The Multi-Disciplinary Design and Optimisation process of products can be supported by automation of analysis and optimisation steps. A 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 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, i.e. compliant with requirements, and desired product behaviour, i.e. that is desired by a designer or design team. To power the automatic analysis in this MDO setting an agent based framework has been developed and a first release is implemented in a real-life DEE called TailorMate. A review of the use of the framework in this DEE has been performed and several positive and negative aspects are addressed. To facilitate the development of the second release agent based framework an analysis of the first release requirements and requirements found in literature has been performed. It is found that the requirements from literature often have a holistic approach towards frameworks, 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. From this literature analysis a new set of requirements is proposed for future framework development, independent of implementation specifics. Conclusively several ideas for the next release of the agent based framework are described, based on the review of the first release and the release of the updated set of requirements.

I. Introduction

THE DESIGN OF AIRCRAFT is an intrinsically complicated process and engineers need a technology that will enable them to virtually access 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 this in a reasonable time and with confidence in the reliability of the results, the concept of a Design and Engineering Engine\textsuperscript{1,2,3,4} (DEE) is proposed to motor the multi-disciplinary design and optimisation (MDO) of aircraft design. In the heart of the DEE a parameterised aircraft product model is implemented in a multi-model generator (MMG). This modelling tool, using Knowledge Based Engineering (KBE) methods, is able to generate many different aircraft configurations and variants, using combinations of specifically developed classes of objects, called High Level Primitives (HLP)\textsuperscript{3}. The HLPs provide designers with a powerful way to capture and re-use not only the geometric aspect of design, but also rules for automatic creation of analysis models for various disciplines.

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 a pilot DEE project. The pilot DEE performs a what-if study of a tail-plane design being subject to dynamic loads\textsuperscript{3}. The project is set up in close collaboration with aerospace industry.
The focus of this paper is to identify and provide solutions to problems related with MDO support frameworks and their implementation into industry. In effect, this paper will review the experience gained with operating the first release Multi-Agent Task Environment and will present a new set of requirements for problem solving frameworks. Finally several ideas for implementation are described for the next release framework agents.

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

II. An overview of the DEE concept

A Design and Engineering Engine (DEE) is defined 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 (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. These elements use loops in order to function.

The definition of the product, i.e. the problem to be solved by the DEE, is based on High Level Primitives (HLPs). These are functional building blocks, which allow the user of the DEE to define a product in a certain product family, which encompasses a structured set of HLPs. These functional blocks are basically sets of rules that use parameters to initiate objects that represent (part of) the product under consideration or to apply an engineering process to the initiated object. The object oriented approach allows the representation of the product and engineering process structure. The KBE environment gives access to a parametric geometric modeller. This allows the rule base to perform all geometric operations normally available in a CAD program.

Figure 1. The Concept of a DEE to support MDO analysis; left the main design process flow; right the Multi-model generator and the analysis tools.
III. Scoping and actors identified within the MDO problem domain

Various levels of abstract 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.

As seen in Figure 2, 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, of which three actors are actively part of the Design and Build Team (DBT). All three actors have close relationship 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 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 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.

IV. Review of first release TeamMate agent framework

This section covers the review of the first release of TeamMate agent framework. It starts with a description of the problems that are addressed and a description of the implementation of the first release. Moreover it addresses several positive and negative aspects that were discovered while using the framework in a real-world design problem, called TailorMate.

B. Framework problem identification and TeamMate design philosophy.

This subsection contains a short overview of the first release TeamMate agent framework. An implementation of this framework is powering a DEE that is capable of performing a manoeuvre load analysis of lifting surfaces (tail empennage) in the preliminary design phase. This DEE is called TailorMate.

The design of the agent based framework is inspired on 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 probably need heavy adapting when a new MDO problem or product is addressed. This problem is defined as the ah-hoc and inflexibility problem.

Moreover when errors in a particular discipline analysis tool emerge, the highly coupled nature of an execution string often leave 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 needs to be executed. Re-executing the whole string is waste resources in the form of CPU time.

Figure 3, (a) DEE process flow for support of MDO. (b) DEE translated into a DBT team layout using a multi-agent system to integrate various tools.
To overcome these 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 domain specific language. Domain specific language is acceptable for internal communication, but engineering language is mandated whenever inter-disciplinary communication is concerned (objective 5).

