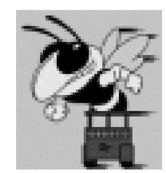
Real-time Cooperative Behavior for Tactical Mobile Robot Teams

Georgia Institute of Technology

and

Honeywell



Technology Thrust Areas

- Fault-tolerant reactive group behaviors
- Communication minimization and planning
- Integrative mission specification and usability testing
- Real-time resource analysis and management

Problem Statement

– Constructing robot control configurations is ad hoc and tedious

– Configurations are difficult to retarget for new vehicles

– Component reuse is difficult yet needed at all levels of abstraction

– Support is needed for evaluation of multirobot configurations

MissionLab

Solution

- Graphical editor supports construction and visualization of configurations
- Allows delayed architectural and vehicle bindings
- Supports multiple code generators, one for each architecture
- Operator console provides faster than real-time simulator
- Libraries of reusable configurations and components maintained for ease of reuse
- Validation by formal usability studies

MissionLab

Constructing a Configuration

– Configuration editor used to graphically place components and connect dataflows

– Select method for state-based coordination

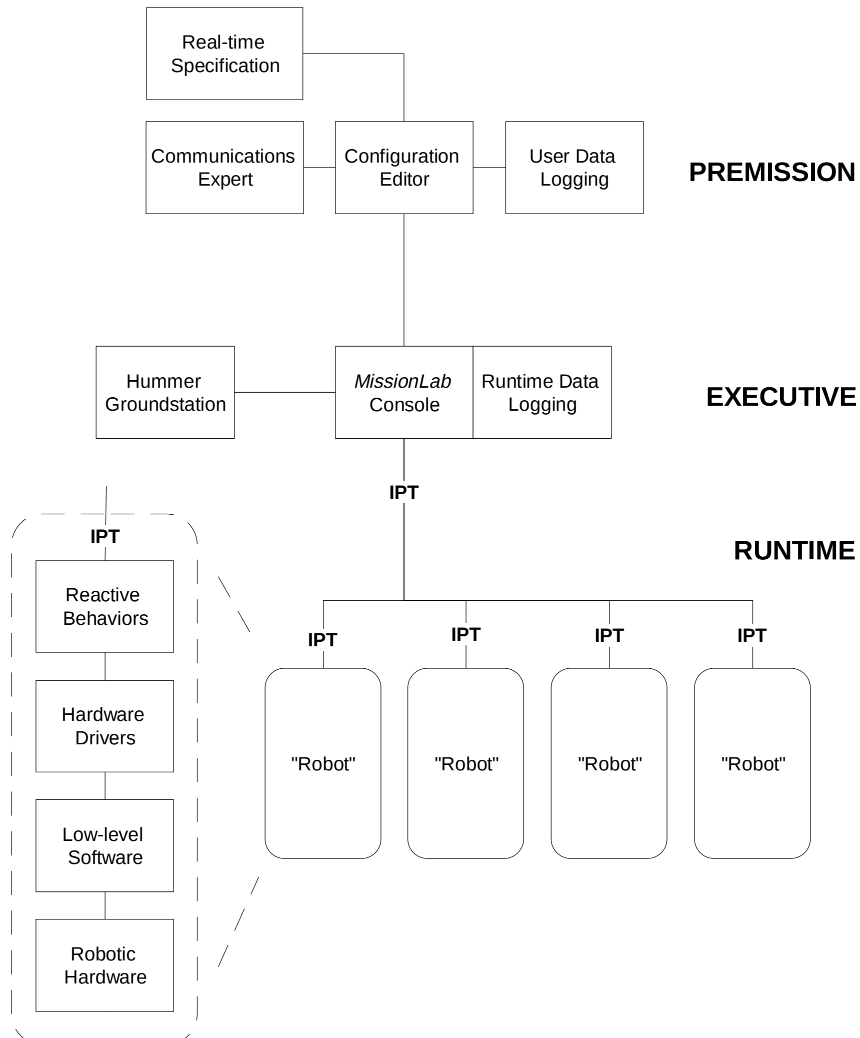
– Bind to the desired architecture and vehicle

