Task Control and Communications

The Robotics Institute, Carnegie Mellon University
February 14, 1994

Submitted to: National Aeronautics and Space Administration
For the period: October 1, 1994 to September 30, 1997

Principal Investigators:
Reid Simmons, William Whittaker, Christopher Fedor

1. Objectives

We propose a three year project to provide reusable software libraries and automated design and analysis tools to facilitate development of interprocess communication and task-level control flow for advanced robotic systems. The results will significantly reduce the development time of distributed robotic software systems, while increasing their reliability and flexibility. The work will combine and extend TCA and TCX, two popular control architectures that are currently in widespread use within NASA and elsewhere.

This project has two major objectives:

1) To provide a well-engineered, flexible, and powerful set of interprocess communication tools tailored to the needs and concerns of roboticists.
   — The challenge is to make the mode of communication transparent to the application program, so that developers can easily utilize different hardware platforms and can trade off flexibility for efficiency in communications protocols.

2) To provide a set of software tools for designing and analyzing concurrent, distributed robotic systems.
   — The challenge is to utilize software engineering methods, formal models, and constraints imposed by robotic tasks to automate many of the tedious, error-prone tasks that arise in developing complex, concurrent robotic systems.

This project will lead to significant increases in software development productivity by providing flexible, powerful communication and task-level control architectures, along with the development tools needed to take advantage of them. It will provide utilities and tools tailored to the needs of the robotics and space robotics communities. The availability of usable, well-engineered software libraries and development tools will significantly decrease the level of expertise needed to build complex systems using TCA and TCX. This work builds on a mature body of existing work to provide a standard for robotic architectures that will both anchor the space telerobotics efforts and extend beyond the NASA community to impact the general state-of-the-art in robotics.

2. Justification

Building complex robotic systems is currently a demanding, time-consuming task. This is partly because most robotic system software today is hand-crafted, one-off creations. To accelerate efforts, the robotics community needs powerful, yet easily usable, tools for developing robotic systems. Interprocess communication and task-level control architectures are elements which can be generalized across many applications and can form a bridge for reusing other software elements such as perception modules and user interfaces. The need is sufficiently widespread (almost every robot system needs distributed, concurrent processing) and the technologies are sufficiently mature, that it makes sense to provide such capabilities to the community at large.

CMU is a leader in the development of robotic architectures: In particular, TCA and TCX have been used in over a dozen robotic projects in NASA and elsewhere. TCX provides interprocess data communication primitives; TCA provides task-level control flow constructs. They are fast becoming standards for data and control communications, and are the architectural frameworks on which complex robotic systems can be built.

TCA and TCX are unique in comparison with other existing robot architectures:

1) The interprocess communication protocols are unmatched in terms of the ease of use in passing data between processes. This minimizes the effort required to initiate and maintain interprocess communication.
2) The range of communication protocols geared toward robotic applications is unparalleled. This allows a wide choice of development environments and control over the efficiency of communications, all transparent to the application.
3) The range and types of task-level control constructs provided by TCA is unique amongst robot architectures. Developers do not need to reimplement such common constructs as scheduling, resource management, execution monitoring and exception handling.
4) The experience with the architectures (over a dozen complete applications) is unprecedented for general-purpose robot architectures. This provides evidence of the general applicability and usefulness of TCA and TCX, and provides a basis for standardizing space robotic applications.

These unique capabilities have made TCA and TCX increasingly popular choices for building robotic systems and form the basis for our proposal.

The development of TCX and TCA has been motivated by different, but complementary, concerns: that of interprocess communication and task-level control, respectively. To meet the growing demand for applications and to foster robotic architecture maturity, it is time to consolidate the efforts, using TCX's communication primitives as the foundation for implementing TCA's control facilities. This consolidation will enable developers to use the architectures at various levels — utilizing TCX's communication primitives when they prove sufficient, or building on top of TCA's more extensive task-level control facilities when necessary.

Design and analysis tools provide a uniquely powerful way to enable developers to utilize the features of task-level control architectures. We envision both text-based and graphic-based tools that will enable developers to specify system designs, including data and control flow, temporal interactions between processes, and exceptional conditions. Other tools will enable users to chart the execution of distributed robotic systems and to analyze their performance. Such tools will enable developers to produce high-quality software systems, while reducing overall system development time.

3. Approach

TCX and TCA are working prototypes for interprocess communication and task-level control. TCX supports a variety of message-based communication protocols on a wide range of hardware platforms. TCA adds a layer of task-level control (e.g, scheduling, monitoring, exception handling) to facilitate the integration and coordination of planning, perception and real-time control.

