The Multi-Disciplinary Design Optimisation (MDO) process of products can be supported by automation of analysis and optimisation steps. The Design and Engineering Engine (DEE) is a useful concept to structure this automation. Within the DEE, a product is parametrically defined using Knowledge Based Engineering (KBE) techniques. The analysis of a particular product instantiation of this product model is performed by discipline analysis tools and a search engine provides a strategy to drive the design toward a feasible design, satisfy functional and performance requirements and constraints. To power the automatic analysis in this MDO setting an Agent Based Framework (ABF) has been developed. From the first release agent implementation several positive and negative aspects were found and addressed. It appeared that the framework requirements from literature often have a holistic approach, striving for integration of the MDO problem on the desktop of the engineer, rather than make use of distributed design capabilities in teams of engineers. In this perspective four levels of scoping are identified: organisational scoping level, framework or integration level, tool or engineering service level and data scoping level. These four scoping levels are a good frame of reference to link the identified actors, the four main established functions of a framework and the recent contributions in engineering framework development. Moreover the four levels of scoping helped to show not only the existence of product and process knowledge but also pointed out that there is a need to glue these independent knowledge areas together by using problem knowledge. With this problem knowledge, in form of a problem definition language, it should be possible to dictate the behaviour of the MDO framework. Conclusively, the second release agent software is described and various industrial use cases and accompanying network architectures are described for which the framework software will be made compatible with. Finally a status update to the implementation is provided.

I. Introduction

DESIGNING ADVANCED ENGINEERING SYSTEMS, like aircraft, is an intrinsically complicated process, essentially a lot of involved and interwoven elements. Teams of engineers need a technology that will enable them to improve virtual access to their ideas, model the multidisciplinary aspect of a product, manipulate geometry and the related knowledge, and investigate multiple what-ifs about their design.

To achieve the above in a reasonable time and with confidence in the reliability of the results, the concept of a Design and Engineering Engine (DEE) is proposed to motor the multi-disciplinary design optimisation (MDO) of aircraft design with engineering teams. In the heart of the DEE a generative aircraft product model is implemented in a multi-model generator (MMG). This modelling tool, using Knowledge Based Engineering (KBE) methodologies, is able to generate many different aircraft configurations and variants, using combinations of specifically developed classes of objects, called High Level Primitives (HLP). The HLPs provide designers with a powerful concept to capture and re-use not only the geometric aspect of design, but also provide capability modules, which include rules for automatic creation of analysis models for various disciplines. Based on the research of the MMG and the HLPs in particular, a framework process primitive has been created and described by Schut et al. This so called Engineering Primitive (EP) integrates methods and knowledge needed to instantiate and Feasilize a design.
All elements in the DEE can be seen as engineering services contributing to a pool of services. An operator actor that needs to determine the behaviour of a possible product solution proposal, selects the services from this services pool. An automation framework through which the behaviour of product solution proposal is evaluated is provided by the multi-agent task environment (MATE)⁶⁷. The article focuses on the design and validation of the second release TeamMate agent framework.

A prototype framework capable of supporting such distributed MDO analysis, using the concept of a DEE, is the TeamMate Multi-Agent Task Environment. This framework is under development and a prototype has been implemented in several DEE projects like a what-if study of a tail-plane design being subject to dynamic loads⁸, a structural optimisation of a wingbox⁴, several master thesis’s and a tool to perform design of electrical wire harnesses.

Since the framework is the enabler for the DEE, this concept is first explained in the next section.

![Diagram of the DEE concept](image)

**Figure 1. The Concept of the DEE to support MDO analysis; left the main process flow; right the Multi-Model Generator and the discipline analysis tools.**

### II. An overview of the DEE concept

A Design and Engineering Engine (DEE) is defined³ as an advanced design environment, where the design process of complex products is supported and accelerated through the automation of non-creative and repetitive design activities. Figure 1 shows the concept of the DEE.

The main components of the DEE are:
- **Initiator**: Responsible for providing feasible starting values for the instantiation of the generative parametric product model.
- **Multi-Model-Generator (MMG)**: Responsible for instantiation of the product model and extracting different views on the model in the form of report files to facilitate the discipline specialist tools.
- **Analysis (Discipline Specialist) tools**: Responsible for evaluating one or several aspects of the design in their domain of discipline (e.g. structural response, aerodynamic performance or manufacturability).
**Converger & Evaluator**: Responsible for checking convergence of the design solution and compliance of the product’s properties with the design requirements and generation of a new design vector. These elements use loops in order to function.

