TIPSTER LESSONS LEARNED: THE SE/CM PERSPECTIVE

Harold Corbin and Aaron Temin
Litton PRC
1500 PRC Drive
McLean, VA 22102
{corbin_hal, temin_aaron}@prc.com

1 INTRODUCTION

The TIPSTER architecture is a domain-specific software architecture (DSSA) for the text processing domain. The primary goal of the architecture was to allow the use of standardize text processing components, enabling "plug and play" capabilities of the various tools being developed. This would permit the sharing of software among the various research efforts and operational prototype applications.

PRC, Inc., was the systems engineering and configuration management (SE/CM) contractor for the TIPSTER program, phases II and III (1994-1998). During this time we had close association with all the TIPSTER participants: Government, research contractors, and project contractors. Our primary role was to support the TIPSTER Architecture Committee and its co-chairpersons. We were able to observe the entire process of creating the TIPSTER architecture and applying it to research, projects, and the Architecture Capabilities Platform (ACP).

The architecture was successful in many ways, though it also fell short of expectations in most areas. This paper describes *our* perspective on the architecture, to help subsequent efforts understand how to benefit from, and efficiently extend, the architecture.

2 BACKGROUND

It will be helpful, in understanding the lessons learned, to bear in mind some background of the TIPSTER program, the intended uses of the architecture, and the process used to develop the architecture.

2.1 Relevant Programmatics

TIPSTER was a DARPA program, with funding provided primarily by DARPA, CIA, and NSA. Coordination among these agencies was formally accomplished by a Memorandum of Understanding (MOU). The MOU specified overall funding contributions, and set the administrative guidelines for the three sponsoring agencies. The TIPSTER Research and Evaluation Committee (REC) was charged with oversight responsibility of the 15 research efforts and the Architecture Committee had oversight of the architecture development and the Architecture Capability Platform effort. Each effort was managed independently by a government Contracting Officer's Technical Representative (COTR) and the goal was to have the COTRs report to the appropriate committee.

Phase III of the TIPSTER Text Program initially was designed for three years. The third year, however, was eliminated due to funding shortfalls experienced by the government sponsors.

2.2 Intended Uses of the Architecture

The architecture was intended to have two direct applications:

1. Provide researchers with guidelines on building their components so that they could easily be used by other researchers within the program. This would also mean that duplication of effort could also be reduced when researchers shared common pre-processing needs, or when the output of one research project could serve as the input to another.
2. Provide developers with guidelines on building entire applications, including interface specifications that would allow the projects to incorporate advances made by the researchers as those results became available.

There were also other intended effects, such as providing guidance to those writing requirements for acquiring text processing systems, and helping shape and publicize the TIPSTER program.

2.3 The Development and Maintenance of the Architecture

The basic development of the architecture was done over nearly two years (April 1994 – January 1996) by the Contractors Architecture Working Group (CAWG) with three-day technical meetings about once a month. The CAWG was composed of one or two people from each of the Phase II contractors. These meetings were essentially discussion and negotiation sessions among the participants. E-mail discussions and additional work was conducted when the members returned home. Later, after a baseline architecture was established, procedures were established to modify and improve the architecture through a Request For Change (RFC) process. Near the end of Phase II several RFCs were submitted and carried over into Phase III, and several more RFCs were submitted during Phase III. RFCs were voted on by the Architecture Committee, and revisions to the design document were made as RFCs were approved.

TIPSTER included a mechanism called the Engineering Review Board, chaired by the SE/CM, which held reviews on TIPSTER projects and reported on differences between the architecture used by the project developers and the documented TIPSTER architecture. This was the major source of RFCs (though RFCs could be submitted at anytime).

3 LESSON LEARNED

This section describes several lessons we believe have been learned as a result of having pursued the development of the architecture.

3.1 Architectures Are "Good Things to Have"

Whatever the specific achievements or failings of the current TIPSTER architecture, the consensus of the program is that having an architecture was a good idea. It provided a central focus to an otherwise somewhat loosely-coupled set of contracts and it provided a forum for the researchers to share concerns and make progress on areas of mutual interest. It made it much easier to describe the program and its goals, both to participants and to outside interested parties.

There was always a lot of great hope expressed in the architecture and a stated desire by all participants that its goals were important to achieve. Though it may be judged that the final TIPSTER architecture did not achieve all these goals, we should be encouraged by the achievements and use the lessons learned for an improved architecture in the future. We should consider architectures as a good idea for text processing systems.

3.2 Programmatic Incentives May Be Necessary To Get The Architecture Used

Usually, an architecture is a model of standards and interfaces used to development common, reusable components or modules in support of various domain operational applications. Applying this concept to the research environment was new and unusual. Normally, researchers are concerned with new algorithms and concepts and are not involved in the bigger application or system picture. They must be properly indoctrinated in the needs for an architecture, provided adequate support tools and *directed* to use the architecture. It can not be an optional consideration except where their work has no bigger contribution to an application.