The multi-agent task environment (Figure 3b) addresses these five primary objectives by including theories from the field of artificial intelligence, social group theories and management. A multi-agent system is using software agents ((Figure 3b, outer circle around the tools) wrapping the individual analysis tool and adding extra functionality to make the agent and tool combination a full team player. The internal design of the agents is based on the steady state model of In’t Veld as shown in Figure 4. This model adds extra elements to any transformation process, which in the case of the agents, is seen as being performed by the enwrapped tool process. Important elements are the filters and the various buffers. The filters are used to determine sanity of the data stream flowing through the agent and the enwrapped tool. The buffers maintain a local record of data used for input and produced by the tools. Control elements provide the necessary control over the transformation process as performed by the agents and the operators.

Figure 4, Steady State Model by In’t Veld in which three zones (input zone, transformation zone and output zone), a filter and a control function is distinguished.

Four main functions are identified for the framework for this release. A resource management function (1) manages the entry, registration and exit of tools and their capabilities (resources) to the Framework. A resource interfacing function (2) supplies a standard for resource wrapping and inter-resource communication. Moreover it provides ways to communicate between the human actors and the agent and tool combination. The process

---

† These four main functions are defined for the first release of the agent framework, a review of these functions is provided in section V in this paper.
execution support function (3) controls the execution of the wrapped tools and finally an information flow control function (4) supplies the DEE framework with a controlled data space.

Using the DEE concept as a baseline for solving MDO problems, the process flow (Figure 3a) followed in a DEE is translated into a framework layout (Figure 3b) in which each major element of a DEE is translated into a discipline tool, embedded and interconnected by a helper agent (Figure 3). In this layout, the discipline tool is owner and responsible for the specific specialist domain, governed by a human discipline specialist. Looking at all human specialists it is obvious they form a human DBT, while the combination of tools and agents can be seen as virtual participants in the DBT.

The most important feature to address the flexibility and no-data-loss problem is the use of a data-pull method within the framework. Each agent within the framework has benevolent social behaviour, meaning that each agent honours and act on other agent requests. An agent does not know the full structure of the process flow, but only knows the patterns of data the enwrapped tool relies on for input and produces at output. These patterns of data are known and published throughout the framework in a unique and consistent fashion in order for other agents to understand the demanded requests. In Figure 5 this data pull process is shown. When an actor requests a certain result from a certain tool (Discipline E), the agent that enwraps this tool is looking in the output buffers, whether or not it can supply the request immediately. If not, it will see if anything related is in its input buffer and when that fails as well, a request is generated for pattern(s) of data that will serve as input for the tool (In case of Discipline E this will be 2 patterns of data C and D). This request is broadcasted to every agent which is a member of the framework and the agents that can provide such a pattern will respond. Than the process of generating requests repeats itself until there is a first agent that is really capable of providing a data pattern. Thereafter the data will flow down again through the tools until the agent from which the start of this process was initiated can provide an output to the agent (human or computer) that originally demanded the information.

![Diagram](image)

**Figure 5.** The request stream flows upwards while the process and data stream flows downwards through the chain of consecutive discipline tools.

C. Review of the use of the first release TeamMate framework

The review of the first release of agents is based on the implementation of the agents in a DEE that is capable of performing a manoeuvre load analysis of a tail empennage for the preliminary design phase. As shown in Figure 6, specialist services selected by operators from the framework to participate in this DEE are a Multi Model Generator.
(MMG), called ACPAM, a finite element mesher (PATRAN) and a set of condensation tools (NASTRAN, MATLAB) to condensate a high fidelity structural model in a low fidelity stiffness and mass model in order to perform the manoeuvre analysis by yet another tool called Var Loads (MATLAB Simulink). A sizing tool is implemented to redesign the tail given the dynamic manoeuvre load. Full Results of this DEE are presented in Cerulli².

As a case study, an existing vertical tail of a long range passenger aircraft is replaced by a new vertical tail, of composite material obtained from the process. The remaining components were represented by a validated baseline model. At first, the attempt is to reproduce the original configuration.

![Diagram of the DEE process for optimisation and redesign of a tail called TailorMate](image)

Figure 6, The DEE process for optimisation and redesign of a tail called TailorMate

The analysis of the Agents focuses on the experience gained with operating DEE and its agents. Several positive and negative aspects of the use of the agents are identified and described in Table 1.