The definition of the product is based on selection (or creation) of High Level Primitives (HLPs). These are functional building blocks, containing an a priori definition of a family of design solutions. These functional blocks are encompassed sets of rules that use sets of parameters to initiate objects that represent the product under consideration. The object oriented approach of the HLPs allows capability modules to specify the representation of the product as desired by various engineering disciplines.

### III. Analysis of the MDO problem domain

Various levels of scoping and several actors have been identified in relation to the MDO problem solving domain. This differentiation in scope and identification of the actors is necessary to focus the development and implementation of solutions for MDO support frameworks.

#### A. Identification of Process versus Product, Scoping and Actors

As seen in Figure 2a, the scoping starts in the top with the *organisational level*. On this level, the design process is executed and managed. The interest of the organisation is that the design problem that needs to be addressed is solved efficiently (within time and budget). All human actors that are identified are part of this organisational level, as this scoping level is the interface between the organisation and the problem solving itself.

Five actors are identified (Figure 2a), of which three actors are actively part of the Design and Build Team (DBT). All three actors have close relationships with another level of scoping. A DBT is characterised by individual members being responsible for their respective knowledge domains and the whole team being responsible for meeting the team objectives and deliverables. The first actor within the DBT is the *operator* actor. This actor is responsible for selecting services provided by the framework to produce a problem solving environment in which a MDO problem is to be solved. This actor does not need to have a full understanding of

---

*Figure 2, (a) Four levels of scoping (Organisation, Agent, Tool and Data) and five actors (maintainer, manager, operator, integrator, specialist) are identified. (b) Relation between process vs product, scoping levels, key advances in framework development*[^10] and technical requirements[^6,9].
all the tools that are involved in solving the problem, this understanding and selection process is carried out on the framework level. An *integrator actor* is responsible for the framework level. The integrator facilitates the cooperation between the organisational level and the tool level. Predominantly this actor is responsible that functions are available on organisational level in order for these functions to operate the framework and that correct interfacing exists between the various tools in the tool level. The third and very important actor is the *specialist*. The specialist is responsible for the correct functioning of discipline analysis tools that provide the engineering services to the framework and consequently to the operator.

The last two actors are placed outside the DBT as they are mainly facilitating actors. The *maintainer* ensures the proper functioning of all software and hardware components within all scoping levels. The *manager* actor ensures that necessary resources are available for the DBT and guards time and resources constraints.

The *framework level* or services integration level is the level for which the *integrator* actor is responsible. There is a 1-many relationship between organisation and framework and a 1-1 relationship between an agent and a tool. Collection of agents and their respective tools form a problem solving environment called a DEE (Figure 5b). The specialist tools are part contained within the *tool level* or (engineering) services level, which is the domain of expertise of the *specialist actor*.

The final scoping level is the *data level*. In essence all data is a product of the tool level and therefore no direct actor is identified. One could say that the specialist actor is indirectly responsible for this level. However, an integrator actor would like to control this level in order to facilitate inter-communication between various tools in order to provide a working framework.

Figure 2b describes the relations found in four important articles related to framework design to support MDO processes. Salas' describes several requirements for framework design that match and overlap with the requirements proposed by Berends' resulting in four requirements groups: Resource Management, Resource Interfacing, Process Execution Support and Information Flow Control. Another interesting observation is that the four key advances in framework development as described by Padula', can be matched on the four scopes described earlier and displayed in Figure 2a.

**B. The missing link between product and process related knowledge: problem relation knowledge.**

All processes and product (family) related knowledge can be developed and viewed as independent knowledge ‘bases’. The main function of an MDO framework is to provide mechanisms to come up with a feasible product (comply with requirement, not necessary optimal), with desirable product behaviour by performing product analysis. To glue the product knowledge and process knowledge a need is identified for a language that can sufficiently describe *how* to integrate these separate bases of knowledge (Figure 3). In this problem definition language, the method to perform automated MDO (search) is described by incorporating various search strategies like Collaborative Optimization (CO), Bi-Level Integrated System Synthesis (BLISS), Concurrent Subspace Optimization (CSSO), or Analytical Target Cascading (ATC), to name a few. This problem definition language does not attempt to create a new search or optimisation strategy but defines concepts and relations to sufficiently control the behaviour of the framework, be able to express and be interpreted by the framework and the specialist tools. Such a problem definition language (Like found in process algebra' and Chi (χ)' is the continued focus of research by the authors.
C. Identification of scoping specific tools

When looking at the role of the four scoping levels within the MDO problem domain, it can be deducted that each level contains a specific part of the MDO solution domain. Moreover, various commercial engineering tool suppliers are active to provide applications used by the engineering intensive industry as can be seen in Figure 4, with a note that the figure with respect to the market suppliers is far from complete.

