EVALUATION OF AVAILABLE OPEN SOURCE TRAFFIC CONTROL SYSTEMS

06-FH1 Phase II Task 1

Advanced Technologies, Incorporated
5703 Red Bug Lake Road, Suite 320
Winter Springs, Florida 32708

Contact Number DTRT57-09-C-10005

Table of Contents

Introduction

According to the Federal Highway Administration there are several standards for traffic signal controllers. One of these systems is the TS 2 standards for traffic controllers, maintained by National Electrical Manufacturing Association (NEMA). This TS 2 lacks requirements that enable interchangeability of sub-components or software between controllers form different manufacturers (FHWA Traffic control system handbook 2005). The TS2 standards assume that the whole controller will be replaced when the system changes. Controllers that follow the TS 2 standards are called NEMA controllers and the manufacturer provide the software along with the controller.

There is also the Advanced Controller Transportation family of standards maintained by a consortium of NEMA, ITE and AASHTO. According to the Traffic control system handbook there are two standards (FHWA Traffic control system handbook 2005):

a. The Advanced Transportation Controller 2070 (ATC 2070)
b. The ITS Cabinet for ATCs

Anyone can develop software for an ATC controller, for any purpose (e.g., traffic signal control, field master unit, ramp metering, count stations, dynamic message sign control, reversible lane control, etc.) knowing that it will operate on controllers from any manufacturer. Most ATC controller software for traffic signals adhere to the functionality specified in NEMA TS 2, and is functionally similar to a NEMA controller (FHWA Traffic control system handbook 2005).

Advanced Technologies, Incorporated is developing an open source base local intersection Signal Control Program Environment (SCoP) for the Advanced Traffic Controller. As part of this effort, we are evaluating currently available traffic signal control programs to see if they are applicable to our effort. The evaluation is based on both system and software criteria. The overall task as defined in our Phase II SBIR proposal can be found in the Appendixes.

Evaluation of Existing Traffic Signal Control Programs

Criteria

The overall goal of this research and development is to build an open source base local intersection Signal Control Program Environment (SCoP). This environment is to be developed so that it runs on an Advanced Traffic Controller. It must be able to incorporate NCHRP 3-66 concepts, interface to the Cooperative Intersection Collision Avoidance System (CICAS) and also support the use of Adaptive Control Software Lite (ACS Lite).

In addition to those requirements, the SCoP must perform in a safety critical manner. There must be mechanisms built into the software to prevent the possibility of collisions due to erroneous preemption routines and/or traffic light state algorithms (i.e., ring barrier breakdown). The software will be open source so it must be well documented and easy to understand. The SCoP should be written using industry known coding standards. If the program is written in a higher level language such as C++ or Ada95, there should be extensive use of exception handling in order to prevent software bugs from halting the system.

Programs under consideration to serve as the core of our SCoP must meet complexity metrics analysis. McCabe metric analysis was chosen over Halstead methods because of the availability of free analysis tools. A structured software procedure, method, or function should have a McCabe Cyclomatic complexity of less than or equal to 12 (McCabe suggests 10, other projects have been very successful using a limit of 15). The McCabe complexity calculations are explained later on. Cyclomatic complexity is a measure of quantity. Another type of metric analysis is Essential Complexity. Essential Complexity measures the “quality” of a software system. It is calculated by removing all primitives from a procedure's control flow and then computing the Cyclomatic complexity on what remains. There is no magic number for Essential Complexity.

If a free analysis tool is available we will use it to determine the nesting levels of the software under analysis. ATI has unique experience with software that contains deep nesting levels. One of the programs we have developed is a predictive system that contains a routine with a variable nesting level. This predictive system takes over 3 hours to run its computations when the nesting level is set at 8. When the nesting level is reduced to 3, the program runs in a few seconds. We are not overly worried but will insure there are no nested loops deeper than 3 levels in evaluated programs.

Summary of Criteria

Summarizing the above, each open source program will be analyzed for the following:

1. Can it be ported to an Advanced Traffic Controller Architecture?
2. Can NCHRP 3-66 concepts be incorporated?
3. Can it be interfaced to CICAS?

4. Can it be interfaced to ACS Lite?
5. Does it have a default steady state?
6. Is the software well documented?
7. Does it make use of exception handling?
8. Is its McCabe Cyclomatic complexity of less than or equal to 12?
9. What is its Essential Complexity and how does that compare to others under analysis?
10. Is the nesting level of loops reasonable?
11. Is the error diagnostic system comprehensive and straight forward?

SCoP Requirements

When evaluating existing systems, we must also take into consideration the capability of the software under analysis to meet all the requirements specified in our Phase II proposal (aka our Phase II Statement of Work). We have determined the following System Requirements from our proposal. These requirements are what we will perform Formal Qualification Tests against. They will be inputted into an open source requirements tool during the beginning of our next task. This will allow us to track software development producing standard requirement traceability matrices. **No changes, additions, or deletions to these requirements will be made without approval and agreement between of Advanced Technologies, Inc. and its VOLPE Contracting Officer Technical Representative.**

SCoP System Requirements				***Applicable to SCoP Evaluation***	***Tested By*[1]** ***Inspection*** ***Lab Test*** ***Field Test (FQT)*** ***Other***
No.	***Id***	***Description***	***Source***		
1	1	Develop a base Local Intersection Control Program Environment	Proposal pg. 5	Yes	Laboratory
2	1.1	All ATI developed critical core software is Dual Redundant	Proposal pg. 5 Proposal pg. 23	No	Inspection
3	1.1.1	Primary ATI Software is Ada95	Proposal pg. 23	No	Inspection
4	1.1.2	Secondary ATI Software is C or C++	Proposal pg. 23	No	Inspection
5	1.1.3	ATI Primary Software provides current time to Secondary Software 10 times per second	Proposal pg. 23	No	Laboratory
6	1.1.4	ATI Primary Software provides heartbeat to ATI Secondary Software once per second	Proposal pg. 23	No	Laboratory

