

TIPSTER Text Phase II Architecture Concept

**Prepared by :
Architecture Committee
for the
TIPSTER Text Phase II Program**

TABLE OF CONTENTS

1.0 EXECUTIVE SUMMARY	1
1.1 The TIPSTER Architecture.....	1
1.2 The Purpose of the Architecture.....	1
1.3 Scope of the Architecture.....	1
1.4 Components of the Architecture.....	1
1.5 Conformance to the Architecture	2
1.6 Background.....	3
1.7 Architecture Mission.....	3
1.8 Architecture Goals	3
1.9 Definitions of Terms	4
2.0 ARCHITECTURE CONCEPT.....	5
2.1 Architecture vs. Application	5
2.1.1 What the Architecture Provides.....	5
2.1.2 What the Application Provides.....	6
2.1.3 Differences between Architecture and Application.....	6
2.1.3.1 Functionality Covered.....	6
2.1.3.2 Computing Environment.....	7
2.1.3.3 Interfaces vs. Internals	7
2.1.4 Interaction between Architecture and Application	7
2.2 Available Help in Building a TIPSTER Application	8
3.0 BENEFITS OF THE ARCHITECTURE FOR INTERESTED PARTICIPANTS.....	11
3.1 End User	12
3.2 COTR/Program Manager.....	12
3.3 Technology Transfer Officer.....	13
3.4 Manager	13
3.5 Application Developer	13
3.6 R&D Researcher	13
3.7 System Support Officer.....	14
4.0 EXTERNAL INTERFACES TO THE ARCHITECTURE.....	15
4.1 Documents	15
4.1.1 Definitions.....	15
4.1.2 Types of Information in a Document.....	16
4.1.2.1 Examples of Information Conveyed Through Text.....	16
4.1.2.2 Examples of Information Conveyed through Other Conventions	16
4.1.2.3 Co-existence of Different Types of Information in a Document	16
4.1.3 Document Parts	18
4.1.3.1 Markup of Documents	18
4.1.3.2 Processing of Document Parts	19
4.1.3.3 Syntax of Markups.....	19
4.2 Interactions with the Application	19
4.2.1 End User Interactions	20
4.2.1.1 Detection Information Requests.....	20
4.2.1.2 Extraction Information Requests.....	20
4.2.1.3 Information Requests Common to Both Detection and Extraction	20
4.2.2 Application Developer Interactions.....	21
4.2.3 System Support Officer Interactions.....	21
5.0 CONFIGURATION MANAGEMENT	23
5.1 Architecture Compliance.....	23
5.2 Configuration Management in a TIPSTER Application Life Cycle.....	24
6.0 SECURITY POLICY	27

TABLE OF FIGURES

FIGURE 1-1	TIPSTER ARCHITECTURE SCOPE	2
FIGURE 2-1	A POSSIBLE TIPSTER-COMPLIANT APPLICATION	8
FIGURE 2-2	TWO WAYS TO "TIPSTERIZE" A MODULE	9
FIGURE 5-1	TIPSTER APPLICATION LIFE CYCLE WITH CM GATES	25

1.0 EXECUTIVE SUMMARY

1.1 The TIPSTER Architecture

The TIPSTER Architecture is a software architecture for providing Document Detection (i.e. Information Retrieval and Message Routing) and Information Extraction functions to text handling applications. The high level architecture is described in an Architecture Design Document. In May 1996, when the initial architecture design is complete, an Interface Control Document will be provided specifying the form and content of all inputs and outputs to the TIPSTER modules.

1.2 The Purpose of the Architecture

The TIPSTER Architecture is intended to facilitate the deployment into the workplace of advanced Document Detection and Information Extraction software. It provides a component and module design which has been jointly developed by a significant number of providers of advanced software of this type. In addition, this design meets the requirements of a number of US Government agencies.

The Architecture was developed to meet the need for US Government agencies with similar text handling requirements to share some software modules and knowledge sources meeting these requirements. Use of the Architecture for Government procurements will also shorten the development process for new text handling applications, because a basis for design already exists and is understood by vendor and customer alike. Finally, the Architecture will allow systems to be upgraded in a modular fashion as new text handling technology becomes available. Similarly, the research community can take advantage of the Architecture to facilitate the testing of new ideas in advanced text handling.

1.3 Scope of the Architecture

The Architecture has been designed to meet a large number of text handling requirements for [the Government](#). However, it meets only those requirements having to do with Document Detection and Information Extraction functions. Requirements for other functions, such as Machine Translation or Optical Character Recognition must be met outside the TIPSTER Architecture. In addition, User Interface (GUI) requirements are not covered by the Architecture. Analytical tools, such as link analysis tools, timelines, or other displays showing document clustering are considered part of the User Interface or the Application. These tools lie outside the Architecture, but use information about document relevancy, relationships between documents, phrase lists, name lists, and relational or object data base records which has been exported by the functionality residing within the TIPSTER Architecture.

1.4 Components of the Architecture

There are four components: Detection, Extraction, Annotation, and Document Management.

- Detection encompasses the technology which does text or message dissemination and text retrieval.
- Extraction encompasses the technology which identifies specific entities and the relationships between entities in free text.
- Annotation allows these two components to share information at a modular level. Primarily, at present, it is the method for recording and passing forward the information developed by modules of the Extraction component. Items of specific types, such as personal names, places, or organization names, for example, can be located in the text by appropriate modules, and the text locations and data types can be passed to any other component or part of the application for further processing or viewing.
- The Document Management component handles the files. This function can be performed by existing document managers or COTS products, such as a standard DBMS, with the addition of a wrapper.

Architecture Overview

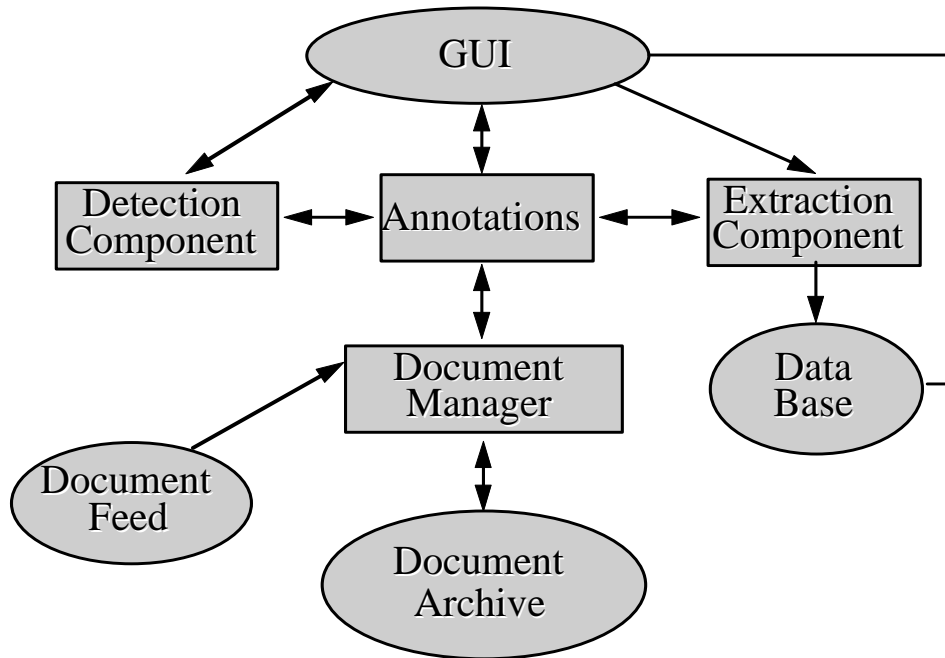


Figure 1-1 TIPSTER Architecture Scope

1.5 Conformance to the Architecture

The TIPSTER Architecture has been completed to the extent that the basic functionality of modules has been determined. However, modular interfaces have not yet been defined to the level of detail which will be required for the Government to meet its goals of software reuse and modular upgrading; i.e., they are underspecified. From May 1995 to May 1996, the Architecture will be tested by use in a number of applications and the lessons learned fed back into the Architecture for the purpose of refining those details which have been determined and specifying those interfaces which remain underspecified in May 1995. These changes will be managed by a Configuration Control process administered by the TIPSTER Program SE/CM contractor.

