MT Summit VIII
Santiago de Compostela (Spain), 18 - 22 September 2001

A Taste of MALT

Ulrike Bernardi, Petra Gieselmann, Steve McLaughlin

SAIL Labs GmbH
Balanstrasse 57
81541 Munich
Germany
ulrike.bernardi@sail-labs.de, petra.gieselmann@sail-labs.de, steve.mclaughlin@sail-labs.de

Abstract

Globalisation is bringing translation and multilingual information processing to areas where it was previously unknown or relatively unimportant. Today, translation is not only important for reaching global audiences, it is becoming an indispensable component inside other systems and workflows. MALT (Modular Architecture for Linguistic Tools) represents a fresh approach to a relatively new problem; how to provide translation capabilities plus any other vital linguistic tools and components inside a common framework, possibly together with other external applications. MALT's modular structure and multi-tier architecture simplify integration into complex workflow scenarios, and the functional separation in the MALT interface permits new components to be added extremely quickly. The applications and components running under MALT can be accessed locally, in a network environment or as engines of a distributed client-server system such as DTS.

Keywords

Integration, Distributed Architecture, Modularity, CORBA, Java

Introduction: The Need for a New Concept

Globalisation is undoubtedly changing the face of business. And one of the main problems globalisation brings with it is how surmount the very real language barriers which divide the peoples of the world. What should have been obvious from the very beginning is now finally being widely recognised - English will never be universally accepted as the sole vehicle of international communication. And rightly so - for hundreds of political, cultural and educational reasons. The only way forward for global players is to be capable of understanding information and providing content in all the native languages of the targeted countries.

The upshot of all this is that language is now an important element in applications where it was previously unimportant. Translation is and always will be a goal in itself, and is an important line of business. But translation and translation-related activities (such as multilingual information retrieval) now play a secondary but vital role in an ever-widening range of other commercial activities.
It is obvious that this is very different to the way translation and linguistic tools have been used up to now.

The classical problem, large enough in itself, was how to get documents into a particular dedicated processing system and how to get the results out. Now the nature of the problem has changed. The new challenge is to build translation and linguistic processing capacity into many different types of workflow. Texts are no longer coming to the translation system - the translation system is coming to the texts!

Piecemeal and one-off integration of linguistic tools into applications is not the correct approach to this new problem. We need a universal solution - one that can be used for a multitude of tasks ranging from classical Machine Translation and Translation Memory systems right through to applications which need linguistics in a supporting role. This is where MALT (Modular Architecture for Linguistic Tools) comes in.
Essentially, MALT represents the framework in which tools and applications can be combined together in order to build simple or complex workflows. The following scenarios will give some idea of the flexibility and potential of MALT.

MALT Scenarios

Scenario 1: Toolbox for Translators and/or MT Developers

First of all, let's take a look at the classical scenario: a translator who, like all other translators, needs to carry out a large number of tasks in a short amount of time, and wants to get as much help as possible from appropriate tools. Imagine that she wants to translate a completely new manual. Of course, she has access to a Machine Translation (MT) engine for draft translations, and access to Translation Memory (TM). Here, she would certainly appreciate an analysis tool which could tell her whether MT or TM (or a combination of both) would be appropriate for the task in hand.

As well as conventional aids such as a Lexicon Editor, she might also need a tool to extract lexicon entries out of a text corpus, a spelling checker, and maybe even a controlled language tool. And other components which make sure that there are no text passages in different languages or check whether the terminology is used consistently would certainly help automate post-translation Quality Assurance work. Our translator may need any of these tools, and does not want to switch from one application to another; that's the reason why she

wants to use all these tools within an integrated framework like MALT (see Figure 1).

And what about the MT developer: He also needs a suite of tools and components; an MT engine, tools for developing lexicons and grammars, a term extraction tool, a tool for evaluating the analysis and translation, a database with test samples and real texts, different lexicons for concordance, lookup, synonyms etc. Maybe he even needs an MT engine from another language pair and the corresponding lexicons and grammars for comparison and standardisation of procedures. It is especially important that he can change grammar rules or lexicon entries and test them immediately without changing programs or needing to restart the whole system.