SCoP System Requirements				***Applicable to SCoP Evaluation***	***Tested By*[1]** ***Inspection*** ***Lab Test*** ***Field Test (FQT)*** ***Other***
No.	***Id***	***Description***	***Source***		
7	1.1.5	The system must fall into steady state after 5 consecutive miscompares of data[2]	Proposal pg. 23	No	Laboratory
8	1.1.6	Watchdog timers will be used to monitor ATI software	Proposal pg. 23	No	Laboratory
9	1.1.7	Protected Types will be used in ATI primary software	Proposal pg. 24	No	Inspection
10	1.2	Software is Open Source	Proposal pg. 5	Yes	Inspection
11	1.3	Software Supported By ATI	Proposal pg. 5	No	Other – Phase III
12	1.4	Use NTCIP standards and protocols[3]	Proposal pg. 6	Yes	Inspection
13	1.5	Abstract all interfaces into separate classes	Proposal pg. 6	Yes	Inspection
14	1.6	Use ATC API Standard, V 2.06b, to interface with Linux ATC OS	Proposal pg. 6	No	Inspection
15	1.7	Use UML for Software Requirements and Design.	Proposal pg. 10	No	Inspection
16	1.7.1	UML Class Diagrams and Associations	Proposal pg. 10, Proposal pg. 11	No	Inspection
17	1.7.2	Interfaces to Simulator/Hardware/Other Software shown on UML diagrams.	Proposal pg. 10, Proposal pg. 11	No	Inspection
18	1.7.3	Auto-Generate Interfaces from UML diagrams.	Proposal pg. 10, Proposal pg. 11	No	Inspection
19	1.7.4	Industry Standard Nomenclature will be used.	Proposal pg. 12	No	Inspection
20	1.8	All requirements/design/code kept under configuration control/management.	Proposal pg. 10	No	Inspection
21	1.8.1	Header Updated automatically when code checked back into CM.	Proposal pg. 12	No	Inspection
22	1.8.2	Automated Scripts build SCoP directly from Configuration Management System	Proposal pg. 12	No	Inspection
23	1.9	All software place in Sourceforge open source repository	Proposal pg. 10	No	Inspection
24	1.10	Safety Critical Software (Intersection Control) tested at the Unit Level	Proposal pg. 11	No	Laboratory

SCoP System Requirements				*Applicable to SCoP Evaluation*	***Tested By*[1]** *Inspection* *Lab Test* *Field Test (FQT)* *Other*
No.	***Id***	***Description***	***Source***		
25	1.11	SCoP is ported to Development Hardware	Proposal pg. 13	No	Laboratory
26	1.11.1	Development Hardware must run Linux	Proposal pg. 14	No	Inspection
27	1.11.2	SCoP to be integrated with the Texas Model	Proposal pg. 16	Yes	Laboratory
28	1.11.2.1	Simulate a 4 leg intersection, 3 inbound, 3 outbound lanes per leg.	Proposal pg. 16	No	Laboratory
29	1.11.2.2	Simulate 3 detectors per lane	Proposal pg. 16	No	Laboratory
30	1.11.3	SCoP to be integrated with CORSIM	Proposal pg. 17	Yes	Laboratory
31	1.12	Ensure SCoP can be integrated with CICAS.	Proposal pg. 18	Yes	Inspection and/or Laboratory
32	1.13	Ensure SCoP IS integrated with ACS-Lite.	Proposal pg. 19	Yes	Laboratory
33	1.14	Integration of SCoP with Advanced Traffic Controller Hardware	Proposal pg. 19	No	Laboratory
34	2.0	Implementation of Selected NCHRP 3-66 algorithms	Proposal pg. 22	No	Laboratory
35	2.1	Integration of NCHRP 3-66 algorithms	Proposal pg. 23	No	Laboratory
36	3.0	SCoP Live Intersection Testing	Proposal pg. 24	No	FQT

Figure 1: SCoP Requirements

Notes:

1. Inspection testing is done by visual analysis and conformation to assure a requirement is met. Laboratory testing is done using simulators and other test drivers on both unit level software and integrated software running on hardware. Field Testing (Formal Qualification Testing) is done at a controlled intersection.

2. Steady state will be a user configurable parameter. It can be flashing red lights or a timed sequence.

3. The current LA-TSCP software is unable to support NTCIP protocols.

Program 1 - California Partners for Advanced Transit and Highways (PATH)

Dr. Marco Zennaro developed the Berkeley Adaptive Traffic Control System Protocol (Berkeley ATCP2070**)** at the University of California Berkeley. It was developed specifically for the Econolite Model 2070 Advanced Traffic Controller. It was released under GPLv2 in May of 2008 and its current (and only) version is 1.0. According to Dr. Zennaro, it is meant to provide interoperability and scalability.

Unfortunately, only the "core" program (batcp.cpp) was available. The core program includes several C++ header files (such as modes.h, types.h, signal.h, and process.h) which are needed for compilation. In addition, the batcp.cpp is coupled to the operating system (OS9) through the include of OS9def.h.

Base Evaluation Criteria

- ***Can it be ported to an Advanced Traffic Controller Architecture?***

  YES. It already runs on an Advanced Traffic Controller. It is not running under Linux but can be ported.

- ***Can NCHRP 3-66 concepts be incorporated?***

  YES, but not easily done. The application software is a single file, batcp.cpp.

- ***Can it be interfaced to CICAS?***

  YES. but not easily done. See above (single file problem).

- ***Can it be interfaced to ACS Lite?***

  YES. but not easily done. Same as above (single file problem).

- ***Does it have a default steady state?***

  YES.

- ***Is the software well documented?***

  YES. Yes, the comment to code ratio is 21%.

- ***Does it make use of exception handling?***

  NO. There are zero exception handlers. Errors are not caught.

- ***Is its McCabe Cyclomatic complexity of less than or equal to 12?***

  NO. The McCabe Cyclomatic complexity for the PATH software averaged 19.94.

- ***Is its McCabe Essential Cyclomatic complexity of less than or equal to 12?***

  NOT PERFORMED. The free tool used to perform metric analysis on the PATH program did not contain an essential cyclomatic function.

- ***Is the nesting level of loops reasonable?***

  YES . Hand inspection of the software showed no nested looping.

- ***Is the error diagnostic system comprehensive and straight forward?***

  NO. There is no diagnostic error system.

Additional Criteria from SCoP requirements

- ***Is the software Open Source?***

  YES. It is released under GPLv2

- ***Does it already contain or use NTCIP standards and protocols?***

  NO. But they can be added (again, not easily)

- ***Can it be integrated or is it integrated with the Texas Model?***

  YES. Before it is integrated into the Texas Model, each procedure in the program would have to be broken out into separate modules. The program does not make use of any object oriented attributes (inheritance, dynamic polymorphism, encapsulation, etc). The program would need to be re-designed and an interface to the Texas Model added.

- ***Can it be integrated with CORSIM?***

  YES. Last response applies here also.

