DARPA Technical Paper: Team Caltech

Richard M. Murray* Joel W. Burdick Pietro Perona
Lars B. Cremean Kristo Kriechbaum Sam Pfister
Tully Foote Jeremy Gillula Jeff Lamb Alex Stewart

Engineering and Applied Science
California Institute of Technology

DARPA Grand Challenge 2005
29 August 2005

Team Members: Lyudmil Antonov, Henry Barnor, Ryan Cable, Eric Cady, Kevin Cossel, Adam Craig, Joe Donovan, Kevin Duncklee, Ryan Farmer, Josh Feingold, Ken Fisher, Laura Fishman, Tully Foote, Jeremy Gillula, Marie Goransson, Rob Grogan, Lily Gruenke, Barrett Heyneman, George Hines, Erik Johansson, Tony Kelman, Dima Kogan, Jeff Lamb, Jeremy Leibs, Gustov Lindstrom, Laura Lindzey, Lisa Nystrom, Ben Pickett, Harmon Pollack, Tami Reyda, David Rosen, Ben Sexson, Alan Somers, Dan Soudek, Alex Stewart, Chris Wetzel, Lusann Yang, Jason Yosinski

Abstract

Team Caltech consists of over 50 undergraduates who have worked to conceive, design, build and optimize *Alice*, our entry in the 2005 DARPA Grand Challenge. Alice utilizes a highly networked control system architecture to provide high performance, autonomous driving in unknown environments. Innovations include a vehicle designed for testing and racing in harsh environments, a highly sensory-driven approach to fuse sensor data into speed maps used by real-time trajectory optimization algorithms, health and contingency management algorithms to manage failures at the component and system level, and a software logging and display environment that enables rapid assessment of performance during testing.

*Address all correspondence to Richard M. Murray, murray@cds.caltech.edu

Figure 1: Caltech's 2005 DARPA Grand Challenge Entry, Alice.

1 Introduction

Team Caltech was formed in April of 2002 with the goal of designing a vehicle that could compete in the 2004 DARPA Grand Challenge. Its 2004 vehicle, Bob, completed the qualification course and traveled approximately 1.3 miles of the 142-mile 2004 course.

This year, Team Caltech has been working hard during the academic year and summer to refine and better integrate our technology so that we can compete in (and win!) this year's race. Through a new course in multi-disciplinary project design, we have had over 50 students participate in conceiving, designing, implementing and testing our new vehicle, named "Alice" (Figure 1). The team consists of a broad range of students from different disciplines and at different skill levels, working together to create a sophisticated engineering system. The final race team is completing the implementation and optimizing its performance over the summer as part of the Caltech Summer Undergraduate Research Fellowship (SURF) program.

1.1 Project Management

Management of Team Caltech's activities is the responsibility of the Integrated Product Team (IPT). Because of the undergraduate nature of the project and the desire to have the students make the primary decisions regarding the project, the IPT functions more as a "coordinating council" than a true IPT. However, the IPT is responsible for ensuring that decisions are made in a timely manner, resolving cross-team issues, and keeping the project moving forward towards its goals.

The primary organizational structure for Team Caltech is through a set of "race teams". Each race team is responsible for implementing a specific set of hardware and/or software capabilities that are part of Alice. For the fall and winter quarters (Oct–Mar), four race teams were used: Vehicles, Embedded Systems, Planning and Terrain. In the spring and summer quarters, the Vehicles and Embedded Systems teams were combined into a single team (Vehicle/Embedded) since much of the detailed work that was required had been completed. In addition, the responsibility for core software infrastructure was shifted to the Planning team, which was the main user of this functionality. The current teams and their responsibilities are:

- Vehicle/Embedded team - responsible for the mechanical and computing hardware in Alice, including software interfaces to actuation subsystems
- Planning team - responsible for guidance, navigation and control, including supervisory logic and core software infrastructure
- Terrain team - responsible for state and terrain sensors (hardware and software) and sensor fusion

In addition to the three race teams, a set of additional teams were used to carry out activities that are required for the race but that do not directly involve hardware or software that is part of Alice:

- Build team - responsible for the software build process and tools
- Documentation team - responsible for the wiki and all documentation submitted to DARPA (application video, site visit materials, technical paper)
- Modeling team - responsible for modeling infrastructure, including player/gazebo
- Operations team - responsible for maintaining support vehicles + shop operations

- Sponsorship team - responsible for soliciting and recognizing support for Team Caltech
- System Administration team - responsible for maintaining the race and non-race operating systems and user environments

Each student serves on one race team and 1-2 additional teams.

