[*Translating and the Computer 23*: papers from the Aslib conference held on 29 & 30 November 2001]

# DTS
# A Delivery System for Translation and Translation-Related Services

**Steve McLaughlin, Angela Schmidt**
SAIL Labs Deutschland GmbH
Balanstrasse 57
81541 Munich
Germany
steve.mclaughlin@sail-labs.de, angela.schmidt@sail-labs.de

**Abstract**

Globalisation is bringing translation and multilingual information processing to areas where it was previously unknown, relatively unimportant, or technically just not feasible. Today, Natural Language Understanding techniques are important for reaching global audiences, and are therefore becoming an indispensable component inside many systems and workflows. The main subjects of discussion continue to be the linguistic approaches used by such tools and applications. Equally important, however, are new types of software that support linguistic systems and that provide the information infrastructure to offer linguistic services to the user. NLU technology often means large and memory-intensive applications, unsuitable for installation and use on smaller computers and mobile clients. DTS is a delivery system ideally suited to providing multilingual and translation services to users within a distributed environment, in networks and on thin clients like mobile phones, PDAs and notebooks. With its modular and scalable architecture and automatic load balancing it provides a flexible basis for delivering linguistic capabilities and other services over the Internet or on an Intranet. Internet translation portals, SMS services, or networked multilingual information systems need a distribution architecture like that provided by DTS.

## 1. Introduction

Even the most sophisticated translation functionality is useless without a means of delivering it to the consumer. In an increasingly networked world, companies like Sail Labs are keenly interested in not only providing applications and services but also in building the systems by which people can make use of these services in a distributed environment. This is where a system like DTS comes in. DTS (Distributed Tasks and Services) is a distributed client-server system offering high-level services to customers in different locations. The system is generic and will be used to make Sail Labs' multilingual communication and information solutions available both on the Internet and in corporate Intranets.

The design of DTS is inspired by Sun Microsystems' JavaSpaces™ technology, which allows communication and sharing of objects in a distributed application: JavaSpaces technology acts as a virtual space between providers and requesters of network resources. An additional intelligent event mechanism in Comprendium DTS results in an automatic load balancing capacity. There is no supervisor component which controls all activities in Comprendium DTS and which could easily become a bottleneck for the whole system. Instead, components (engines) ask for more jobs as soon as they are out of work, thereby minimizing administrative overhead and communication costs.

A state-of-the-art delivery system like DTS must offer the following features:
*Modularity:* Functionality should be encapsulated into pluggable components. A core configuration should contains vital functionality only. Any additional utilities should be built on top of the core configuration.
*Scalability:* It must be feasible to extend the system in order to get a better performance (throughput of requests). Especially the administration and communication overhead must be minimized. It must be possible to offer a number of configurations depending on the customer requirements - starting from all system components on a single machine right up to large machine parks
*Portability:* The system has to run on UNIX and Windows NT/2000 environments and must be capable of being used by a wide variety of clients.
*Security:* Secure communication must be guaranteed, and at the same time, conventional communication protocols must be supported.
*Service independence:* The software should be generic, i.e. it should be able to work with any type of service without containing service specific functions. This guarantees a simple integration of new services without software modifications.

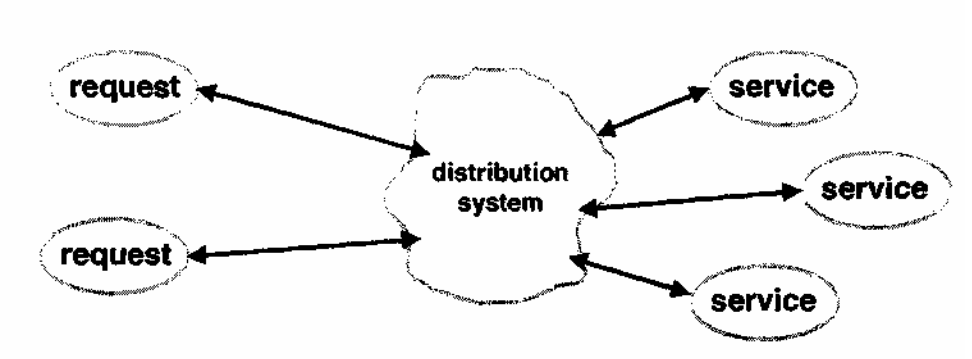


**Figure 1: Service-Request Distribution System**

## 2. DTS Scenarios

Before we delve any deeper into the details of the DTS system, let's take a look at some of the scenarios in which we envisage that a system of this kind could be effectively employed to provide multilingual communication and information services.