Program 2 - Advanced Technologies, Incorporated Dual Redundant Base Software

Advanced Technologies, Incorporated developed a dual redundant base traffic intersection controller prototype as part of the Phase I effort for 06-FH1. This prototype could control an intersection and contained the railroad preemption concept of NCHRP 3-66. The prototype used a configuration file to "define" the intersection. The number of intersection approaches, traffic signals, lanes, crosswalks, etc are all modifiable without software constraints.

Base Evaluation Criteria

- ***Can it be ported to an Advanced Traffic Controller Architecture?***

  YES. The software developed by ATI can be ported to any software or hardware environment. It already runs under Linux and Windows.

- ***Can NCHRP 3-66 concepts be incorporated?***

  YES. The software developed by ATI has already incorporated the train preemption concept of NCHRP 3-66. The software is modular and object oriented. The employees of ATI who are working on this Phase II effort know the software well because they wrote it.

- ***Can it be interfaced to CICAS?***

  YES. Currently, there is not a formal specification for the interface between CICAS-V and the traffic controller. ATI's primary Ada95 software is object oriented. Adding an interface module is doable.

- ***Can it be interfaced to ACS Lite?***

  YES. Current implementations of ACS Lite use the NTCIP standard. ATI's Statement of Work states we will be NTCIP compliant.

- ***Does it have a default steady state?***

  YES. The current Phase I prototype includes steady state processing of an intersection. However it does not include transitioning to a default state upon detection of errors. Many different error detection techniques are included in the software. A detected error was displayed as a warning (miscompare and/or log message) but the prototype did not drop into the steady state.

- ***Is the software well documented?***

  YES. The primary software has a comment to code ratio of 22%. The comment to code ration of the secondary software is 23%. The code is easily understood. Both the secondary software and primary software were written using formal coding standards.

- ***Does it make use of exception handling?***

YES. There are **395** exception handlers in the primary software alone.

- ***Is its McCabe Cyclomatic complexity of less than or equal to 12?***

  YES. The McCabe Cyclomatic complexity measurements for ATI's auto-generated and redundant software are well under 12 (1.5 and 1.21 respectively).

- ***Is its McCabe Essential Cyclomatic complexity of less than or equal to 12?***

  YES. The McCabe Essential Complexity measurements for ATI's auto-generated and redundant software are well under 12 (1.5 and 2.27 respectively).

- ***Is the nesting level of loops reasonable?***

  YES. The analysis provided by our software case tools showed looping levels of less than 1 for both the auto-generated code and the ATI written code. This implies there is no delay induced because of nested loops.

- ***Is the error diagnostic system comprehensive and straight forward?***

  YES. Errors are handled by a central error handling system.

Additional Criteria from SCoP requirements

- ***Is the software Open Source?***

  YES. All software developed by ATI under this contract is by definition open source.

- ***Does it already contain or use NTCIP standards and protocols?***

  NO. NTCIP standards and protocols were not used in Phase I because we did not have to interface to outside systems.

- ***Can it be integrated or is it integrated with the Texas Model?***

  YES. ATI's primary Ada95 software is object oriented. Adding an interface module is simple.

- ***Can it be integrated with CORSIM?***

  YES. We will dynamically link interface libraries with CORSIM on a Windows based PC to allow CORSIM to drive our software (containing the TSCP) executing on the development board. This is the way the University of Idaho uses CORSIM.

Program 3 - The LA Traffic Control Signal Program

During the Phase II Proposal process, Ed Fok of the Federal Highway Authority tried to obtain a copy of the LA Traffic Control Signal Program (LA TSCP). At the time of the proposal writing the software was still not open source. ATI recently recontacted Ed Fok to determine the current status of the software. Mr. Fok said there is no plan to pursue making the LA TSCP program open source.

Program 4 - The InSync Adaptive Traffic Signal Controller

InSync is an adaptive traffic signal system developed by Rhythm Engineering©. The system is claimed to automatically optimize local traffic signals and coordinates signals along roadway arterials based on real-time traffic demand. The system utilizes cameras coupled with image processing of vehicles queues to adjust traffic signal timings in an adaptive fashion. The software is written is written in C++ language and it is a proprietary software (not open source system). The software is capable of communicating with NEMA and 2070 controllers alike (InSync Traffic-Adaptive System White Paper).

When a sensor of this system is placed in emergency/fog mode, InSync will access 4-weeks of historic green split data for specific TOD/DOW at that particular approach. This data is then normalized into a split time to place in the controller until the sensor is functioning again properly. If communications between networked intersections fail, individual processors will continue to perform local optimization functions.

Because this is not open source, we are not considering it for SCoP. However, we did look at it for its functionality.

Program 5 – MIT Intelligent Transportation System Program (MITSIMLab)

MIT's Intelligent Transportation Systems (ITS) program developed the MITSIM Lab to evaluate the impact of the alternative of the traffic management system design. According to the MIT Intelligent transportation systems web site, http://mit.edu/its/mitsimlab.html, the software incorporates a traffic management simulator (TMS) that can be used to evaluate:

1. Ramp control (ramp metering)

2. Freeway mainline control
    a. Lane control signals (LCS)
    b. Variable speed limit signs (VSLS)
    c. Portal signals at tunnel entrances (PS)

3. Intersection control

4. Variable Message signs (VMS)

5. In-vehicle route guidance

The software has an open source version that requires the Linux operation system. It calls for the "Redhat Linux 7.3 distribution" to compile the source code.

The files can be downloaded from the MIT's Intelligent Transportation System Program website at: http://mit.edu/its/MITSIMLabOSnew.html.

MITSIM was examined to see how other open source traffic programs are implemented. Just like the SCoP we are building, it has an online user's group -

http://tech.groups.yahoo.com/group/MITSIMLab/

The software is under configuration control and uses the same underlying tool used by ATI (the Concurrent Versioning System, or CVS) . Some files contain excellent headers with attributes -> Class Name, File Name, Class Type, Derivation, Layered, Friends, C++ Version, Calls to, and Library. Some do not. The software has detailed installation instructions and a 116 page users manual explaining how to use it. Like ATI's Phase I prototype, there is a way to simulate an eight-phase dual-ring traffic signal controller .

ATI considered using the Traffic Management System (TMS) part of this software for the core logic of our system. However, the software is extremely complex. The average McCabe complexity figure for the TMS C++ classes is **23.92**. That might be overlooked if the code was adequately commented. But, the comment to code ratio is only 10 percent. This is way less than both the PATH software and the ATI Phase I prototype software under consideration.