A framework such as MALT gives the developer all the required tools and the necessary flexibility (see Figure 2).

offers a serious of technologies to support these goals. Which combination of these technologies our user will actually employ, will depend on the task in hand.
He can use **Information Retrieval** tools to find relevant documents and extract relevant data from them (see Figure 3). **Directory Services** and **Document Clustering** can be used to categorize documents according to a given ontology and to search those directories which best match a particular query.
He could use **Information Extraction** tools to find objects and attributes and prepare them for structured processing, or generate text from structured data in different languages.
Or he could use an **Intelligent Dialogue System** which links his queries to answers (prepared manually or using extraction tools) and Natural Language generation tools.
Maybe he needs to automatically scan incoming texts and needs to know if certain patterns, templates or topics appear in the information stream. For this he would need **Filter&Notify** technology.

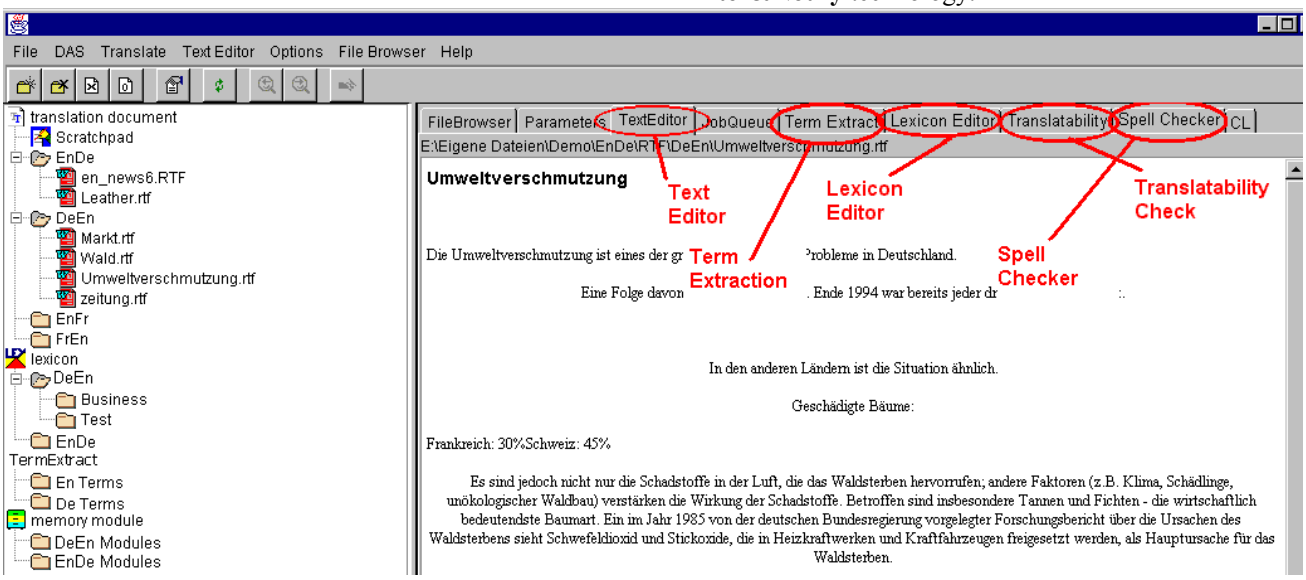

Figure 1: MALT Toolbox for Translators (Configuration Study)

Scenario 2. Toolbox for Multilingual Content Technology

The Content Technology scenario is one that is becoming increasingly familiar.
In this scenario, our user is an ”information broker” in any of a hundred roles; industrial analyst, paralegal, investigator, IT support worker etc. etc. Information, knowledge about it, and access time to it represent money and competitive advantage in very real terms, and our user needs to utilize powerful technology to find answers to his questions as rapidly as possible. The challenge to our user lies in the gigantic amount and diversity of information to be searched. As early as mid-1999, there were more than 800 million pages on the Internet. Looking for a certain piece of information easily turns into trying to find a needle in a haystack.

Content Technology (CT) automates the process of assembling, indexing, extracting, cataloguing, retrieving and even summarizing the information. Traditionally CT

For multimodal information sources, he might well need **Audio and video mining** tools, or **audio and video transcription** tools**.**

Although these are clearly separate technologies, the same components (**tokenizer**, **lemmatizer**, **namer**, **pattern matcher**, **language identifier**, **topic identifier**, **document classifier**, **summarizer**, **query expansion and answer generation** modules etc.) are used in different systems and applications. Our user may even want to call and use some of these components individually, requiring a modular system architecture.
If our information broker needs to consult information sources in more than one language, multilingual and crosslingual CT extensions are necessary (**document and query translation, term substitution** and **multilingual text generation**).

