TMC Simulator

By Stuart Heater

Advisor: Dr. Clark Turner

Summer 2013

Abstract

The goal of this project was to design and implement a graphical user interface which simulates TriTech's VisiCad Inform computer-aided dispatch well enough for trainees to learn how to efficiently and accurately use the software in a risk-free environment. The simulator should also allow the training proctor to actively create new incidents during training in order to ensure that the trainees are able to respond properly. The structure of this project allowed me to work with both more- and less-experienced programmers, particularly those who are far more experienced with networking and hardware than myself. It was my first time taking the lead of part of development, and my first time developing a large-scale GUI. The interface involves approximately fifteen panels that can be accessed using hotkeys or a Photoshop-like toolbar and contain a variety of different information – traffic collision reports, physical descriptions for persons of interest, emergency vehicle maintenance reports and data, and more.

Contents

1 Introduction

Computer-aided dispatch – or CAD – is a means of dispatching mobile resources, such as field service technicians or ambulances, using computers. While the means of doing so varies from system to system, all forms of computer-aided dispatch provide dispatchers with interfaces and tools meant to handle requests for resources in the most efficient way possible. A CAD system typically provides assistance in most – if not all – aspects of a dispatch center, helping dispatchers assign units to events in the field, track the status (inactive, available, on-scene, etc) and location of those units, schedule unit maintenance, and record and log all calls for resources. Computer-aided dispatch systems are regularly used by public transit services and on-site technicians, as well as by police and emergency medical services. [5]

TriTech Software Systems' VisiCAD Inform (also known as Inform CAD) is a CAD software suite specially designed for use by emergency services personnel. From TriTech's VisiCAD Inform product page:

> ...helps communications center personnel manage a large amount of information—unit locations, unit statuses, pending and active calls, and other critical data—while serving as a voice of reassurance to callers and providing vital information that links police officers, firefighters, and paramedics. Inform CAD dispatch software captures, manages, and prioritizes mission-critical data to enable rapid decisions in situations where every second counts.

VisiCAD Inform was designed with mission-critical applications in mind, and even provides the System Administrator with alerts should the system's performance decay to the point where it could place lives at risk. [8] However, TriTech does not provide any specialized software for training emergency dispatchers in the use of VisiCAD Inform, despite the suite being used in situations where a few minutes can be the difference between life and death. [7] Because of this, CalTrans contracted Cal Poly Corporation (CPC) to create a simulator room and training program to allow dispatchers to practice using VisiCAD Inform in a risk-free environment.

VisiCAD Inform is sold like a commercial product – the source code is not made available to the purchaser, only the finished product. The goal of the

project is to reverse-engineer a VisiCAD Inform simulator that accurately represents its real-world analogue well enough to provide dispatchers with a means to practice using the system in an environment where a mistake does not put lives at risk. The simulator must represent the real-world version of VisiCAD Inform well enough that the individual running the simulator - referred to as the 'proctor' - is able to judge whether or not the trainees are skilled enough in its use to be ethically permitted to do so in situations where mistakes could result in the loss of life.

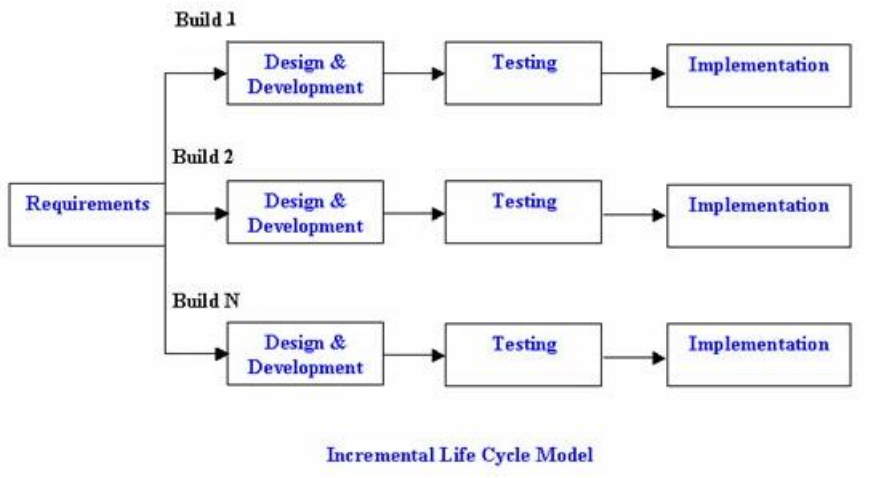
The simulator's proctor must be able to load the system with a variety of different dispatch incidents (from traffic jams to overturned semi trucks, all the way to toxic chemical spills and terrorist attacks), in order to assess how well the trainees are able to respond to both low- and high-priority situations, as well as their ability to deal with various levels of incoming calls. The personnel responsible for maintaining the simulator room and running the training sessions are members of the development team, and will act as supervisors as well as de facto customers, since they will be able to judge whether or not the simulator is ready to be put into use.

2 Process Model

2.1 The Incremental Model

The Incremental Model divides the software system into smaller, more manageable iterations called cycles or builds. In each cycle, several smaller modules go through all of the phases of the Waterfall Model's software life cycle (requirements, design, implementation, testing). Each module may undergo several iterations, until it is fully functional and ready to be used in the completed system. [9]

When working *incrementally*, priority is first given to quality, *then* to functionality. By dividing the software system into smaller modules and only focusing on developing a small number of them at once, each module can be given more attention than if they were all being developed simultaneously.



Incremental Life Cycle Model

2.1.1 Advantages

- Speed - Generates working versions of software early in the software life cycle.
- Flexibility - Adding or removing non-essential features is relatively low-cost compared to other process models. [9]
- Ease of Testing - It is much simpler to test and debug smaller modules that only encapsulate a very specific section of functionality.
- Adaptability - Simple to adapt to changing requirements because of fast iteration cycles. [3]
- Customer Feedback - Each build produces something that can be presented to the customer, which actively engages them in the development process and helps further refine requirements. [6]

2.1.2 Disadvantages

- Front-loaded - Large amount of time must be spent on the initial design/documentation before the system can be successfully dividing into smaller modules. [9]
- Refactoring - Substantial redesign may be necessary between iterations. [3]

- Customer-dependent - Depends on communication with the customer in order to develop the best possible understanding of the system. [4, 6]
- Size-limited - Not suitable for very large projects or large teams. [3]

2.1.3 When is the Iterative Model Appropriate?

- The customer(s) and/or end users are easy to contact with and are willing to work with developers. [3]
- The system is clearly designed or easily understood. [9]
- The development team is relatively small, or the project is small enough that refactoring is not unnecessarily costly. [3]

2.1.4 Conclusion

The TMC simulator development team only consists of four members, and each team member has a relatively isolated section of the project to work on. Because of this, back-end development can take place on previously completed GUI elements as new GUI elements are in-progress, resulting in a 'staggered Waterfall' style of development. The simplest (and most fundamental) parts of the system can be developed first, and then integrated into the larger, more complicated parts of the system (such as the Vehicle Information window that has a button which opens the License Plate Information window) once they have already been thoroughly designed and tested.