Program 6 – Software Controller Interface Device (CID) II:

The National Institute for Advanced Transportation Technology, University of Idaho, developed a real-time interface between a 170, 2070 and NEMA TS 1 and TS 2 traffic controllers and application software running on Windows 98, Windows ME or Windows 2000 (Brian Johnson et al, 2001). Listed below are applications of the software:

(1) A real-time interface between the TSIS/CORSIM traffic simulation running on a computer and 170, 2070 and NEMA TS1 and TS2 traffic controllers (hardware-in-the-loop simulation). The simulation runs with the real traffic controller instead of a generic model in the simulation, resulting in more realistic simulations that can be used to test traffic signal plans or train new engineers.

(2) A suitcase tester, in which a laptop computer and a CID are used to test the settings of a traffic controller and simulate full operation of the controller. This allows signal timing and progression to be checked under multiple scenarios prior to field installation.

(3) A hardware tester that can be used to test the operation of the CID periodically and test the continuity in the cables connecting the CID to the traffic controller.

In addition, the AASHTO Green book and the MUTCD were reviewed, both books only include suggestions for the logic to be used in the signal operation and the signal timing, but there was no mention of the software operating traffic signal controllers.

We may obtain this software during the port of our software to the Advanced Traffic Controller to aid in the testing of our software.

Summary of Research Findings (Pros and Cons)

- **LA TSCP** - Not open source, will not be used as base software for SCoP.

- **INSYNC ADAPTIVE TRAFFIC SIGNAL CONTROLLER** - Not open source, will not be used as base software for SCoP.

- **MITSIMLab –** Open source, however, not considered for core base logic because of the complexity of the software and lack of extensive comments that might overcome the complexity.

- **PATH SOFTWARE:**

  PROS:

  - The main benefit to the software developed by Dr. Marco Zennaro is it has been run on an Advanced Traffic Controller (Econolite Model 2070).
  - The software is well commented.
  - The software creates an ATCP sensor server, an ATCP actuator server and a "lookup" server.

  CONS:

  - The software uses hard-coded strings to specify paths.
  - There is no error handling.
  - It is not POSIX compliant.
  - It does not run under Linux but instead is tied to OS9.
  - All initialization logic is hard-coded.
  - Magic numbers are used.
  - Threading not used. Not interruptable (preemptable)
  - Most of the logic is contained in a single file that has an C++ extension but does not use C++ the way it is meant to be used.
  - No easy method to scale software.

- **INTERSECTION SOFTWARE DEVELOPED BY ATI:**

PROS:

- Safety Critical (Dual Redundant, Software Watchdog Timers, Protected Types, Exception Handling).
- The software is well commented.
- Auto-Generated from UML Design.
- Object Oriented techniques used (inheritance, encapsulation,and association)
- Tasking model allows easy incorporation of preemption.
- The software is not complex based upon metric analysis.
- The software is modular and can be easily interfaced to other systems.
- The software uses an initialization file to define an intersection. This makes scalability simple.
- The software is portable. The primary software already runs under Linux and Windows. It should also run under any POSIX compliant operating system.

CONS:

- The software uses terminology unfamiliar to subject matter experts.
- The dual-redundant approach, while it promotes safety, requires additional independent programmers for the redundant software.
- Headers are currently missing from the redundant implementation.

Recommendation

ATI spent a few weeks researching and analyzing existing open source base intersection control programs. Our Phase II proposal included the possibility of using the LA TSCP program, the California PATH by Marco Zennaro or the software developed by ATI during our Phase I effort. During this task, two of our subject matter experts, Dr. Essam Radwan and Mr. Noor Elmitiny, researched additional programs that could be applicable to our effort. The most promising of these was the TMS component of the MIT MITSIM program.

Based upon the ability to meet SCoP requirements and metric analysis, we believe the best forward approach is to enhance the software developed by ATI during Phase I with the following caveats derived from the Task 1 research:

1) Use Canny Quach's in-depth knowledge of the LA-TSCP software to ensure our software contains the same base functionality.

2) Use the interfaces developed by Dr. Marco Zennaro to guide us when porting our finished software to an Advanced Traffic Controller.

3) Use the MIT developed MITSIM documentation as a guide when developing our SCoP installation instructions and user's manual. This is some of the best open source installation instructions we have seen.

Advantages of Approach

1) ATI's principle investigator and other engineers wrote the software and are intimately familiar with it.
2) Unlike other software examined, the ATI code itself has safety mechanisms built in.
3) The ATI software is the least complex and best documented of all programs evaluated.
4) The ATI primary software is written in Ada95, the same language used in flight control systems, nuclear power plants, and other safety critical applications.
5) The ATI software is extremely portable and is POSIX compliant.

Appendix A - CICAS-V Interfacing

As part of the evaluation criteria, we studied current CICAS prototypes and concepts. We determined how easy it would be for each of the intersection control systems under evaluation to interface with these prototypes.

We examined the Intersection Collision Avoidance-Violation (ICAV) project completed by the Virginia Tech Transportation Institute. This system is designed to warn drivers if they are in danger of running a stop sign or a red light. We will interface to the ICAV signal-violation system.

If SCoP is to interface with the ICAV testbed, it must be able to interface with the infrastructure controller contained in the testbed. This controller was custom built to control the ICAV test intersection. The system uses wireless UDP packets to communicate between the signal (infrastructure) controller and the algorithm processor. SCoP should be able to provide an interface the custom built infrastructure controller OR the Algorithm Processor can access.

We looked at the overall architecture defined by the CAMP partners. It is the "Traffic Signal Interface" that SCoP must integrate with.