| Table 1, Positive and negative aspects of the first release TeamMate agent as experienced |
|---------------------------------------------------------------|------------------------------------------------------------------|
| **Positive aspects**                                         | **Negative aspects**                                              |
| 1. The agents can support discipline tools in order for      | Configuration of the Agents itself is difficult (raw XML).        |
|     them to operate.                                          |                                                                  |
| 2. The agents sort out problems and solves them on their     | Configuration of the Framework as a whole is difficult.           |
|     own with respect to data retrieval and message handling. |                                                                  |
| 3. Wrapping of commercially of the shelf tools works fine.   | Implementation of in-house built analysis tools is sometimes      |
|                                                           | difficult if the methodology used for development does not      |
|                                                           | align with the framework methodology.                            |
| 4. Handling of various data for various DEE's works good.    | Difficult to see what is happening within the components of the  |
|                                                           | DEE at a certain time and place.                                 |
| 5. Handling of multiple DEE's simultaneously works good.    | Difficult to picture the workflow of the analysis.                |
| 6. Handling of multiple Analysis on various DEE's simultaneously. | Multiple agents on a single computer requires active processes   |
|                                                           | and 'screens' pop up, which can be distracting and awkward to a  |
|                                                           | user.                                                            |
| 7. Data pull system works.                                  | No 'real' version control of data per analysis cycle.             |
| 8. Failover of master function works.                       | Communication via web interface is rather limited, due to the     |
|                                                           | way data is presented.                                            |
| 9. Easy to interact with the framework remotely via a web   | Communication via web interface is not actor specific.            |
|     interface.                                               |                                                                  |
This article does not provide the platform to address all positive and negative aspects in great detail. However, some positive and negative aspects are important to highlight and discuss further.

Positive Aspect 1: The agents can support discipline tools in order for them to operate.
This aspect is the core idea of a framework: integrate tools into a framework in order to solve MDO problems. It can support the analysis tools sufficiently to be able to complete a run in batch using correct input data and producing correct (ergo: sane) output data. Not saying that the data contents should be correct, the datatype and transfer between agents is performed without loss of content or any other bitwise differences. Consistency and correctness of the data is a responsibility belonging to the tool domain users: Specialists. Moreover the agent system is globally capable of completing a DEE loop.

Positive Aspect 2: The agents sort out problems and solve them on their own with respect to data retrieval and message handling
This aspect is also part of the core idea of any framework. In this case the agents supporting the analysis tools are able to handle data. They are able to retrieve data that serve as input for its encompassed tool and are able to serve tool output data to other agents. Moreover agents are capable of serving the correct data files in the correct input directories and retrieve data from output directories.

The agents solve data problems by actively pursuing the data needed in order for the tool to reach a desirable goal state. The term ‘on their own’ reflect the autonomy and authority the agents have to decide and initiate request and pursuit actions in search for data. In this case, all agents act aggressive, in the sense that the agent takes action without provocation or without waiting for other agents to make the first move. This behaviour is supported by using a communication system which entails short messages. All agents are open to all other agent communication and are willing to supply any requests. This requires a message handling capability which is successfully implemented.

Positive Aspect 5 Handling of multiple DEE’s simultaneously works good
The Agent and Tool combination (DEE Component) that are specific created to perform within a certain DEE can peacefully live next to other agents and tools in a single framework, being the collection of all agents that communicate with each other. Within a LAN there is normally only one framework. The dynamic workflow functionality will take care that the right agents and hence the right tools are included in a DEE.

Negative Aspect 1: Configuration of the Agents itself is difficult (raw XML)
It is cumbersome to edit the raw XML file in which the configuration of each agent is depicted. These files are mainly accessible through the file/console interface and with a lot of agents working on multiple computers, difficult to manage and check for consistency (matching of in/output patterns). A better user interface targeted for an integrator needs to be provided in the next release.

Negative Aspect 2: Configuration of the Framework is difficult
Due to the implementation of a data pull scenario and request based process flow, (see section IV-A) the configuration of the framework is not centrally managed and depicted but solved dynamically at problem run-time. Each agent has a limited field of view also called fringe. The agent only knows its required input and output patterns and by matching up those, the process flow in implicitly created. As all agents are independent entities, running on various computers and locations, it is cumbersome to manage these pattern matchings. A better user interface targeted for an integrator needs to be provided.

Negative Aspect 3: Implementation of in-house built analysis tools is sometimes difficult if the methodology used for development does not align with the framework methodology
This aspect surfaced while using the popular scientific tool Matlab, but is true for all inhouse-developed specialist tools. It is demonstrated that Matlab can be wrapped and included as DEE component in the framework. However, in some cases it is also made clear that when you don’t take into account that separate modules inside

<table>
<thead>
<tr>
<th>Positive aspects</th>
<th>Negative aspects</th>
</tr>
</thead>
<tbody>
<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>
MATLAB, that should be able to run independently but, despite make use of global variables are very difficult to integrate using agent frameworks. (Figure 7)

Figure 7, Modules within a monolithic Matlab tool (a) cannot be reused in other DEEs when not made available through agents (b).

Figure 8, In the web interface it is currently difficult to track what is happening, moreover no facilities are available to provide a picture of the implicitly created workflow..