Under these circumstances, conformance to the TIPSTER Architecture cannot be rigidly defined. For the purposes of the year of Architecture testing (May 1995-May 1996), conformance will be defined as follows:

Designs of applications or products will be submitted to a TIPSTER Engineering Review Board (ERB). The ERB will produce a TIPSTER Architecture Conformance Assessment Document (TACAD) detailing the ways in which the design complies with the Architecture and those where it does not. Regarding those places within the design which do not comply, the ERB will issue a recommendation, that the Architecture be changed, that the design be changed, or that the exception be allowed. This recommendation will be reviewed by the TIPSTER Configuration Control Board. All designs will be kept on record, both in Design-to and As-built form, with the TIPSTER Program Office.

This process will result in an enrichment of the Architecture with the experience gained from specific implementations as well as the beginnings of a library of information about what TIPSTER compliant modules and components exist throughout the Government community.

1.6 Background

The TIPSTER Architecture describes a common framework for the development of text processing applications. By defining common components and interfaces, it will facilitate the development of both operational and research applications, as well as the rapid transfer of new text processing technology into the field.

The purpose of this Architecture Concept is to serve as an introduction to the TIPSTER Architecture for potential users. It encompasses ideas which will be realized during the TIPSTER Text Phase II Program, as well as ideas which will be developed over a longer period of time. This document addresses the following subjects:

- the Mission and Goals of the TIPSTER Architecture;
- the concept of the Architecture;
- benefits of the Architecture for various users;
- inputs and outputs to the Architecture;
- maintenance of the Architecture.

Appendix A of this document discusses various implementation-related issues.

Other documents which describe the Architecture and its associated concepts include:

- TIPSTER Text Phase II Architectural Requirements
- TIPSTER Text Phase II Architecture Design Document
- TIPSTER Text Phase II Interface Control Document (ICD)
- TIPSTER Text Phase II Configuration Management Plan.

If there is a conflict in the information in these five sources, the order of precedence should be:

- Interface Control Document
- Architecture Design Document
- Architecture Requirements Document
- Architecture Concept (this document)
- TIPSTER Configuration Management Plan.

1.7 Architecture Mission

The Architecture will provide a vehicle for delivering TIPSTER Text document detection and data extraction methods to today's analysts in [the Government](#). Two fundamental criteria must be met to do this:

- Costs must be controllable and minimized for development, deployment and maintenance; and
- Functionality must be sufficient to handle a variety of analytical tasks.

The Architecture has a secondary purpose which is desirable, but not of equal importance to the first, namely, to provide a convenient and efficient research framework for research in document detection and data extraction.

1.8 Architecture Goals

Provide Application Program Interfaces (APIs) so that document detection and data extraction can either jointly or singly import, process, and export data

Allow the building of applications to handle one or more human languages.

- Provide for modular substitution of functionally similar Architectural parts.

- Apply to a wide range of software and hardware environments
- Be scaleable to a large number of document archives and high document flow rate.
- Support appropriate application response time
- Support incorporation of multi-level security

1.9 Definitions of Terms

Some basic terms as they are used in the rest of this document are given below. Additional definitions may be found in the Glossary of the TIPSTER Architecture Design and in the Architecture Requirements Document.

An **Architecture** is a description of all functional activities to be performed to achieve the desired mission, the system {components} needed to perform the functions, and the designation of performance levels of those system {components}. An architecture also includes information on the technologies, interfaces, and location of functions and is considered an evolving description of an approach to achieving a desired mission.

An **Application** is a group of modules, both internal and external to the TIPSTER Architecture, that operates on documents to answer a User's request for information from documents. An application may contain a detection component, an extraction component, a clustering component, or any combination thereof.

A **TIPSTER Application** is an application which uses TIPSTER technology.

A **Component** is an aggregation of software that satisfies an end use function, such as detection, extraction, clustering, or user interface.

A **Detection Component** is a component that selects documents from a collection or routes documents to a User.

An **Extraction Component** is a component that extracts information from documents.

A **Clustering Component** is a component that groups similar documents together without user-specified detection criteria.

A **Module** is a logical element in the design of a component. Examples of modules are a parser and a stemmer.

A **TIPSTER module** is a module whose input and output specifications are defined in the TIPSTER Interface Control Document.

Persistent Knowledge is knowledge which is retained (i.e., stored) from one run to the next of an application. Examples of Persistent Knowledge are a lexicon, a set of grammar rules, and a gazetteer.

2.0 ARCHITECTURE CONCEPT

The TIPSTER Architecture is a flexible, cost-effective, technology-independent framework for building text-processing applications for Government analysts. It is designed to accommodate a wide range of text-processing requirements, in particular:

- document detection needs, information extraction needs, or both combined;
- target text in any human language;
- domain- or subject-matter independence;
- different computing environments;
- different security requirements.

The TIPSTER Architecture is documented in an Interface Control Document (ICD), which specifies the form and content of the inputs and outputs to TIPSTER modules. The TIPSTER Architecture Design Document describes the design underlying the Architecture.

The TIPSTER Architecture is modular. Modules are loosely coupled, communicating by means of shared data and control messages. The module interfaces are unambiguously defined in the ICD.

The TIPSTER Architecture is open. It is designed to be easily extended and enhanced as text-processing technology advances, and as modules with new functionality are developed in response to the needs of particular applications.

Modules that comply with the TIPSTER ICD specifications may be sharable by other applications. A list of TIPSTER-compliant modules will be maintained under Configuration Management. As a related service, the TIPSTER program will also maintain a catalog of other sharable resources, such as lexicons, gazetteers, data dictionaries, and query libraries, as well as tools to assist in building compliant applications.

Procedures for establishing an application's compliance with the Architecture, as well as procedures for extending the Architecture itself, have been established. They are set forth in the Configuration Management Plan. The TIPSTER Configuration Control Board has the responsibility for determining compliance with and extensions to the Architecture.

Because the main benefits from the Architecture derive from its commonality, its usefulness will be directly proportional to the number of applications which are built upon it. The more applications use and contribute sharable modules and data, the better and more extensive the Architecture will become, and the more cost savings will be realized.

2.1 Architecture vs. Application

Care should be taken to distinguish the TIPSTER Architecture from an application which is built in compliance with it. The following section outlines some of the differences between the Architecture itself and an Architecturally-compliant application. This is followed by a description of how the Architecture and an application interact with one another, and by a definition of an "Architecturally-compliant application".

2.1.1 What the Architecture Provides

The TIPSTER Architecture provides:

- functional descriptions for each of a set of modules
- for each included module, the form and content of inputs to it and outputs from it.

Architecture-related services include:

- maintenance of a catalog of previously-built TIPSTER modules and Persistent Knowledge bases.
- maintenance of the Architecture itself, through the Configuration Control Board.