As Dalbey states, an iterative development method is most effective when the customers will have a large amount of involvement with the process. This is definitely the case with the TMC Simulator; the primary proctor of the dispatch training is one of the managers of the development team, so the developers will have easy access to one of the most experienced users of the previous version of the system.

One of the most appropriate times to use an iterative process - as defined by an ISTQB (International Software Testing Qualifications Board) certification preparation website - is when 'the requirements of the system are clearly defined and understood.' [9] Because VisiCAD inform already exists, the real-life system itself serves as a fairly firm definition of everything that

is necessary for the simulator to accurately represent it.

The specific situation regarding the development of the TMC Simulator minimizes the disadvantages of using an iterative development model, as well as fulfills several of the requirements that makes such a model most effective. For these reasons, I believe that an iterative model would be appropriate for the development of the TMC Simulator.

3 Requirements

3.1 Functional Requirements

3.1.1 Activity Log Viewer

Form that allows user to view various information about the goings-on of a particular agency, jurisdiction, division, etc

- Date-Time Period – Allows the user to view all activity that occurred within a specific date-time range.
- Show Units by Abbreviated Name – Toggle abbreviated names for jurisdictions, divisions, battalions, stations, and units.
- Select/Unselect All – Select or deselect every entry associated with a particular agency for the chosen date-time range.
- Agency – Dropdown menu containing all CalTrans and CalTrans-associated agencies that use VisiCAD to report/log activites.
- Jurisdictions – Table of jurisdictions associated with a particular agency.
  - Select/Unselect All – Select or deselect every jurisdiction associated with a particular agency.
  - Show AVL Data – Toggle AVL (Automatic Vehicle Location) data on/off
- Divisions – Table of divisions associated with a particular jurisdiction.
  - Select/Unselect All – Select or deselect every division currently in the Divisions table.

- Battalions – Table of battalions associated with a particular division.
  - Select/Unselect All – Select or deselect every battalion currently in the Battalions table.
  - Show Radio Log Data – Toggle a battalion's radio logs on/off.
- Stations - Table of all stations associated with a particular jurisdiction.
  - Select/Unselect All – Select or deselect every station currently in the Stations table.
- Units – Table of all units associated with a particular agency, jurisdiction, division, battalion, or station.
  - Select/Unselect All – Select or deselect every unit currently in the Units table
- Activity Log – Table of all activities logged based on the date-time period/agency/jurisdictions/divisions/battalions/stations/units selected.
  - Each entry contains the date/time, the type of incident, the unit number of the vehicle involved, the activity undertaken, location, and a section for comments.
- Refresh button – Refresh the activities currently listed in the Activity Log based on the parameters selected in the other tables.
- Print button - Print current Activity Log.
- Exit button - Exit the Activity Log Viewer

3.1.2 BOLO Entry Form

Form that allows the user to create a BOLO (Be On the LookOut) alert, also known as an APB (All Points Bulletin).

- General Info – Section containing the general information associated with every BOLO.
  - Case Number - The case number the BOLO is associated with.

  - BOLO Type - The type of BOLO issued.
  - Request Date/Time - The date and time at which the BOLO alert was requested.
  - Requested By - Who initiated the BOLO request.
  - Unit – The ID number of the unit who requested the BOLO.
  - Other Info – Miscellaneous information that may be necessary.
  - Expires - The date/time at which the BOLO expires.
  - Expired checkbox – Indicates whether or not the BOLO is an expired one.
  - Incident Number – Incident number associated with the BOLO.
  - Entry Date/Time – The date/time at which the BOLO was entered into the system.
  - Entered By – The dispatcher who entered the BOLO request into the system.

- Person of Interest - Section containing descriptive information about a specific person for agencies to 'be on the lookout' for.
  - Last Name – Last name of the person of interest.
  - First Name – First name of the person of interest.
  - Alias – Any known aliases associated with the person of interest, if applicable.
  - DL# – Driver's license number of the person of interest, if applicable.
  - State – The state where the driver's license was issued, if applicable.
  - Race – Ethnicity of the person of interest.
  - Gender – Gender of the person of interest.
  - DOB – Date of birth of the person of interest.
  - Age – Age of the person of interest.
  - Height – Height of the person of interest. Weight – Weight of the person of interest.

  - Hair – Hair color of the person of interest.
  - Eyes – Eye color of the person of interest.

- Vehicle of Interest - Section containing descriptive information about a specific vehicle for agencies to 'be on the lookout' for.
  - Year – The vehicle's year of manufacture.
  - Make – Name of the manufacturer that produced the vehicle.
  - Model – Name of the specific type of vehicle.
  - Style – Descriptive information about the vehicle (2-door, 4-door, sedan, SUV, hatchback, etc).
  - Color – Primary paint color of the vehicle of interest.
  - VIN – VIN (Vehicle Identification Number) of the vehicle of interest.
  - License Plate Info – Section containing information about the vehicle of interest's license plate, if applicable.
    * Plate – License plate number.
    * State – The state the license plate was issued in.
    * Expires – Month/year in which the vehicle's license plate expires.

- Table containing a log of BOLOs that have been issued
  - Date – Date the BOLO was issued.
  - Time – Time the BOLO was issued.
  - User – User who issued the BOLO.
  - Comment – Additional information about the BOLO.
  - Comment Field – Text field where the user can enter comments about a BOLO.
  - Add – Add a BOLO to the BOLO Log.
  - Cancel – Do not add BOLO to the BOLO Log.
  - Save – Save a BOLO Log.

- Add - Add a current BOLO to list of active BOLOs (unless expired).

- Cancel - Clear all fields.
- Save - Save in-progress BOLO request.

3.1.3 Cardfile

Form that contains contact information for companies, agencies, and services used by dispatchers.

- Agency/Service Selector – Collection of tabs that allows the user to select the particular service required in order to get further contact information/activity logs
  - Agency/Service Tabs...
    - Federal Agencies
    - Ranches/Livestock
    - Fire/EMS
    - Jails
    - CHP Offices
    - State Agencies/Facilities
    - Government Officials
    - Public Transportation
    - GG Other
    - MY Misc
    - VL Misc
    - Coastal Division Units
    - Police/Sheriff/Coroner
    - Courts
    - Gate Access Codes
    - VT Call Signs
    - SLCC Employees
    - SL County Services
    - SL Resources
    - Truck/Tire Repair
    - MCC Employees

    * Ranges
    * Hotlines
    * Hwy Patrol OOS
    * Parks/Recreation
    * Shelters
    * Utilities
    * Animal Control
    * Airports
    * Credit Cards
    * GG Crisis Shelters
    * Hospitals/Med Centers
    * Tow Companies
    * CalTrans
    * County Roads
- Agency Contact Info
    * Name – Name of the agency.
    * Address – Address of the agency.
    * City – City in which the service is located.
    * State – State in which the service is located.
    * Zip – Zip code in which the service is located.
    * Phone #1 – Service's primary phone number.
    * Phone #2 – Service's secondary phone number (if applicable).
    * Fax Number – Service's fax number (if applicable).
    * Comments – Log of comments for a particular agency/service.
        · Date – Date comment was entered.
        · Time – Time comment was entered.
        · Initials – Initials of the commenter.
        · Comment
        · Comment Field – Text field where user enters a comment.
    * Add – Add a comment to an agency's/service's comment log.
    * Cancel – Clear the comment field.
    * Save – Save the current comment.