– Generate robot executables

– Test configuration in simulation

– Deploy on robots

Subsystems


Real-time Specification
Communications Expert
Configuration Editor
User Data Logging
PREMISSION
Hummer Groundstation
MissionLab Console
Runtime Data Logging
EXECUTIVE
IPT
RUNTIME
IPT
Reactive Behaviors
Hardware Drivers
Low-level Software
Robotic Hardware
IPT
IPT
IPT
IPT
"Robot"
"Robot"
"Robot"
"Robot"

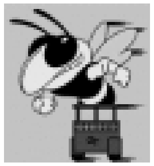
Executive Subsystem

The executive subsystem consists of the

- *MissionLab* console
- faster-than-real-time simulator, and
- runtime data logging components.

MissionLab Console

- Loads precompiled robot control programs and overlay description files
- Configures the display
  - generating obstacles for simulations
  - altering the scale of the display
  - changing the virtual time for simulations
  - scaling the size of the display (zooming)
- Provides a Command interface that permits interactive step-by-step command issuance by the operator using CMDL, a structured English language
  - has the ability to execute, stop, pause, restart, rewind, single step, and abort missions during execution
  - has the ability to use team teleautonomy by directing robots to particular regions of interest or by altering their societal personality ().



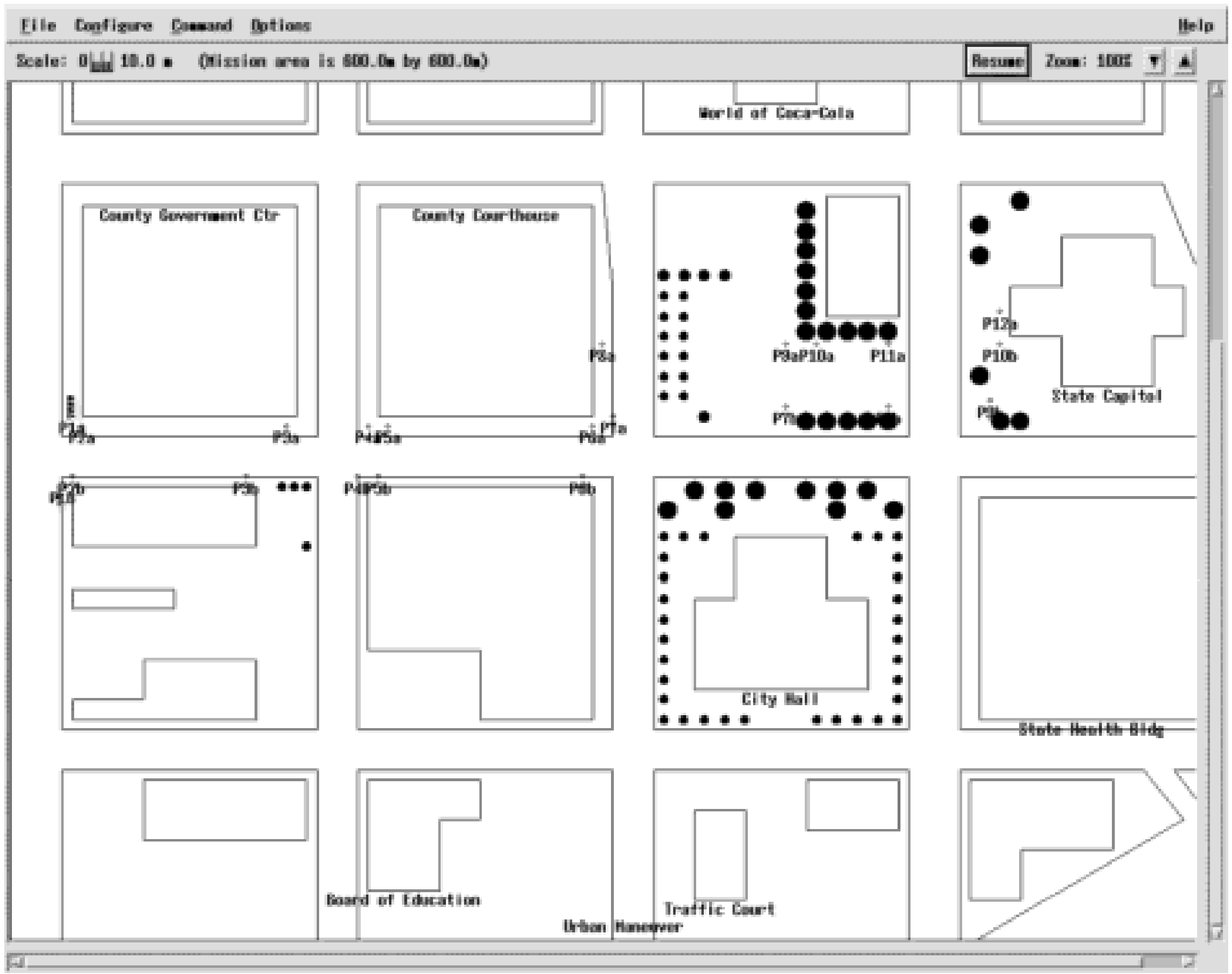
MissionLab Console

- Provides display options
  - leave trails where the robots have been
  - highlight obstacles that affect the robot
  - show instantaneous directional reaction of robot to its environment

The *MissionLab* console also provides a display that shows:

- The output of a simulated robotic mission that is run faster than real-time
- An operator mission display screen where robots in the field report back their position and relevant mission data that is shown on the mlab display to provide situational awareness and context for higher level decisions regarding aborting, continuing, or biasing the mission in various ways

Console Display



Runtime Data Logging

- The executive subsystem will include a runtime data logging capability that will be used to provide *a means to evaluate the performance and effectiveness of a mission*.
- This will include measures regarding the risks that the robots undertook during the mission, other related safety factors, time and distance to completion, etc.

Notional Hummer Groundstation Deployment

Premission Subsystem

The *premission* subsystem involves:

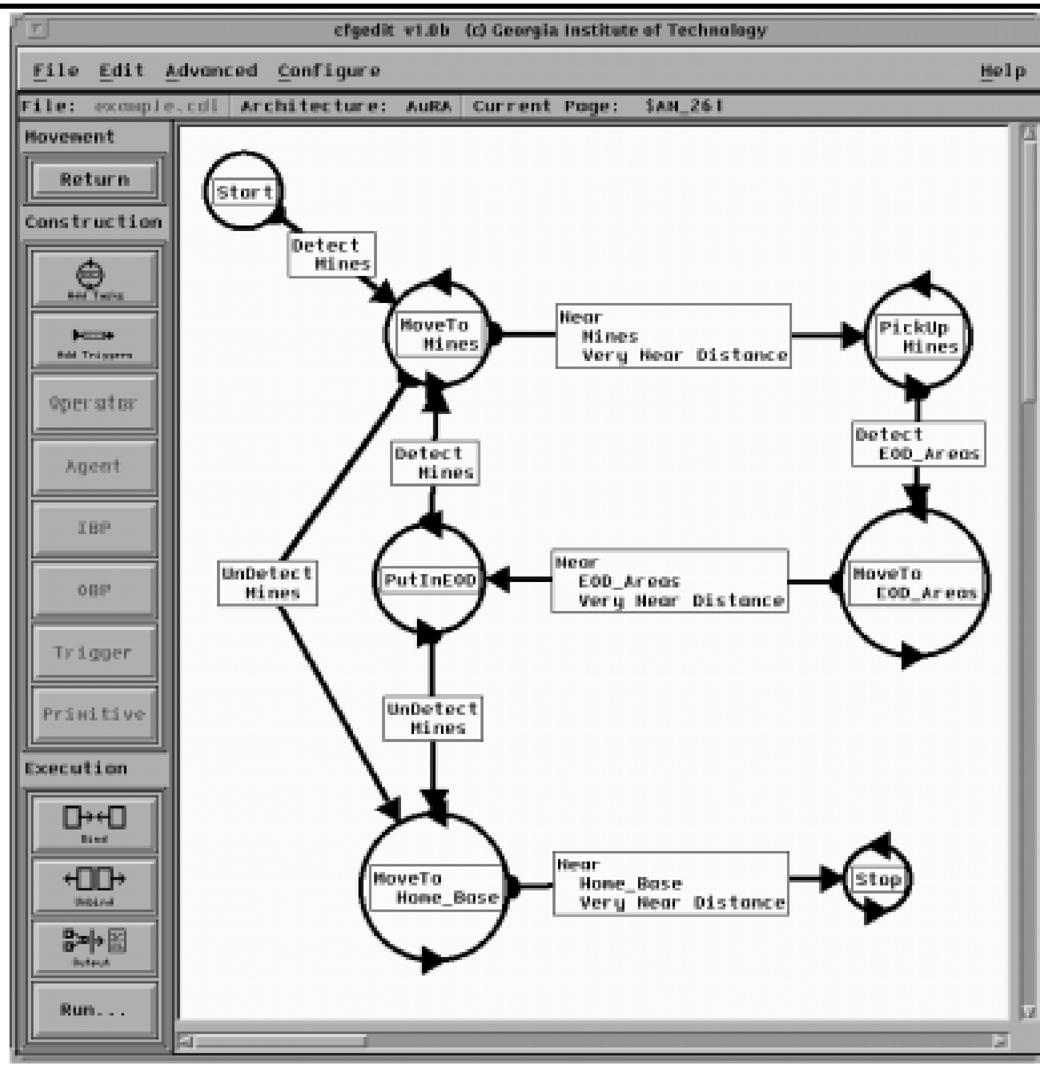
- The specification, creation, and construction of behavior-based robots suitable for specific missions.
- It provides a user-friendly graphical programming environment
- A series of language compilers used to transform the high-level iconic description into executable code suitable for the executive subsystem.
- Data logging tools that are geared for usability studies leading to the enhancement of the user interface.

Configuration Editor

The configuration editor (*cfgedit*):

- Provides a visual programming entry point into the system.
- It is geared to average end-users and requires limited training to use.
- The interactive iconic interface generates configuration description language (CDL) code which, when compiled and bound to a particular architecture and robots, generates a meta-language.
- In this project this is CNL, the configuration network language, that serves as a precursor to the C++ code that is ultimately generated when the CNL code is compiled.
- This resulting C++ code forms the executable code for the robot controller itself. Within the executive subsystem, this code is then directed either to the simulation or the actual robots for execution.

Cfgedit Interface



Georgia Tech / Honeywell

Communication minimization and planning

- Analysis of
  - role of communication in tactical situations
  - effects of organization on communication requirements
  - effects of task type and complexity
- Generate design recommendation
- Communication planning tool

Communications Expert

- The communications expert (under development) will provide mission-specific recommendations for communications methods and topology (e.g., broadcast, point-to-point).
- It will enable a user to automatically configure the communications links between robots necessary to support a mission.
- This follows the same pattern as our earlier work in the design of a formation expert used in the DARPA UGV Demo II program

Usability Data Logging

- Additional software is used to record user actions during premission planning.
- This includes data such as the number and type of keystrokes and mouse clicks, time to create certain objects, and other relevant data.
- These data are then used to interpret the skill by which a user is capable of achieving within the system, and after subsequent usability analysis, is used to refine the design interface itself.
- It is a support tool geared for formal usability studies.