The IPT drives activities within the team based on a quarterly test plan. The test plan is developed based on student input, typically through a planning session in the first week of each quarter. For the winter and spring quarters (Jan-Mar, Apr-Jun), the test plan was based on field tests approximately every three weeks, with goals and objectives for each test that match the project-level goals and objectives. In the planning session, a rough set of high level goals are developed by the IPT and these are used as input to a sequence of breakout sessions in which students get together in alternating race teams and test teams to develop a more detailed plan. The output from the planning session is used by the IPT to develop the final test plan for the term.

For Summer 2005, the test plan was much more detailed, broken into four spirals (of approximately three weeks each), with weekly goals and specific objectives leading up to completion of each spiral. Starting at the end of Spiral 4.1, three days of tests were scheduled for each week to allow sufficient time for desert testing:

<table>
<tr><th>Type</th><th>Mon</th><th>Tue</th><th>Wed</th><th>Thu</th><th>Fri</th><th>Sat/Sun</th></tr>
<tr><td>Main team</td><td rowspan="2">▪ 9a: IPT
▪ 11a: Project
▪ 1p: Planning
▪ 2p: Terrain
▪ 3p: Veh/Emb
▪ 4p: Project</td><td rowspan="2">▪ Clear blockers for week N test
▪ 5p: Test team status meeting</td><td colspan="2">▪ Develop and test new capabilities
▪ Prepare for week N+1 testing</td><td>▪ 5p: Week N+1 test team</td><td rowspan="2">Optional field testing (local)</td></tr>
<tr><td>Test team</td><td colspan="3">Desert testing</td></tr>
</table>

1.2 System Specification

Team Caltech' strategy for winning is embedded in its overall system specification, which describes the performance characteristics for Alice and the team's operations. This system specification is used to drive the specifications for individual components that are maintained by the teams. The current system specification contains the following requirements:

1. 175 mile (282 km) range, 10 hours driving, 36 hours elapsed (w/ shutdown/restart).

2. Traverse 15 cm (∼6") high (or deep) obstacles at 15 mph, 30 cm (∼12") obstacles at 1–2 mph, 50cm (∼18") deep water (slowly), 30cm (∼12") deep sand, 15 deg slope. Detect and avoid situations that are worse than this.

3. Operate in dusty conditions, dawn to dusk; 1 CPU failure, 2 sensor failures.

4. Average speed versus terrain type:

Terrain type	Distance (%)		Speed (mph)			Expected		Time (hr)
	Min	Max	Min	Max	Exp	mi	%	Time (hours)
Paved road	1	10	20	40	30	18	10%	0.6
Dirt road	40	60	10	40	25	132	75%	5.3
Trails	20	30	5	15	10	18	10%	1.8
Off-road	5	20	1	5	5	5	3%	1
Special	n/a	n/a	2	2	2	2	1%	1
Total			1	40	25	175	100%	9.7

5. Safe operation that avoids irreparable damage, with variable safety factor.

6. Safety driver w/ ability to immediately regain control of vehicle; 20 mph crash w/out injury.

7. Commercially available hardware and software; no govt-supported labor.

8. $120K total equipment budget (excl. donations); CS/EE/ME 75abc + 24 SURFs.

9. Rapid development and testing: street capable, 15 minute/2 person setup.

2 Vehicle Description

Alice is a 2005 Ford E-350 van that has been modified by Team Caltech sponsor Sportsmobile, Inc. for offroad usage. Modifications include custom suspension and transmission, plus electrical power and air handling systems.