![Diagram showing scoping levels and market suppliers](image)

**Figure 4.** For each scoping level a set of market supplier and applications is identified. On the left the four key advances in framework development by Padula are listed.

On the data level, product lifecycle management tools are found like Dassault Systemes Enovia MatrixOne, Dassault Systemes Enovia SmarTeam, Siemens UGS TeamCenter, and Oracle Agile PLM. These data level tools provide an enterprise integrated management of product data, storage and versioning control, often integrated with product modelling tools.
The Tool level is the scoping level on which most applications and their (market) suppliers can be linked to. Also Padula\(^{10}\) identified the tool scoping level (*modularity*) as the first level to mature (chronologically) before the data level (*data handling*), framework level (*parallel processing*) and eventually the organisation level (*user interfaces*) matures (see Figure 3, Figure 2b). Several suppliers can be linked when considering solely the structural design and analysis domain, product modelling domain and aerodynamic design and analysis domain. For the structural design and analysis domain products like MSC MD Nastran, Dassault Systemes Simulia Abaqus, Ansys Structural, Siemens UGS Femap, and Siemens NX Nastran are available. For the aerodynamics design and analysis domain these are AMI VSAero, and Ansys Fluent. Finally for the product modelling domain Siemens UGS, Dassault Systemes Catia, Dassault Systemes Solidworks and eventually GDL Genworks can be linked. In this scoping level most of the specialist analysis and modelling tools are placed.

The framework or integration level has its own set of tools. The most common engineering frameworks to date within the industry are LMS/Noesis Optimus, Phoenix Integration Modelcenter and Engenious iSight. These frameworks are all equipped with various design space search tools like optimisers, convergers, Design of Experiments (DoE), full factorial or gaussian search and so forth. The TeamMate research framework as under development by the authors is also linked to this scoping level, however not portrayed in Figure 4. As bridge between the product and process related knowledge (Figure 3), the TeamMate agent framework should cater the interface for the problem definition description as discussed in Section III-B.

The last and top level, organisational level, is the most interesting level. Padula\(^{10}\) describes the creation of user interfaces as the last advancement in design support frameworks, and yet to be discovered. On enterprise integration level high level tools and application are to be found. Suppliers for knowledge engineering and management tools Epistemics PCPack and Mondeca are linked to this level. This level is in active development and in a feutal stage. The next release TeamMate framework software provides technologies in order to integrate intuitive design environments in a later phase. Developing such a design environment is continued focus of research by the authors.

The next section handles the design of the TeamMate framework in an abstract way, with the actors and scoping levels described in this section as a background.

### IV. Design of the Multi-Agent Task Environment Framework

The design of the agent based framework is inspired by the problem that earlier generation design support frameworks address the automation of MDO problems often as a top-down execution of a string of individual discipline analysis tools. These strings are executed from start to finish. These support frameworks are often created by (a team of) engineers during the design process in an implicit way and need heavy adapting when a new MDO problem or product is addressed. This problem is defined as the ah-hoc and inflexibility problem\(^{6}\).

Moreover when errors in a particular discipline analysis tool emerge, the highly coupled nature of an execution string often leaves no other possibility than to re-execute all or parts of the tool chain, even when this is not always necessary. In theory, only those tools that are dependent on output data from the discipline tools that produced an error need to be executed. Re-executing the whole string is a waste of resources in the form of CPU time.
To overcome the identified obstacles a multi-agent task environment is developed that addresses the aforementioned problems in a structured and consistent way: decoupling the knowledge of the product from the process and able to handle a family of design problems (objective 1). Moreover, the framework should prevent waste of resources when partial re-execution of tools is needed (objective 2) and should avoid channelling all data through a single bottleneck (objective 3). Moreover, instead of depicting up-front to each tool its address and freezing this in the chain definition, the problem is communicated to the framework and each agent and tool combination is using its communication skills and knowledge of the problem to request information through a specified, but not tool and address specific, request (objective 4). Entities in the virtual team of agents and tools become Knowledge Workers: respecting their own responsibility for data handling and acquisition within and between disciplines. Finally, when working in a multidisciplinary problem domain, a language should be used to facilitate the clear communication, avoiding engineering domain specific language. Engineering domain specific language is acceptable for internal communication, but a common engineering language is mandated whenever inter-disciplinary communication is concerned (objective 5).