2.1.2 What the Application Provides

A TIPSTER-compliant application is responsible for:

- selecting the programming language and operating system, and making other related hardware/software decisions
- determining what functionalities are needed to meet the User's needs
- for those functionalities covered by TIPSTER specifications, building (or obtaining from previous TIPSTER applications) the necessary modules, ensuring that the interfaces between them conform to the ICD
- for those functionalities not covered by the Architecture, building the necessary modules
- building (or obtaining from previous TIPSTER applications) all necessary Persistent Knowledge bases
- building any necessary interfaces to external databases and systems
- building a User Interface Component, ensuring that its output conforms to the input specifications for the TIPSTER module which receives it
- pre-processing of documents if necessary (i.e., document segmentation). (See Section 4.1 for more details on documents.)

2.1.3 Differences between Architecture and Application

As can be seen from the above, the differences between the Architecture and a TIPSTER-compliant application fall into three main areas: functionality covered, computing environment, and internals vs. interfaces. These are discussed below.

2.1.3.1 Functionality Covered

In the majority of cases, the set of modules in a given application will not be the same as the set of modules covered by the ICD.

On the one hand, it is likely that the Architecture will define specifications for more modules than are needed by any particular application. For example, an application may be needed for document detection only, even though the Architecture also provides specifications for extraction-related modules. The selection of which Architecturally-specified modules to use is the responsibility of the application.

On the other hand, an application will also need some functionalities not covered by the Architecture. For example, it may require the parsing of aircraft tail numbers, even though no ICD specifications exist for such a module. Obviously, if an aircraft-tail-number parser were to be linked to TIPSTER modules, its input and output must conform to the output- and input specifications of the TIPSTER modules surrounding it.

Every application will need at least one component not covered by the ICD: the User Interface Component. While the Architecture provides a means for handling a wide range of User needs, as well as a specification for the way in which the relevant data are input into the Architecture, it does not specify which of those User needs a particular application must satisfy, nor does it specify the manner in which the User Interface Component should operate. It is the application's responsibility to select the appropriate capabilities, to build the User Interface Component (including user commands, screen layout, and sequence of user operations), and to ensure that the output of this component meets the TIPSTER ICD specifications.

In order to utilize the features of the Architecture to the fullest, most applications will also need to provide an additional module which is not covered by the Architecture: the Document Setup Module. The Architecture provides a means for handling many types of information within documents (e.g., different types of formatted text vs. free text; text in different languages; and text at different security levels). It also provides a specification of the way these different types of information should be marked as input into the Architecture. In other words, the Architecture specifies an internal structure to which documents being

input into TIPSTER modules should conform, in order to ensure optimal processing. Since documents, as originally received, are not likely to conform to this structure, it will usually be necessary to convert documents as they exist in the external world into documents that conform to the internal document structure used by the Architecture. Ensuring that this happens ("document setup") is the responsibility of the application. (Section 4.1.3 provides more details on documents and document setup.)

2.1.3.2 Computing Environment

The TIPSTER Architecture is general and designed for use in a variety of software and hardware environments. The selection of the computing environment is the responsibility of the application.

2.1.3.3 Interfaces vs. Internals

The TIPSTER ICD defines interfaces (inputs and outputs) between modules. The internal operation of modules is left completely to the application. As long as the inputs and outputs of a module conform to ICD specifications, it is considered to be a TIPSTER module.

It should be noted that a TIPSTER module may, in fact, comprise several modules from the point of view of a particular application. These sub-modules may be modules in themselves, or complete systems, or databases. Individually, they may or may not be TIPSTER-compliant. The existence or nature of sub-modules is irrelevant for the Architecture, as long as the module itself accepts TIPSTER-compliant input and outputs TIPSTER-compliant output. From the point of view of the Architecture, only one TIPSTER module is involved, and only its interfaces must conform to the ICD.

Non-TIPSTER modules may be made TIPSTER-compliant by the use of wrappers which provide TIPSTER-compliant interfaces. In this case, the module plus the wrapper is equivalent to a single TIPSTER module.

2.1.4 Interaction between Architecture and Application

The relationship between the TIPSTER Architecture and TIPSTER applications is expected to be a close and mutually beneficial one. The Architecture will facilitate the development of text-processing applications, and new text-processing applications will contribute to the development of the Architecture.

As noted above, the TIPSTER ICD, which defines the Architecture, specifies inputs and outputs to components and modules. Initially, those ICD specifications will be based on the module interfaces in the first applications built under the TIPSTER Text Phase II program.

Developers of new TIPSTER-compliant text-processing applications will follow the existing ICD specifications for those modules whose functionalities are included in their application. Application development may be expedited by adapting previously-developed Architecturally-compliant modules.

It is likely that a new application will also require additional functionalities beyond those for which the ICD specifies an interface, especially in the early phases of the Architecture. It is expected that any newly-implemented functionality which is of potential use to other applications will be submitted to the TIPSTER Configuration Control Board (CCB) as an extension to the Architecture. If accepted, its specifications would be included in the ICD. (The TIPSTER Configuration Management Plan describes the procedures involved.)

Application developers may also find that individual ICD specifications need to be modified and extended to meet their needs. Such modifications and extensions may be submitted to the CCB as proposed changes to the ICD.

Because changes and improvements to the Architecture are expected, the Architecture will be under version control, administrated by the TIPSTER Configuration Control Board. TIPSTER applications will be associated with a particular version of the Architecture.

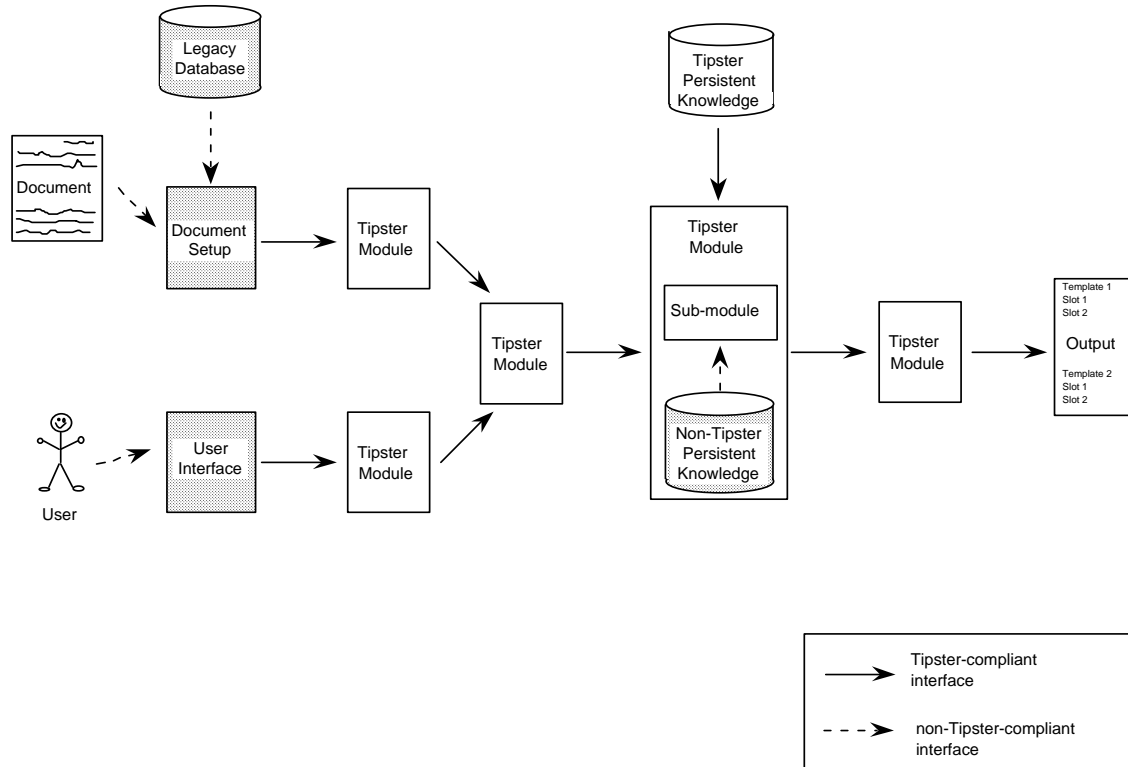


Figure 2-1 A Possible TIPSTER-Compliant Application

2.2 Available Help in Building a TIPSTER Application

Let us suppose that a developer has been given the task of building a TIPSTER-compliant text processing application that accomplishes certain tasks. The TIPSTER Architecture gives the developer several sources of help.