        * Delete – Delete the currently selected comment.

- Add – Add a new service/agency to the Cardfile.
- Delete – Delete a service/agency from the Cardfile.
- Cancel – Clear the Agency Contact Info fields.
- Save – Save the current agency/service contact information.
- Print – Print the currently selected agency's/service's contact information.
- Exit – Exit the Cardfile.

3.1.4 Cardfile Search

Search form that allows a user to search the entries in the Cardfile using any of the criteria of a Cardfile entry.

3.1.5 Search

Form that allows the user to search through the database of logged dispatch incidents.

- Basic search – Simplified search form that allows users who aren't experienced with databases to perform incident searches based on more familiar criteria
  - Priority – Narrow the search based on the priority levels assigned to incidents.
  - Nature/Problem – Narrow the search based on the nature of the incident.
  - Response Location – Narrow the search based on the location that services were dispatched to.
  - City – Narrow the search based on the city in which the incident occurred.
  - Address – Narrow the search based on the primary address associated with the incident.

- Apt # – Narrow the search based on the primary apartment number associated with the incident (if applicable).
- Bldg # – Narrow the search based on the primary apartment building number associated with the incident (if applicable).
- State – Narrow the search based on the state in which the incident occurred.
- Zip Code – Narrow the search based on the zip code in which the incident occurred.
- Cross Street – Narrow the search based on the nearest cross street to the address at which the incident occurred.
- Map Coordinates – Narrow the search based on GPS map coordinates.
- Unit – Narrow the search based on the unit dispatched to the incident.
- Alarm Level – Narrow the search based on the alarm level assigned to the incident.
- Incident Number – Search for a particular incident number.
- Base Response Number – Narrow the search based on the number of the base that responded to the incident.
- Case Number – Search for a an incident associated with a particular case number.
- Incident Type – Narrow the search based on the type of incident.
- Jurisdiction – Narrow the search based on the jurisdiction in which the incident occurred.
- Division – Narrow the search based on the division that dealt with the incident.
- Battalion – Narrow the search based on the specific battalion that dealt with the incident.
- Response Area – Narrow the search based on the response area.
- Response Plan – Narrow the search based on the particular response plan used when responding to the incident.
- Command Channel – Narrow the search based on the command channel that was used when responding to the incident.

  - Primary/Alternat Tac Channel – Narrow the search based on the tactical channel utilized during the response.
  - Sector – Narrow the search based on the sector in which the incident occurred.
  - Optional Search Criteria – Fields that allow a user to further narrow the search using criteria that is not applicable to all incidents.
    * Call Backs
      · Caller Type – Narrow the search by the type of caller (business, residential, etc)
      · Caller Name – Narrow the search by the name of the person who made the call.
      · Called from Location – Narrow the search by the location the call was made from.
      · Called from Address – Narrow the search by the specific address the call was made from.
      · Phone – Narrow the search by the phone number used to make the call.
    * Transportation
      · Location – Narrow the search by the location of a type of transportation (bus, train, etc).
      · Address – Narrow the search by the specific address of a type of transportation (bus, train, etc).
    * Time Stamps - Narrow the search by the time stamps associated with each incident.

- Advanced search – Search form that allows users to interact more directly with a database, performing searches based on actual database fields/tables.

3.1.6 Incident Editor

Field that allows a user to create a new incident or edit one that has been previously created.

- Incidents – Table containing information associated with incidents.
  - Date – Date of the incident.

  - Type – Type of incident.
  - Address – Address of the incident.
  - Location – Location of the incident.
  - City – City the incident occurred in.
  - Incident # – Identifying number of the incident.
  - Units – Units dispatched to/associated with the incident.
  - Dispo – Codes for final dispositions (gone on arrival, unable to gain entrance, arrest made [particular type of arrest], domestic incident, patient removed to hospital, etc).

- Search Criteria – Fields that populate the Incidents table based on specific criteria.
  - Agency Type – Narrow the incidents displayed based on the agency (police, EMS, etc).
  - Current Database – Only show incidents stored in a particular database.
  - From.../To... – Only show incidents that occurred between a specific calendar range.

- Reopen – Reopen an incident.
- Duplicate Cell – Duplicate the currently selected cell(s) in the Incident table.
- Search – Refresh the Incidents table based on the specified criteria.
- Print – Print the information currently populating the Incidents table.
- View – Open the Incident Info window, displaying more detailed information about the currently selected incident.
- Refresh – Refresh the Incidents table without altering any search criteria (will display any newly-added incidents that match the currently selected search criteria).
- Exit – Close the Incident Editor.

3.1.7 Incident Info

Form containing highly detailed information about a particular incident.

- Incident Number – The identification number of the incident.
- Call Initiated – The time at which the incident was first created.
- Call Taken – The time at which an available unit was assigned/dispatched to the incident.
- Time in Q – The amount of time that the incident has been in the incident queue.
- Last Updated – The most recent time the status/information of the incident was updated.
- Total Elapsed Time – The total amount of time that the incident has existed for. (From the initial call to the present time [if an active incident] or when the incident was resolved [closed incident]).
- Status Bar – Display that allows a dispatcher to easily and quickly determine the current status of an incident (In queue, assigned, en route, inactive, complete, etc).
- Address – The primary address associated with the incident.
- City – The city in which the incident occurred.
- Apt – The apartment number at which the incident occurred (if applicable).
- Building – The apartment building/apartment building number at which the incident occurred (if applicable).
- Phone – The phone number associated with the incident (if applicable).
- Ext – The phone extension associated with the incident (if applicable).
- Cross Street – The cross-street nearest to the address of the incident.
- Map Info – Map coordinates or other map information to aid dispatched units in finding the location at which the incident occurred.

- Resp Area – The response area in which the incident occurred.
- Division – The dispatch division in which the incident occurred.
- Sector/Sector Code – The sector in which the incident occurred and its associated dispatch code.
- Caller Type – Type of caller who reported the incident.
- Caller Name – The name of the caller who reported the incident.
- Problem/Problem Code – The type of problem reported and its associated dispatch code.
- Priority Desc – The priority descriptor of the incident.
- Primary Unit – The call sign of the primary unit responding to the incident.
- Backup Units – The call signs of any backup units called in to assist with the incident.
- Incident Comments – Table containing additional comments about the incident.
  - Date/Time – Date and time the comment was made.
  - Disp – The dispatcher who made the comment.
  - Comment – The comment itself.
- Print - Print the currently displayed incident.

3.1.8 Incident Supplement - Person Form

Form containing highly detailed information about a person of interest associated with a particular incident.