**Scenario 1: Translation Portals on the Internet or in a Corporate Intranet**

First of all, let's take a look at a classical multilingual service: an Internet translation portal. It typically serves the purpose of gist translating foreign language information sources. A business analyst in a bank, for example, regularly screens the Internet to find information on European and Asian investments. Rapid access to this type of information represents a real competitive advantage. Occasionally, these sources are not available in English or any language she is able to understand. A translation portal can help her to get a first impression of a document and to assess, whether it might merit the effort and cost of human translation.

It might be especially helpful if specialized resources (like special terminology from the banking and investment area) were available. A translation portal has to offer an intuitive client interface, to support the upload and download of source and target documents and the selection of translation parameters. In such a scenario, security would be a vital issue, too.

Already, there are translation companies that offer specialized portals for fast gist translation of domain specific texts to large numbers of subscribers. As commercial services, they have to ensure that they can flexibly extend their systems if the demand arises. For times of heavy loads, a load balancing solution is a must. To broaden the offered services, it should be easy to integrate new MT language pairs and additional linguistic applications.

A framework such as DTS supports all operations and configurations necessary for this scenario. Sail Labs is currently working on several customer projects based on these types of scenarios.

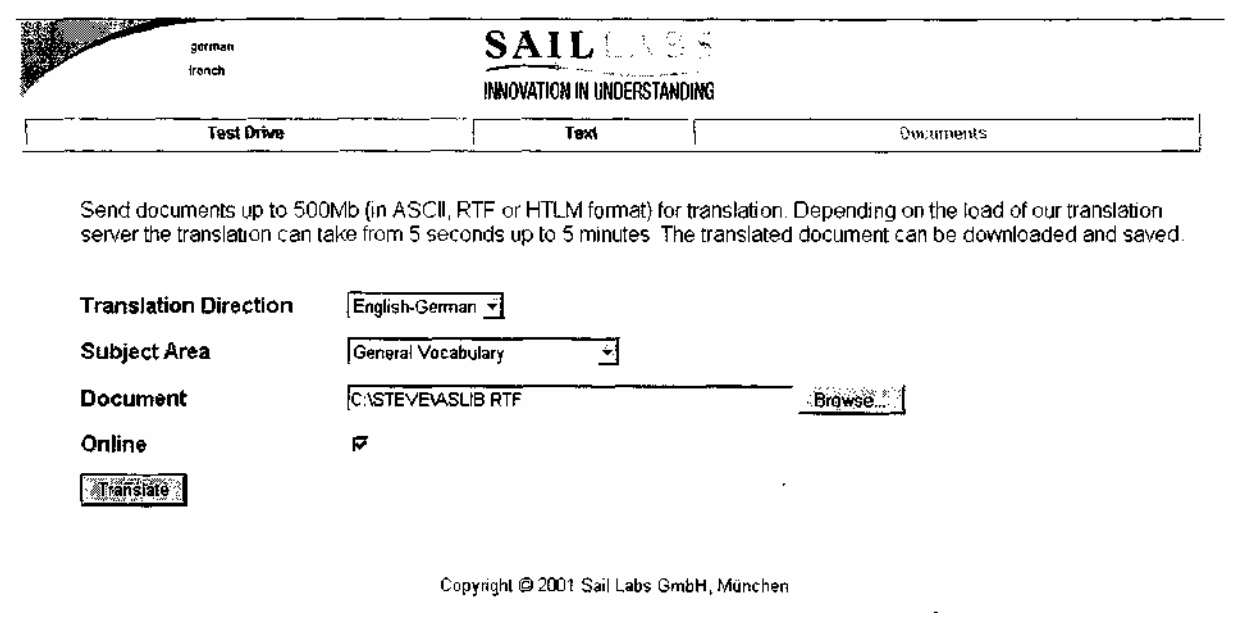


**Figure 2: Browser Client of an MT Portal (Sail Labs's Test Drive)**

### Scenario 2. Information Access on Thin Clients

With thin clients like mobile phones, PDAs and palm tops, ubiquitous computing is within reach. However, these platforms still have their limitations. It is hardly possible to install applications which need large storage space and a high processing capacity, as many NLU programs do. What's more, especially on mobile phones, there is a strict limit to the length of information output. An SMS message, for example, must not exceed 164 characters. This calls for an environment that supports central computing and decentralized distribution, like DTS. If length of output is fixed, there must be summarizing or specialized text generation components.

Let's take a look at the following scenario: Pedro Garcia from Madrid plans to invest in new entries on the Milan Stock exchange. He registers with a specialized information service via a multilingual portal site. There, he chooses to receive daily information on new Milan stocks. He selects the e-mail mode, for delivery. After having carefully analysed the daily Spanish information on the new Milan stock market, he decides to invest on-line in a certain stock. So he configures his profile in order to receive news on it via GSM and/or SMS whenever it becomes available. In addition, he defines an "alert" that notifies him whenever his stock doubles in price. For notification purposes, he chooses the automatic phone-based voice announcement.