From these objectives the four main functions where drafted and embedded in a set of requirements on which the first release framework software is based. The four main functions are resource management, resource interface, process execution support, and information flow control (Figure 2 and Figure 6). These four established functions form the backbone of the framework design and implementation.

A first generation framework software was developed and tested within several projects and the next chapter handles the review of the first generation software.

V. Review of the First Generation Multi-Agent Tasking Environment

A review of the first generation agents has been carried out. The findings can be found in Table 1 and are described in detail in Berends'.

<table>
<thead>
<tr>
<th>Positive aspects</th>
<th>Negative aspects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The agents can support discipline tools in order for them to operate.</td>
</tr>
<tr>
<td>2</td>
<td>The agents sort out problems and solves them on their own with respect to data retrieval and message handling.</td>
</tr>
<tr>
<td>3</td>
<td>Wrapping of commercially of the shelf tools works.</td>
</tr>
</tbody>
</table>

![Figure 5](image_url) (a) DEE process flow for support of MDO. (b) DEE translated into a hybrid team layout using a multi-agent system to integrate various tools and discipline specialists.
<table>
<thead>
<tr>
<th>Positive aspects</th>
<th>Negative aspects</th>
</tr>
</thead>
<tbody>
<tr>
<td>fine.</td>
<td>sometimes difficult if the methodology used for development does not align with the framework methodology.</td>
</tr>
<tr>
<td>Handling of various data for various DEE’s works good.</td>
<td>Difficult to see what is happening within the components of the DEE at a certain time and place.</td>
</tr>
<tr>
<td>Handling of multiple DEE’s simultaneously works good.</td>
<td>Difficult to picture the workflow of the analysis.</td>
</tr>
<tr>
<td>Handling of multiple Analysis on various DEE's simultaneously.</td>
<td>Multiple agents on a single computer requires active processes and 'screens' pop up, which can be distracting and awkward to a user.</td>
</tr>
<tr>
<td>Data pull system works.</td>
<td>No 'real' version control of data per analysis cycle.</td>
</tr>
<tr>
<td>Failover of master function works.</td>
<td>Communication via web interface is rather limited, due to the way data is presented.</td>
</tr>
<tr>
<td>Easy to interact with the framework remotely via a web interface.</td>
<td>Communication via web interface is not actor specific.</td>
</tr>
<tr>
<td>Can support human teams with this implementation.</td>
<td>The authentication and protection of data is non-existing.</td>
</tr>
<tr>
<td>Implemented Security System is sufficient for the prototype operation.</td>
<td>Communication between agents and actors is not multi-modal.</td>
</tr>
</tbody>
</table>

The most important conclusion that can be drawn on the positive side is that the data pull and communication between the various discipline analysis tools can be integrated in a framework. On the negative side the main conclusion is that the lack of a decent and intuitive user interface prevents further acceptance among engineers.

The main goal for the next release software is to bring the framework closer to the end-users by employing the technical strengths from the first release. The focus of the next release is to improve usability of the framework by the end-user.

Following this review of the first release agents and proposals for a second release, a new set of requirements has been determined based on the excellent work of Salas and Townsend⁹, Padula and Gillian¹⁰ and earlier work by the authors⁶. The review of the requirements and the outcome is described in other work by the authors⁷, the result is displayed here in Figure 6 for completeness. In this figure several colours and fill patterns are used to denominate the origin of the various requirements as found by the mentioned sources.
Based on this set of requirements and findings of the first release framework software, a second generation software is designed and described in the next sections.

VI. Second generation TeamMate agent design and implementation

Based on the review of the framework requirements and first release TeamMate agent, a new generation agent is designed and implemented.