- Incident No – The identification number of the incident the person of interest is associated.
- Involvement Type – How the person of interest is/was involved with the incident in question.

- Last Name – Last name of the person of interest.
- First Name – First name of the person of interest.
- Middle Name – Middle name of the person of interest.
- DOB – Date of birth of the person of interest.
- Age – Apparent age of the person of interest.
  - Age Min - Lowest feasible apparent age.
  - Age Max - Highest feasible apparent age.
- Weight - Apparent weight of the person of interest.
  - Weight Min - Lowest feasible apparent weight.
  - Weight Max - Highest feasible apparent weight.
- Height - Apparent height of the person of interest.
  - Height Min - Lowest feasible apparent height.
  - Height Max - Lowest feasible apparent weight.
- Race – Apparent race of the person of interest.
- Gender – Apparent gender of the person of interest.
- Build – Physical build of the person of interest.
- Hair – Hair color of the person of interest.
- Facial – Any distinguishing facial characteristics of the person of interest.
- OLN – Person of interest's driver's license number (Operator's License Number).
- OLS – The state in which the person of interest's driver's license was issued (Operator's License State).
- SSN – Person of interest's social security number.
- Shirt – Color of the shirt the person of interest was last seen wearing.

- Pants – Color of the pants the person of interest was last seen wearing.
- Shoes – Color of the pants the person of interest was last seen wearing.
- Hat – Color of the hat the person of interest was last seen wearing.
- Glasses – Color of glasses the person of interest was last seen wearing.
- Jacket – Color of the jacket the person of interest was last seen wearing.
- Flight Direction – Last direction the person of interest was seen traveling (if applicable).
- Flight Mode – Last mode of transportation the person of interest was seen using.
- Weapon – Any weapons that the person of interest is known to be carrying.
- Place of Residence – Contains information about person of interest's primary known residence.
  - Street – Address of residence.
  - Apt – Apartment number (if applicable)
  - City – City of residence.
  - State – State of residence.
  - Zip – Zip code in which residence is located.
  - Phone – Primary phone number of person of interest.
- Comments – Text fields for any comments regarding the person of interest.
- Cancel – Clear fields and close Person Supplement Form.
- Save – Save information currently entered in Person Supplement Form.
- Save with Records Check – Save information and check the description against agency records.

3.1.9 Incident Viewer

Form containing detailed information about a particular incident, similar to the Incident Editor. However, also links incidents to persons/vehicles of interest (Incident Supplement – Person Form, BOLO Entry Form, and Vehicle Information Entry form), maintenance requests for response vehicles (Rotation Service Request form), license plate information (License Plate Information form), transport information, call backs, unit assignments, and more.

- Map Loc – Primary map location of the incident.
- Apt – Primary apartment number associated with the incident (if applicable).
- Location – Primary street address associated with the incident (if applicable).
- Cross St – Cross-street nearest to the primary street address (if applicable).
- City – City in which the incident occurred.
- County – County in which the incident occurred.
- RP – Reporting Party. The person responsible for reporting the incident.
- Phone – Primary phone number associated with the incident.
- Ext – Phone extension for the phone number associated with the incident (if applicable).
- ALI – Automatic Location Identification. Provides for an address display of the subscriber calling 911. Includes the subscriber's address, community, state, type of service and if a business, the name of the business.
- RP Type – The type of reporting party.
- Sector – Response sector in which the incident took place.

- Beat – Police beat in which the incident took place.
- Agency – The agency responsible for responding to the incident.
- Type Code – Dispatch code indicating the specific type of incident.
- Pri – Priority level of the incident.
- MEDIA – The specific news media to which a comment was made.
- Confidential Comment – Indicates whether or not the currently selected comment is confidential or not.
- Comments – Field displaying comments made to the media about an incident.
- Hub Xter – Request a transfer of resources from a particular dispatch hub (one not normally responsible for responding to an incident in the area)
- Add FSP/CHP – Assign Freeway Service Patrol (FSP) or California Highway Patrol (CHP) to the incident.
- ANI/ALI – Add an ALI (Automatic Location Identification) or ANI (Automatic Number Identification) to the incident.
- Unit Rec – Recommand unit.
- Update Map Loc – Commit any updates made to the incident's map location.
- Exit/Send – Send incident to the incident queue and close the Incident Viewer.
- Send to Q – Send incident to the incident queue.
- Person of Interest button – Open the Incident Supplement - Person form in order to associate a person of interest with the incident.
- Vehicle of Interest button – Open the Vehicle Information Entry form in order to associate a vehicle of interest with the incident.

- Tow Rotation button – Open the Rotation Service Request form in order to request a tow or other rotation services.
- LAW – Additional information for/about law enforcement dispatched to the incident.
- FIRE – Additional information for/about firefighters dispatched to the incident.
- EMS – Additional information for/about emergency medical services dispatched to the incident.
- Call Backs – Table containing call-back information.
  - Date – Date the call-back was made.
  - Time – Time the call-back was made.
  - Initial – When the initial call was made.
  - Comment – Comment about a specific call-back.
- Assignments - Table containing information about unit assignments.
  - Unit Assignment table
    - Unit – Call-sign/unit number of the unit in question.
    - Alarm Level – Alarm level of the incident the unit in question is responding to.
    - Type – Type of assignment the unit is responding to.
    - Status – Current status of the unit.
    - Responding From – Where the unit is responding from.
    - Elapsed – How long the unit has been on an assignment.
    - Response Number – Identification number of a particular response.
  - Available Units table
    - Resource - The type of unit available.
    - Capability - What the capabilities of the available unit are.
  - Recommand – Recommand an available unit to an incident.

- Activities - Table containing information about current dispatch activities.
  - Date/Time – Date and time of a particular activity associated with an incident.
  - Vehicle/Unit – Vehicle/Unit involved in activity.
  - Activity – What the activity being undertaken actually is.
  - Location – Location where the activity is to take place/is taking place/took place.
  - Comment – Additional comments about the activity.
  - Dispatch – The dispatch center that is responsible for creating/dealing with the activity.
- Additional Information - Contains expanded information about an incident.
  - Incident Number – Identification number of the incident.
  - Incident Type – Full description of the incident type (expanded information based on type code).
  - Call Taken – When call that initiated the incident was received.
  - Call Taker Phone Ext – Phone extension of the dispatcher who originally took the call.
  - Call Status – Current status of the call (received, in queue, complete, etc)
  - Alarm Level – Alarm level of the incident.
  - CC/Jurisdiction – The call center that received the call/the jurisdiction in which the incident occurred.
  - Area Ofc/Division – Area of command/division in which the incident occurred.
  - Area/Battalion – The battalion responsible for the area in which the incident occurred.
  - Beat/Response Area – The specific police beat/unit response area in which the incident occurred.

  - Response Plan – Plan for responding to a particular type of incident.
  - Rotation Provider Area – Which provider of unit rotation/replacement/maintenance is responsible for the area in which the incident occurred.