Modifications. Team Caltech's system specification emphasizes ease of testing, so we elected early on to employ a street capable vehicle as our entry in the Grand Challenge. The vehicle of choice is a modified Ford E-350 named Alice, which has been extensively modified. Sportsmobile, Inc. performed the 4WD conversion, including the installation of a Dynatrac front axle, long-travel off-road suspension components, skid-plates, an electrical power generation capability and centrally mounted server case in the rear of Alice. Aluminess, Inc. fabricated the front bumper with bumper LADAR mount, in addition to the adjustable unistrut sensor mount situated on the roof. Additionally, the members of Team Caltech

(a) Suspension

(b) Bumper and differential

(c) Rear workstations

Figure 2: Offroad modifications by Sportsmobile, Aluminess and Team Caltech.

have completed the transformation of the interior into its current form as a fully-featured software development lab.

To support the fact that Alice is intended to function as a mobile workstation, she has been designed for safety as well as functionality; developers are held into their custom racing seats by 5-point harnesses. The selection of restraints was chosen to keep any developers inside Alice firmly secured for both personal safety, and ease of typing, even over rough off-road terrain at high speeds.

Power. Alice has a six liter diesel engine, selected because of its ability to idle for extended period of time at very low rates of fuel consumption. The electrical power generation system consists of the vehicle alternator, a 3 kW auxiliary generator mounted on the rear of Alice, and a set of four deep-cycle batteries suspended in a shielded bay underneath the rear of Alice. The electrical system can power the vehicle's computing, actuation and sensing systems for well over 24 hours with the engine and generator running.

Actuators. The steering actuator is mounted underneath the steering wheel, and it is directly linked to the steering column by a chain and sprocket drive. The arrangement is such that when disabled, the motor can be easily back-driven, keeping the vehicle road-legal, and not impairing human control. The steering motor can be easily disabled through a switch on the dashboard, allowing a human driver to take control whenever necessary.

The 2005 E-350 uses electronic throttle controls, which removes the requirement for mechanical actuation. In manual operation, moving the accelerator pedal with your foot moves three potentiometers that modulate three reference voltages which in turn control the throttle of the vehicle. The throttle controller implemented in Alice mimics the voltages output from these potentiometers through a computer interface. A throttle switch located on the dashboard allows the driver to choose the source of the reference voltages fed to the engine, which allows the driver to change between manual and computer control with the

(a) Steering Motor

(b) Throttle Switches

(c) Brake System

Figure 3: Actuation subsystems.

flick of a switch.

The brake actuator is mounted beneath the drivers seat and consists of four pistons, each approximately twice as powerful as the next smallest. This combination of pistons enables the actuator to support 16 discreet brake pedal positions; 0 (no brake) to 15 (full brake), all of which can be applied by the brake actuator faster than they would be if a human were driving. In addition there is a 5th override cylinder in Alice which will be set-up to fire in the event of power loss, providing Alice with failsafe stopping ability. When not in use, the rod connecting the piston to the brake is easily removed, avoiding any risk of the brake actuator interfering with normal driving.

Computer and Interface Hardware. Alice's software systems run on six Dell PowerEdge servers and two quad-core IBM Opteron-based servers running Gentoo Linux. Developers access the servers via a 4-user KVM (keyboard, video, mouse) switch, allowing each person access any of the servers from any of the workstations within Alice. The servers themselves are housed in a shock-isolated, climate-controlled box fitted with mil-spec connectors.

The interface electronics allow the throttle, brake and steering actuators to be commanded by the software running on the servers. These controllers are mounted in a shock-isolated case behind the drivers seat, and are connected to Alice's internal gigabit Ethernet network through a serial device server (SDS). The SDS allows any computer to communicate with a given actuator controller over the network, providing redundancy of actuator control in the event of a server failure.