First, the TIPSTER Architecture Committee will maintain a list of previously-built TIPSTER applications, the contractors who built them, and detailed information about each module in those applications. The developer may be able to obtain one or more of those modules from the contractor(s), and to modify them for his purposes with minimal effort.

Even if no modules have yet been built which serve his purposes, the developer may be able to find help in the TIPSTER Interface Control Document. This document may contain specifications for some of the modules he needs to implement. By following those specifications, the developer is assured that his modules will be TIPSTER-compliant.

It may be the case, however, that the ICD does not include specifications for the modules the developer needs. In that case, he may look in the Architecture Design Document for guidance. Since this document gives the overall design of the TIPSTER Architecture, the developer may be able to determine where his modules would fit in the TIPSTER design. Once his modules are implemented, he may be able to submit them for consideration as extensions to the TIPSTER ICD. Details about the Configuration Management Policy are given in Section 5.0 of this document, as well as in the Configuration Management Plan.

Finally, even if the developer can find no reference to his proposed modules in the Architecture Design Document, he may look in the Architecture Requirements Document. This document describes the goals of the TIPSTER Architecture over the long term, and would give the developer an indication as to whether or not his modules would even be considered relevant to TIPSTER.

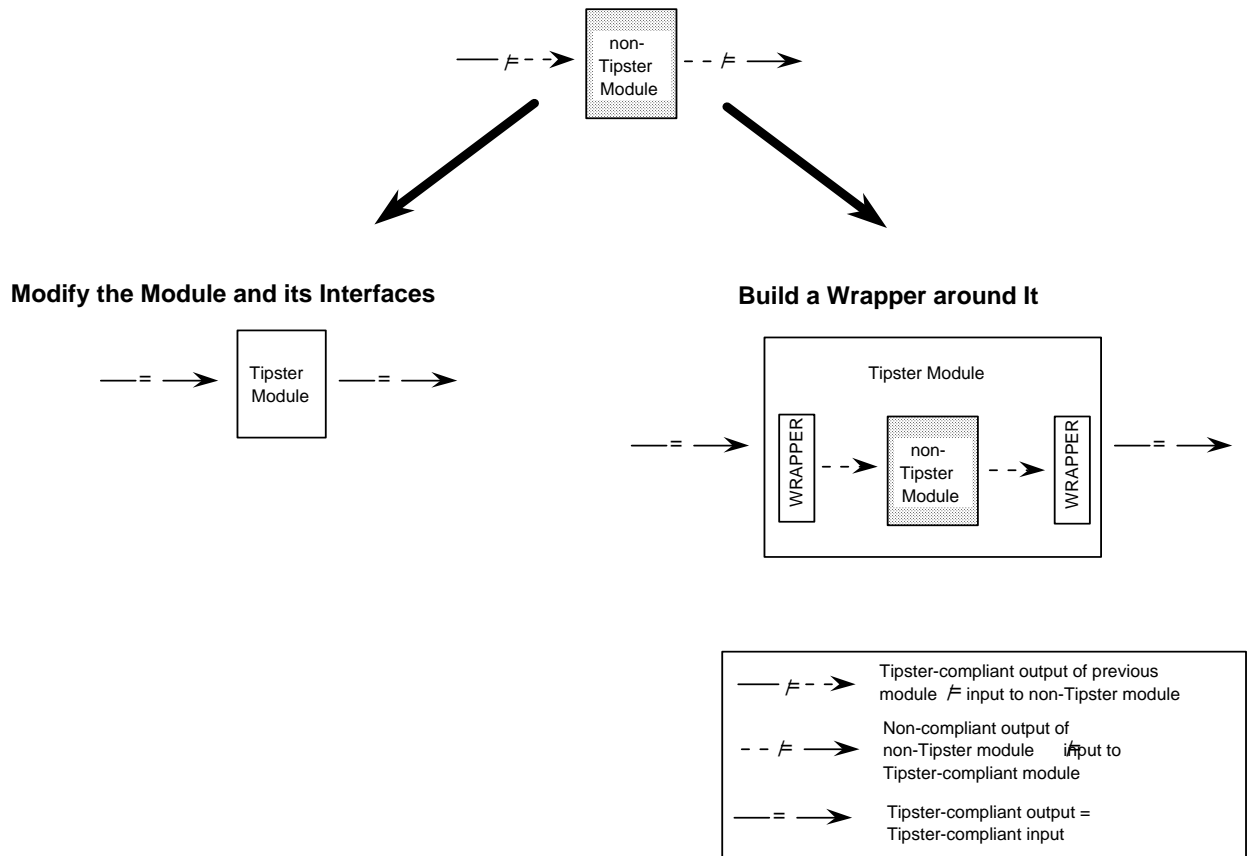


Figure 2-2 Two Ways to "TIPSTERize" a Module

3.0 BENEFITS OF THE ARCHITECTURE FOR INTERESTED PARTICIPANTS

Many people are involved in many different ways in working with a text processing application. Each has a different set of interests. Seven major groups of participants are identified below, according to their interests regarding a text processing application. (Depending on the agency or the contract, these roles may be combined.)

Participant	Interests
End User	<p>Wants to be able to specify application needs efficiently</p> <p>Needs operational, useful, efficient applications</p> <p>May need application which can be developed in a short time and modified quickly as requirements change</p>
COTR/Program Manager	<p>Must facilitate the identification of application contractors and potential teaming members</p> <p>Desires ability to specify application which meets End User needs at lowest cost in a timely manner</p> <p>Must evaluate application technical and cost alternatives</p> <p>Needs clear and unambiguous understanding of application design</p> <p>Must reduce the risk of fielding applications with new technology</p>
Technology Transfer Officer	<p>Would like flexible applications which can be easily integrated, reconfigured, and extended.</p>
Manager	<p>Concerned with the cost of a new application</p> <p>Needs to know immediate staffing impact of a new application</p> <p>Needs accurate estimates of life-cycle support required (cost and staffing)</p>
Application Developer	<p>Must quickly and easily specify and implement applications</p> <p>Desires availability of wide range of functional pieces of code with detailed interfaces</p> <p>Desires to create successful and meaningful text processing 'product'</p>
R&D Researcher	<p>Would like to quickly and easily examine new technical approaches for text processing</p> <p>Would like to facilitate identification of areas of potential weakness in text processing</p> <p>Would like to demonstrate and provide proof of concept for new and novel ideas</p> <p>Desires to demonstrate state-of-the-art advancement</p>
System Support Officer	<p>Would like applications which minimize data maintenance</p> <p>Would like reliable, robust applications</p>

The following sections discuss how the Architecture will benefit each of these users.

3.1 End User

The End User is expected to be someone in a United States Government agency who uses text processing applications.

The End User is the person whose needs the application is designed to meet. The specification of those needs is usually made by the End User and the Technology Transfer Officer working as a team. Typically, at the procurement stage and during the early stages of development, more technologically-oriented input from the Technology Transfer Officer is required; as the project develops and is deployed, the End User provides more input regarding needs. The Architecture Requirements Document provides a standard framework and terminology in which to specify and discuss needs efficiently. This document is a compilation of all the text processing needs which are envisioned to be required by the U.S. Government during the life of the TIPSTER project. It is not necessary that the End User be familiar with the Architecture Requirements Document in order to benefit from it: the availability of this document, whether or not the End User is aware of it, will reduce the chances of misunderstanding or overlooking critical requirements.