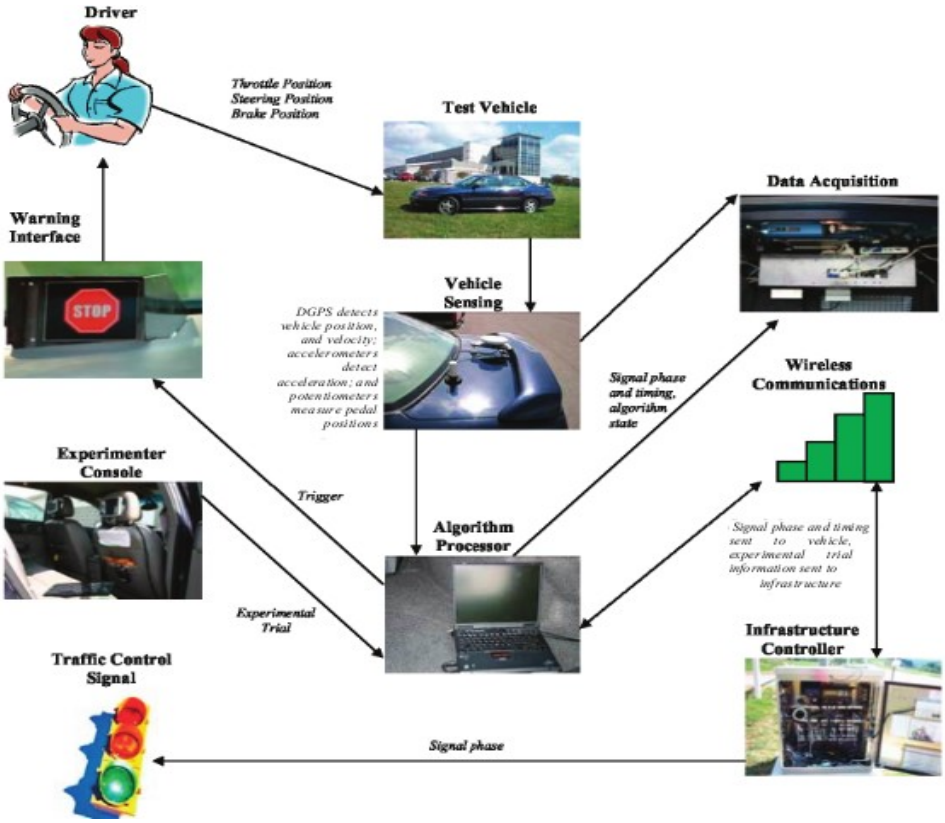


Figure 2: ICAV testbed

Appendix B - Texas Model Interfacing

As part of this task we re-installed the latest Linux version (6.0) of the Texas Model on one of our development systems. We were able to create a simple intersection. However, we were more interested in the software itself and how we plan to make use of it during our SCoP development.

The Texas Model is a combination of Java and FORTRAN based programs. If the model is to be rebuilt, the main FORTRAN programs must be modified depending upon the platform they will be executed on. There are scripts (such as compaq_run_removec_on_z_texas_src_to_f_ files_linux.bat) to perform this modification. For our purposes, we decided to stick with Linux and use the already modified Linux source files.

The Texas Model for Linux has the following components:

- `gdvsim: Enter or change specifications for intersection geometry or traffic, traffic control, duration of simulation process, or request creation of animated graphics file. This must be the first command if you are beginning a new problem.`
- `gdvpro: Process your specifications for geometry and traffic.`
- `geoplot : Optionally display/plot the intersection geometry and vehicle paths.`
- **`simpro: Executes the actual simulation using specifications from files created by the gdvsim and gdvpro commands.`**
- `dispre: Optionally prepare an animation graphics file for display.`
- `dispro: Optionally view the animated graphics.`
- `reptol: Perform replicate runs until the specified tolerance is achieved and create simstat.rep statistics file.`

ATI will replace the “simpro” portion of the Texas Model with our SCoP. This will allow the Texas Model to drive our system. **Our current simulation GUI will be totally replaced by the Texas Model.**

Appendix C - The ATI Dual Redundant TSCP Metrics and Dependencies

Ada95 Primary Software

```
Line metrics summed over 61 units
  all lines                          : 17018
  code lines                         : 10921
  comment lines                      : 2931
  end-of-line comments               : 111
  blank lines                        : 3166
Element metrics summed over 61 units
  all statements                     : 3573
  all declarations                   : 2720
  logical SLOC                       : 6293
 102 public types in 26 units including
    4 tagged types
    20 private types
    2 task types (there are more private task types)
 114 type declarations  in 32 units
 254 public subprograms in 30 units
 304 subprogram bodies  in 30 units
```

Figure 3: Ada 95 Primary Software Metrics

C++ Redundant Software

Lines	Code	Comment	Blank	Strs	AvgLen	Filename
85	39	26	20	3	5	./clock.cpp
271	99	116	56	1	10	./clock.h
54	19	22	13	1	10	./common.h
212	184	5	23	28	14	./entrance_lane.cpp
117	48	39	30	5	8	./entrance_lane.h
368	263	43	62	36	24	./intersection.cpp
39	24	0	15	1	14	./intersection.h
118	66	33	19	3	10	./interval.cpp
42	27	1	14	1	10	./interval.h
3	1	0	2	1	13	./lane_list.cpp
22	15	0	7	1	17	./lane_list.h
42	32	0	10	0	0	./list_node.h
87	63	2	22	14	17	./main.cpp
171	126	24	21	8	11	./ring.cpp
47	29	2	16	1	17	./ring.h
92	74	0	18	1	8	./road.cpp
46	31	2	13	3	15	./road.h
118	97	0	21	9	13	./road_group.cpp
46	34	0	12	4	11	./road_group.h
123	88	10	25	22	14	./socket.cpp

```
     52      34       6      12       0       0      ./socket.h
      0       0       0       0       0       0      ./train_track.cpp
     31      15       0      16       1      10      ./train_track.h
   2186    1408     331     447       1      10 Total
Percentage Code:         64%
Percentage Comment:      15%
Percentage Blank:        20%
Percentage Cmt/Code:     23%
Average Code/File:       61 lines
Blocks:                  99
Lengths (lines):         min: 0 max: 169
Strings:                 144
Size (bytes): total: 2296 average: 15
```

Figure 4: ATI Secondary Software Metrics

McCabe Metrics

These are the metrics produced by the GNAT Programming System toolset for ATI's Phase I primary Ada95 **core** intersection control software. The software makes use of inheritance and auto generation of code. Simulation/GUI code was not included in the analysis (it will be replaced in Phase II by the Texas Model and CORSIM).