E-Stop. Team Caltech has implemented an E-Stop device, capable of putting the vehicle into pause, run, and disable modes as specified by DARPA. The E-Stop device is primarily designed to seamlessly integrate with the E- Stop controller which will be provided by DARPA. However, when not connected to the DARPA E-Stop, run, pause, or disable modes can still be specified using our own remote control system. In addition to the ability to trigger an E-Stop in hardware, an E-Stop pause can also be triggered in software, either by a

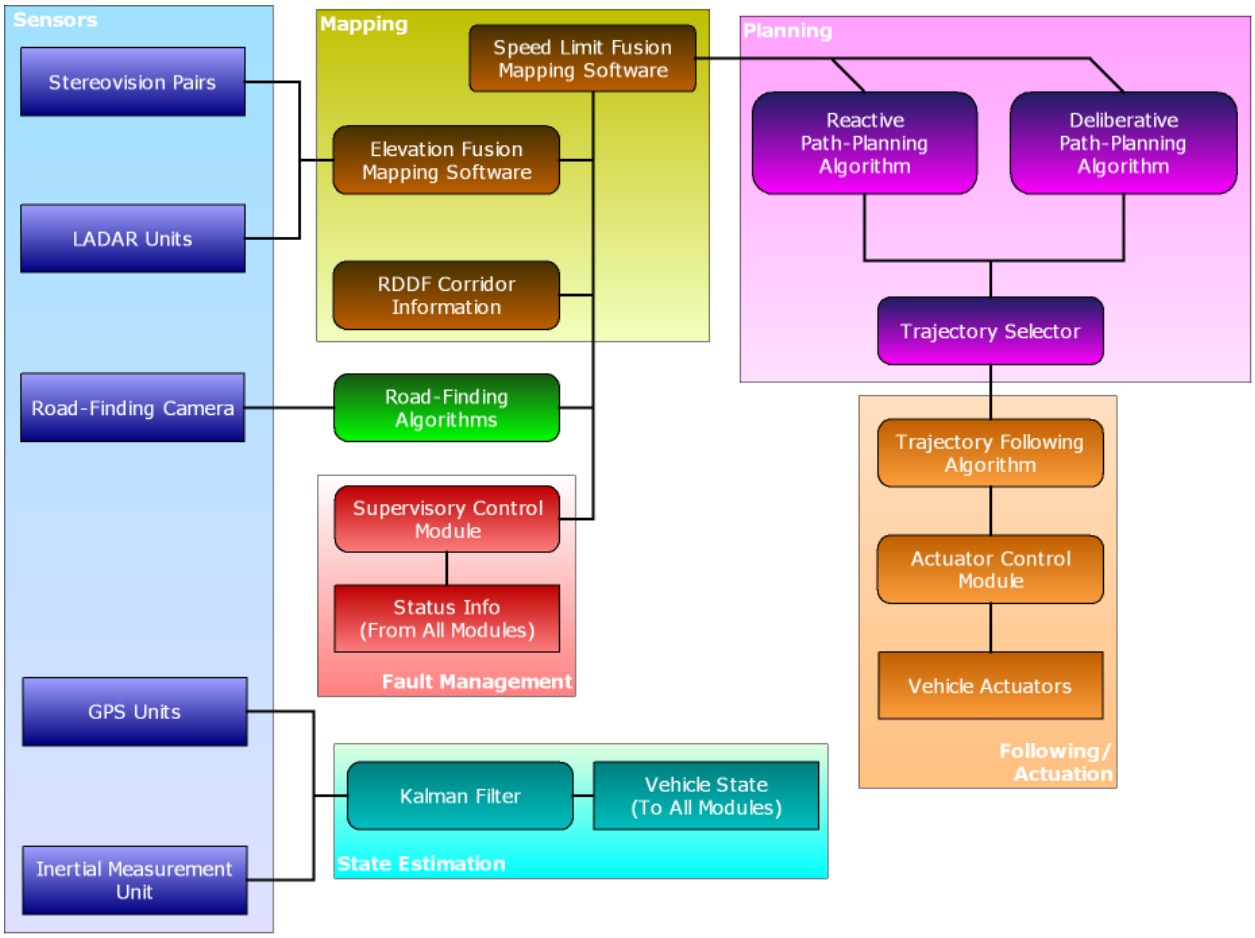

Figure 4: Team Caltech Software Architecture.

human operator or by one of the software modules; this capability has been used extensively during testing of the planning architecture.