After weeks of receiving mostly encouraging SMS messages from different European sources concerning his stock, he gets a call. A friendly voice announces in Spanish that the stock of his choice has doubled in price. Back to his office, he sells all his stocks and plans a trip to the Canary Islands. He feels confident about the fact that even on Lanzarote, the information portal will still provide all the sufficient Spanish information to decide on his next investment - this time maybe in Paris.

A scenario like the one described above would need components like user profile administration, information classification, Natural Language Generation, summarization to access and prepare the information and multilingual capabilities to process and translate foreign language sources. With the help of DTS, all these components could be flexibly installed on a central or distributed configuration. DTS would receive requests from mobile clients and would distribute the output of all the engines involved back to the user.

**Scenario 3. Multilingual Information Systems**

For large international institutions dealing with cross-border problems (such as health, defence, finance, crime-fighting etc.), with dependencies in a large number of countries and centralized data coordination centres, multilingual communication and information is a daily necessity.

In an organization of this kind, for which Sail Labs is currently helping to build a powerful DTS-based system, national units in various European countries collect information in their own language and send this data to a central agency for collation. Once it is stored in

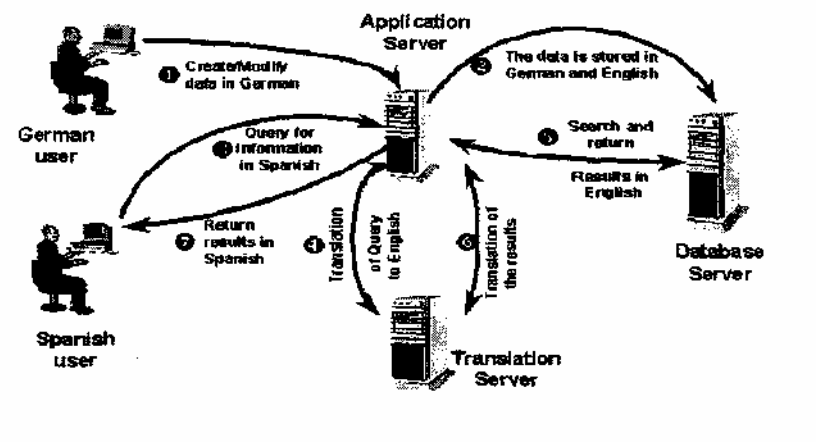


**Figure 3: DTS-Based Multilingual Information System**

databases at this central office (in the original language and in English) the information can be queried from all other national units.

Language, of course, represents an important obstacle in processes of this kind which involve users speaking a wide range of languages. The system allows users in all countries connected to the system to send queries for information in their native language. These queries are then translated into the supported languages and searches carried out for relevant documents in the

central databases. If such documents are found, the information is presented to the requesting unit in a form they can understand.
The DTS system in this application will support a large number of engines running a range of Sail Labs technologies such as Information Retrieval, Query Translation, Translation Memory and Term Substitution. Machine Translation may also be added in a further phase.