- Comments/Notes – Table containing additional miscellaneous information associated with an incident.
- Edit Log – Table containing information about any edits made to an incident.
  - Date – When the edit was made.
  - Edits/Updates – What actually comprised the edit to the incident.
  - Reason – Why the edit was made.
  - Changes By – Who made the edit.
  - Terminal – Computer terminal from which the edit was made.
- Times – Times associated with particular events involved with an incident.
  - Times table - Table containing unit response times.
    * Unit – Call-sign/unit number of the unit in question.
    * Alarm Level – Alarm level of the unit's assignment.
    * Assigned – When the unit was given an assignment.
    * Enroute – When the unit was designated as 'en route.'
    * Staged – When the unit
    * Arrival – When the unit arrived on the scene.
    * Access – When the unit gained access to the scene (if applicable).
    * Depart – When the unit departed the scene.
    * At Dest. – How long the unit spent at its current destination.
    * Status 5 – How long a unit spent in a Code 5 (indicates that an officer is observing a location for possible criminal activity and advises other units to avoid the area).
    * Available – When the unit became available for dispatch again.
    * Resp Num – Identification number of the response in question.

  - Ring - When the incident was initially called in.
  - In-Queue - When the incident entered the queue.
  - All Available - When all units dispatched to the incident were re-designated as available once again.
  - Call Closed - When the incident was closed/completed.
  - Page Times - Log all Time information of an incident.

- Transport Info - information about scheduled transport of persons/objects/vehicles/etc.
  - Name – Name of person(s) being transported.
  - Transport to City – City to which the person(s) are being transported.
  - Transport to Location – Location to which the person(s) are being transported
  - Address button – Open a map from which an address can be selected.
  - Address – Address to which the person(s) are being transported.
  - Transport to State – State to which the person(s) are being transported.
  - Zip – Zip code within the state to which the person(s) are being transported.
  - Room, Apt, etc – Additional information about the specific location to which the person(s) are being transported, including room number, apartment number, etc.
  - Building # - Building number to which the person(s) are being transported.
  - Mode tab – Information about how the transportation should be carried out.
    * Transport Protocol... - Specific transport protocols that must be observed.
    * Transport Priority... - Priority of the transportation in question.
    * Assisted By... - Any departments/services that will be assisting in the transportation.

  - Times tab - Information about the timetables of the transportation.
  - Odometer tab - Odometer display of transport vehicle before/after the transportation.

- User Data - Allows users to apply time stamps and descriptions to data that they have entered.
  - Time Stamps table - Table containing information about time stamps associated with an incident.
    * Date - Date of a particular time stamp.
    * Time - Time of a particular time stamp.
  - Date field - Text field where a user can enter a date to be stamped.
  - Time field - Text field where a user can enter a time to be stamped.
  - Stamp - Commit a time stamp to the Time Stamps table.
  - Data Fields table - Table containing information about various data fields associated with an incident.
    * Data Field Description - Description of the data field in question.
    * Data - The actual data entered in said data field.

- Attachments - Allows a user to associate data files with an incident.
  - Attachment table - Table containing information about all files attached to an incident.
    * Data - Preview of the data contained in an attachment.
    * Attachment Type - The type of file attached.
    * Size - The size of a file attachment.
    * Description - User-dictated description of the attached file.
  - File Name – Allows user to indicate the name of the file to be attached.
  - Browse – Opens a browser window to allow the user to search the file system for the file to be attached.
  - Description – Text field for entering the description of a file attachment.

- Add – Attach the selected file.
- Delete – Delete the currently selected file attachment.
- Cancel – Clear 'File Name'/'Description' fields.
- Save – Save a file attachment in progress.

3.1.10 License Plate Information

Form containing information about the license plate of a particular vehicle.

- Plate number – License plate number.
- State – State in which the license plate was issued.
- Year – The year the license plate expires.
- Type – Type of license plate issued (commercial, personal, etc).

3.1.11 Vehicle Information Entry

Form containing information about a vehicle of interest that is/was involved with a particular incident.

- Incident No – Incident number that the vehicle of interest is associated with.
- Involvement Type – How the vehicle of interest is involved with the incident.
- Towed By – Company that towed the vehicle of interest (if applicable).
- Make Year – The vehicle's year of manufacture.
- Make – Name of the manufacturer that produced the vehicle.
- Model – Name of the specific type of vehicle.
- – Descriptive information about the vehicle (2-door, 4-door, sedan, SUV, hatchback, etc).
- Color 1 – Primary color of the vehicle.

- Color 2 – Secondary color of the vehicle (if applicable).
- License Plate No – License plate number of the vechile.
- License Plate State – The state the license plate was issued in.
- License Plate Year – Year the license plate expires.
- License Plate Type – Type of license plate issued (commercial, personal, etc).
- VIN – Vehicle Identification Number of the vehicle.
- Characteristics – Text field for any additional descriptive information about the vehicle in question.
- Comments – Text field for any additional comments the user may have.

3.1.12 Rotation Service Request

Form that allows a user to request rotation service on a response vehicle.

- Pending/Active Requests tabs - Contains information about service requests that are scheduled to happen or in progress.
  - Entered – Date the request was entered into the system.
  - Incident Number – Incident number that necessitated the service request.
  - Unit – Unit number of the vehicle in question.
  - Rotation Category – What type of service the vehicle in question requires.
  - Location – Where the vehicle in question is currently located.
- Rotation Provider Area – Areas covered by Rotation Providers.
- Rotation Category – The type of service necessary.
- Location – Location of the vehicle.
- Incident/Case Number – The number of the case/incident that necessitated the service request.

- Entered Date/Time – Date and time the service request was entered.
- Request Date/Time – Date and time the service was initially requested.
- OnScene Date/Time – Date and time the vehicle arrived at the service provider.
- Completed Date/Time – Date and time the service was completed (if applicable).
- Cancel Date/Time – Date and time the service request was canceled (if applicable).
- Unit – Identification number of the unit in need of service.
- Requested By – The person who initially made the service request.
- Entered By – The person who entered the service request into the system.
- Vehicle Information – Expanded information about the vehicle in question.
  - Year – Year of the vehicle's manufacture.
  - Make – Name of the manufacturer who produced the vehicle.
  - Model – Name of the specific type of vehicle.
  - Style - Descriptive information about the vehicle (2-door, 4-door, sedan, SUV, hatchback, etc).
  - Color – Primary color of the vehicle.
  - License Plate Number – License plate number of the vehicle.
  - License Plate State – The state the license plate was issued in.
  - License Plate Year – Year the license plate expires.
  - Release date – Date the vehicle is scheduled to be released by the service provider.
  - VIN – Vehicle identification number.
  - Hold for Evidence – Whether or not the vehicle should be held as evidence.