Our ”information brokers” require access to any or all of these CT tools, and needs a primary system which has the flexibility to be configured on a daily basis according to

the current needs in a given scenario. This is the strength of a system like MALT.

Other Scenarios

There are any number of other scenarios where multilingualism is involved and/or huge volumes of information have to be dealt with, and where a wide variety of linguistic and non-linguistic tools could usefully be employed within a common framework and common workflows and without needing to leave the user's normal working environment. They all call for the open architecture and modular approach that MALT provides.

These guiding principles are reflected in the general and component architectures and in the GUI design.

General Architecture

The MALT Framework hosts two main components: the **Document Administration System** (DAS) and the **Operation Area** (OA). The Operation area may contain components such as a File System Browser, a machine translation component (MT), a translation memory component (TM), an information retrieval tool, or even completely different applications.

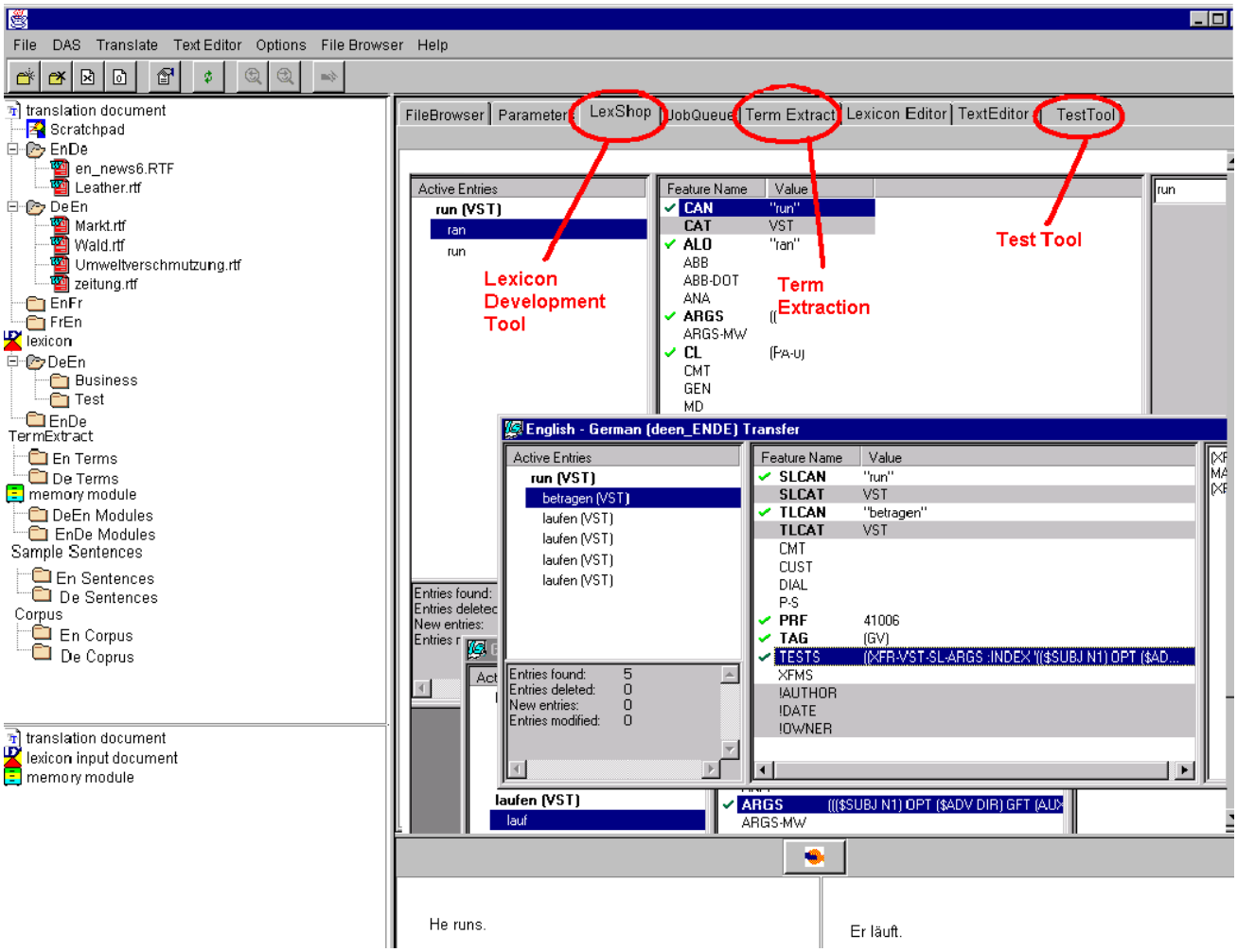

Figure 2: MALT Toolbox for MT Developers (Configuration Study)

Inside MALT

How is MALT able to deal with all these different scenarios? The MALT concept has four guiding principles:

- **Modularity**: MALT is a framework for the integration of diverse components and tools
- **Distributed architecture**: Components can be accessed locally or in a network environment
- **Functional separation**: The MALT interface is divided into functionally separated areas
- **Configurability** on demand

The components in the Operation Area are GUI components. These access **Business Objects** (BOs) like the Machine Translation Object (in this case an MT engine) which do the actual processing work. The MALT system is configurable in different ways to provide access to MT or any other Business Object:

- Local solution: One or more engines are installed on the same computer as MALT.
- Distributed solution: Engines can be accessed through the network.
- Combined solution: The main engine (a frequently used application) is installed locally. Engines used

only occasionally are accessed through the network and can thus be shared by multiple users.
- A sophisticated client server system such as DTS (see below) can be integrated as an engine.

MALT and its engines communicate through CORBA interfaces which bridge the programming 'language gap' between Java (used for MALT) and C++ (used for the engines).

Component Architecture

All components are based upon a three-tier architecture consisting of a **GUI layer**, **a logic layer** and a **persistence layer**. The **GUI layer** is the user interface of the component; it displays a window. The only thing it 'knows' is how to display data. Even logic rules like "control X has to be disabled when checkbox Y is checked" have been eliminated. The GUI layer forwards requests like "Event A has occurred" (in this case: "option Y has been changed to 'true'") to the logic layer.

