduling decisions in the current DTC implementation packaged with JAF, but it is not known whether a real integration ever took advantage of these concepts yet. The explaining part might not be sufficient and it might lack possibilities to control and drive the scheduling process into a certain direction. Starting for instance with simple explanation and control support for the concept of hard deadlines and investigating these extension might give evidence on their usefulness.

On the other side of DTC lies the execution and monitoring component. Some improvements there have already been discussed in [Wagner et al., 2001b], but the only interface specified by now is the schedule description. This schedule description is reduced to DTC relevant information and lacks any information passed through the scheduling process by the domain planner. Doing so might provide essential information directly to the execution and monitoring component and can also be used to make domain-dependent contextual decisions.

6.4 Integrating new features in TÆMS/DTC

The integration of the above mentioned extensions ought to be realised without destroying the existing core of TÆMS and DTC. Nevertheless, it should lead to a more open software design of DTC that might reduce performance for gains of modularity and extensibility.

These extension should improve understanding of TÆMS in the context of DTC and upgrade DTC as a planner for hand-crafted task-structures with domain context decision support and adjustable complexity handling.

7 Summary

The TÆMS/DTC framework has proven to be successful in many applications in different areas. But as shown in this paper, there exist some shortcomings in semantic clarity and some unresolved and unhandled inconsistencies.

The features and extensions proposed here could help to overcome these shortcomings and to simplify integration of TÆMS/DTC in agent systems. However, it can not be fully foreseen what the actual consequences of the proposed changes are. Therefore, further work on these proposals, on specification and implementation of the meta language, on a new DTC scheduler and on testing the concepts in concrete MAS is required.

Our DTC implementation is Java based and follows the extensible design proposed, aimed to allow easy integration of new features and providing a configurable run-time executable. The simulation of new special purpose QAFs is supported as well as adding of new ordering heuristics that can base their decisions on domain-dependent context. Furthermore, many interpretation problems have been disambiguated either by defining a certain interpretation for the implementation or by supporting of switching between several modes. The design has been prepared to allow for multi-dimensional quality, cost and durations and the two aspects QAFs — quality accumulation and ordering constraints — are already handled separately. The work on the implementation of these two new features is currently in progress.

References

[TÆMS] TÆMS: A Framework for Task Analysis, Environment Modeling, and Simulation
http://dis.cs.umass.edu/research/taems/

[DTC] Design-to-Criteria Scheduling
http://mas.cs.umass.edu/research/dtc/

[GPGP] Generic Coordination Strategies for Agents
http://mas.cs.umass.edu/research/gpgp/

[JAF] JAF - The Java Agent Framework
http://mas.cs.umass.edu/research/jaf/

[Ash and Dabija, 2000] David W. Ash and G. Vlad Dabija. ”Evolution of Planning” in Planning for Real Time Event Response Management. Prentice Hall. 2000.

[Brezillon, 2003] Patrick Brézillon. Context dynamic and explanation in contextual graphs. In: Modeling and Using Context (CONTEXT-03),

P. Blackburn, C. Ghidini, R.M. Turner and F. Giunchiglia (Eds.). LNAI 2680, Springer Verlag. pp. 94–106. 2003.

[Garvey et al., 1994] Alan Garvey, Keith Decker and Victor Lesser. A Negotiation-based Interface Between a Real-time Scheduler and a Decision-Maker. UMASS Department of Computer Science Technical Report TR-1994-008.

[Garvey and Lesser, 1995] Alan Garvey and Victor Lesser. Design-to-time Scheduling with Uncertainty. UMASS Department of Computer Science Technical Report TR-1995-003.

[Garvey and Lesser, 1996] Alan Garvey and Victor Lesser. Design-to-time Scheduling and Anytime Algorithms. SIGART Bulletin 7:2, pp. 16–19. 1996.

[Horling et al., 2001] Bryan Horling, Regis Vincent, Roger Mailler, Jiaying Shen, Raphen Becker, Kyle Rawlins and Victor Lesser. Distributed Sensor Network for Real Time Tracking. Proceedings of the 5th International Conference on Autonomous Agents. ACM Press. pp. 417–424. 2001.

[Horling et al., 2002] Bryan Horling, Victor Lesser, Regis Vincent and Thomas Wagner. The Soft Real-Time Agent Control Architecture. UMASS Technical Report TR-2002-014. 2002.

[Horling et al., 2003] Bryan Horling, Victor Lesser, Regis Vincent, Tom Wagner, Anita Raja, Shelley Zhang, Keith Decker and Alan Garvey The Taems White Paper.
http://dis.cs.umass.edu/research/taems/white/. 16.04.2003.

[Lesser et al., 1999] Victor Lesser, Michael Atighetchi, Bryan Horling, Brett Benyo, Anita Raja, Regis Vincent, Thomas Wagner, Ping Xuan and Shelley XQ. Zhang. A Multi-Agent System for Intelligent Environment Control. Proceedings of the 3rd International Conference on Autonomous Agents (Agents-99), May, 1999.