The End User is the person who most directly benefits from an operational, useful, efficient application. To ensure that an application meets these criteria, it must be tested. Because an Architecturally-compliant application will follow standard conventions for internal and external interfaces, it will be possible to build standard test suites. Easier and better testing will contribute to an application which better meets the End User's needs.

The End User generally has immediate needs which must be met as quickly as possible. When the needs change, the technology should be able to adapt quickly. One of the goals of the Architecture is to provide a catalog of previously-developed TIPSTER modules which may be adapted for use in other applications, thus saving time in developing that module.

3.2 COTR/Program Manager

The COTR/Program Manager's responsibility is to ensure that the most suitable developer(s) are selected for a project, and to oversee the progress of the project. (For simplicity, the term COTR will be used in the following discussion.) For some projects, the COTR and the Technology Transfer Officer are the same person.

The TIPSTER Architecture will help the COTR choose an Application Developer team, since a review of previous applications and modules will identify developers that provided/support a particular technology. This will also allow the COTR to recommend sources for specific technology and encourage teaming arrangements.

Like the End User, the COTR/Program Manager must be able to specify requirements, but at a more technical level. The availability of standards for describing text processing components and interfaces should greatly facilitate COTR/contractor discussions about requirements.

Once the requirements are established, the COTR must be able to evaluate the technical and cost alternatives for meeting them. Again, the availability of standards will provide a common ground for discussions. Also, the COTR will have access to information about any TIPSTER compliant components which have already been developed and which may be appropriate for use.

The COTR must monitor and oversee the application design. The modular nature of the Architecture supports review packages that have well defined characteristics.

The COTR must identify and focus attention on the more risky parts of his development effort. Because the list of TIPSTER applications and their modules will be available, the COTR will be able to identify what areas of his application development are new and under-tested, and thus probably more risky.

3.3 Technology Transfer Officer

The Technology Transfer Officer is responsible for the integration of an application into the End User's environment and for major upgrades to the application. For some projects, the Technology Transfer Officer and the COTR are the same person.

The Architecture will promote easy insertion of new technology into existing environments because it defines a common set of external interfaces. For the same reason, reconfiguration and extension of an Architecturally-compliant application will be easier than for a non-compliant application.

3.4 Manager

The Manager is primarily concerned with the cost and staffing issues (both immediate and long term) which are associated with an application.

Because the Architecture is modular and identifies equivalent components, a basis on which to compare these variables may exist. If a component has previously been used in another application, its associated costs, both to develop and to maintain, may be known. Its impact on staffing is also likely to be known. This provides a basis from which to estimate the cost and staffing required for a similar, new component.

The more widely the Architecture is used, the better will be the basis on which to estimate cost and staffing requirements.

3.5 Application Developer

For the Application Developer, the Architecture will provide most of the same support that is available for the COTR in identifying appropriate TIPSTER technology, identifying teaming partners, monitoring the project, making risk assessment, and measuring performance. In addition, the Architecture will assist the Application Developer in the actual implementation of the application.

The Architecture will assist the Developer in communicating with the End User/ Technology Transfer Officer team, because everyone will have the same reference points for conceiving and evaluating the capabilities of TIPSTER modules, and everyone will be using the same terminology for those modules.

The Architecture will help the Developer get an application to the End User more quickly and will provide a more flexible application design. It will provide a starting point for application development, because many (if not all) of the required components will be identified and their interfaces defined. Some previously-developed TIPSTER modules may be available which can be easily adapted for a new application. When new modules must be implemented, the standards for the interfaces will be pre-established, so that a minimum of investigation about related application modules will be required.

Developers who do not produce entire applications will still be able to produce Architecturally-compliant components or modules in their areas of expertise. Since their interfaces will be standard, these pieces can readily be used by other developers.

3.6 R&D Researcher

The TIPSTER Architecture will support the R&D Researcher with many of the same capabilities as are provided for the Application Developer.

The ability to test new ideas without having to develop all the components of an application is particularly attractive to the R&D Researcher. For example, if a Researcher has a new concept for extracting data, he may be able to use existing document management capabilities, Persistent Knowledge bases, matching functions and defined user interface to test the idea quickly and cost effectively. In this way, proof-of-concept and hypothesis testing can be performed with more comprehensive, realistic applications, and thus more effectively. The Researcher, operating in an established environment, is free to concentrate on specific, well-defined areas and not be burdened with developing the necessary infrastructure to test the new component. This will facilitate new research ideas, increase interest and expand the technology and the associated funding.

The Architecture may also help the Researcher to look for gaps in the technology since all modules in TIPSTER applications will be documented. The Researcher who is considering implementing a new idea will thus be able to determine whether or not a similar TIPSTER module already exists.

As for Application Developers, the Architecture will aid the Researcher in identifying collaborators to fill their technology gaps.

The Architecture will make it easier for the Researcher to evaluate component interactions. With interfaces clearly defined, it will be easier for the Researcher to insert measurement tools around new and supporting components.

3.7 System Support Officer

The System Support Officer is mainly concerned with the life-cycle support of the application. The Architecture helps System Support Officers perform their work more effectively in several ways.

Over the long term, it is envisaged that multiple applications will be able to use a set of common Persistent Knowledge Repository items. This would simplify the task of maintaining the data in the Repository. For example, adding words to a lexicon should occur less often than if every application had its own lexicon. Also, the maintenance procedures should be more consistent across different types of Persistent Knowledge Repository items.

The use of common shared modules in the Architecture will require the System Support Officer to become knowledgeable about fewer application parts. This will expedite problem identification and reporting, facilitate direct user support, and in general provide a more structured and consistent operational environment. Also, if the same components have been used in different applications, it is likely that they have been exercised more extensively than non-shared components. A more robust TIPSTER application, requiring less maintenance, should result.

4.0 EXTERNAL INTERFACES TO THE ARCHITECTURE

The External TIPSTER Interfaces are the links between the outside world and the TIPSTER environment. Three external interfaces are identified:

- the document
- the information request
- the output

Section 4.1 discusses documents in more detail. Discussion of information requests and outputs are combined in Section 4.2. The reason for this is that often, different types of output need to be manipulated, thus becoming input. From a user's point of view (the point of view taken here), the distinction blurs.

4.1 Documents

Documents are one of the links between the outside world and the TIPSTER environment. The purposes of this section are:

- to define the types of documents and document parts, which will be exploitable within the TIPSTER Architecture;
- to describe types of setup processing which might be performed on a document to facilitate and improve the output of TIPSTER;
- to illustrate some of the ways in which the TIPSTER Architecture might be extended for particular applications, in order to exploit a wider range of documents and document parts.

4.1.1 Definitions

Below are definitions of some terms as used in the discussion of TIPSTER documents.

For TIPSTER purposes, it is useful to distinguish between different Forms of a document, on the basis of the types of processing performed on it. Forms 0 - 4 are defined below. Note that, for any given document, Forms 1 - 4 may or may not be distinct from one another, depending on the amount of processing performed on the document.

Document - Information conveyed through some standard set of conventions and saved.

Form 0 Document - The document as originally composed by the writer(s). A Form 0 document is not necessarily in written form. Examples might be a news broadcast, a handwritten note, a photograph.

Form 1 Document - The document in machine readable form. A Form 1 Document is a Form 0 document plus:

- it has been saved, and
- it is in machine readable form.