3 Autonomous Operations

Team Caltech's software architecture is shown in Figure 4. This architecture is implemented on a high-bandwidth, multi-computer network that is capable of handling the large amounts of data and processing required for autonomous navigation.

3.1 Processing

Team Caltech has chosen to make use of sophisticated algorithms for sensing and navigation, hence it uses a substantial amount of processing power for its operations. Alice's software

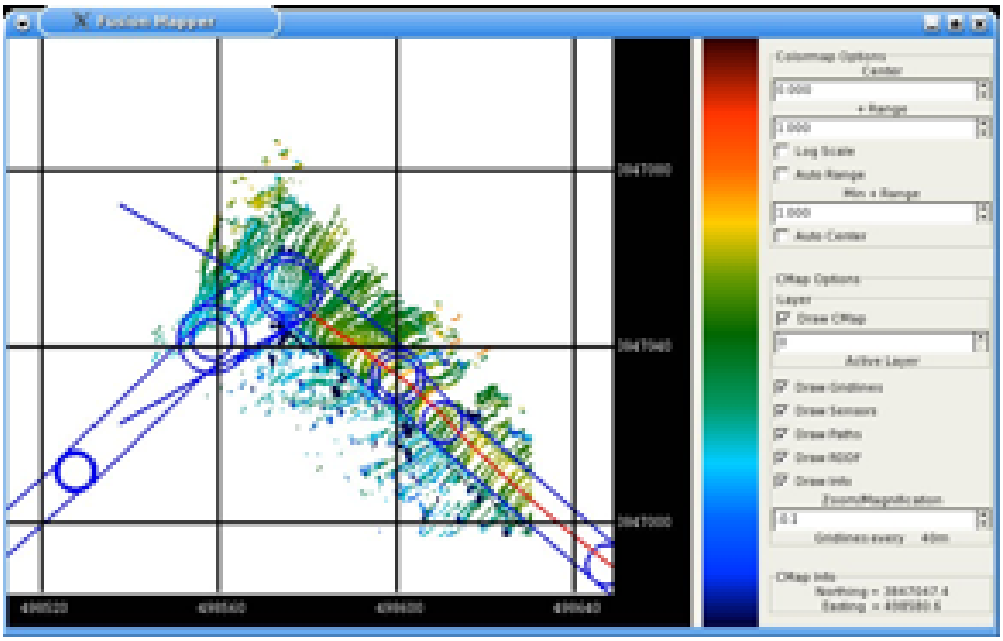

Figure 5: Team Caltech GUI, showing elevation estimates from one of the LADAR units.

systems run on six Dell PowerEdge servers and two IBM quad-core Opteron servers, all running Gentoo Linux. All computers are connected via two independent 1 Gb/s ethernet networks, with high bandwidth data restricted to one network and other inter-computer communications running on the second interface.

A major element of Caltech's software infrastructure is the GUI and logging capability. These features allow for rapid determination of conditions that lead to errors and also offline testing of software on pre-stored sensory data. Two simulation environments are also used: a dynamic model of the vehicle motion (including traction) that is used for testing without sensory input and a Gazebo simulation environment.[1]

3.2 Localization

State estimation is accomplished through the combination of Navcom SF2050G and Novatel ProPak-LBplus differential GPS units to measure absolute position, and a Northrop Grumman LN-200 IMU to measure relative position and orientation. The inputs from these sensors are combined through Kalman filtering to produce an optimal estimation of state. This state estimation is then broadcast across the network, where it is buffered and inter-

[1]The Gazebo simulation environment was used relatively lightly due to the team's decision to focus on desert testing.

polated as necessary, providing reliable state information to all of the modules running in Alice.

Alice has the capability of storing previously driven terrains and incorporating this information into its terrain and cost data.

3.3 Terrain Sensing

All of Alice's sensors are mounted either on the roof or in the bumper. Aluminess has fabricated a custom sensor mount with unistrut (visible in Figure 1). This enables us to easily mount new sensors or change the position of current sensors as needed for rapid development.