3.3 Direction and Support

In Phase II, the Government and COTRs, would meet once a month for architecture discussions. This provided common grounds to support architecture development. In Phase III, these meetings did not occur, with the result that architecture development slowed because of lack of common guidance to the researchers as to the importance of the architecture. Also, the work statements and funding for some of the researchers may not have supported their needed contributions to the architecture through the Technical Working Groups.

The message here is that when there are so many contracts, a high level of coordination and cooperation is necessary.

3.4 The Architecture Should Be Available At The Beginning

Since the architecture was designed in parallel with the researchers' main tasks it could not serve as a framework on which they could do their work since it changed frequently. Thus, there was uncertainty as to the environment and structure into which their work should fit and how their components would work with other components.

The architecture development process should have proceeded quite differently. The architecture should have been essentially complete after two or three months so the researchers would know about the framework in which they were expected to work. This improved schedule could have been achieved by having an Architecture Design Team (ADT) consisting of no more than five people working together continuously, at one location, for two or three months. The ADT should be comprised of domain specialists AND system specialists. A suggested composition would be:

- Two expert Document Detection computer linguists
- Two expert Extraction computer linguists
- System Engineer and Chief Architect (knowledge of, and experience in, building real applications is important)

The major focus of the architecture for Phase III was the development of a COmmon Request Broker Architecture (CORBA) compliant Architecture Capabilities Platform (ACP) to host TIPSTER-compliant software components and modules. The ACP provides a software platform to test individual TIPSTER tools and capabilities. Developers will be able to demonstrate to the Government the modularity of their text handling systems by plugging components and modules into the ACP and interacting with the other TIPSTER components on the platform. In addition, the ACP will demonstrate the capability to interact with systems based on Z39.50 standards. The ACP will also have various supporting components such as document collections, standard detection needs, lexicons, a document manager and a default graphical user interface (GUI).

The unavailability of the ACP in Phase III is very similar to the unavailability of the architecture in Phase II. Both were needed early so the researchers could properly design and test their products during development. Even though there was a limited budget for the ACP it probably would have been more effective if most of the money was loaded to the beginning of the project so it could "get on the air" sooner.

3.5 Architectures Can Promote Sharing and Increase Efficiency

Text processing systems are complicated systems composed of many components. Broadly speaking, these components are arranged in a serial pipeline, each component building on the output of the preceding component. While researchers concentrated on developing individual components, they all had need for input data. In the extreme case shown in Figure 1, a version of each component m in an N-stage pipeline gets built N-m+1 times (by that many researchers). Ideally, each component could be built once and then shared (Figure 2).

TIPSTER demonstrated this sharing with several components. The most highly shared component was the Document Manager. Several versions (but far fewer than the number of research efforts) were built and shared throughout Phases II and III. Lexicons, semantic nets, and some part-of-speech tagging components were also shared.

There was also a great deal of discussion of sharing that we feel would have materialized had there been more time (even simply the third year of Phase III). This component sharing was primarily hampered by the delay of fielding the ACP.

3.6 Architecture Design and Application Development Experience Are Critical

The CAWG approach could have been more tightly controlled, directed, and limited in duration. The contributors were experts in their particular domain, but the group as a whole would have benefited from additional expertise in system architecture design.

Early on it was apparent that the CAWG could not agree on the scope, selection, design or utility of numerous small modules. This resulted in an architecture of large components (the equivalent of a Computer Software Component in lifecycle terminology). The issues the CAWG faced with small modules were ownership of algorithms and software, module interfaces, which modules would be designed and whether small modules were technically feasible. There also appeared to be some resistance to the concept of a larger, generalized systems approach for TIPSTER. This is somewhat understandable since many of the researchers were used to working independently on small algorithmic pieces of code.

The end result was that the architecture was made of three large components. Document Manager, Document Detection and Information Extraction. Since all of the researchers could use a Document Manager, it became the center-piece of