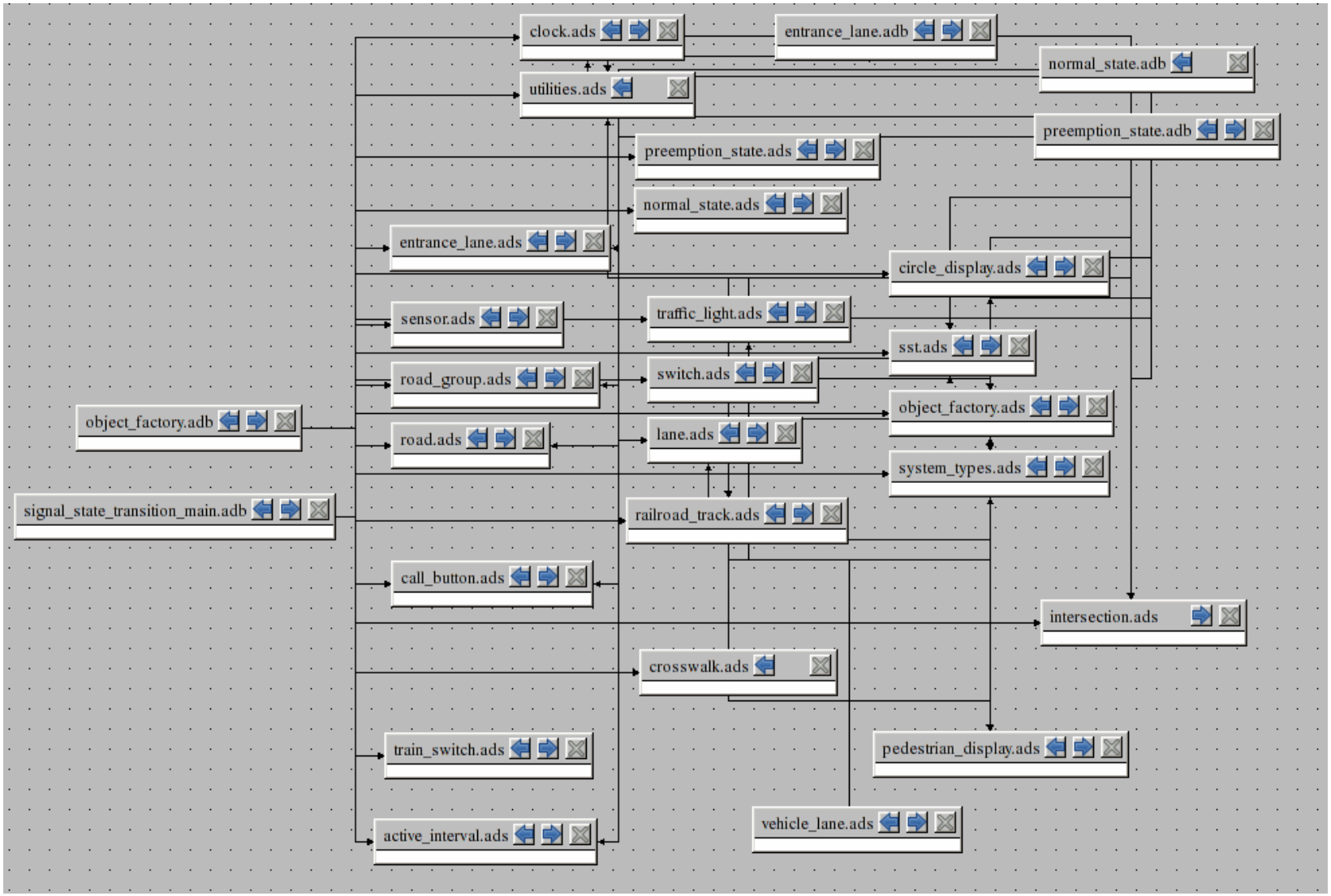
	McCabe and Other Metric Averages					
Code Type	**Statement Complexity**	**Short-Circuit Complexity**	**Cyclomatic Complexity**	**Essential Complexity**	**Max Loop Nesting**	**Extra Exit Points**
Inherited or Auto-Generated	1.5	0	1.5	1.5	0.33	0.17
ATI Phase I Core Software Methods (functions/procedures)	2.23	0.04	2.27	1.21	0.21	0.06

Figure 5: ATI McCabe Metric Summary

Dependency Graph (ATI PHASE I)

ATI's dependency graph would take several pages. The software IS NOT tightly coupled, however, it is object oriented and highly modularized and therefore contains many packages. The following is just a brief part of our overall graph.

Figure 6: Partial ATI Dependency Graph



Appendix D - PATH Software Metrics and Dependencies

C++ (C) Software Metrics

The GPS Toolset does not yet produce metrics for C++. These metrics were produced by the free tool cccc found on Sourceforge. The tool automatically highlights moderate problems in yellow and severe deficiencies in red.

LOC = Lines of Code
MVG = McCabe's Cyclomatic Number
COM = Comment Lines

Function prototype	*LOC*	*MVG*	*COM*
ATCP_receive(statusDataType *) declaration batcp.cpp:152 definition batcp.cpp:827	*66*	*26*	16
FIO_sensors_read(statusDataType *) declaration batcp.cpp:156 definition batcp.cpp:912	*48*	9	33
Set_Control(statusDataType *) definition batcp.cpp:1118	*84*	*13*	20
Set_Lights(statusDataType *, char, char) definition batcp.cpp:1060	*42*	7	17
Set_Up_Socket(statusDataType *, int) declaration batcp.cpp:150 definition batcp.cpp:283	*31*	7	18
Update_Control(statusDataType *) definition batcp.cpp:1272	**385**	**146**	18
bin_prnt_byte(int) declaration batcp.cpp:158 definition batcp.cpp:164	17	2	3
finalize_status_data(statusDataType *) declaration batcp.cpp:151 definition batcp.cpp:268	11	3	3
fio_init(statusDataType *) definition batcp.cpp:1222	*35*	6	8
initialize_status_data(statusDataType *, int, char) declaration batcp.cpp:149 definition batcp.cpp:184	*64*	4	12
main(int, char **) definition batcp.cpp:1727	*43*	5	38
parse_atcp_packet(packet *, statusDataType *) declaration batcp.cpp:155 definition batcp.cpp:335	**299**	**98**	57
send_actuator_list_packet(packet *, statusDataType *)	*35*	2	6

definition batcp.cpp:780			
send_sensor_data_packet(packet *, statusDataType *) declaration batcp.cpp:154 definition batcp.cpp:990	*53*	4	6
send_sensor_list_packet(packet *, statusDataType *) declaration batcp.cpp:153 definition batcp.cpp:693	*66*	5	11
set_control(statusDataType *) declaration batcp.cpp:157	1	0	0
sig_handler(int) declaration batcp.cpp:159 definition batcp.cpp:1823	11	2	2
Totals (TOT), Averages (AVG)	1291 (TOT)	19.94 (AVG)	268 (TOT)

Figure 7: PATH Software Metrics

Dependency Graph (PATH)

The dependency graph below shows the PATH Software to be a single file dependent upon the OS9 operating system. Unlike the ATI modular approach, the core logic is contained in a single program, batcpp.