Negative Aspect 5: Difficult to picture the workflow of the analysis

The limited user interface does not show a ‘traditional’ workflow. This workflow knowledge is not known on ‘framework’ level. Each agent knows what to do, but the workflow is not ‘programmed’ into the agents itself. This should be derived from querying the agents. A human agent (actor) can appreciate it a lot when he is presented with a workflow representation.

Negative Aspect 7: No ‘real’ version control of data per analysis cycle

Multiple DEE’s can co-exist next to each other within the framework and are separated using an identification tag called a Design and Engineering Engine Identification (DEEID). This tag is used to label (and thus separating) data and requests from all simultaneously operating DEE’s on the framework. Data handling is currently organised per DEEID only and these DEEID are fixed. Due to the inflexibility of this implementation, real version control of...
data within a certain DEE and a certain execution round is not achieved. In the next release it should be possible to relax the strict implementation and provide better user interface for an operator to produce a new DEEID each time an operator wished to execute a workflow.

**Negative Aspect 8: Communication via web interface is rather limited, due to the way data is presented to this page.**

The presentation of data via web interface (Figure 8) is limited with respect to graphical representation of workflow and current activity in the framework. This relates with negative aspect 5. It is advisable to dynamically generate graphical elements on the page, just as one would do in a normal Graphical User Interface (GUI). Web techniques that are able to include more graphical (client-side generated) content like Scaled Vector Graphics (SVG) and provide more dynamics such as Asynchronous JavaScript and XML (AJAX) could support such graphical interfaces.

**Negative Aspect 9: Communication via web interface is not actor specific**

The current web interface is not actor specific, which clutters the interface for actors with information element not relevant for their role. It is wise to provide such features.

**Negative Aspect 10: The authentication and protection of data is non-existing**

The good thing of having a flat authentication system is that no authentication is required. The flip side is that no data is protected and that everyone can intercept problem data flowing through the framework. In a research environment this is not a problem, but when enabling agent framework functionality across the internet, this becomes an issue. Attention should be given in the next version with respect to authentication and information security.

**Negative Aspect 11: Communication between agents and actors is not multi-modal**

The communication between the agents and the human agents (actors) is either via a web interface or via text (console). Humans (and agents) can do much better like more graphical, speech or represent themselves in other forms like SMS or via an internet messenger (Skype Chat, MSN, ICQ, Jabber, Google Talk). When it is claimed that computer and human agents can work together to solve design problems, agent should be able to interface humans in the most natural way.

Conclusively it can be said that there is room for improvement for the next release of framework agents. A lot of experience is gained by operating the first release and some good positive aspects have proven itself. The demonstration of the data-pull and request based functionality of the agent framework in the first release was proven successful. The main thing lacking is usability by others that the creator of the framework and extension of functionality with respect to security and data authenticity.

### V. Review of published requirements for frameworks.

This section handles an analysis of the published requirement by Salas et al in 1998, in relation to the objectives and functions used to create the first release TeamMate agent framework. Moreover, the review of Engineering Framework Development performed by Padula et al in 2006 will be incorporated as well.

The requirements from Salas and the four key ideas described by Padula have the flavour of creating a monolithic engineering desktop in which a super-engineer has control over all the workflow and has intimate knowledge of the processes involved. This is not a desirable approach and impractical for implementation in industry. In research environments the scale of the MDO problems are rather limited to an intimate group of engineers, where in industry problems can be distributed over various groups, often even dislocated in place (and time).

When a merge of the Salas and the Berends requirements are made, the lessons learned from the first release agent framework are to be taken into account as well.

The four groups of requirements as were proposed by Salas are presented in Table 2.
Table 2. Framework requirements as proposed by Salas

1. Architectural Design
   a. Provide a intuitive GUI
   b. Design using OO principles
   c. Modular and extensible
   d. No unreasonable amount of overhead
   e. Handle large problems
   f. Support collaborative design
   g. Design based on standards

2. Problem Formulation Construction
   a. Allow user to configure branching and iterative MDO problem formulation without low-level programming
   b. Easily re-configurability of MDO problem formulation.
   c. Integration of legacy and proprietary codes
   d. Support several optimisation methods, including subdomain optimisation
   e. Provide debugging facilities

3. Problem Execution
   a. Automate the execution of processes and movement of data
   b. Execute multiple processes in parallel
   c. Execution using heterogeneous computer over a network
   d. Provide User interaction during the design cycle
   e. Operate in batch mode

4. Information Access
   a. Provide database management features
   b. Visualise intermediate and final optimisation and analysis results
   c. Monitor system and execution status
   d. Provide mechanism for fault tolerance

The objectives or requirements proposed by Berends as presented in Table 3.