[Lesser et al., 2000] Victor Lesser, Bryan Horling, Frank Klassner, Anita Raja, Thomas Wagner and Shelley XQ. Zhang. BIG: A Resource-Bounded Information Gathering and Decision Support Agent.

Journal of Artificial Intelligence, May-2000, vol 118, issue 1-2, pages 197-244.

[Omicini and Ossowski, 2003] Andrea Omicini and Sascha Ossowski. "Objective versus Subjective Coordination in the Engineering of Agent Systems" in Intelligent Information Agents: The AgentLink Perspective. LNAI 2586 (State-of-the-Art Survey). Springer-Verlag , March 2003.

[Phelps et al., 2003] John Phelps, Thomas Wagner and Valerie Guralnik. TAEMS Agents for Distributed Emergency Response Coordination. Honeywell Labs Technical Report ACS-R03-002, January, 2003.

[Raja and Lesser, 2001] Anita Raja and Victor Lesser. Towards Bounded-Rationality in Multi-Agent Systems: A Reinforcement-Learning Based Approach. UMASS Technical Report TR-2001-034. 2001.

[Wagner et al., 1996] Thomas Wagner, Alan Garvey and Victor Lesser. Satisficing Evaluation Functions: The Heart of the New Design-to-Criteria Paradigm. UMASS Department of Computer Science Technical Report TR-1996-082.

[Wagner and Garvey, 1997] Thomas Wagner and Alan Garvey. Leveraging Uncertainty in Design-to-Criteria Scheduling. UMASS Department of Computer Science Technical Report TR-1997-011.

[Wagner et al., 1997a] Thomas Wagner, Alan Garvey and Victor Lesser. Design-to-Criteria Scheduling: Managing Complexity through Goal-Directed Satisficing. In In the Proceedings of the AAAI-97 Workshop on Building Resource-Bounded Reasoning Systems, July 1997. Also available as UMASS Department of Computer Science Technical Report TR-1997-16.

[Wagner et al., 1997b] Thomas Wagner, Alan Garvey and Victor Lesser. Complex Goal Criteria and Its Application in Design-to-Criteria Scheduling. UMASS Department of Computer Science Technical Report TR-1997-010.

[Wagner et al., 1998] Thomas Wagner, Alan Garvey and Victor Lesser. Criteria-Directed Task Scheduling. Journal for Approximate Reasoning–

Special Scheduling Issue, 19:91–118, 1998. Also available as UMASS Technical report TR-1997-059. 1997.

[Wagner and Lesser, 1999] Thomas Wagner and Victor Lesser. Design-to-Criteria Scheduling: Real-Time Agent Control. UMASS Technical Report TR-1999-058. 1999.

[Wagner, 2000] Thomas Wagner. Ph.D. Thesis: Toward Quantified Control For Organizationally Situated Agents. University of Massachusetts Amherst. Februar 2000.

[Wagner et al., 2001a] Thomas Wagner, Anita Raja and Victor Lesser. Modeling Uncertainty and its Implications to Design-to-Criteria Scheduling. Under review, 2001.

[Wagner et al., 2001b] Thomas Wagner, John Phelps, Yuhui Qian, Erik Albert, and Glen Beane. A Modified Architecture for Constructing Real-Time Information Gathering Agents. Agent Oriented Information Systems (AOIS), 2001.

[Wagner and Horling, 2001] Thomas Wagner and Bryan Horling. The Struggle for Reuse and Domain Independence: Research with TÆMS, DTC, and JAF. 2nd Annual International Workshop on Infrastructure for Agents, MAS, and Scalable Multi-Agent Systems, 2001.

[Wagner et al., 2002a] Thomas Wagner, Valerie Guralnik and John Phelps. Software Agents: Enabling Dynamic Supply Chain Management for a Build to Order Product Line. Agents for Business Automation, 2002.

[Wagner et al., 2002b] Thomas Wagner, Valerie Guralnik and John Phelps. Software Agents for Dynamic Supply Chain Management. Proceedings of the Workshop on Agents for Business to Business Ecommerce, AAAI02, 2002.

[Wagner, 2003] Thomas Wagner. Tom's Scheduler Notes and Other Documentation. http://mas.cs.umass.edu/ wagner/research/. 16.04.2003.

[Wagner et al., 2003a] Thomas Wagner, Bryan Horling, Victor Lesser, John Phelps, and Valerie Guralnik, The Struggle for Reuse: Pros and

Cons of Generalization in TÆMS and Its Impact on Technology Transition. 12th International Conference on Intelligent and Adaptive Systems and Software Engineering (IASSE-2003), San Francisco, CA, USA, 2003.

[Wagner et al., 2003b] Thomas Wagner, Valerie Guralnik and John Phelps. TAEMS Agents: Enabling Dynamic Distributed Supply Chain Management. To appear in the Journal of Electronic Commerce Research and Applications, Elsevier, 2003.