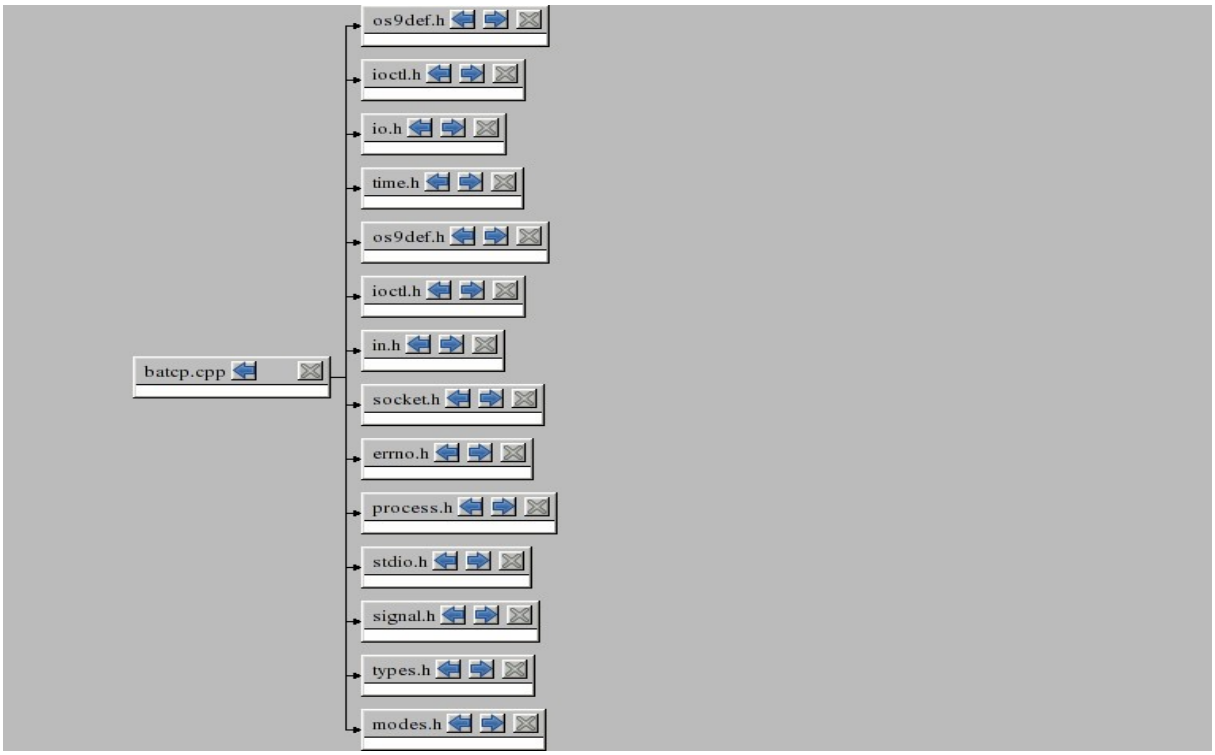


Figure 8: PATH Dependency Graph

Appendix E - MITSIMLab TMS Software Metrics and Dependencies

C++ (C) Software Metrics

The GPS Toolset does not yet produce metrics for C++. These metrics were produced by the free tool cccc found on Sourceforge.

LOC = Lines of Code
MVG = McCabe's Cyclomatic Number
COM = Comment Lines

Procedural Metrics Summary

Module Name	*LOC*	*MVG*	*COM*
TMS_ApidDetector	259	81	23
TMS_ApidParameters	33	3	8
TMS_ApidPrmTable	66	9	0
TMS_CmdArgsParser	6	0	5
TMS_Communicator	376	103	30
TMS_CtrlLogic	86	16	4
TMS_DetectedInc	105	31	8
TMS_Engine	*508*	79	84
TMS_Exception	18	2	0
TMS_FileManager	118	54	6
TMS_Guidance	109	16	6
TMS_Incident	196	39	32
TMS_IncidentDetector	100	27	2
TMS_IncidentDialog	74	11	19
TMS_Interface	22	3	1
TMS_Lane	2	0	4
TMS_Legend	7	0	7
TMS_Link	127	52	14
TMS_LusAction	215	59	9
TMS_McmasterDetector	233	89	8
TMS_McmasterParameters	50	4	8
TMS_McmasterPrmTable	62	7	0
TMS_Menu	49	0	9
TMS_Modeline	37	1	8

TMS_Network	2	0	5
TMS_OutputDialog	76	9	13
TMS_Parameter	42	14	1
TMS_PsAction	102	17	7
TMS_ResponsePhase	69	12	11
TMS_Segment	181	53	18
TMS_Sensor	203	54	19
TMS_SensorDataDialog	90	4	13
TMS_SetupDialog	131	7	11
TMS_Signal	8	1	0
TMS_Status	81	7	0
TMS_Symbols	5	0	8
TMS_TollBooth	5	0	5
TMS_VslsAction	220	45	16
Totals (TOT), Averages (AVG)	4073 (TOT)	23.92 (AVG)	422 (TOT)

Figure 9: MITSIMLab TMS Software Metrics

Appendix F - Phase II Task 1 SOW Description

From our Phase II Proposal (that is also our Statement of Work):

Objective: Evaluate Available Open Source Traffic Signal Control Programs.

ATI will evaluate existing open source traffic signal control programs for use in our system. This evaluation will be presented to our FHWA COTR and project advisory team.

Work Plan 1 (WP1): Evaluation of Existing Traffic Signal Control Programs.

ATI will attempt to examine the Los Angeles Traffic Signal Control program (LA TSCP) for our core intersection logic. The LA TSCP consists of three separate tasks that interact with each other. The TSCP's low-level control task interacts with input and output device buffers. The high level control task determines the system behavior. It is this task which our software will interface to. The task is written in C.

Until we have the source code or an IRS/IDD in our possession it is not possible to definitively state the interface method. Based on our past experiences, we will use one or more of the following methods:

- Direct Coupling: Directly linking our C++ and Ada95 software to the TCSP high level control task. Ada95 interface pragmas will be used for the primary software's interface. The secondary software can directly link the task in.

- Wrappers: We will develop wrappers for any interfaces exposed in the TSCP software.

- TCP/IP: We will use industry standard interface methods to loosely couple the LA TSCP software with our system. CORBA has been ruled out because it is not an ITS standard (Canny Quach).

We will examine the LA TSCP program for compliance to the ATC API standard. If possible, we will try to upgrade the source code to comply with the standard by adding translation objects at the interface level.

As part of this integration task we will examine the way the CALTRANS TSCP incorporated the LA TSCP as their baseline. This will help us to better understand interfacing to the LA TSCP.