Table 3. Framework Objectives as proposed by Berends

1. Operational Objectives
   a. Decouple product from process (o1)
   b. Handle a large family of design problems (o2)
   c. No loss of data from earlier successful runs in the workflow (o3)
   d. No single point of failure and bottlenecks (o4)
   e. Multi Modal communication between framework and actors (o5)
   f. Target use in Industrial Research and Research Institutes (o6)

2. Technical Objectives
   a. Use cutting edge technologies for implementation (OO, Web services) (o7)
   b. Be able to run transparent on multiple architectures (Win, Mac, Linux) (o8)
   c. Use standards (o9)
   d. Apply Data security (o10)
   e. Be able to remotely access and control the framework (o11)

From the Berends objectives, four main functions are deduced resulting in the following table.

Table 4. Four Main functions the Framework should provide according to Berends

1. Resource Interfacing.
2. Resource Management
3. Process Execution Support
4. Information Flow Control

The following four Key Advances in framework development have been identified by Padula and is presented in following table:
Between the objectives proposed by Berends and Salas, there are many similarities, although not clearly and directly visible. Looking at the four functions in Table 4, these are in some parts overlapping and the same as the four requirement groups proposed by Salas in Table 2. 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 in section III and the four function of Table 4. This is made visual in Figure 9.

The requirements of Salas and the requirements of Berends are then merged and grouped together. From that six requirements groups emerge, of which are four technical requirements and two groups with implementation constraints. This is in line with standard systems engineering practise, where such requirement trees are often built up similar. The culmination of the merge results in a requirements discovery tree for MDO frameworks and is shown in Figure 10.

**Table 5, Four Key Advances in Framework Development**

<table>
<thead>
<tr>
<th>Key Advance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Modularity</td>
</tr>
<tr>
<td>2. Data Handling</td>
</tr>
<tr>
<td>3. Parallel Processing</td>
</tr>
<tr>
<td>4. User Interface</td>
</tr>
</tbody>
</table>

**Figure 9, Relation between process vs product, scoping levels, key advances in framework development and technical requirements.**
A detailed description of all requirements is included in the appendix due to space limitations within the body of this paper.

VI. Design Solution Proposals

This section covers various implementation proposals for the next release of framework agents based on the addressed positive and negative aspects as discussed in section IV-B and the requirements from section V. The order of the proposed design solution is random.

A. Shadow Tool development environment

In order to facilitate the tool specialist to develop the code that is incorporated in a framework, some helper functions must be available to easily make a shadow environment for the specialist to work on (Figure 11). This
shadow environment can benefit from data in the production environment and when the development version is bug free it can eventually replace the production version.

B. Multimodal Agent and Tool integration

Based on the (programming) experience level of the engineer, several methods of tool and agent interaction can be incorporated.

Python API
By using python to set up a quick agent enabled capability (from agent import SimpleAgent) a python adept developer can integrate very quickly an agent that can connect to a nearby (read: local on the machine? or via a proxy agent to the world) framework.

XML-Remote Procedure Call (RPC), SOAP based interaction
Using XML-RPC based service request to the agent interface a well integrated tool or specialist tool developer can communication with the agent. Think of Asynchronous JavaScript and XML (AJAX) based agent interactions.

XML file based interaction
Using a standard written XML file in a 'visible' directory for the agent that is continually checked, a communication between tool and agent can take place. This is the current mode of operation for any loop tool like Converger, Analyser and Optimiser. Such XML standard should be close to the XML-RPC message to be consistent in implementation and to simplify the implementation for other parties.

Semaphore file based interaction
This is a very limited method of communication that is also the most simple to apply. The tool can signal the agent some status information messages by writing out special files that are continuously searched for by the agent.

C. Multimodal Interaction

Several good and bad things, as well as requirements stated that human agents (Actors) can do better in other modality of communication than text. The easiest way of communication by an actor with a human is via text, layed-out graphically and supported by images in a web-page setting. However, there are tools available to add modalities to the agents. Examples of multimodal communication by the agents are presented below.
Speech
Using speech synthesizer software, an agent could use a messenger functionality to actually call a human agent (actor) to address him and make him aware of a fault that requires the actors attention.

Web GUI
By using AJAX and XML-RPC enabled agents, the web based GUI can be much more interactive than currently. With this technology, the web interface can have almost the same capabilities with respect to usability than a normal programmed user interface. Modification and interaction with 3D objects via a web interface is still awkward.

By using messenger functionality, it is possible to interact with the human developer pro-actively. It connects to a user internet messenger like Skype, MSN, Google Talk or any messenger client. The technology of connecting to these messenger types depends a little bit. The technology behind Jabber, which powers Google Talk as well, is available Open Source and can be used as a communication framework as well. Essentially making all agents messenger clients.