## 3. DTS Architecture and Workflow

DTS is a loosely coupled distributed environment based on CORBA architecture and using a model analogous to JavaSpaces. CORBA is a platform- and language-independent "middleware" providing common interfaces for interoperation of heterogeneous systems. In a Java Spaces model, a virtual space is used to exchange tasks, requests and information among participants in a distributed solution. This virtual space is called a *Pool* in DTS. Clients, Engines and other components synchronize their behavior by writing entries to and reading entries from such a Pool. The DTS term for such an entry is *Task.* Changes of state of tasks in the Pool *(Waiting, Running, Done)* are communicated to connected components via what is known as a *Filtering Queue* (see diagram below).

If, for example, a user wishes to use an SMS Translate Sentence service (from English to German) offered by a DTS system, a Client (which there can be many different kinds, C++ applications, CGI programs, applets, servlets etc.) first writes an appropriate Task to the Pool with which it is connected. The Task has the state *Waiting.* The next step is for a suitable engine to take this Task. However, as not every Engine can carry out a Translate Sentence operation from English into German, the Pool "pushes" tasks into different Filtering Queues.

A Filtering Queue is a type of event channel which communicates changes of Task states only to those components which are interested in a specific event. This is illustrated in the diagram below; here task A is put into the first filtering queue and task B into the second one. There is a Filtering Queue for every type of event. An Engine which can carry out Translate Sentence operation from English to German, for example, only needs to monitor the Filtering Queue which handles events connected with this kind of task. An "interested" Engine thus learns that there is a *Waiting* task of the kind it can deal with and "pulls" the task from the Filtering