D. A collection of dummy specialist tools used for development.

With a collection of tools, simulating input and output of representative engineering tools, a test setup can be created. This collection of tools is designed to demonstrate most requirements of the agents without getting in the way of complexity.
These tools include (1) dummy tools that provides a delayed creation of prescribed output when it is started, simulating processes like NASTRAN, ICAD, or VSAERO. (2) Various optimisation test functions used to test optimisation algorithms (e.g. Rosenbrock, Six-hump Camel Back test function, Michalewic’s test function, Goldstein-Price test function). Finally, (3) various search algorithms like Sequential Quadratic Programming and Genetic Algorithm. These tools serve as a very controlled environment and replacements for the specialist tools and are either implemented in Matlab or Python.

In order to support the implementation the focus is to complete several test benches to demonstrate compliance of the requirements as presented in Figure 6.

1. A debug test bench implementation

The first step for debugging purposes, a debug test bench is implemented using a single agent and a single tool (Figure 7). This setup is not for checking compliance with requirements directly, however this debug test bench is important to step to more complicated set-up like the Simple Test Bench (Section VI-D-2) and the Parallel Execution test bench (Section VI-D-3).

![Figure 7, Debugging Test Bench implementation using a dummy tool.](image)

2. A Simple Test Bench Implementation

This testbench implementation uses tools from the dummy tools library to form the most simplest of workflows as depicted in Figure 8.

![Figure 8, Simple Test Bench implementation using an optimisation test function.](image)

The simple test bench consists of two agent software implementations (a1,a2) with two wrapped tools. The a1 agent can wrap various search algorithms like a converger or an optimiser, implemented in MATLAB. The a2 agent wraps various optimisation test functions, computing objectives and constraint values, implemented in MATLAB and as described in the start of this section. A user interface in the agents is provided to the operator (actor) in order to demonstrate compliance with various resource interfacing requirements.

This simple workflow is used to demonstrate a fair amount of requirements. Compliance to the following requirements is demonstrated with this collection of tools: The numbers of the requirements corresponds to the numbers in Figure 6.
3. A parallel execution test bench

This testbench is designed to tackle various problems related to parallel execution of tools. The case is to compute objective and constraint values in a parallel fashion, while an optimiser steers a design variable vector to a minimal objective (Figure 9).

![Parallel Execution Test Bench implementation using an optimisation test function and distributed evaluation of objective and constraint functions.](image)

This testbench is implemented in MATLAB and Python. An optimiser function is implemented and wrapped via agent a1. Agent a2 and a3 compute the objective and constraint values respectively. Several iterations are necessary to reach an optimum objective within the desired constraints. Optimisation test functions as described in the start of this section can be implemented (a2). Moreover, this testbench is demonstrating the multi-platform capability so the agents will run in multiple operating systems.

Via this testbench, the following requirements can be demonstrated, next to other requirements already demonstrated in the Simple Test Bench setup (subsection 2).

3.2 Support parallel execution of tools
6.4 Cross Platform (Windows, Mac OSX, Linux, Unix)
6.6 No single point of failure or bottlenecks

This concludes this section on the various benchmark test setups to demonstrate compliance to all requirements presented in Figure 6. The next section provides an overview of the design of the next generation framework software.
E. Release 2.0

The design of the new generation agent builds on the experiences with the first release agents, based on the renewed set of requirements (section V), inspired on web services and a phenomenon called Web 2.0. The focus for the release 2.0 is an inverse from the focus in the first release. The main goal of the next generation is usability and openness where the first release focussed on the demonstration of the technical capabilities. The focus on technical capabilities resulted in a reduced attention for the underlying technology how to communicate this to the end-user, resulting in mediocre critique in the review (Section V). The second release is using the technology and methodology developed and successfully demonstrated for the first release only for the agent implementation part as can be seen in Figure 10.

The second release focuses on usability by the end-user within industry. The way of implementation for industrial applications is very important. It is found that in the last years the IT landscape has changed drastically due to the demand for higher security and data protection against fraud, massive abuse and unauthorised computer intrusions. Industry often secures their corporate network at a higher standard for security than the (until recently) relatively open networks of the research and educational institutes. In order to gain acceptance in industry for corporate wide integration of engineering tools, the software framework must be made compatible with these network architectures. Section VI-F addresses the various industrial architectures for which the TeamMate release 2.0 will be compatible with.