- Service Provider Information - Contains information about the service provider.
  - Company Name – Name of the service provider.
  - Contact Name – Name of the department(s)'s contact at said service provider.
  - Phone 1 – Primary phone number with which to contact the service provider.
  - Phone 2 – Secondary phone number with which to contact the service provider.
  - Pager – Pager number of the service provider (if applicable).
  - Paging Provider – The provider of the service provider's paging service.
  - Override button –
  - Skip button – Skip a scheduled service request.
  - Assign Provider button – Assign a service provider to the vehicle in question.
  - Cancel Request button – Cancel a scheduled service request.
  - Provider On Scene button – Designate a service provider as 'on scene.'
  - Provider Complete button – Mark a service request as complete.
- Comments - Table containing comments made by dispatchers regarding a rotation service request.
  - Date – The date when the comment was made.
  - Time – The time at which the comment was made.
  - Dispatcher – The dispatcher who made the comment in question.
  - Comment
  - Comment field – Text field where a dispatcher can enter new comments.
  - Add Comment button – Add an in-progress comment to the Comments table.
  - Save – Save an in-progress comment.
  - Cancel – Clear the comment field.

3.1.13 Powerline

A command line-style interface that allows a user to perform actions within VisiCAD without opening other forms or using a mouse.

- TO command – TO <Mailbox List> Message [Subject] – Send a message.
- TOP command – TOP <Mailbox List> Message [Subject] – Send a high-priority message.
- FI command – FI <Incident list> [cancel code].[disposition code].[comment] File an incident.
- DU command – DU <Unit list> - Unit 10-10 (off duty).
- OFC command – OFC <Unit list> [Station/Post code].[comment] – Place a unit at the area office (use * for list of offices).
- OOS command – OOS <Unit> [OOS reason].[comment] – Unit out of service for a 10-6 or 10-7 (10-6: busy unless urgent, 10-7: general out of service).
- ROOS command – ROOS <Unit> [comment] – Remove out of service reason.
- PUG command – PUG <Unit list> [location].[comment] – Manually update a unit's position (geo-validated).
- PU command – PU <Unit list> [location].[comment] – Manually update a unit's position (non-geo-validated).
- PUSH command – PUSH <Sector list> PersonnelID – Push control of area.
- PULL command – PULL <Sector list> - Pull control of area.
- U command – U <Unit list> Incident.[comment] – Assign a unit to an incident.
- EN command – EN <Unit list> [comment] – Unit is responding (en route to incident).

- UE command – UE <Unit list> Incident.[comment] – A unit is assigned an en route.
- ENA command – ENA <Incident ID> [comment] – Mark all assigned units as en route.
- ADD command – ADD <Backup unit list> Target Unit.[Incident ID].[comment] – Add additional units to the working log.
- 2LEG command – 2LEG <Unit list> [location].[Incident ID].[comment] – Unit is en route to alternate (geo-validated).
- 2LE command – 2LE <Unit list> [location].[Incident ID].[comment] – Unit is en rout to alternate location (non-geo-validated).
- 2L97G command – 2L97G <Unit list> [location].[Incident ID].[comment] – Unit 10-97 (arrived at scene) at alternate location (geo-validated).
- 2L97 command – 2L97 <Unit list> [location].[Incident ID].[comment] – Unit 10-97 (arrived at scene) at alternate location (non-geo-validated).
- BTS command – BTS <Unit> [comment] – Unit is back at scene from alternate location.
- 98 command – 98 <Unit list> [disposition code].[exception reason code].[comment] – Unit 10-98 (assignment completed).
- 98K command – 98K <Unit> [except-for list].[disposition code].[exception reason code].[comment] – Make all units 10-98 with an except-for option.
- 108 command – 108 <Unit list> [disposition code].[exception reason code].[comment] – Place unit in the Available status.
- FU command – FU <Unit list> [cancel code].[disposition code].[comment] – Cancel a unit from an incident (files the log if 1 unit is assigned).
- RTC command – RTC <Incident list> [comment] – Request to cancel beat unit.
- IU command – IU <Unit name code> [comment] – Inquire unit status.

- UUA command – UUA <Unit> [comment] – Update unit activity log.
- UU command – UU <Unit> [comment] – Enter comments in the incident record by unit ID.
- STACK command – STACK <Unit> [Incident].[comment] – Unit stack incident.
- SODN command – SODN <Unit> [Incident].[comment] – Unit stack original and dispatch new.
- RODN command – RODN <Unit> [Incident].[comment] – Reassign original incident and dispatch unit to new incident.
- US command – US <Unit> [Unit].[reassign code].[disposition code].[comment] – Swap unit assignments.
- USE command – USE <Unit name code> [Unit name code].[reassign code].[disposition code].[comment] – Swap responding units.
- US97 command – US97 <Unit name code> [Unit name code].[reassign code].[disposition code].[comment] – Swap units at scene.
- RUD command – RUD <Unit> [Incident].[comment] – Reassign unit to new log.
- RU command – RU <Unit> [reassign code].[disposition code].[comment] – Unit reassignment.
- CIM command – CIM <Unit> [command code].[Incident ID].[NEW].[comment] – Critical incident mode (pre-populates the powerline).
- ECIM command – ECIM [comment] – End critical incident mode.
- FC command – FC <Unit> [capability code].[comment] – Find closest capability.
- FCR command – FCR <Unit> #Capability code list [/].[comment] – Find closest capability w/ recall.
- FR command – FR <Unit> [resource code].[comment] – Find closest resource.

- FRR command – FRR #Resource code list [/].[comment] – Find closest resource w/ recall.
- RBU command – RBU [Incident list].[comment] – Recommend beat unit.
- RR command – RR [Incident ID].[comment] – Recommend unit w/ recall.
- R1010 command – R1010 [Division code list][/].[New] – Show 10-10 units (off duty units) by area (default will be the whole hub).
- TU command – TU <Unit list> [sector].[comment] – Transfer unit to another sector.
- LU command – LU <Unit name code> - VisiCAD: find unit.
- ZU command – ZU <Unit> - Zoom explorer map to unit location.

3.2 Nonfunctional Requirements

3.2.1 Operating System

The software must function on Windows XP/Vista/7. Compatibility with Mac/Linux operating systems will never be necessary.

3.2.2 Appearance

The software must replicate the look-and-feel of software native to Windows operating systems, whether XP, Vista, or 7. The system must also display properly on native 1280 x 960 (UVGA) resolution monitors.

3.2.3 Size

The software must be no larger than 10MB, uncompressed.

3.2.4 Response Time

The software must either respond to inputs in less than 1 second, or when it cannot, provide an indication that it is processing rather than frozen (progress bar).

3.2.5 Accurate Simulation

The software must simulate the real VisiCAD dispatch software with enough fidelity that the dispatch proctors (Jeff Gerfen and Neil Hockaday) are satisfied that they can successfully use it to train dispatchers in VisiCAD's use. If the proctors feel that the system does not simulate VisiCAD well enough that they can ethically run the dispatcher training program and accurately gauge whether or not a trainee is ready to use VisiCAD in real-world situations, the system will be rejected.

3.2.6 File Type

The software's executable must be a .jar file.