Console
A console can be used for debugging the internal processes of the agents and framework. This facility is primarily for an advanced integrator actor.

D. LAN and WAN integration
Connecting agents through commercial firewall systems should be possible, in order to streamline the flow of messages from one part of the framework to the other. This is possible with a commonly known and easy accessible directory agent as shown in Figure 12 (also known as management, helper or broker agent). By using web services standards, most firewall systems and IT department are acquainted with this business to business communication, it should be easy to achieve cross organisation (WAN) integration.

![Figure 12, When agents have to form groups across the internet or through firewalls a common known management agent should be available to facilitate](image)

E. Agent types
Currently, all agents have the same code base. A three-class distinction between the agents is preferable (Figure 13). This is based on the negative aspect 6.

Micro or Nano agent
This agent contains a very simple transformation and has minimal capabilities in communicating with the framework. There can be a multitude of this class of agents living in the network. Communication over a wider network is always supported by a proxy agent.
Capability agent
This agent is equivalent to the current agents support and support the monolithic commercial off the shelf products like Matlab environments, Patran and ICAD.

Proxy agent
This agent is only once available on a single computer and it controls the network connections with the outside world. Other capability agents, nano and micro agent connect to this proxy agent internally and reroute any messages through this proxy. It does not mean that normal capability agents cannot access the network on their own, but it is preferred to do this via a proxy agent. The proxy agent helps to hide workflow complexity.

F. Distributed Knowledge and Knowledge Bases
Agents can produce locally a part of a wider ontology and distribute this part to the rest of the framework. A broker agent that performs a master function for this functionality aggregates pieces into a global ontology and redistribute this over the framework. The ontology can be used to access the correctness of data input and output and help the agents finding the related tools to which they are dependent on for data. Moreover, it can be used to distribute information with respect to cost prediction on framework level.

G. Automatic documentation
There are several tools available to provide documentation of both the code, the workings of the framework and the framework contents. One could think of links inside the agent webinterface that redirect to a documentation Wiki in which any user can alter the current documented information. Other actors can access, modify or add information in this system.

Tools like Trac support wiki and documentation of the development process. Trac also includes integration with subversion source code versioning repository as provides bug tracking facilities.

H. Dynamic workflow and cost prediction
Using some meta information stored on agent level and querying the network a workflow modelling function in the framework could predict the running cost of a certain workflow.

I. Tender or Auction system for distribution of loads
In order to let multiple agent with the same capabilities cooperate on a problem analysis a nice auctioning system or tender system could be incorporated. The value that will be bid is the amount of (available) resources an agent is able to spend on a task. The social characteristic of the agents is such that these agents want to win a contract. A reward system (or penalty system which is in fact the negation of a reward system) should be in place to prevent non-compliance behaviour.

A tender system is a time limited closed bids auction in which bidder(s) make a single bid on an auction item. The bidder with the lowest or highest bid after the time limit is awarded the contract.

An auction system is a time limited open bid auction in which bidder make a single or multiple bids on an auction item (contract). The bidder with the lowest or highest bid after the time limit is awarded the contract.
VII. Implementation

Implementation of the new release of the TeamMate agents incorporating the requirements and the described design solutions is currently underway.

Appendix

This appendix contains a detailed description of all requirements as described in section V and the requirements discovery tree as provided by Figure 10. For each requirement statement a small description is provided, its scope and the reference from which the requirement originated.

A. Technical Requirement 1: Resource Management

<table>
<thead>
<tr>
<th>1.1</th>
<th>Manage a list of framework components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>A list of framework or resource components that are available in the framework should be managed by an entity and made available to the rest of the framework.</td>
</tr>
<tr>
<td>Scope:</td>
<td>Organisational Scope</td>
</tr>
<tr>
<td>Reference:</td>
<td>Berends, f1.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1.2</th>
<th>Support several optimisation (loop) methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>In MDO design support optimisation is the key feature. Several optimisation codes and type of loops should be available in the framework. The actual integration of the specific optimisation code is a tool in itself.</td>
</tr>
<tr>
<td>Scope:</td>
<td>Organisational Scope</td>
</tr>
<tr>
<td>Reference:</td>
<td>Salas, Problem Formulation (2d)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1.4</th>
<th>Easily reconfigurable for various MDO problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>In a framework it should be easy to change the set of tools used to solve an MDO problem of a particular product. It should support the user in creating alternative analysis of the same product model by combining other tools, or tools on other levels of fidelity or for a different product altogether.</td>
</tr>
<tr>
<td>Scope:</td>
<td>Organisational Scope</td>
</tr>
<tr>
<td>Reference:</td>
<td>Salas, Problem Formulation (2b)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1.5</th>
<th>Provide Debugging Facilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>The framework should provide feedback to the user when MDO problems cannot be executed due to gaps in the workflow definition.</td>
</tr>
<tr>
<td>Scope:</td>
<td>Organisational Scope</td>
</tr>
<tr>
<td>Reference:</td>
<td>Salas, Problem Formulation (2e)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1.6</th>
<th>Handle Multiple problems simultaneously</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>The Framework should be able to handle multiple problems simultaneously. Tools could be shared amongst the problems if necessary, but data should be strictly separated.</td>
</tr>
<tr>
<td>Scope:</td>
<td>Organisational Scope</td>
</tr>
<tr>
<td>Reference:</td>
<td>Berends, Operational objectives, Large family of design problems (o2) Salas, problem Execution (2b)</td>
</tr>
</tbody>
</table>