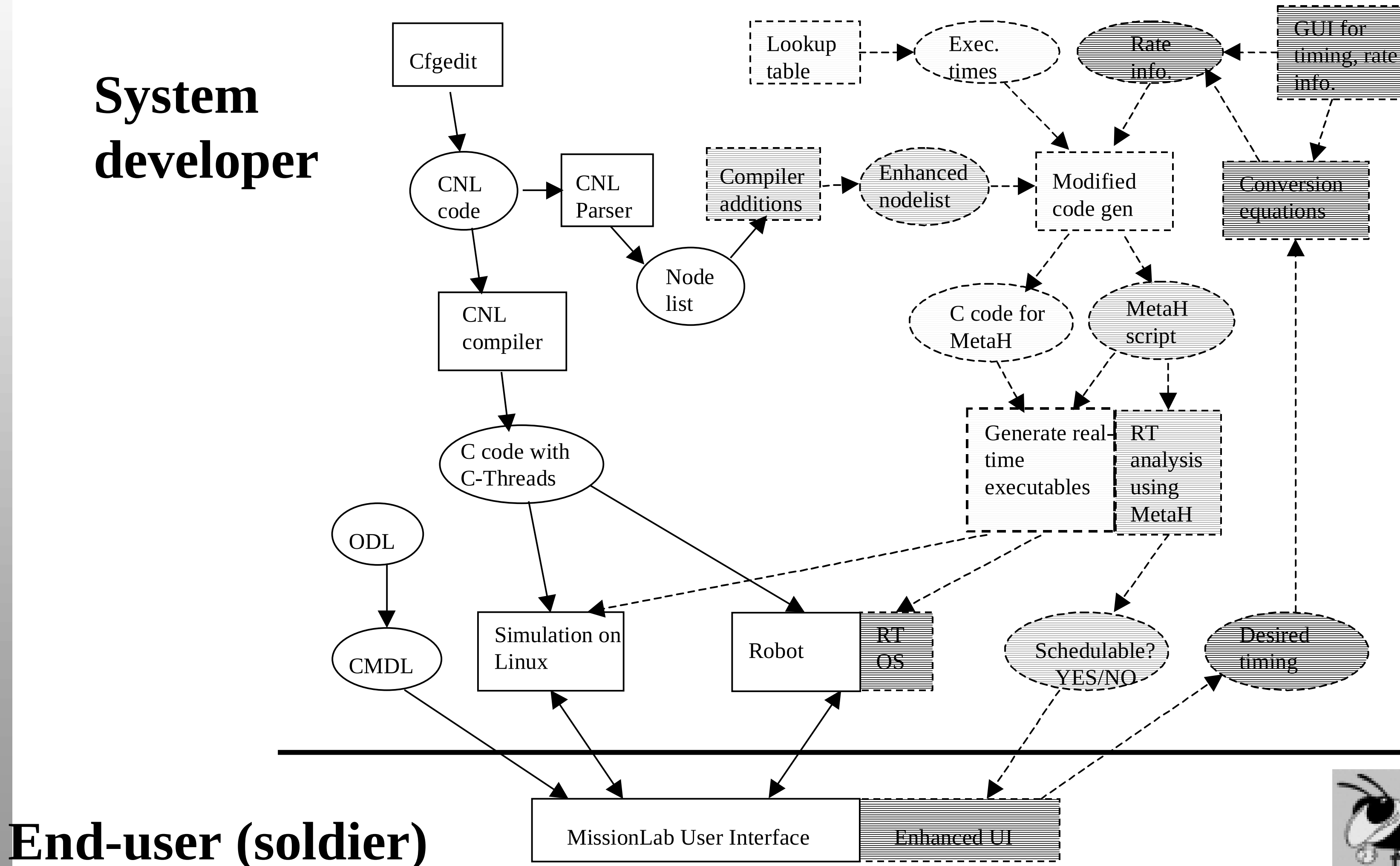
Run-time Subsystem(s)

- One runtime subsystem is located on each robot required for the mission.
- IPT provides interprocess communication between the robots and the mlab console.
- Consists of:
  - a set of reactive behaviors and sensor strategies to interpret and react to the world
  - hardware drivers customized to interface designated robots to the *MissionLab* system
  - low-level robot control code generally provided by the robot manufacturer
  - actual robotic and sensor hardware.