Examples of a Form 1 document might be the portion of the news broadcast which was saved digitally or transcribed, the OCR'd note, the scanned photograph. In the case, e.g., of a document originally composed in machine readable form, Form 0 = Form 1.

Form 2 Document - The document as the End User's site receives it. A Form 2 document is a Form 1 document plus:

- anything done to it before the End User's site receives it.

A Form 2 Document might be a piece of a larger document; it might be encrypted or compressed. It may be a Form 1 document with additional information (e.g., header); it may be a fragment of a Form 1 document; or it may have the same amount of information.

Form 3 Document - The document as received by the TIPSTER Architecture. A Form 3 Document is a Form 2 document plus:

- anything done to it between receipt by the End User's site and input into the TIPSTER Architecture.

A Form 3 document may contain markups (e.g., the results of a non-TIPSTER document retrieval application). It may be reassembled, decrypted, decompressed.

Form 4 Document - The internal TIPSTER document. A Form 4 document is a Form 3 document plus:

- anything done to the document (e.g., identifying parts) to prepare it for the appropriate TIPSTER algorithms.

4.1.2 Types of Information in a Document

The basic TIPSTER Architecture distinguishes between two types of information:

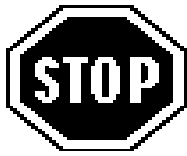
- information conveyed through the conventions of a particular natural language ("text"), and
- information conveyed through other conventions.

4.1.2.1 Examples of Information Conveyed Through Text

THE FORMER EMPLOYEE SAID THAT AUDON AND THE AYARZAS HAVE AN EFFICIENT SYSTEM FOR MOVING THEIR MARIJUANA SHIPMENTS.

Quot homines, tot sententiae.

4.1.2.2 Examples of Information Conveyed through Other Conventions



(def-load-pointers disable-dead-keys ())
(set-dead-keys nil)

This category would also include video, sound, graphics; other non-natural "languages" (e.g., C, mathematical notation); format based conventions (e.g., spreadsheets, tables).

4.1.2.3 Co-existence of Different Types of Information in a Document

It is important to note that information is often conveyed through a combination of text and non-text conventions.

Examples are:

- tables containing words:

OWNER	PET
John Smith	rabbit
Mary Jones	cat
Pete Michaels	armadillo
Ramon Suarez	llama

- communication headers:

```

ENVELOPE
CDSN = LGX898 MCN = 93027/26204 TOR = 930271638
RTTEZYUW RUEKJCS0923 0271637-EEEE--RUEALGX.
ZNY EEEEE
HEADER
R 271637Z JAN 93
FM DIA WASHINGTON DC
INFO RUEAHQA/CSAF WASHINGTON DC
RUEAIAQ/MPC FT GEORGE G MEADE MD
RUEHC /SECSTATE WASHINGTON DC
RUEAIIA/CIA WASHINGTON DC
RULKQAN/MARCORINTCEN QUANTICO VA
RUDMONI/ONI SUITLAND MD//NAVATAC//
RUEALGX/SAFE
R 271141Z JAN 93
FM COGARD INTELCOORDCEN WASHINGTON DC
TO RUEKJCS/DIA WASHINGTON DC//DSP-2B//
INFO RUDMGRD/COGARD INTELCOORDCEN DET SUITLAND MD
RUCBSAA/USCINCLANT NORFOLK VA//J24//
ACCT CG-W2GARC
R 252345Z JAN 93 ZUI ASN-CAA026000296
FM COMLANTAREA COGARD NEW YORK NY//A//
TO COGARD INTELCOORDCEN WASHINGTON DC
ACCT CG-W2GDRC
BT
CONTROLS

UNCLAS E F T O FOUO //N03821//
SECTION 01 OF 02
SERIAL: IIR 4 111 2204 93.

/***** THIS IS A COMBINED MESSAGE *****/
BODY
COUNTRY: JAMAICA (JM).
SUBJ: IIR 4 111 2204 93/PORT ANTONIO, JAMAICA
WARNING: THIS IS AN INFORMATION REPORT, NOT FINALLY
EVALUATED INTELLIGENCE. THIS REPORT IS UNCLASSIFIED FOUO.

-----
DEPARTMENT OF TRANSPORTATION
-----

DOI: 930116.
REQS: D-MAH-20401-690; T-3CX-19000-190A; R-CGE-43837;
D-DC4-44151; S-ULA-00266.
SOURCE: USCGC MOHAWK PERSONNEL.

```

- text in more than one language:

Terence wrote, "Quot homines, tot sententiae."

- formatting conventions embedded in text:

E. GELASIO LABORI (TECS# B0646308700J01, P0683667700J01) IS A DEA SUSPECT. HE WAS ASSOCIATED WITH THE VESSEL "YOGI" DOC# 577019 WHICH WAS SUSPECTED OF SMUGGLING DRUGS IN 1990.

- graphics with titles:



Map of North America

- outlines:

8. ADDITIONAL INFORMATION FOUND ONBOARD.

A. A PAMPHLET "PARTIDO UNIDAD NACIONAL DEMOCRATIC (P.U.N.D.) - MANIFIESTO PROGRAMATICO".

B. BUSINESS CARD FOR - JIM KLINE INC.

MOBILE MARINE REPAIR

DIESEL,GASOLINE & ELECTRICAL REPAIRS

PO BOX 741

MARATHON, FL 33050

305-743-9041

664-7330 (HAND WRITTEN)

4.1.3 Document Parts

Viewed in this context, a particular natural language is one of many possible sets of conventions used to convey information within a document. Each set of conventions requires a different type of processing in order to exploit information represented in it. For example, the portion of a document that is contained in a table will require different processing from the portion of a document that is written in English, which will, in turn, require different processing from the portion of a document that is written in Arabic.

A Document Part will be defined as a portion of a document which requires a particular type of processing, in order to exploit its information.

In case no Parts are identified, the basic TIPSTER Architecture will make the default assumption that all Parts of a given document are text Parts of one particular, user specified language. Note that under this assumption, information conveyed through a combination of text and non-text may be partially exploited. For example, some information may be gleaned from the words in a table, without actually processing the table structure.

4.1.3.1 Markup of Documents

A document markup will be defined as information which has been added to a document before it becomes a Form 4 document. Markups may be added manually or by automatic means, such as a message handling

application or a detection application. The most common markup is probably to indicate Parts of a document. Markups may be embedded within the document or they may co-exist with it in the form of annotations, possibly containing pointers to specific locations in the document.

To exploit Document Parts, markups must indicate, at a minimum, boundaries between the Parts and identify the Part type. Then the appropriate type of processing can be applied to each Part. An application must be able to ignore any markups it does not use; that is, unused markups should not cause it to break.

The TIPSTER program has undertaken to identify several types of Document Parts. These include some message headers and two languages, English and Japanese. The identification of other Part types will be added on a case-by-case basis by Architecture users who require the exploitation of those parts.

4.1.3.2 Processing of Document Parts

As noted above, if no Parts are identified, the basic TIPSTER Architecture will make the default assumption that all Parts of a given document are text Parts of one particular, user specified language. Obviously, if that document is to yield information, that default language must be one of the languages for which TIPSTER modules and components exist.

In addition, the TIPSTER Architecture will process Part types that are traditionally viewed as part of Natural Language Understanding (NLU), such as communication headers.

As the TIPSTER Architecture grows, processing capabilities for additional generic Part types can be added. Although the Architecture will theoretically be extendible to accommodate the processing of any Part type (e.g., LISP, spreadsheets), it is expected that the emphasis for the TIPSTER program will be primarily on processing textual information. In particular, Parts in any natural language will be processable within the Architecture, provided that the language can be identified and indicated through markups, and that Architecturally-compliant modules and components for the language exist.