TCX and TCA are gaining wide acceptance partly because the communication and task control primitives provide a development environment which hides the need to worry about hardware architectural differences in the finished robot system. A user of this software is guaranteed that, except for having to recompile for the target machine, he/she can debug a system on a desktop workstation and run the same code in the field on different hardware.

As prototypes, implementation of TCA and TCX was driven by the need to understand robot software integration problems, and less by concerns for efficiency and usability. Enough experience and understanding of these issues now exists to produce a well-engineered combination of TCA and TCX for increased efficiency, portability, usability, and reliability.

The first phase of the project is to ensure that TCX and TCA meet the needs of robotic applications. We will do this by producing a requirements and specification document of TCA and TCX based on analysis of existing and projected robotic projects. Particular emphasis will be placed on the needs of the CMU Lunar rover, the Ames virtual reality system, DOSS, and the Ranger project. Other groups expected to be involved include the Martin Marietta UGV integration project and Sandia National Labs use of GISC.

Based on these requirements, we will re-engineer and combine TCA and TCX, with the communication primitives of TCX forming the basis for TCA's task-level control.

Re-engineering TCX's communication system will follow a traditional software engineering model including problem analysis, requirements specification and design reviews. Problem analysis and requirements specification will be facilitated by the large number of existing systems currently using TCX. Issues to be resolved include the final application interface and concerns about methods for handling exceptional conditions at the data transport layer. The communication primitives in TCX will be reimplemented in a layered fashion to hide operating system and specialized hardware differences. The software will be designed to minimize future changes required to port to additional hardware platforms. In the out years, additional communication primitives will be provided to deal with satellite communications and multiple cooperating robots.

We will leverage the assistance of master of software engineering students from the Software Engineering Institute at Carnegie Mellon University to address some of the communication problems of encoding and decoding data for different computer hardware. The encode-decode task is the core hardware dependent task for both TCX and TCA. This work, which will be undertaken as part of a studio project, will introduce formal software engineering techniques to this critical area.

Similarly, TCA's task tree representation will be re-engineered. Task trees are ubiquitous for task-level control since TCA uses them for for task decomposition, scheduling, monitoring, and exception handling. The task tree representation will be reimplemented with maintenance and efficiency (especially memory utilization) in mind. We will also explore the alternative of distributing the task tree representation, to eliminate the potential bottleneck a centralized server entails, and to make the architecture more compatible with the point to point communications used in TCX. In addition, we will analyze how TCA is used in actual applications and use that analysis to update the application interface to make it easier to utilize the more advanced features of TCA.

As the project progresses, we will examine the requirements and procedures necessary for flight qualifying the TCA and TCX software. Our approach is to provide well engineered software for use in both developing robot systems and in actual planetary or orbital missions. While we may not be able to fully flight qualify TCA and TCX, given the limited amount of time and budget for this proposal, our requirements and specification document will list the necessary steps and how we intend to follow those procedures during this proposal.

Many existing architectures utilize subsets of the communications primitives found in TCX. We will reimplement several common architectures to utilize TCX, in order to a) provide a standard basis for interprocess communications, b) increase the portability of the architectures, and c) allow different robot systems using different architectures to cooperate directly. The idea is to base these architectures on TCX primitives in such a way that the user of the combined system would not notice a major functional difference. There may be some small operational differences, however, such as allowing modules to start before a master server or router is started and providing the target architecture with a new set of hardware it can run on.

TCA will be the first architecture to be combined with TCX, because TCX was originally derived from TCA to explore communication issues. TCA will be fully implemented on top of TCX. The combined system will be demonstrated on several significant applications that currently use TCA (such as the Tile Inspection Robot and the Lunar Rover).

For the other architectures (telRip and GISC), we will concentrate on a representative example system to minimize the time required replace their communication primitives with TCX and demonstrate a non-trivial working system. telRip was chosen as a test example because it was a NASA project and has recently become a commercial product. GISC was chosen because there are a number of DOE robotics projects based on this collection of programs and it is centered on a communications core. A description for how to implement the complete system in TCX will be provided for any missing parts. A complete implementation of these systems using TCX will depend on the interest of those projects currently using GISC and telRip.

Developing a complex distributed, concurrent robotic system is typically a demanding task. We argue that some of the more tedious aspects of the design and development can be automated. In particular, we propose to develop a number of design and analysis tools, specialized for TCA and TCX, that can be used to significantly reduce system development time, while producing more reliable, bug-free software systems.