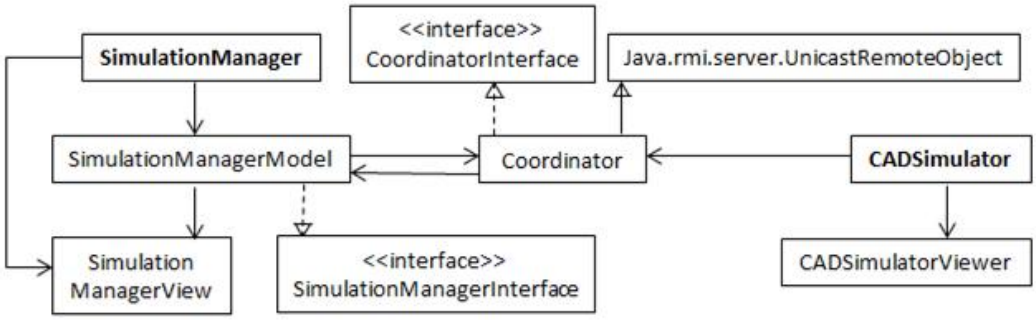
3.2.7 Portability

The software is meant to work on a local network of machines that are not connected to the internet. The system of machines has several custom-built elements involving land-line telephones to simulate dispatch calls, video screens to simulate traffic cameras, etc. The software will never need to be moved to a different system.

4 Design Notes

4.1 Analysis of Existing System

4.1.1 Connection Between the Simulation Manager and the CAD Simulator



UML Diagram featuring major classes in the communication between the Simulation Manager and CAD Simulator.

The *Simulation Manager* and *CAD Simulator* are connected through a *Coordinator*. As shown in Figure 1, *Simulation Manager* hasA *Simulation Manager Model* hasA *Coordinator* and *CAD Simulator* hasA *Coordinator* as well. The *Coordinator* also hasA *Simulation Manager Model*.

When running the *CAD Simulator* and *Simulation Manager* main method properties files

sim_manager_config.properties and *cad_simulator_config.properties*

define a port to be 4445, allowing them to communicate through the *Coordinator*.

The *Coordinator* is initialized by the *CAD Simulator*, while the *Simulation Manager* attempts to look for the coordinator variable through the port ID defined in the properties. For this reason, *CAD Simulator* must run first to create the *Coordinator* variable for *Simulation Manager* to use, otherwise *Simulation Manager* will fail to run.

4.1.2 Communication between the CAD Manager and the CAD Client

The CAD Manager and CAD Client use Sockets to send information between the two. Their properties files specify their Socket ports to each be 4444. Both the Sockets used by the Client and the Manager receive Documents through their *ObjectInputStreams*, which is sent to the other through their corresponding *ObjectOutputStream*. The root Node of the Document represents the type of message delivered, with the child Nodes including some additional information in some cases. For each type of message, a corresponding action is taken.

On the CAD Client side, it receives messages through the ObjectInputStream of the Socket in the form of a Document. Possible messages received by the CAD Client are:

1. UPDATE_SCREEN (along with the type of screen to update it to [INCIDENT_INQUIRY, INCIDENT_SUMMARY, INCIDENT_BOARD, ROUTED_MESSAGE, and BLANK_SCREEN])
2. UPDATE_STATUS
3. UPDATE_TIME
4. UPDATE_MSG_COUNT
5. UPDATE_MSG_UNREAD
6. CAD_INFO
7. APP_CLOSE

On the CAD Manager side, the socket receives messages in the form of a Document through the ObjectInputStream of the Socket. Possible messages received by the Manager are:

1. TERMINAL_REGISTER (with first child Node being the position Node and the second being the use ID Node)
2. SAVE_COMMAND_LINE

3. TERMINAL_COMMAND_LINE (with first child Node being the type of command requested (INCIDENT_BOARD, INCIDENT_UPDATE, INCIDENT_INQUIRY, INCIDENT_SUMMARY, ROUTED_MESSAGE, ENTER_INCIDENT, TERMINAL_OFF, or APP_CLOSE)

4. TERMINAL_FUNCTION

4.1.3 Data Stored by Existing CAD Manager and CAD Client

As part of the CAD Manager, Incidents are stored in the IncidentManager class. The IncidentManager contains several private data structures (TreeMaps and Vectors) to keep track of the data included in Incidents:

private TreeMap¡Integer, Vector<IncidentEvent>> completedEvents;

private Vector<Incident> incidentList;

private Vector<IncidentBoardModel_obj> IncidentBoardModelObjects;

private Vector<IncidentInquiryModel_obj> IncidentInquiryModelObjects;

private Vector<IncidentSummaryModel_obj> IncidentSummaryModelObjects;

Each of these Vectors holds sets of data for specific requests from the user. The descriptions below are summarized from the code documentation.

In the IncidentEvent class (completedEvents TreeMap):

- public long secondsToOccurInIncident - Time when the event will occur in relation to the start of the simulation.
- public long secondsOccuredInSimulation - Time when the event occured.
- public IncidentInquiryModel_obj eventInfo - Model object that contains data that is filled when the event occurs.
- public String waveFile - Audio wav file.
- public int waveLength - Length of wav file in seconds.
- public Vector<XMLIncident> XMLIncidents - (XMLIncident class includes: private String incidentID; private INCIDENT_STATUS incidentStatus; private INCIDENT_TYPE incidentType; private Vector<String> lanes; private IncidentLocation theLocation).

- public Vector<CCTVInfo> cctvInfos - (CCTVInfo class includes: public int cctv_id; public CCTVDirections direction; public boolean toggle).
- public EVENT_STATUS eventStatus - Current status of the event. Can be WAITING, TRIGGERED, COMPLETED, or FINALIZED.

In the Incident class (incidentList Vector):

- public String description - Description that appears in the Simulation Manager GUI
- public Integer logNumber - CHP Incident Log Number.
- public IncidentInquiryHeader header - IncidentInquiry header information (includes: public Integer logNumber; public String logStatus; public String priority; public String type; public String callBoxNumber; public String beat; public String fullLocation; public String truncLocation; public String origin; public String incidentDate; public String incidentTime; public String dispatcher).
- public TreeMap<String, IncidentLocation> locationMap - Contains IncidentLocation objects with a String identifier (IncidentLocation class includes: public String locationID; public String incidentRoute; public String incidentDirection; public String incidentPostmile; public String incidentLocType).
- private long startTime - Corresponds to the time that the event starts, in seconds.
- private long secondsIncidentStarted - The time in seconds that the incident occurred, whether it was as scheduled or manually triggered.
- private boolean incidentOccured - Whether or not the incident has occurred.
- private Vector<IncidentEvent> eventList - List of IncidentEvents that will occur in successive order.

In the IncidentBoardModel-obj Class (IncidentBoardModelObjects Vector):

- public int bulletinNum - The Bulletin number.

- public String date - The Bulletin creation date.
- public String time - The Bulletin creation time.
- public String message - The Bulletin message text.