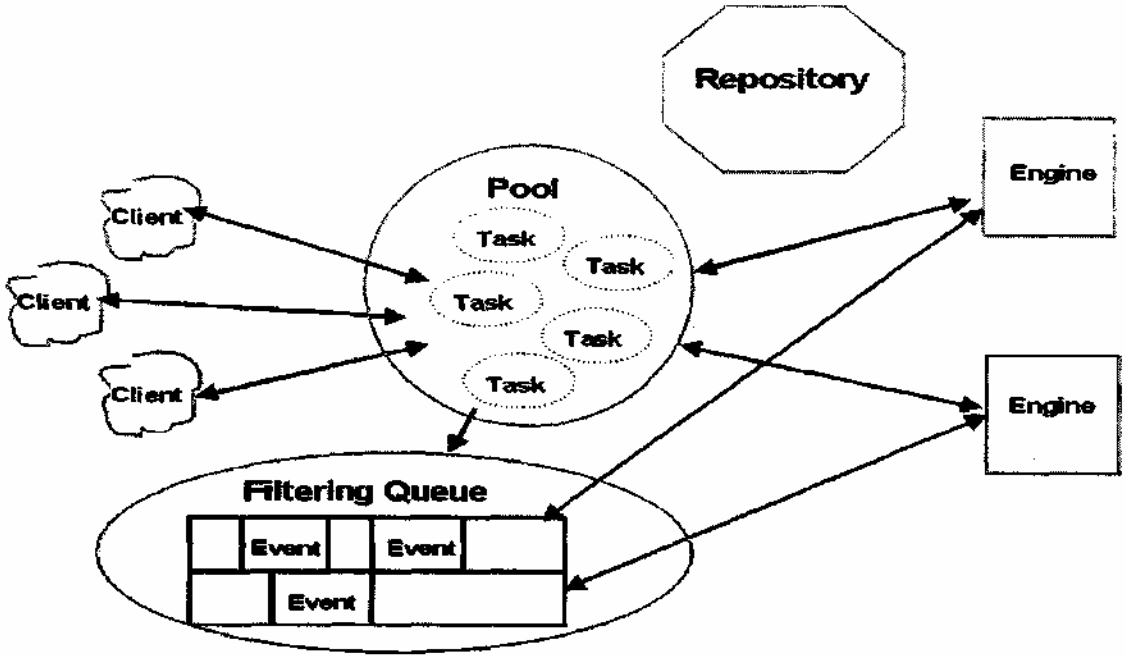


**Figure 4: DTS Architecture**

Queue. The Task status now changes to *Running*. A typical engine will not compete for new tasks while processing a request, but as soon as processing is done. This leads to automatic load balancing, which improves the overall system performance substantially. Once the Engine has completed the Translate Sentence task, the result is written to the Pool and the Task status changed to *Done*. The Engine now monitors the appropriate Filtering Queue for the next task. Results can be returned to Clients automatically or Clients can monitor the filtering queue which informs them about such events. Where speed is essential - as in the case of Sentence Translation via SMS for example - the first method would probably be used; in this case the results are immediately returned to the clients.

## 4. Components in the DTS Core

The DTS Core contains the components mandatory to any DTS Installation. Unlike the Engines and Clients, which are variable, the components in the DTS Core are essential to any DTS system. These components are the Pool, Filtering Queue and Repository.

As already described above, the *Pool* is a virtual storage and exchange space. This central component accepts tasks from clients, stores them in memory until requests are processed, and passes results back to the clients. Changes in task states result in events that are passed by the pool to a corresponding *Filtering Queue.*

The idea behind a Filtering Queue has also already been outlined above. It represents an elaborate event mechanism, which minimizes network traffic by filtering events for consumers. A consumer can be any component that registers itself with the Filtering Queue. In Comprendium DTS, a Filtering Queue stores events that are triggered by changes in the status of tasks. A queue server implements distinct queues for each state transition. Components that are interested in a particular event should register with the corresponding filtering queues. This mechanism means that the workload of the pool is further diminished and the overall performance is increased.

The DTS *Repository* is a network wide registry (Name Service). It is used to configure the components and Engines of DTS. All components and Engines that can be accessed through a CORBA interface have to be registered here. The central Repository enables persistent host and component configuration.

## 5. Engines and Clients

An Engine in DTS is a CORBA object and responsible for the execution of a specific task or tasks. The DTS framework does not define the engines themselves; it provides an API with functions that can be used to implement such engines. The DTS Core has no restrictions on the technology (language, platform, database, etc.) used to implement the particular Engines. They simply have to be able to communicate with the DTS Core using one of the methods provided.

The actual DTS clients will be customer-specific, i.e. implemented for a specific customer and designed depending on the requests of that customer. Of course, implementing clients on demand can only work if those clients can be developed without significant effort. Since more and more applications are becoming available over the Internet and in Intranets without being

locally installed, it is essential to offer DTS user interfaces that are browser enabled, i.e. that run within an Internet browser.

## 6. Security and Authentication

For security reasons, many companies refuse to open up ports in their firewalls other than those for the "standard" communication protocols, e.g. HTTP, FTP, etc. DTS can be installed behind a Web Server, enabling access for large numbers of users via HTTP. Such a configuration allows Internet solutions with standard security mechanisms like firewalls and SSL. The use of Java is also an enormous advantage with regard to secure communication.

DTS eases security worries by employing a 3-tier architecture. The DTS core configuration itself is meant to run behind a firewall. A typical installation will consist of a pool server and a number of engines, all residing in the same LAN. Instead of implementing the user clients