Developing a TCA-based system involves describing both the data flow (the messages the system supports), and the control flow (task decomposition, scheduling, monitoring, error handling). We propose to develop automated code generation tools for both these aspects of system design. The initial tools will be text-based (although at some point we may develop graphical interfaces for them): we will develop a simple to use, yet comprehensive, language that enables developers to succinctly describe the messages used in the system. The tool will take text files containing message descriptions and C header files, and will produce C code files that perform the message definition and registration, as well as producing code stubs for message handling procedures.

This style of code generation is analogous to commonly used tools such as devguide (which is used to develop complex X-based user interfaces), and has been shown to have great utility in software development. In particular, this message-definition tool will have several benefits:

a) It will simplify the specification of data flow in the system;
b) It will decrease errors, especially by providing a simple means for maintaining the correspondence between message formats and the C data structures that are sent along with those messages;
c) It will help enforce a standard style for defining and registering messages, which increases code understandability and maintainability.

We will also develop a design tool to address the problem of specifying task-level control. The tool will again be based around a descriptive language for specifying task control strategies, that is, a "programming language" for concurrent, hierarchical plans (perhaps graphically oriented, using Harel's state chart representation). The developer will specify task-decomposition, task scheduling, and exception handling strategies using this language, and the tool will generate the appropriate TCA calls to implement the strategies. Note that while the developer will still have to supply robot-specific code to implement the message handlers, this tool will provide the control framework for embedding that code.

This tool will have the same advantages as the message-definition tool described above. In addition, there is the added advantage that the task-control descriptions can be analyzed to guarantee certain desirable features, such as absence of deadlock and resource contention. This will be done by utilizing a formal model of TCA (as is currently being developed in the "Autonomous Rover Technologies" program) that represents how systems develop over time. Then, standard verification techniques, such as Petri Net analysis (as is used at RPI) or temporal model checking (as is used by Ed Clarke at CMU) can be used to check for correctness and soundness of the task-level control descriptions. Thus, problems with the system design can be detected at design-time, rather than having to wait until they arise during system testing.

Such design tools cannot always produce bug-free code, however. In particular, subtle timing interactions present in distributed, concurrent systems make run-time analysis tools extremely beneficial. We are currently developing a tool that analyzes TCA message log files to graphically reconstruct task trees: this enables developers to see how plans are evolving and to detect certain classes of timing problems. We propose to develop a number of other tools that analyze the log files to detect other commonly occurring classes of problems, including a tool that profiles system resource utilization and detects bottlenecks, one that provides a replay mechanism by resending messages to recreate a particular system run, and one that enables users to visualize system execution in real time. In combination, these graphically-oriented tools should significantly decrease the time expended in tracking down timing and data-interface bugs among processes.

Once a robot software system has been implemented and is successfully operating, a common desire is to duplicate the system on a second robot. Coordinating behavior among multiple robots is a challenging task. Part of that task involves the communication necessary between multiple systems running almost identical software. We will select a representative cooperating robot task from those currently using TCX and provide the software necessary for coordinating behavior among several robots and monitoring such behavior from a user interface. Communication problems to be resolved include identifying separate systems among identical robot software systems and initializing the task from a common interface.

In a similar vein, other task-level control architectures (such as NASREM) make use of independent, hierarchical controllers. Once the ability to connect multiple communication servers is in place, we intend to extend TCA to handle multiple task-level controllers (currently, TCA can only deal with a single centralized server). In particular, given suitable interest in the user community, we will use a multiple-controller version of TCA to develop an architecture that can be used in implementing systems that conform to the NASREM guidelines.

4. Participants

The proposed work will be undertaken primarily at the Robotics Institute, Carnegie Mellon University, and will be led by the principal developers of TCA and TCX. We will also have participation from Master of software engineering students at the Software Engineering Institute.

Levels of participation from outside of CMU varies from cooperative development to simply making use of the software provided. The needs of the NASA Ames virtual environment work for robotics is a close match to the proposed interprocess communication TCX will provide and much interaction is expected. Integration work for the unmanned ground vehicle program by Martin Marietta and Sandia National Labs use of GISC and Genasis architecture in DOE programs will provide strong demonstration testbeds for TCX in other programs. Finally, our user community continues to expand. The large user community quickly identifies and corrects errors and continues to apply TCA and TCX to robotics and other projects requiring interprocess communication.

In addition, we have firm commitments from the following institutions and people to support our efforts by providing applications and testbeds for ideas and developed products.