B. Technical Requirement 2: Resource Interfacing
<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Provide multi-modal communication</td>
<td>Multi-modality in communication is to provide more than a single way of communicating between the software and the actors. Moreover communication between the software and the actors can be made more actor specific, leaving uninteresting information out for certain actors types.</td>
</tr>
<tr>
<td>2.2 Provide tool integration into framework</td>
<td>The actual integration of a tool so that it can be used in the framework is required through this requirement.</td>
</tr>
<tr>
<td>2.3 Provide communication between tools</td>
<td>In order for the tools to work together in a framework, the integration part on the communication side should be implemented as well.</td>
</tr>
<tr>
<td>2.4 Provide user interaction during problem analysis</td>
<td>During an execution of a workflow user interaction should be provided and be possible to start, stop, restart and inspect the workflow products and processes.</td>
</tr>
<tr>
<td>2.5 Visualise intermediate and final results</td>
<td>In order for actors to inspect results (or intermediate results) it is required to provide basic visualisation of product data. Either in textual format or graphical. It is advisable to be able to generate trend graphs.</td>
</tr>
<tr>
<td>2.6 Provide intuitive GUI</td>
<td>An intuitive Graphical User Interface (GUI) is an interface that is usable without digging into a manual to use it properly. The GUI should contain enough information to guide the user to the right information and interaction possibilities that is required by other requirements. Moreover it should be possible to remote access the framework (WAN, LAN) through such a GUI and perform control actions on it.</td>
</tr>
<tr>
<td>Child Values: Provide Remote Access Monitoring and Control</td>
<td></td>
</tr>
<tr>
<td>Scope: Framework Scope</td>
<td></td>
</tr>
<tr>
<td>Reference: Salas, Architectural Design (1a) Berends, Technical Objective (o11)</td>
<td></td>
</tr>
<tr>
<td>3.1 Automate the execution of tools</td>
<td>The execution of tools needs to be started without any user intervention. Smart rules when to start the tool transformation process should be in place.</td>
</tr>
<tr>
<td>Scope: Tool Scope</td>
<td></td>
</tr>
<tr>
<td>Reference: Salas, Problem Execution (3a)</td>
<td></td>
</tr>
</tbody>
</table>
### 3.2 Support parallel execution of tools

**Description:** Both coarse grained parallelism and fine grained parallel execution of a set of problems should be supported by the framework, if the workflow permits parallel execution of tools without introducing faults in the final results.

**Scope:** Tool Scope

**Reference:** Salas, Problem Execution (3b)

### 3.3 Monitor tool status and availability

**Description:** The framework should contain mechanisms to detect the status of a tool that is connected to it and publish that to the rest of the framework. E.g., When a tool is reported as busy, new executions of that tool should be prohibited. Also time in the execution should be monitored. Moreover if the tool produced an error, this should be picked up as well.

**Scope:** Tool Scope

**Reference:** Salas, Information Access (4c)

### 3.4 Operate in Batch mode

**Description:** Provide execution of a set of problems without any user intervention. E.g., nightly test runs could be implemented to check consistency of the framework tools.

**Scope:** Tool Scope

**Reference:** Salas, Problem execution (3e)

### D. Technical Requirement 4: Information flow Control

#### 4.1 Automate the movement of data

**Description:** The framework should provide mechanisms to move data around without user action. The correct data should be send or retrieved by the correct tools.

**Parent:** Information Flow Control

**Scope:** Data Scope

**Reference:** Salas, Problem Execution (3a)

#### 4.2 Filter data

**Description:** The framework should provide the ability to filter incoming and outputted data between tools. This to prevent erroneous data of spilling in the framework by which the final results cannot be trusted.