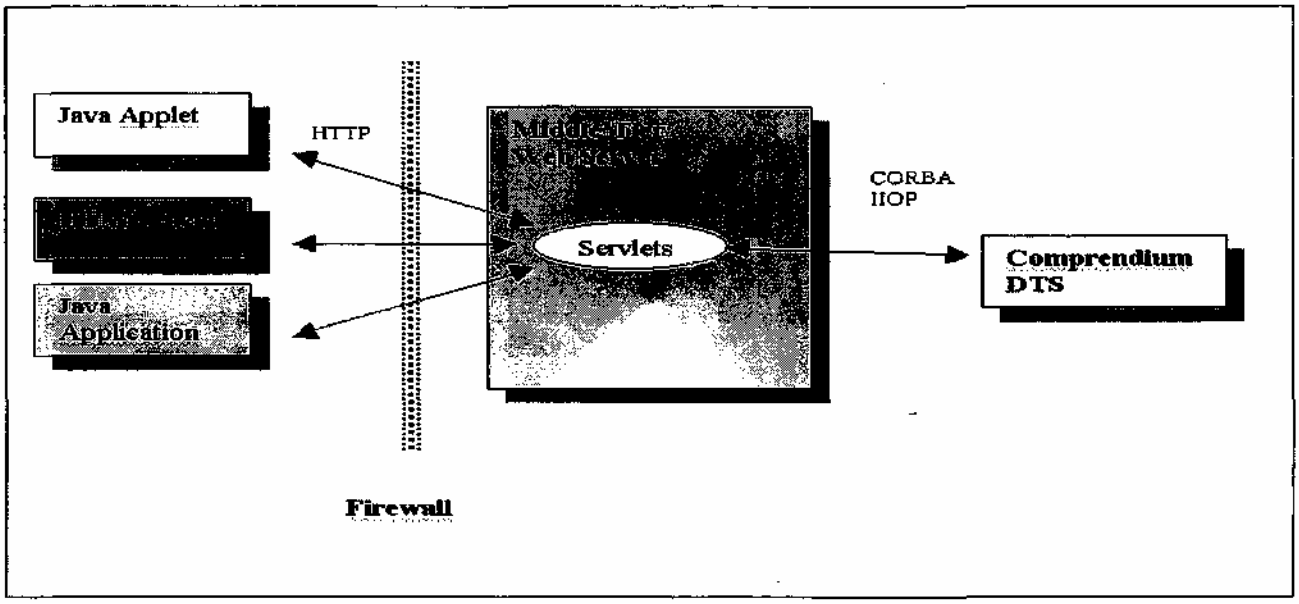


**Figure 5: 3-Tier Communication Architecture**

as core configuration clients (i.e. as Java applets directly accessing the pool), in this 3-tier architecture a Java applet or an HTML client connects to a servlet, and it is up to the servlet to communicate with the pool. The servlet can also perform input checking, error recovery, logging, etc. The use of servlets means that HTTP is employed as the communication protocol, and the security settings of the Interbet Browser and web servers are included automatically, guaranteeing secure communication.

## 7. Performance Aspects

DTS is an extremely performant system that can reach throughput speeds of hundreds of tasks per second on standard hardware configurations. As the performance of the DTS Core alone, although interesting for the developers, is of limited practical value, the graph in the figure below shows the throughput in combination with "dummy" engines. Different measurements were made using engines with processing times of between 10 ms and 100 ms per task. These figures clearly show that the DTS Core system is hardly likely to be the bottleneck in any real life application.

Let's look at performance aspects of a typical DTS-based application such as an MT Portal. Although the translation speed of the Sail Labs' MT engines is quite high, MT engine performance is the most obvious potential bottleneck in such an application. However, adding MT engines is simple and the achieved performance increase is almost linear, i.e. given identical document sizes and identical configurations, doubling the number of MT engines will double the performance of the system. The goal is to keep the overhead small and have the engines running continuously. As already described above, the concept of Filtering Queues and event pulling reduces communication overhead, as well as the absence of any central controlling instance. Engines try to keep themselves at work; no one allots work to engines. This leads to automatic load balancing.

The number of engines is probably the only variable that a DTS system administrator will need to worry about in terms of total system performance. However, should the need arise in a particular application, performance increases can also be achieved by increasing the number of Pools or Filtering Queues in the DTS Core.