The **logic layer** contains a set of Business Objects. These provide the data to be shown and supervise the rules and constraints to be followed. Thus when a BO receives requests like "option Y has been changed to 'true'", it checks the logical constraints and reacts on them by requesting the GUI component to disable control X. This layer, however, has no knowledge at all about how to display the data.

The **persistent data** can be stored in a file or a database system. If the data were mirrored on a server machine it would be possible to access user-specific information from any machine.

De-coupling the GUI completely from the business logic makes it easier to distribute the Business Objects over a network. Each time the user requests a service from a GUI object, the GUI object can load different BOs. The BOs can exist on server machines anywhere in a network, whereas the GUI objects are downloaded onto the client machine. Another advantage of separating the GUI completely from the business logic is that the complete GUI can be ported much faster, from JAVA to MFC, for example.

The MALT Framework

During startup the MALT Framework sets up a common menu, a set of toolbars and a list of shortcuts for all components. The Framework then arranges the windows of the components and dispatches any command selected from the main menu, the toolbars or the shortcuts. A **configuration file** provides information about all the components which are to be loaded. This configuration file is a simple text file and can be easily adapted to cater for new components and applications.

The Document Administration System and the Operation Area have fixed relative positions within the Framework. The configuration file contains a separate section for each component, defines additional parameters such as the tab within the Operation Area. On program startup the framework tries to load all components listed in the configuration file. A failure when loading an optional component does not affect the rest of the system. Therefore, the system can be easily configured in different ways.

Document Administration System

The DAS is a browser-like hierarchy that contains folders and documents. The user can add, delete and move folders. For import from the Windows Explorer, folders and documents can be inserted into the DAS using the drag-and-drop function. There are different kinds of documents, depending on the task to be carried out in MALT. For MT, for example, users would import documents in RTF, ASCII or HTML format for translation. But they could also add lexicon files, translation memory modules, grammar files etc. Each Document type has a "primary document", which can be understood as the source file of all related commands. Dependent files and documents are called "satellites".