The two main types of terrain sensors Alice makes use of are LADAR and camera-based stereovision. We currently have 2 SICK LMS-221-30206 LADARs mounted on Alice. These LADARs have a maximum range of 80 meters and a scanning rate of 75 Hz. One of these LADAR units is enclosed in the front bumper, providing reliable detection of large obstacles independent of range. The other unit is mounted on the roof, pitched downwards such that it scans at approximately 30 meters from the vehicle. This provides us with medium range terrain detection as it sweeps out a path on the ground. A pair of Point Grey Dragonfly cameras mounted on the roof are used in combination with SRI's Small Vision System to generate 3D pointclouds. These pointclouds cover a broader range in front of the vehicle. An additional single camera mounted on the vehicle also feeds a road-finding algorithm. This algorithm identifies vanishing points and uses dominant pixel orientations within an image to predict both the direction and location of roads in the image. This will enable us to take advantage of any road networks, whether they be simple dirt grooves from other offroad vehicles, established desert roads, or paved highways.

An additional SICK LMS 291-S14 LADAR has been mounted on the roof. This LADAR is pitched to look approximately 40 meters out, providing us with medium to long range LADAR scans at 75 Hz. We have also mounted a Reigl LMS Q-120i LADAR on Alice. This LADAR provides .04 degree resolution with a maximum range of 150 meters. This LADAR is be pointed out at approximately 65 meters, providing us with long range LADAR scans. A second pair of stereo cameras with zoom lenses will provide an additional stereo pointcloud at a range of approximately 40 meters.

Data from each LADAR and each stereo pair is fused at the elevation level. Each LADAR scan or stereo point cloud is first converted into elevation values associated with their x, y coordinates in a global reference frame. These individual maps are then combined on a

cell-by- cell basis using a fusion algorithm to generate a single unified map which contains an optimal estimate of the elevation at each cell in the grid. The mapping software then analyzes that map to determine the locations of obstacles as well as the speed at which non-obstacle cells can be driven. This data is then fused with information from Alice's road-finding algorithm to generate a final speed limit map for evaluation by the planning software.

3.4 Navigation

Navigation is accomplished using two different planning techniques: a high speed reactive planner and a more sophisticated deliberative planner.

The reactive planner works by evaluating a large number of "trial" trajectories composed of S-bend primitives which diverge from the vehicles current position, and converge to a series of points that span the RDDF a specified distance further along the RDDF. The reactive planner then discards any "trial" trajectories that are impassable, defined as passing though a cost map cell whose cost is greater than a specified threshold. Of the remaining trials, the reactive planner evaluates the "cost" of each trajectory using a function of the number of cost map cells through which the trajectory passes, and the recorded costs in each of these cells.

The deliberative planner maintains its own version of the cost map and determines an optimal trajectory through the RDDF and detected obstacles. The planned trajectories are optimal in the sense that they are dynamically feasible and time-optimal for the current initial conditions. Team Caltech uses a receding horizon motion planning algorithm, with a kinematic bicycle model of the vehicle to determine approximate dynamic feasibility. The task is stated as a nonlinear programming problem, which is then solved using SNOPT, a commercially available method. Once a trajectory has been created by the deliberative planner, it is sent to the trajectory selection module for evaluation prior to execution to protect against rapid changes in circumstances such as large obstacles appearing in the cost map.

The trajectory selector receives trajectories from the two different sources: the deliberative planner and the reactive planner. Each of these trajectories are evaluated through the current cost map to check for recent changes. First the selection algorithm determines whether a trajectory has become impassable. Next a comparison in of the cost associated with each of the trajectories which were found to be passable is performed. Total cost per trajectory is a function of the number of cells, and costs associated with each of the cells

passed through by the trajectory being evaluated. The trajectory selection algorithm then selects the trajectory that is passable, and has the lowest cost. If neither trajectory meets the passability criteria, then a failure mode is generated, which is detected by the fault management executive and command Alice to come to an abrupt stop and then take action to identify and rectify the fault.