Reactive Behaviors

- A collection of reactive behaviors is compiled and downloaded to each robot for execution of the mission that embody the mission specification designed within *cfgedit*.
- They process sensor data as rapidly as possible and issue commands to the lower level software for timely execution on the robot.
- These behaviors include activities such as obstacle avoidance, waypoint following, moving towards goals, avoiding enemies, and seeking hiding places, all cast into mission-specific reusable assemblages.
- The output of these behaviors is sent to the groundstation for monitoring purposes as well as to the robot for execution.

Integrating MetaH with MissionLab



Initial Activities - MetaH

- Real-time analysis - by hand
  - Selected sample CNL code to add three numbers
  - Hand-modified compiled CNL code to delete Cthreads code and added MetaH code
  - Hand-wrote timing information
  - Hand-wrote MetaH script
  - Ran script through MetaH for real-time analysis
  - Generated real-time executables of adder CNL code

Recent Activities - MetaH

- **Real-time analysis - fully automated; arbitrary timing data**
  - Modified CNL compiler
    - added code to enhance node list
    - modified code generator to generate MetaH scripts
  - Hand-wrote table of timing information using arbitrary values for execution time and rates
  - Ran script through MetaH for real-time analysis
- **Real-time code generation - by hand**
  - Compiled sample CNL code to generate C++ code
  - Replaced Cthreads related code with MetaH communication code
  - Generated executable that can run on MetaH

Future Tasks (short to medium term) - MetaH

- Real-time code generation - fully automated
  - Port MetaH to Linux
  - Compiler changes
    - automatically replace Cthreads code with MetaH communications code
    - other changes/additions needed for MetaH compatibility
- Real-time analysis - using real timing information
  - Measure computation times
  - Store computation times and rate values in a table that can be read directly by the real-time code generator

Future Tasks (long term) - MetaH

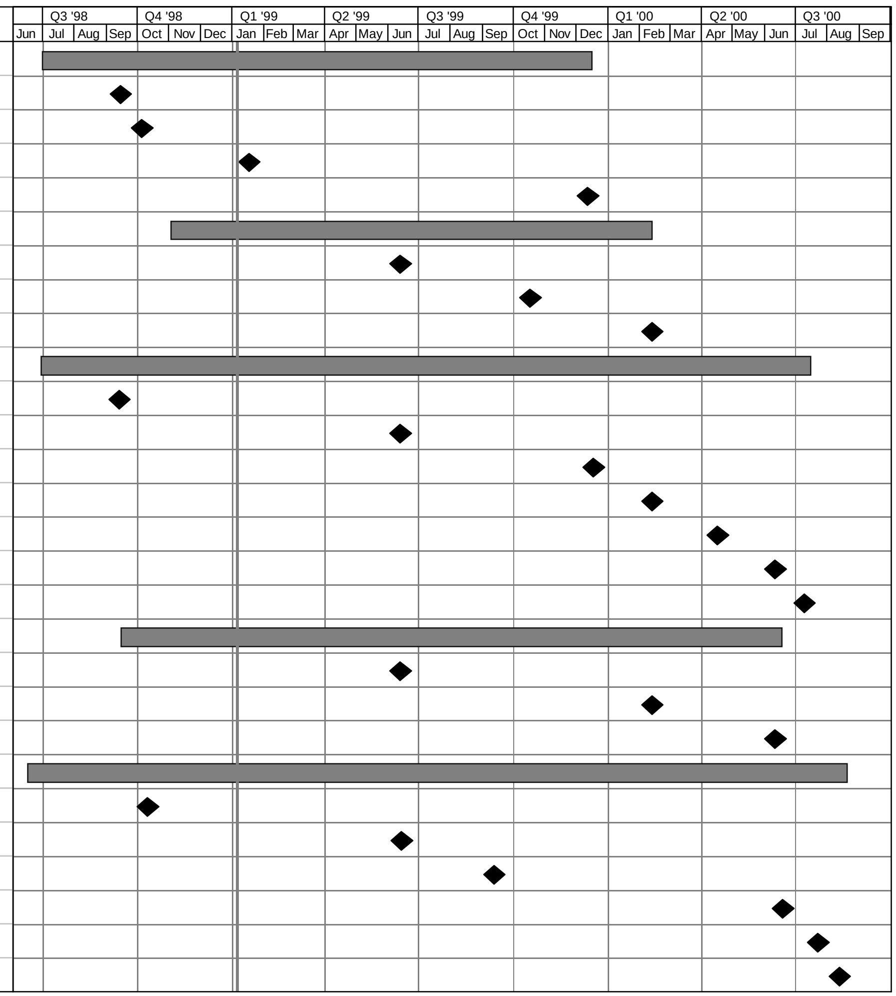
- GUI development
- Selection of real-time OS
- Change parts of code that might cause unpredictable RT behavior