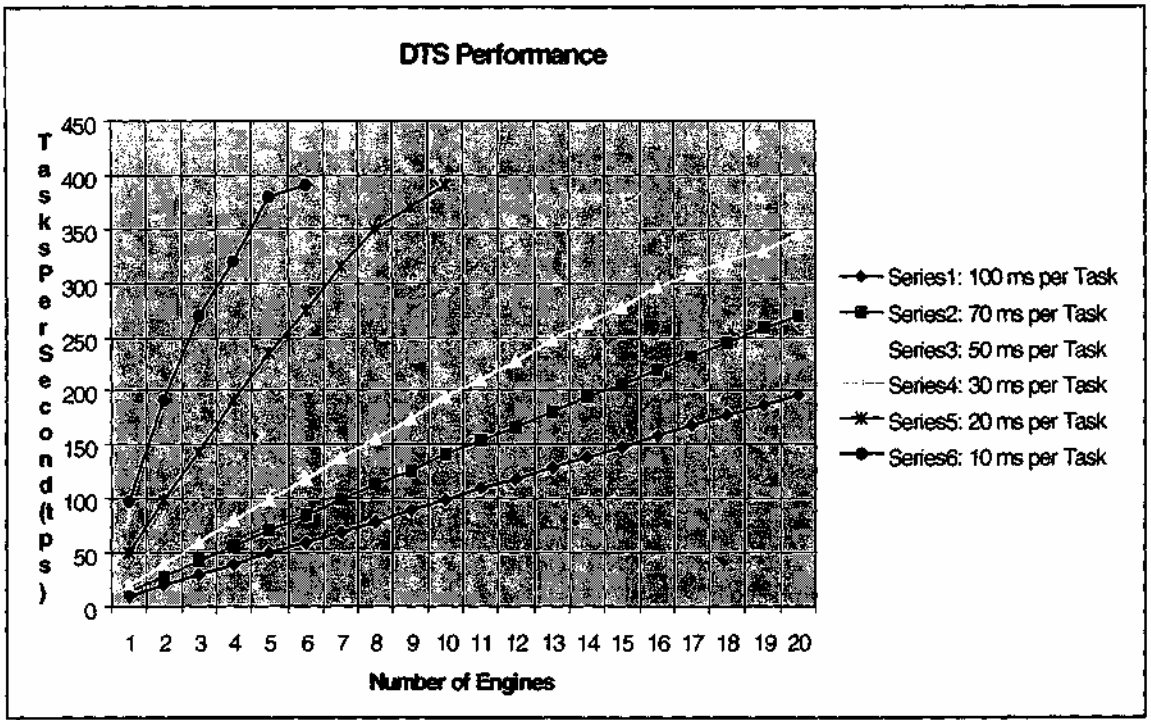


**Figure 6: DTS Performance with "Dummy Engines"**

## 8. Possible Services & Further Enhancements

The DTS system was designed and implemented with NLP applications in mind, and Sail Labs is currently implementing engines running its own multilingual communication and information applications; (Translate Sentence, Translate Document, Dictionary Lookup, Query Translation, Pattern Matching, Information Extraction Filter&Notify etc.). Figure 7 shows the MT engines that are already available from Sail Labs, and those under development or in the planning stage.

Sail Labs is currently working on APIs to allow other vendors to create services based on their own NLP technology; multi-vendor systems could easily be built on the DTS platform. What's more, there is no intrinsic restriction to translation or translation-related services. By the very nature of the system, DTS can be used to offer almost any type of service, and the APIs could even be employed to build engines for checking bank accounts or ordering pizzas!

Version 2 of DTS (due to be completed in the first quarter of 2002) will include a number of enhancements to the existing functionality. These will include:

- Remote configuration of components
- Remote control (start/stop) of components
- Performance and load monitoring
- Logging and report generation
- Visualization of configuration via an Administration GUI
- Error recovery and persistency.