3.5 Vehicle Control and Contingency Management

Once a desirable trajectory has been generated, verified, and selected by the trajectory selection module, the problem becomes one of actually following the path. The trajectories coming from the planner include the Northing and Easting UTM coordinates, as well as their first and second derivatives. This allows an open loop feedforward term to be computed based on known vehicle dynamics, around which a control loop is then wrapped. Currently, there are separate feedback loops around the steering (latitudinal), and throttle and brake (longitudinal) actuators. The longitudinal accelerations are controlled based on the error between desired and actual speed, while lateral accelerations are controlled based on perpendicular distance from path and the difference between desired and actual heading. Once the feedforward and feedback terms have been combined, they are passed directly to the actuator control module which translates the commands and passes them to the actuators.

Alice is equipped with a supervisory controller that can modify the system operation if a fault is detected. In this context, a fault is defined as a state of the system in which she does not perform as intended. A failure mode is any series of faults that have a characteristic effect on the performance of the system. For example, pushing up against a barbed wire fence which cannot be seen by the available sensors is an instance of the unseen blocking obstacle failure mode. Team Caltech is developing an approach to fault management architecture design involving a combination of distributed monitoring and reporting of component health and distributed adaptive rectification logic. Similar examples of this type of approach have been demonstrated at JPL as part of NASA's Mission Data System (MDS).

4 Implementation and Optimization

Team Caltech makes use of a spiral development process to guide our efforts as we make progress toward the completion of the DARPA Grand Challenge. Spirals are defined phases in the projects development moving outward from the initial point. Each spiral outward adds a new layer of functionality that future layers can build upon. A given spiral passes through

the following phases: define, design, build and test. A spiral process is far more useful for a project like ours that requires multiple components that all depend on one another to be developed in parallel since any given component can always depend on the level of functionality of the other components in the previous spiral.

4.1 Development Tools

Team Caltech uses proven open-source tools as a critical part of our code-development. All code is written in C/C++. The source tree is managed with the subversion source control system, allowing for versioning control. Additionally, HTML documentation of the source tree is automatically generated by doxygen. The bugzilla tool from the Mozilla project is used to track the different bugs we inevitably encounter in the team source tree. Bugzilla is also used to manage tasks assigned to different members of the team. The team also maintains a wiki for general documentation. This extent of this documentation ranges from meeting minutes to sub- system documentation and status. The HTML format of the wiki makes our documentation easily accessible to the members. Finally, the team also maintains a web based discussion board for its members to further discuss any new ideas or large issues that come up when its hard to get everyone together for a meeting.

As part of the plan to gain a competitive advantage, Team Caltech invites industry experts to review the progress of the team at the end of each spiral of development and milestone (once per term). This review committee includes people from JPL, Northrop Grumman, STI and other companies. As part of the review, the industry experts submit requests for action (RFA's) on any component of the architecture that they feel needs work. This puts an obligation on the team to have a response ready on the next review session and helps to guide some of our work over the next term.

An important feature of all modules is their ability to log raw data and reply the data for offline debugging and testing. This capability is used frequently in testing and allows a detailed analysis of failures and the ability to replay data through the system to verify that modifications solve the intended problem.

4.2 Test Plan

Development and testing of individual modules and full system integration is achieved through an extensive test plan. During the academic year, multi-day field tests were scheduled approximately every three weeks, starting in December. With the beginning of the

summer, this schedule was accelerated and Team Caltech spent 3-5 days testing each week starting at the end of July, beginning with local testing in Pasadena and increasing the amount of time spent on desert testing as the summer progressed.

Each field test is organized by a "test manager". The test manager is responsible for setting the objectives for the test, tracking the preprations for the test, and maintaining the schedule during the test. This person is not assigned to any other tasks, otherwise they get distracted and stop coordinating the various activities that are taking place. The specific responsibilities of the test manager include:

- Insuring that everyone at the test is aware of their role and the timing of the tests that they are participating in.
- Keeping track of what activities are required to start a test and the making sure the various activities will be completed in time to perform the test in the specified slot (if not, the test manager should either ask more people to help on the bottleneck activity or adjust the schedule) .
- Rearrange the test schedule if needed to accomodate events of the day.
- Run the end of test team meeting, but going through the objectives for the testand seeking input from all team members on what was accomplished and what we should do differently the next time.

The students who participate in a test are chosen based on which modules will be tested.