Technical Issues

- Calculating computation times
  - How to automate generation of computation times (modify compiler?)
  - Do they vary significantly (use average time or worst case)?
  - Are they dependent on source of input?
- Other real-time scheduling issues
  - Aperiodic (event-driven) processing that can provide faster response time
  - Incremental processing (anytime algorithm)
  - Multiple processors per robot
  - Multicriticality scheduling - say which behaviors are more important in the case of processor overload

Timeline

ID	Task Name
1	**Fault-tolerant multi-robot behaviors**
2	Prototype behaviors for two specific military scenarios
3	Acquisition of DARPA-compatible robot testbed
4	CNL extensions for DARPA testbed
5	Completion of tactical behavioral repertoire
6	**Communication minimization and planning**
7	Specification of communication requirements and protocols
8	Preliminary integration of communication tool
9	Complete integration
10	**Mission specification and user interface**
11	Begin simulation testing
12	Release of MissionLab 3.0
13	Start usability testing
14	End simulation testing
15	Complete usability testing
16	Complete analysis of usability results
17	Release of MissionLab 4.0
18	**Real-time requirements**
19	Real-time schedulability requirements on lab testbed
20	End-user GUI completed for feasibility analysis
21	Demonstration of effectiveness
22	**Operational tasks**
23	Subsystems Specification
24	Demo A - Lab demo of military scenario
25	Demo Plan
26	Demo B - DARPA-specified scenario and site
27	Adaptation of tech transfer of software
28	Final Report / ICD



Fault-tolerant multi-robot behaviors

Milestone	Description	Benefit
Prototype Behaviors for two specific military scenarios (9/98)	Several MOUT-motivated scenarios have been developed and tested in simulation, including: outdoor advances on an objective building; and interior room clearing.	Demonstrates rapid prototyping ability of behavioral control in DARPA-relevant scenarios.
Acquisition of DARPA-compatible robot testbed (10/98)	4 Pioneer AT robots purchased	Provides compatibility with other efforts in TMR program
CNL extensions for DARPA testbed (1/99)	Provides low-level hardware interface to Pioneer robots, completing bridge to *MissionLab* software	Provides end-to-end mission generation for real robots.
Completion of tactical behavioral repertoire (12/99)	Full scale MOUT relevant behaviors established at primitive, assemblage, robot, and team levels.	Entry point for usability testing. Should enable complete coverage of standard TMR MOUT missions for use by all team members.