The TIPSTER Architecture is not envisioned to encompass the processing of non-generic Parts, such as specific types of tables or outlines. Such processing could be included in an application, but outside the Architecture compliant portion.

4.1.3.3 Syntax of Markups

Markups present in Form 3 documents require a key before the information they represent can be used. For example, if the markups were in the Standard Generalized Markup Language (SGML), then the syntax and definitions (e.g., embedded tags <NAME> </NAME> mark a name) would need to be available in the form of "Document Type Definitions" (DTDs).

The Architecture will provide a standard for document markups. During document setup, pre-existing markups must be converted to the standard, in order to be used by the TIPSTER parts of the application. Markups which are not needed by the application need not be converted.

It may be the case that the Architecture does not include a standard for some pre-existing markup which would be useful for some other part of the application. For example, formatting markup for bolding, which may not be specified in the Architecture, might be used by the (application-specific) User Interface Component. Because the "original" document is always retained, all pre-existing markups are available to the application, whether or not they are converted to the Architecture standard.

4.2 Interactions with the Application

This section discusses some ways in which users will need to be able to interact with a text handling application. An application need not provide all of these capabilities to be Architecturally compliant. However, if it does provide any of these capabilities, in order to be compliant, it must follow the TIPSTER standards for user/application interfaces. Of course, the Architecture also does not prohibit an application from providing additional capabilities which are outside the scope of the Architecture.

Different users of a TIPSTER application will typically have different types of interactions with the application, to accomplish different tasks. This discussion is organized around typical interactions of the

End User, the Application Developer, and the System Support Officer. It is not meant to imply that only those individuals should be able to make those inputs.

4.2.1 End User Interactions

The End User will need to have the ability to interact with the application in many ways. Most obviously, he will submit "information requests": if the application is a detection application, this will most likely be a retrieval request or a routing request, and if the application is an extraction application, it will probably be a request to fill templates with information from documents. The End User must be able to control many other aspects of the application as well. The Architecture will provide standards for handling the following types of End User interactions.

4.2.1.1 Detection Information Requests

- a. creating detection criteria in the form of a statement of relevance (text), an example (text), a Boolean expression, keywords (including negative operators), or a combination of these elements
- b. manipulating and refining detection criteria (e.g., reformulating them, annotating them with priorities which are then transferred to the returned documents, weighting parts of the detection criteria)
- c. using external databases (e.g., word lists) to help formulate a query. If such a list is used frequently by several applications it may be added to a Persistent Knowledge base.
- d. storing, searching, and retrieving detection criteria
- e. comparing results of using different versions of detection criteria
- f. specifying the following aspects for the returned document list: criteria for prioritizing (e.g., relevance, author, date of document, date of receipt, subject), destination (e.g., file, extraction application), cutoff point for ranked document list, additional document related information to be returned with the document list (e.g., classification, bibliographic information, abstract), order in which the information returned with the document list is to be presented
- g. saving and manipulating the document list (e.g., re-ordering it manually)
- h. searching, simultaneously, an archival index and an index for newly acquired documents and merging their results
- i. searching foreign language documents using an English statement of relevance and receiving a single ranked list of documents
- j. annotating foreign language documents with English glosses and viewing the annotated document
- k. associating specific portions of the Detection Criteria to specific zones of a document

4.2.1.2 Extraction Information Requests

- a. accessing a library of templates (for pattern based extraction applications) to reuse or to use as a basis for a new template
- b. writing, or selecting from a library, fill-rules and patterns to accompany a template
- c. combining fill-rules, templates and documents in various languages
- d. editing filled templates

4.2.1.3 Information Requests Common to Both Detection and Extraction

- a. viewing the original document
- b. creating ordered and unordered groups of documents

- c. specifying multiple sources from which to get information (e.g., multiple collections, message streams, databases)
- d. narrowing a search by using attributes of the document or part of document (e.g., date, source, language)
- e. clustering documents by similarity; identifying duplicate documents
- f. viewing document with reason for selection/slot fill (e.g., for detection, highlighting the text that caused the document to be selected for detection, or presenting a bar graph with the weighting of terms; for extraction, highlighting the source text for any given slot fill)
- g. viewing data about documents (e.g., size, classification, author, dates) and collections (e.g., concordances, KWIC indexes)
- h. finding documents containing identical passages and annotating them
- i. requesting and modifying items from the Persistent Knowledge bases (e.g., collections of detection criteria, template and template component collections, indexes, lexicons)
- j. specifying whether processing occurs in the background or foreground
- k. specifying the time at which to begin processing a collection
- l. adding user specific annotations to be used for future processing
- m. viewing the annotations to a document
- n. obtaining an abstract or summary of the document

4.2.2 Application Developer Interactions

- a. requesting Persistent Knowledge (e.g., query and profile collections, template and template component collections, document collections, indexes, lexicons, document lists, data dictionaries which list the annotations and attributes related to each module/component)
- b. manipulating Persistent Knowledge Repository (e.g., adding lexicon, modifying templates)
- c. testing individual modules and components using standard evaluation tools
- d. modifying existing modules and components
- e. creating templates for extraction from scratch
- f. modifying existing templates by deleting slots, adding slots (for pattern based extraction components). These templates would be objects, and each of the slots in the templates would be linked to the parts of the application that are used to fill it. This implies the ability to view and modify each of the parts of the application that are linked to those slots.
- g. grouping annotations into sets

4.2.3 System Support Officer Interactions

- a. ensuring that each part of the application is marked and handled at the proper level of security classification. This would include: documents and parts of documents, application modules, Persistent Knowledge Repository items (e.g., lexicon, saved queries and profiles)
- b. ensuring that access control is properly handled. This would include being able to access and change User and User Group permissions to read/write/execute on a module level
- c. requesting auditing and usage information (e.g., number of documents and sources accessed; how often certain types of data are searched; source and classification of each document)
- d. combining detection criteria from multiple users for a single pass against a collection
- e. applying detection criteria against multiple document lists simultaneously

- f. maintaining an archival index and a corresponding index for new documents
- g. viewing diagnostic information about problem documents, document processing and errors
- h. viewing the results of multiple diagnostic runs

5.0 CONFIGURATION MANAGEMENT

Responsibility for maintenance of the Architecture resides with the Configuration Control Board. The TIPSTER Configuration Management Policy is documented in the Configuration Management Plan. It describes procedures for:

- obtaining the current version of the ICD
- submitting ICD specifications for a new TIPSTER module
- proposing modifications to existing ICD specifications
- obtaining certification for a TIPSTER-compliant application
- obtaining information about previously-developed TIPSTER-compliant modules.

The Executive Summary of the Configuration Management Plan is reprinted below for reference.

5.1 Architecture Compliance

The CM review process will result in a document which details the ways in which an application or vendor product conforms to the Architecture Design Document and is in agreement with the TIPSTER Architecture design. This document is a TIPSTER Application Conformance Assessment Document (TACAD).

In order for an application or vendor product to successfully acquire a TACAD, the following conditions must be met:

For TIPSTER Application development:

The TIPSTER Application development complies with the TIPSTER CM process, the details which are contained in this document. In short, the TIPSTER Application must undergo a Preliminary Design Review (PDR) and a Final Operating Capability (FOC) review. At these reviews, any discrepancy or deviation from the TIPSTER Architecture must be documented and justified/explained.

Any new code or capabilities for the TIPSTER Application must be developed in accordance with the TIPSTER Architecture. Failure to do so will be documented and justified in the TACAD.

To the extent possible and in the Government's best interest, existing code and capability to be incorporated into the TIPSTER Application will be re-engineered in accordance with the TIPSTER Architecture. Failure to do so will be documented and justified in the TACAD.