**Scope:** Data Scope

**Reference:** Berends, f3.2 and f3.5

#### 4.3 Provide mechanism for data fault tolerance

**Description:** To prevent data spill of a tool producing erroneous results into the rest of the framework, a mechanism should be in place to prevent this. Error messages should be fed back to the user. Re-execution of correct data in upstream parts of the workflow should be prohibited. Data produced downstream in the workflow should be flagged suspicious and probably be re-executed.

**Scope:** Data Scope

**Reference:** Salas, Information Access (4d)

Berends, Operational Objectives (o3)
### 4.4 Provide data management techniques

**Description:** Specialist and Operators should be provided with data produced by respectively their tools and the framework. Moreover it should be able to move, copy or delete stored data and remaining data storage space should be visualised.

**Scope:** Data Scope

**Reference:** Salas, Information Access (4a)  
Berends, Information Flow Control (f4.2)

### 4.6 Provide data security

**Description:** Provide data security in order that data can only be retrieved by authentic agents (human or computer) that are authorised to receive such data. Not all data should be provided with a security level.

**Scope:** Data Scope

**Reference:** Berends, Technical Objectives (o10)  
Berends, Recommendations

---

### E. Organisational Constraints

#### 5.1 Limit Target Use

**Description:** Constraints the Target Use of the framework

**Child Values:** Industrial Research  
Academic Research

**Scope:** Organisational Scope

**Reference:** Berends, Operational Objectives (o6)

#### 5.2 Integrate Legacy Codes and Proprietary Codes

**Description:** Require the integration of Legacy Codes and Proprietary Codes (COTS) that are available in target use organisations. E.g. Fortran Aerodynamic Code, Matlab Tools, ICAD, CATIA, Patran, Nastran (various releases) etc.

**Scope:** Organisational Scope

**Reference:** Salas, Problem Formulation (2c)

#### 5.3 Support Collaborative Design

**Description:** Provide collaboration between various discipline specialists to come to a final design solution of a stated problem analysis.

**Scope:** Organisational Scope

**Reference:** Salas, Architectural Design (1f)
### 5.4 Enable Communication with other Process Management Systems

**Description:** It is realised that the TeamMate framework is not the only MDO process management system in the world. There should be (limited) possibilities to interact and provide basic information interchange with other process management systems or other frameworks. E.g. Integration with other framework which can execute parts of the workflow. Extract model and problem data from a PDM system.

**Child Values:**
- Product Data Management
- Product Lifecycle Management Systems
- Requirements Management Systems
- Other Frameworks

**Scope:** Organisational Scope

**Reference:** Berends, Marriage project proposal [Berends et al, 2006b]

### F. Architectural Constraints

#### 6.1 Use standards

**Description:** For interoperability between tools, it is wise to make as much use of Open Data Standards as possible. Not only for data transferred in the framework, also for communication protocols and report contents.

**Child Values:**
- Data Standards
- Communication Standards
- Reporting standards

**Scope:** Organisational Scope

**Reference:**
- Salas, Architectural Design (1g)
- Berends, Technical Objectives (09)

#### 6.2 Use Object Oriented programming techniques

**Description:** Any software component of the framework should be programmed according to Object Oriented programming techniques in order to benefit from the inherit modularity and extensibility.

**Scope:** Organisational Scope

**Reference:**
- Salas, Architectural Design (1b)
- Berends, Recommendation

#### 6.3 Be modular and extensible

**Description:** In order to be modular and extensible in the present time and in the future, make sure that a modular software framework is in place. Define interfaces between these modules and use standard programming patterns where possible to enable flexibility and extendibility. E.g. Deploy a plug-in interface with automatic updates.

**Scope:** Organisational Scope

**Reference:**
- Salas, Architectural Design (1c)
- Berends, Recommendations
6.4 Be cross platform

| Description: | Be able to operate the framework incorporating a multitude of common used computer platforms |
| Child Values: | Microsoft Windows |
|             | Linux |
|             | Unix |
|             | Mac OS X |
| Scope: | Organisational Scope |
| Reference: | Salas, Problem Execution (3c) |
|            | Berends, Technical Objectives (o8) |

6.5 No unreasonable amount of overhead

| Description: | The overhead time used by the framework for management and other actions should be not unreasonable in relation to the time it takes for the tools to complete. |
| Scope: | Organisational Scope |
| Reference: | Salas, Architectural Design (1d) |

6.6 No single point of failure or bottleneck

| Description: | Single point of failures of the framework should be avoided by using fail safe mechanisms. Bottlenecks in data flow schemes should be avoided by using distributed data management and flow control. |
| Scope: | Organisational Scope |
| Reference: | Berends, Operational Objectives (o4) |

References

11Trac, Trac Integrated SCM and Software Management, Edgewall Software, URL: http://trac.edgewall.com