F. Industrial Architectures

Industry and corporations have stringent security policies and consequently have compartmentalised network architectures in place to protect corporate data. Based on a review amongst various commercial partners cooperating in the TeamMate research, several network architectures are identified. Derived from this, several network architecture use cases are drafted to serve as the benchmark for the integration of TeamMate into these architectures. Network architectures outside the scope of the described ones, can be derived from the ones described or the framework software can be re-configure to suit the alternative architecture.

---

§Web 2.0 is a trend in World Wide Web technology, and web design, a second generation of web-based communities and hosted services such as social-networking sites, wikis, blogs, and folksonomies, which aim to facilitate creativity information sharing, collaboration, and sharing among users.
1. **Single agent architecture**
   In Figure 11a a simple architecture is displayed in line with the most simple of use cases as expected a single engineer would use. On a single machine in a corporate Local Area Network (LAN) a single release 2.0 package is installed. Each package can contain multiple agents, so the engineer, identified as operator actor, can generate a simple collection of engineering services. No connections outside the LAN are necessary.

2. **Multi-agent architecture**
   When multiple computers within a corporate LAN need to work together (Figure 11b) this is possible by installing release 2.0 TeamMate software on each computer. These computers can have different system architectures. As long as firewalls on these computers allow bi-directional connections between the agents, the framework is operational. The agents are utilising standardised ports and protocols for their network traffic. No traffic outside the corporate LAN is present.

3. **Hybrid agent architecture.**
   When combining the two LAN’s in Figure 11 the landscape changes drastically. Between the corporate LAN’s there should be a direct connection between all agent installations, which most of the time is not possible due to effected security policies. In order to bridge the two LANs a need arises for an agent installation which acts as a proxy and is accessible by anyone within the connected LANs (Figure 12). This agent, denominated in Figure 12 as MATE server, automatically becomes a master node in the framework and performs master functions, such as distribution of messages, a list of capabilities available within the framework and, in rare cases, data.

   This architecture is currently enforced by recent changes in the IT infrastructure of the Delft University of Technology. Features in the hybrid architecture are:
   - Introduction of a dedicated agent acting in a server fashion. This MATE server in a, so called, demilitarized zone (DMZ), reachable for all agents within the enterprise MATE framework.
   - The MATE server can provide other services as well and open up the web interface (via HTTP or HTTPS) to outside clients. In effect, any agent can provide this web interface service.
   - Bidirectional traffic (firewall opened) between MATE server and various LAN within the enterprise
   - Bidirectional traffic (firewall opened) between different LAN and their contained Agents
   - Introduction of a polling client in a remote LAN segment, possibly integrated in the enterprise infrastructure via a secure Virtual Private Network (VPN).

---

Figure 11. (a) Single Agent Architecture. (b) Multiple Agent Architecture within a corporate LAN: bidirectional traffic via XML-RPC over HTTP(S).
G. Open Standards and Application Programming Interface (API)

As discussed in section VI-E, the TeamMate 2.0 design is based on webservises and open standards. It was discovered during operating and integrating DEEs and tools in the first release software, that a need to communicate direct with the framework by various discipline tools (Matlab, PyCoCo – An application to perform automated FEM analysis) would be beneficial. Several features were introduced to enable this communication. The main feature requested was to instruct the agent to initiate a search request and the ability to instruct the agent that new data was available (Figure 13).

![Figure 12, Hybrid Multi Agent Architecture within enterprise network architecture.](image)

![Figure 13, A feature request by tool developers was to be able to initiate data search requests like the green dotted arrow by the “actor” (in this case another agent) and announce data pattern availability by the tool to the agent.](image)
These features were necessary to be able to integrate search tools (optimisers, convergers) within the framework. Search tools produce a new dataset (variable and parameter vector) for tools within the search loop and need to request the output of the analysis of this dataset.

The need for an open interface, which was in use widespread and integrated in several programming languages was the basis for the choice of an XML-RPC interface for the second release. All communication between the agents is exclusively performed via the XML-RPC interface except data transfer. This interface is known and can be made available to any tool developer that wants to interface with the framework. It might even be possible for any tool developer to mimic the behaviour of the agents by only using the calls to the interface.

VII. Implementation Status

Currently the implementation of the second release Multi Agent Tasking Environment software framework is well underway and scheduled to be able to demonstrate a subset of the capabilities in the second quarter of 2008.

The second release framework will be tested within several National and European funded projects in very close collaboration with industry.

References