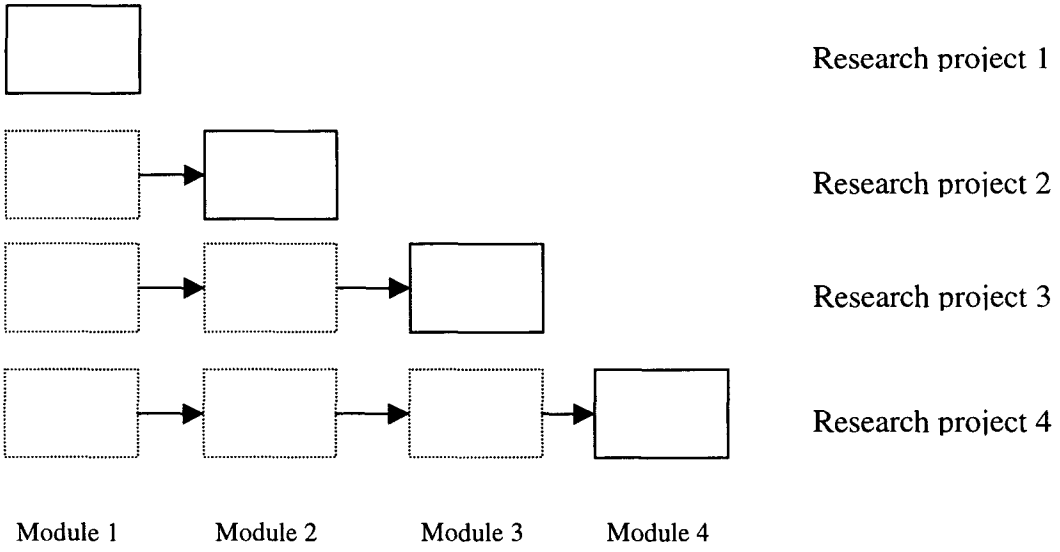

Figure 1: Each project builds its own copy of each module.

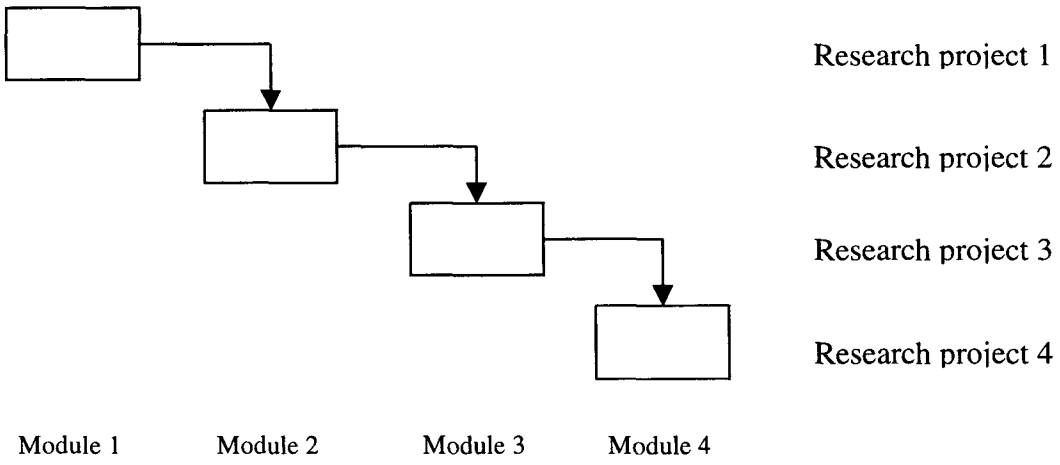

Figure 2: Each project reuses modules from other projects.

the architecture and controlled much of the remaining work of Phase II. In our opinion, this significantly weakened the concept of an architecture that could be used to build a variety of domain applications. If the document manager functions were smaller and more flexible, they would likely have been able to support a broader set of needs (e.g., detection researchers were unable to develop a compliant document manager that was also fast enough for their needs). The program also would have benefited from greater focus on the detection and extraction components.

4 UNFINISHED WORK

The architecture currently is a work-in-progress. Some things that might be considered for a future program are:

- The existing architecture is a mixture of standards, interfaces and implementation approaches. This causes confusion as to what parts are really architecture and which parts are module and component code. An overhaul of the architecture is needed to separate these things into a document which provides organization and structure through standards augmented with compliant modules which are built to the architecture standard as components in a toolbox. The ACP is a partial step toward the toolbox; however, more tools are needed.

- The interfaces need clarification so that they specify only what is needed for compatible interfaces, allowing for different implementations that allow systems to optimize for different constraints. Many of the current interfaces are overly constrained.

- A standardized storage method for documents should be established. This would allow different and possibly more efficient Document Manager components to be used in the architecture on an interchangeable basis. If an architecture is as general, flexible and open-ended as the TIPSTER architecture, it becomes nearly impossible to have a application built which can have interchangeable components.

- Code should be provided which supports the Detection Need and Queries function. Since this is a generic function, a tool to support it is appropriate.

- The Pattern Specification Language capability should be completed and tested. This could be a critical area in standardizing rules to bring Information Extraction technology up the level of Document Detection technology.

5 DOCUMENTS

Four basic TIPSTER document were prepared during Phase II and updated during Phase III. These documents established the baselines for TIPSTER and were placed under configuration management control. The current versions of the documents are:

TIPSTER Text Architecture Concept, Version 1.12

TIPSTER Text Architecture Requirements, Version 2.01

TIPSTER Text Architecture Design, Version 3.1

TIPSTER Text Configuration Management Plan, Version 1.3

6 CONCLUSION

In presenting a brief history of the development of the TIPSTER architecture and an assessment of its use and value, we hope to encourage future programs to undertake the development of a domain specific architecture and to enable those programs to capitalize on the TIPSTER experience so that a working-level architecture may be developed.