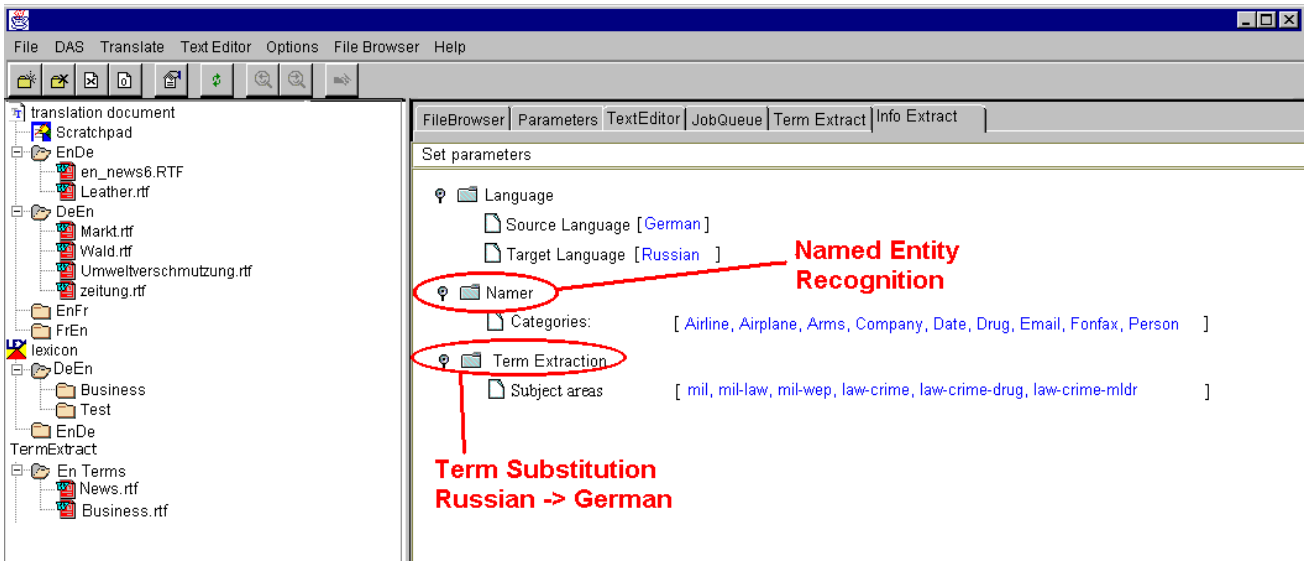

Figure 3: MALT Toolbox for Information Retrieval (Configuration Study)

Satellites are created by executing a command on the primary document. An MT translation, for example, creates a target document and perhaps a Glossary or an Unknown Words list as satellites of the original source document.

Importing Documents into the DAS

The user builds up the DAS by importing files or folders from the File Browser component in the Operation Area or from external programs like Windows Explorer. If the user drops an individual file onto a folder of the DAS, the DAS tries to create a new document of the type that corresponds to the respective root folder. As users may drop any file type onto any folder, the DAS must be able to identify the file type. If the user drops a file system folder onto the DAS, this has the same effect as if all files in this folder, including subdirectories, had been dropped separately. The folder's directory structure is recreated below the node. However, only those documents and files that can be processed by a component in the MALT are actually imported.

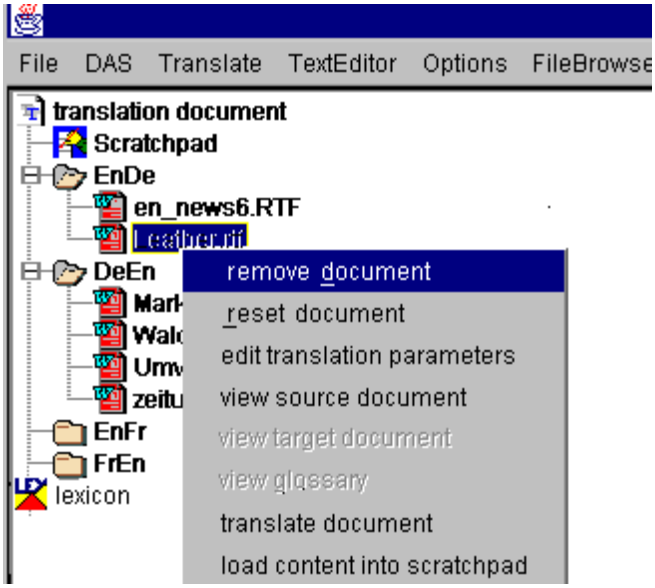

Figure 4: Context Menus in the DAS

Users are free to build up their own folder hierarchy. To provide maximum flexibility, the documents are maintained within a pool. Each display object keeps a reference to one of the pool objects. A pool document may be referenced more than once within the document administration tree.