Communication Minimization and Planning

Milestone	Description	Benefit
Specification of communication requirements and protocols (6/99)	Provide common standard language and protocols for interrobot and robot/operator command and control.	Can serve as basis for minimal communication standard for TMR effort.
Preliminary integration of communication tool (10/99)	Realization of protocols in a user-friendly planning tool that facilities the establishment of operator-robot and robot-robot communication channels.	Essential for effective use of Commsbot robot.
Complete integration (2/00)	See above.	See above.

Integrated Mission Specification and Usability Testing

Milestone	Description	Benefit
Begin simulation testing (9/98)	Mission scenarios developed as specified. These include 97-20 urban scenario (implemented), airport incursion (in progress), hospital surveillance, and other missions as specified in the CDG scenario document.	Show feasibility of behavioral controlled teams of robots in DARPA scenarios. Proof of concept.
Release of *MissionLab* v 3.0 (6/99)	Next major release incorporating all new improvements to date. (Leverage: *MissionLab* 1.0 and 2.0 were developed under DARPA RTPC program.)	Provides solid documented version for use by all TMR team members at midway point of Phase I of program.
Start usability testing (12/99)	Testing and analysis of human subjects of mission specification, communication planning, and feasibility analysis tools developed.	Provides documented scientific evidence of the utility of the developed system. The methodologies developed and used are extensible to other similar efforts in the TMR program in Phase I and beyond.
End Simulation Testing (2/00)	See “Begin simulation testing.”	See “Begin simulation testing.”
Complete usability testing (4/00)	See “Start usability testing.”	See “Start usability testing.”
Complete analysis of usability results (6/00)	See “Start usability testing.”	See “Start usability testing.”
Release of MissionLab v 4.0 (7/00)	Release incorporating all modifications and additions in year two of the contract.	Springboard into Phase II of TMR program and other DARPA efforts.

Real-time Resource Analysis and Management

Milestone	Description	Benefit
Real-time schedulability analysis on lab testbed (6/99)	Provide analysis of behavioral controller relevant to resource availability on run-time robot controller.	Establishes go/no-go conditions for a specific mission.
End-user GUI completed for feasibility analysis (2/00)	User friendly tool for conducting real-time feasibility analysis.	Provides feedback to the warfighter regarding potential for mission success in terms of run-time operations on robot.
Demonstration of effectiveness (6/00)	See above.	See above.

Operational Tasks

Milestone	Description	Benefit
Subsystem Specification (10/98)	Preliminary description of overall system architecture.	Disseminates, at an early point in time, the structure of our effort. Provides guidance to Part B contractors.
Demo A – Laboratory demo of military scenario (6/99)	Laboratory demo (at Georgia Tech) of MOUT Scenario (e.g., room clearing and/or surveillance)	Proof of concept in hardware.
Demonstration Plan (9/99)	Documented plan for conducting Demo B.	Provides clear goals and metrics for Demo B.
Demo B – DARPA-specified scenario and site (6/00)	Actual demonstration of TMR scenario incorporating multiple robots for a DARPA-specified mission at a DARPA-specified site.	Proof of concept on actual location. Integration of perception, action, and locomotion in a manner that can motivate military end-users.
Adaptation of Tech transfer of software (7/00)	Consultation with Part B contractors and transformation of *MissionLab* where appropriate into formats that are compatible with their efforts.	Assures continuity of programmatic effort.
Final Report/Interface Control Document (8/00)	Documents software and entire project.	Provides understanding of both accomplishments and future directions.

Changes to contract since inception

- None

Interactions with other TMR contractors

- Discussions and correspondence with all Part B contractors
  - provided detailed technology approach to SAIC
  - provided references and information to J. Borenstein (for Raytheon)
  - held conference call with Draper, with emphasis on interface and integration issues
  - providing additional information as requested, and via Web site
- Investigated use of SRI vision (availability of hardware and suitable interfaces)
- Informal discussions with other Part A contractors, mostly at IPR