For Vendor Products:

- If the vendor's product is used in a TIPSTER Application, the criteria stated above in "For TIPSTER Application development" will apply.
- A vendor's product may be determined to be TIPSTER compliant with the use of a TACAD independent of actually being part of a TIPSTER Application. To support the development of this TACAD, the vendor will demonstrate, by inspection, module-by-module compliance with the TIPSTER Architecture.

On the basis of the TACAD, the Configurations Control Board (CCB) will determine that a TIPSTER Application is conformant or non-conformant, if it exhibits sufficient overlap with the Architecture Design Document.

The extent of TIPSTER Conformance will be determined on a "per module" basis and documented in the TACAD. As a result of the TIPSTER Application reviews (described in section 1.2, below) a summary matrix will be available as shown in Appendix A, Figure A-1, below.

The TACAD may be used by Vendors to facilitate teaming with other Vendors or insertions of new capability into existing TIPSTER systems.

5.2 Configuration Management in a TIPSTER Application Life Cycle

The TIPSTER CM process imposes two control gates, PDR and FOC, on the TIPSTER Application development lifecycle, as shown in Figure 5-1. In preparation for these control gates, it is expected that the developing contractor and the SE/CM will work together to prepare the documentation and to identify any discrepancies between the Architecture Design as detailed in the Interface Control Document (ICD) and the TIPSTER Application's design. This cooperation will be in the form of an Engineering Review Board (ERB).

At the PDR control gate, the following TIPSTER Application documents are expected to be put under TIPSTER CM control:

1. TIPSTER Application Design Document
2. A Document detailing any design discrepancy or deviations from the TIPSTER Architecture on a per-module basis
3. A Document justifying the discrepancy or deviations in #2, above
4. Any Requests For Change (RFC) to the TIPSTER Architecture to cover the discrepancy or deviations in #2, above.

At the FOC control gate, the following TIPSTER Application documents are expected to be put under TIPSTER CM control:

1. TIPSTER Application As-Built Document
2. A Document detailing any As-Built discrepancy or deviations from the TIPSTER Architecture. This document will also explain the discrepancy or deviations
3. Any Requests For Change (RFC) to the TIPSTER Architecture to cover the discrepancy or deviations.

It is expected that the TIPSTER Application's project will be requiring control gates at the time of design and at the time of final delivery. The TIPSTER PDR and FOC control gates will 'piggy-back' on the projects control gates, to the maximum extent possible.

The Architecture Committee will determine whether a TIPSTER Application has successfully passed a PDR and FOC control gate. In practice, the Architecture Committee will appoint a small group of people to sit on the Configuration Control Board (CCB) to examine, in detail, the documentation and justifications provided at the PDR and FOC control gates. The CCB will then make a recommendation to the Architecture Committee as to whether or not the control gate has been satisfied. The specifics of how to run the CCB and the control gates is in the CM Plan.

The CCB will assign and use an Engineering Review Board (ERB) to compile the documentation necessary for the CCB's review. The ERB will be comprised of the SE/CM and the developing contractor's representatives. The specifics of how to run the ERB is in the CM Plan.

Discrepancies will originate from the ERB and will primarily be presented by the developing contractor's representatives. The discrepancies will be submitted to the CCB for disposition, which can be one of the following:

- An RFC can be initiated to incorporate the discrepancy into the Architecture Design Document and the into the Interface Control Document (ICD).
- The discrepancy can be "Noted and Documented". This would assume that it is in the best interest of the Government to deviate from the Architecture in this particular case.
- The discrepancy can be sent back to the developing contractor and the Government Contracting Officer's Technical Representative (COTR) with a request that the discrepancy be cleared up.

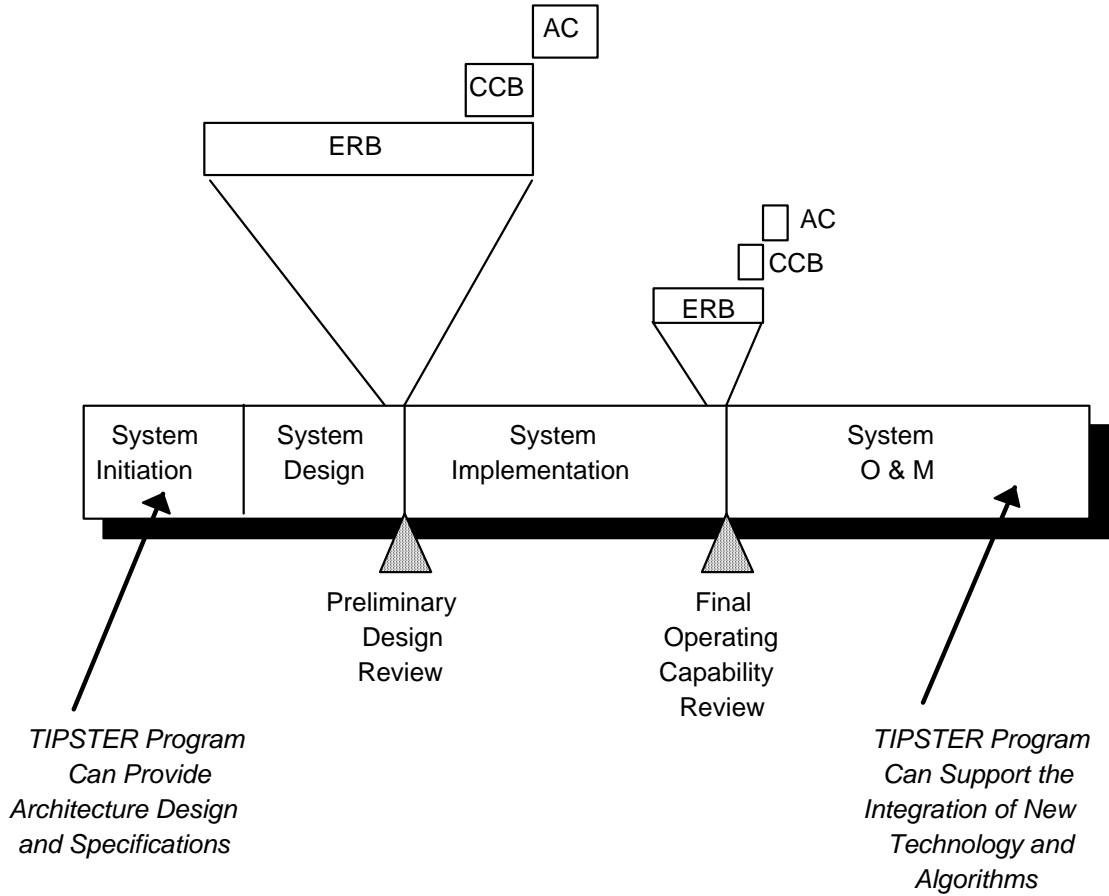


Figure 5-1 TIPSTER Application Life Cycle with CM Gates

6.0 SECURITY POLICY

The Security Policy which guides the development of the Architecture is documented in the TIPSTER Text Phase II Architecture Design Document and reprinted below for reference.

- Architectural choices should recognize the desirability of incorporating multi-level security in component implementation. The Architecture shall support the Application security requirements of the organization responsible for the Application.
- The Architecture shall support physical and software boundaries to devices where documents and data are stored. The boundaries will ensure that persons only have access to the appropriate level of classified information. These may be implemented at the API level of the software components.
- Auditing and administrative support shall be available for marking and filtering data to ensure proper document data access, distribution and viewing and also to record improper access attempts. The scope of marking of data may be at the document, paragraph, data item or object level.