When the user selects a DAS document and presses the right mouse button, a context menu opens that contains all commands available for the selected document. Some commands are available only when a certain satellite actually exists. For example, the translation of a document can obviously only be viewed after the translation has carried out. Otherwise, the corresponding command is greyed out and disabled. When MALT is started, the system reads the configuration file in order to be able to offer all the commands that can be used with the currently loaded modules.

The Operation Area (OA)

The Operation Area is loaded into the framework as a mandatory component. It hosts a variable number of optional GUI components that serve as editors, viewers or other types of components for the entities that are maintained by the DAS. The components that can be included in the Operation area are defined in the system configuration file. All components mentioned above are GUI components connecting to Business Objects that actually perform the different types of tasks. Each GUI component is represented as a tab folder and it can be opened by clicking on the tab.
The following components would be typically included in the OA:

- **File System Browser:** Allows the user to browse through the file system of the current computer including network connections. To move a file from here to the Document Administration System on the left of the screen, the drag-and-drop function is used.
- **Parameter Settings:** The user selects the settings for the individual jobs. For MT this such would be language direction, stylistic preferences, etc. For information retrieval, the parameter settings would be completely different.
- **Job Queue:** The Job Queue holds jobs of any kind for later processing. Priorities can be assigned to each individual job.
- **The Document Viewer:** MALT also contains a viewer for the displaying documents in formats such as HTML, RTF, and TXT.
- **Lexicon Editor:** Used to add terms to lexica in MT or other applications.
- **ScratchPad**: Translation area of the OA. Used for fast, ad hoc translations of text entered by the user and for testing new lexicon entries or grammar rules.

Adding Components

As we have learned, MALT is a framework for the integration of both linguistic and non-linguistic tools. What's more, MALT is the ideal platform for the creation of multi-vendor systems. Sail Labs plans to fully integrate the majority of its own tools into MALT. This integration work, and indeed the integration of components from other vendors, can take place on various levels.

The top level offers full integration into the Operations Area, i.e. full interaction with the application is available within MALT. Sail Labs' Lexicon Editor or Memory Alignment tool would be an example of this type of integration.

A second level of integration offers the possibility of steering components or applications from within the Operations Area (but without full interactive support), and the viewing of results within the OA. An example of this could be setting parameters (source files, language, objects etc.) to control an information extraction tool and the subsequent possibility to monitor the data found.

At the third level, there is the possibility of calling applications or components from MALT that process

DAS objects but are otherwise self-contained external programs. Examples of these could be tools or programs from other vendors where either no APIs are available or where it would be superfluous or overly complex to fully integrate their interfaces into the Operation Area.

The number of components that can be integrated into the Operation Area is practically unrestricted. To integrate a new component into the MALT, a developer has to carry out the following steps:

- Update the ”optional components” section of the MALT configuration file.
- Implement the Ipage Context interface located in the GUI package. This requires:
  - a Java component class that is hosted by the Operation Area
  - a title to be used for the tab in the Operation Area
  - entries for the main application menu
  - entries for the DAS pop-up menu

The Ipage interface is used to call the components for initialisation and de-initialisation, changing the display language and changing the container where the component is hosted.

DTS under MALT

Business Objects can also be added to MALT as engines running under DTS. DTS, also developed by the Core Components group at Sail Labs, is a loosely coupled distributed environment based on CORBA architecture and using a model analogous to JavaSpaces. DTS is a modular, high-performance and state-of-the-art delivery system providing any type of service within Intranets or the Internet.

Please contact Sail Labs for more information on MALT or DTS.

Bibliographical References

Core Component Department Sail Labs (2001). MALT 1.0: Detailed Functional Specification. Munich

Technology Office Sail Labs (2001). DTS: Distributed Tasks and Services. Munich