The FHWA has been actively trying to have the LA TSCP classified as open source. As of this date, **there has not been a formal open source release of the LA TSCP**. This is why we are considering additional core logic options. During our research we found that on May 9th, 2003 the Los Angeles City Council adopted the following. We've highlighted the relevant parts:

Roll Call #5 - Motion (Bernson - Reyes) Adopted, Ayes (15)

03-0859 - TRANSPORTATION COMMITTEE REPORT relative to the City's Automated Traffic Surveillance and Control System (ATSAC) Distribution and Licensing

Agreements. Recommendations for Council action, SUBJECT TO THE APPROVAL OF THE MAYOR:

1. AUTHORIZE the General Manager, Department of Transportation(DOT), or designee on behalf of the City, to enter into a **software distribution agreement with the Federal Highway Administration to grant a non-exclusive and non-transferable right to distribute the Traffic Signal and Control Program (TSCP),** Adaptive Traffic Control System (ATCS), Smart Transit Priority Manager (STPM), and Transit Priority System (TPS) traffic signal control software, subject to the submission of a certification of compliance form or waiver request relative to Equal Benefits Ordinance Certification, acceptance of the Contractor Responsibility Ordinance described included in the Standard Provisions for City Contracts and approval of the City Attorney as to form and legality.

2. AUTHORIZE the General Manager, DOT, or designee, on behalf of the City, to **enter into a software licensing agreement with the McTrans Center of the University of Florida to grant an unlimited, non-exclusive right to list and sell licenses to use the TSCP**, ATCS, STPM and TPS traffic signal control software, subject to the submission of a certification of compliance form or waiver request relative to Equal Benefits Ordinance, acceptance of the Contractor Responsibility Ordinance described in the Standard Provisions for City Contracts and approval of the City Attorney as to form and legality.

3. AUTHORIZE the DOT to receive funds from the distribution of the traffic signal control software and deposit said funds into a new ATSAC Trust Fund Software Licensing Revenue Account, account number to be determined by the Controller.

4. AUTHORIZE the DOT to expend funds from the Software Licensing Revenue Account for the ongoing development of new applicable areas of traffic signal control technologies, transportation management strategies, staff training, professional development and other related investments.

5. DIRECT the DOT to report to the Transportation Committee regarding revenue received in and funds expended from the Software Licensing Revenue Account within one year of establishing the Account.

Fiscal Impact Statement: The City Administrative Officer reports that there is no impact to the General Fund. No funds are required to engage in these contracts. **DOT reports that the subject software will be distributed for amounts between $15,000 and $30,000** and DOT, on behalf of the City, would collect half the selling price. It is unknown how much software will be distributed; therefore, it is recommended that the Department report to the Transportation Committee regarding any proceeds

The LA Traffic Control Signal Program (LA TCSP) was not free nor open source as was expected during Phase I development. **As of now, it costs $1,500 per copy.** This is from the McTrans website:

TRAFFIC SIGNAL CONTROL PROGRAM (TSCP) SINGLE

TSCP7 allows the Model 2070 Controller to function as a two- through eight-phase, six-overlap, dual-ring traffic signal controller. The TSCP can operate as a stand-alone actuated or non-actuated controller, or as part of an interconnected system to either an ATSAC type traffic control system with second-by-second communications, or a hard-wire or modem field master.

Operating System: NA Level Of Support: NA **$1500** Product ID: TSCP.S

A more realistic plan is to examine the California Partners for Advanced Transit and Highways (PATH) control software to determine if it is suitable for our program. The PATH software was developed by Marco Zennaro as part of his graduate thesis for the University of California at Berkeley. According to his website, Zennaro.net, it was to be released in May of 2008. It is still not posted. We have tried to contact him though e-mail and his cell phone. We will ensure that Mr. Zennaro's software is included as part of our evaluation.

We will present our findings from the examination of these and any other available open source traffic signal control programs to the FHWA COTR and our project advisory team. They will assist in making a decision whether or not to use one of these programs **or develop our own**.

IF A PRE-EXISTING PROGRAM IS CHOSEN, ATI will evaluate associated design documents. ATI prefers to use the Unified Modeling Language (UML) to specify system/software requirements and design. If existing UML models are not available for the chosen program, then ATI will use the reverse engineering capabilities of Umbrello (our UML open source case tool of choice) to build class diagrams and associations from existing code. We will add the interfaces to the Texas Model, CORSIM, the MPC885 Application Development kit, and a generic ATC to this model. All interfaces developed by ATI will be modeled in UML, auto-generated, coded using strict standards, and tested. All requirements, design, and code for any interfaces will be kept under configuration control.

All of ATI's software will eventually be placed on Sourceforge, the largest open source data repository.

References

1. Gardinier, Mark, Romanowich, Donna M., “Signal State Transition Software SBIR 06-FH1 Final Report”, Advanced Technologies, Inc., San Diego, California, 2007

2. Johnson, Brian, Wells, Richard, Kyte, Michael, Bullock, Darcy, Li, Zhen, Zhou, Ying, Richards, James, Fisher, John, Remus, Jeremiah, Miller, Cody, Bordenkircher, Eugene, Duldulao, Richard, Jacob, Thomas, Gordon, Dan Lee, Matt *“CONTROLLER INTERFACE DEVICE (CID) II”*. National Institute for Advanced Transportation Technology, University of Idaho, 2001.

3. Lee, Suzanne E., Perez, Miguel A., Doerzaph, Zachary R., Stone, Scott R. , Neale, Vicki L., Brown , Sarah B., Knipling, Ronald R., G., Holbrook, Thomas , and Dingus , Thomas A. “Task 5 Final Report Intersection Collision Avoidance—Violation Project: Final Project Report ”, Virginia Tech Transportation Institute, Blacksburg, VA, 2007

4. Lombardo, Louis V., “Cooperative Intersection Collision Avoidance System (CICAS-V) to Avoid Violations at Stop Signs & Signals” , SAE Government/Industry Meeting , Washington, DC , 2006

5. McCabe, T.J., "A Complexity Measure," IEEE Transactions on Software Engineering, vol. 2, no. 4, pp. 308-320, 1976

6. Maile, Michael, “Cooperative Systems for Intersection Crash Avoidance Presentation”, 2008

7. Zennaro, Marco, “Berkeley Adaptive Traffic Control System Protocol”, http://www.zennaro.net/projects/transportation.php, 2008

8. *“TRAFFIC CONTROL SYSTEMS HANDBOOK”*, Federal Highway Administration Office of Transportation Management, October 2005.

9. MIT Intelligent Transportation Systems program web site, http://mit.edu/its/mitsimlab.html

10. InSync, Next Generation Adaptive Traffic Signal System, Dr. Reggie Chandra, Rhythm Engineering©. http://www.rhythmtraffic.com/