- Kennedy Space Center
- Ames - Butler Hine's group.
- University of Maryland (Ranger project)
- Martin Marietta - UGV integration
- McDonnell Douglas Aerospace, Huntington Beach — connection with KSC and Johnson Space Center.
- Sandia National Labs - GISC and Genasis architecture

5. Deliverables

Year 1:

- Produce a Software Requirements and Specification document for TCX and TCA. This document will summarize the problems of task level control and interprocess communication. It will provide a detailed description of the external behavior of the software being developed to eliminate inconsistencies and ambiguities.
- Provide a re-engineering of the TCX communication primitives. The existing code will be rewritten to be more efficient and extensible. The system will be ported to a variety of hardware architectures, including Unix, VxWorks, MacIntosh and PC (DOS and Windows).
- Provide a re-engineering of TCA task trees. The code will be redesigned to be more efficient and less memory intensive (goal is to halve processing time and memory usage). The application interface will be updated to be more usable.

Year 2:

- Develop a suite of design and analysis tools. Design tools will include automated code generation for data and task-level control flow. Analysis tools will include performance evaluation and run-time viewing of system progress.
- Demonstrate ability of design and analysis tools to reduce by order of magnitude the development time needed for task-level control portions of robotic systems. Potential robotic system candidates include Lunar rover, DOSS, or Ranger.
- Implement TCA, telRip and GISC using TCX communication tools. The current TCA communications will be fully replaced by TCX. Degree of reimplementation of telRip and GISC will depend on user needs.

Year 3:

- Provide additional design and analysis tools. Tools will include code generation for monitoring and error handling capabilities and identification of possible deadlocks and resource contention.
- Formally validate and verify TCA and TCX in preparation for following software flight quality rules and guidelines.
- Provide communication primitives that enable multiple identical cooperating robots to communicate effectively. Communication primitives will also be added to make effective use of intermittent low bandwidth satellite links for remote robots.
- Implement a NASREM style architecture which makes use of a separate controller at each level.

6. Commercialization Potential

There is a large potential in terms of rapid-prototyping of commercial-grade robotic systems, and for validating system design. In particular, the demand for flexible interprocess communication tools is substantial, as witness the recent marketing of telRip. In fact, both TCA and TCX already have a wide, and growing, user community beyond CMU and NASA.

This proposal could significantly decrease the time and cost of developing, integrating and coordinating complex robotic systems by providing an easier to use, more extensive set of utilities. The use of common control architectures could spur the development of robotic applications, in much the same way as the use of common operating systems (Unix, MS-DOS) spurred development of application software in the computer industry. In fact, the work proposed here has potential utility beyond the robotics industry: a recent article in MacWeek stated that the development of techniques for interprocess communication and coordination was sorely needed in order to take advantage of the growing increase in computers and computer networks.

7. Funding

Funding is for three years, providing partial support for Reid Simmons and Red Whittaker, full support for Christopher Fedor, and support for one additional research programmers to assist in the development and testing of the architectures and associated design and analysis tools. In addition, funding is sought to enable visits to NASA sites to install TCA/TCX and to present tutorials on its use. Modest amounts of equipment money are being sought to facilitate the development on alternative platforms (e.g., PC and Mac)

	Year 1	Year 2	Year 3
Personnel & Overhead	$260,000	$275,000	$275,000
Equipment	10,000	5,000	

Total Budget: $825,000

8. Facilities

The Robotics Institute at Carnegie Mellon University is a unique resource. Established in 1979 as the nation's first comprehensive robotics research center to conduct research in robotic technologies relevant to scientific, industrial, and governmental needs, the Institute has grown into a powerful, interdisciplinary research organization, which draws upon Carnegie Mellon's strengths in computer science, engineering, science, and industrial administration.

The Institute has a staff of over 120 personnel, over 50 graduate students enrolled in affiliated programs, and 60 graduate students enrolled in the Robotics Ph.D. program. The Institute is organized into thirteen dedicated research laboratories and three program centers, which cover the spectrum of robotic research initiatives and encompass a wealth of robotic expertise. The Institute pursues a comprehensive research agenda that culminates in competent, working robots.

9. Principal Investigators

Reid Simmons is a Research Computer Scientist in the School of Computer Science and Robotics Institute at Carnegie Mellon University. Dr. Simmons has been a faculty member at CMU since 1988.
(phone: 412-268-2621; fax: 412-621-5739; e-mail: reids@cs.cmu.edu)

William "Red" Whittaker is a Principal Research Scientist in the Robotics Institute at Carnegie Mellon University. He is founder and director of the Field Robotics Center at CMU.
(phone: 412-268-6559; fax: 412-682-1793; e-mail: red@frc2.frc.ri.cmu.edu)

Christopher Fedor is a Research Programmer at the Field Robotics Center at CMU.
(phone: 412-268-8208; fax: 412-682-1793; e-mail: fedor@cs.cmu.edu)

The mail address of the principal investigators is:
Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213