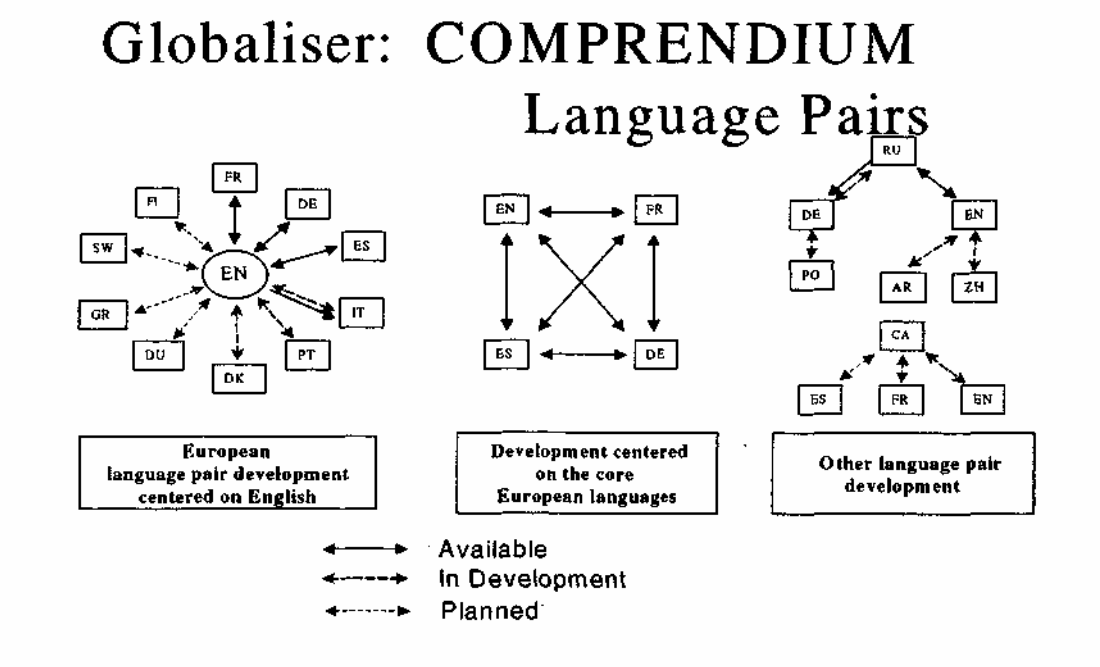


**Figure 7: MT Language Pairs from Sail Labs**

## 9. DTS under MALT

MALT (Modular Architecture for Linguistic Tools) is a framework (and platform) for the integration of diverse components and tools. These include Machine Translation, Translation Memory, lexicon editors, development tools, term extraction tools and many more. It is a freely configurable system with distributed architecture, and the MALT interface is divided into functionally separated areas.

Sail Labs envisages that MALT will be used as a framework for combining linguistic tools and applications for a wide variety of purposes and for controlling both simple and complex workflows. MALT can directly communicate with its own engines on a local machine or in a local area network. But, more importantly, MALT has been designed to employ a DTS system as an "engine", permitting access to further components and applications in a widely distributed environment.

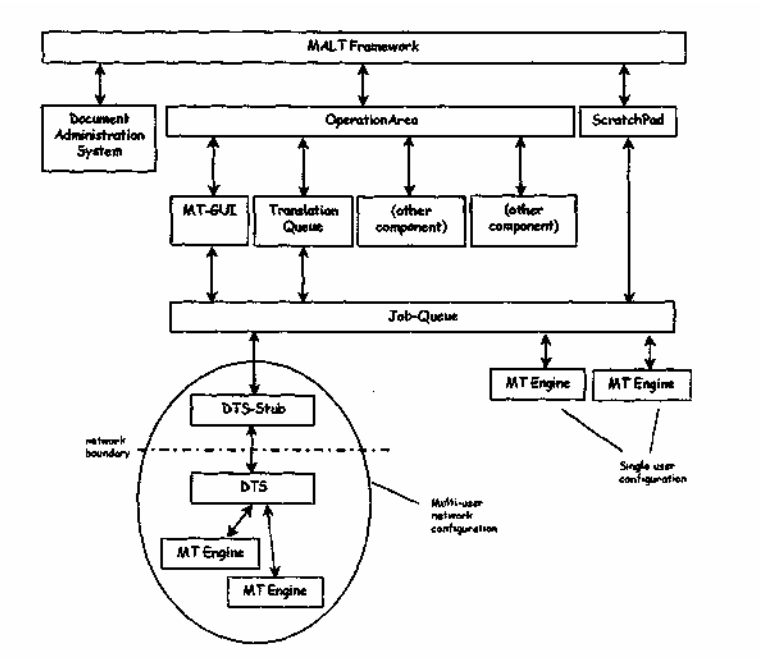


**Figure 8: DTS as an Engine under MALT**

## Bibliographical References

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