In the IncidentInquiryModel-obj Class (IncidentInquiryModelObjects Vector):

- private IncidentInquiryHeader header - Header data.
- private Vector<IncidentInquiryDetails> details - List of IncidentInquiryDetails objects (includes: public String details; public Boolean sensitive).
- private Vector<IncidentInquiryUnitsAssigned> units - List of IncidentInquiryUnitsAssigned objects (includes: public boolean isPrimary; public String beat; public String statusType; public boolean isActive).
- private Vector<IncidentInquiryWitnesses> witnesses - List of IncidentInquiryWitnesses objects (includes: public String reportingParty; public String telephoneNum; public String address).
- private Vector<IncidentInquiryTows> tows - List of IncidentInquiryTows objects (includes: public String towCompany; public String confPhoneNum; public String publicPhoneNum; public String beat; public String statusInfo).
- private Vector <IncidentInquiryServices> services - List of IncidentInquiryServices objects (includes: public String serviceName; public String confPhoneNum; public String publicPhoneNum).

In the IncidentSummaryModel_obj Class (IncidentSummaryModelObjects Vector):

- public Integer logNumber - The log number.
- public String logStatus - The log status.
- public String date - The log creation date.
- public String time - The log creation time.

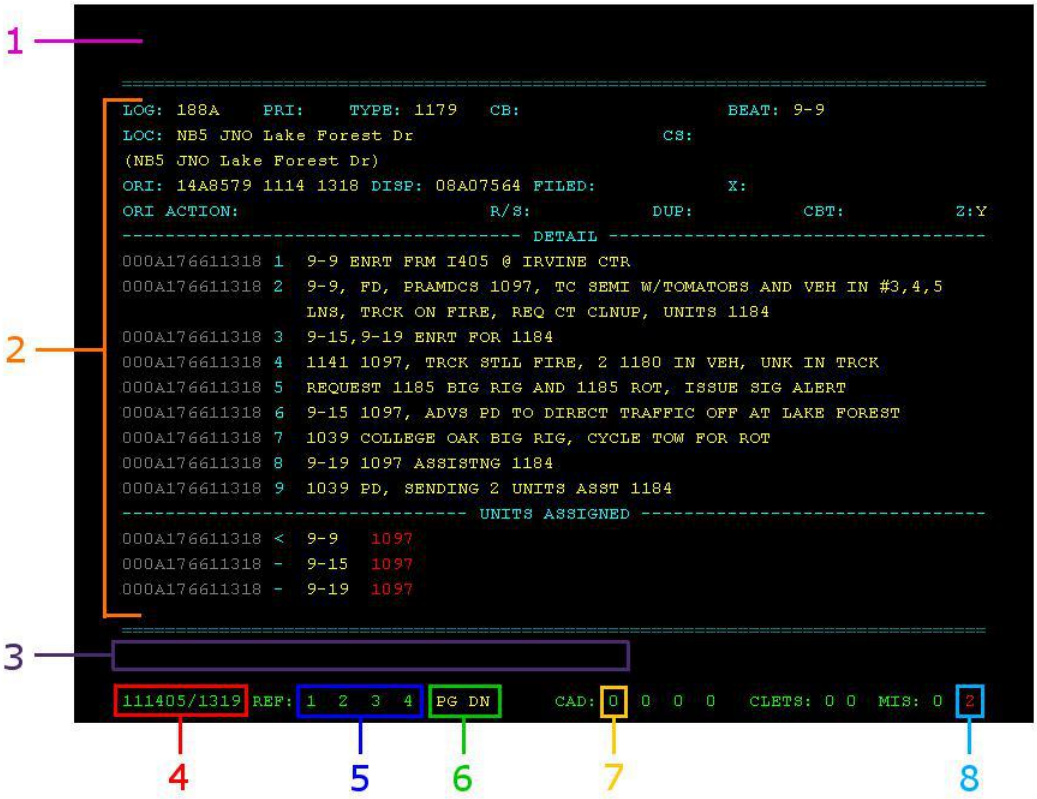
- public String priority - The log priority.
- public String callType - The log type.
- public String beatArea - The beat area.
- public String location - The location.
- public String beatAssigned - The beat assigned.

The CAD Client has the following instance variables:

- private CADClientSocket theClientSocket - Object to handle socket communication between the Client and CAD Simulator.
- private CADClientModel theClientScreenModel - Object handling data communication between the Client and CAD Simulator.
- private CADClientView theClientScreenView - Object to handle display panels CADClientSocket and CAD ClientModel are for handling communication.

4.1.4 Screens and Controls for Existing CAD Client

CADClientView primarily deals with screens, so below is a screenshot of the CADClient and some functionality for reference.



1. Command Line - User input.
2. CAD Data - CAD output.
3. Error Messages - CAD error messages.
4. Time/Date - The current date and time (MMDDYY/hhmm)
5. Screen Update - Page update notifications, colored yellow when available.
6. Page Scrolling - 'PG DN', 'PG UP', and 'UP DN' show possible scrolling.
7. Routed Message - Current number of routed messages.
8. Screen Number - Current screen number.

CAD Command Summary

- Incident Summary
  - SA. - Request an abbreviated list of current incidents.
- Incident Inquiry
  - II.<log number> - Request the current log for an incident.
- Update Incident
  - UI.<log number>.D/<text> - Update indicated incident.
  - UI.D/<text> - Update currently viewed incident.
- Route Message
  - TO.<CAD Position(s)>.M/<message> - Transmits a routed message to another CAD operator.

4.1.5 CAD Client Model Instance Variables

CADClientView has the following instance variables:

- CADClientModel theModel - Used for communication between CAD Client and Manager.
- CADCommandLineView CADCommandLinePane - The command line(seen as the #1/"header" at the top of the screenshot in Figure 1).
- CADMainView CADMainPane - The "body" or displayed information(seen in as #2 /"body" in the screenshot in Figure 1).
- CADFooterView CADFooterPane - The bottom displayed information(the rest of of the screenshot in Figure 1).
- CADCommandParser cmdParser - CAD Command line parser.
- boolean shiftKeyPressed - Boolean to designate whether the shift key is being pressed.
- CADScreenNum currentScreenNum - Current CAD Screen Num.
- TreeMap¡CADScreenNum, Integer> pageLocationMap - Map of CAD-Screen numbers and the current page displayed.

- boolean pageLocationSaved - Boolean to designate whether the screen's page location has been saved.

These classes(found in *tmcsim.client.cadscreens*) extend CADMainView:

- IB_IncidentBoard.java
- II_IncidentInquiry.java
- SA_IncidentSummary.java
- TO_RoutedMessage.java

There are commands shown in Figure 1 that correspond to these classes(II, SA, TO), so this information is displayed as the CADMainView when the corresponding command line commands are input (All these instance variables can be found in *tmcsim.cadmodels*):

- IB_IncidentBoard has the instance variable IncidentBoardModel.
- II_IncidentInquiryModel has the instance variable IncidentInquiryModel.
- SA_IncidentSummary has the instance variable IncidentSummaryModel.
- TO_RoutedMessage has the instance variable RoutedMessageModel.

4.1.6 Conclusions and Analysis Regarding the Existing CAD System

The classes in the CADClient are responsible for displaying the information obtained from the classes in the CADModel/Manager. The Incident/IncidentEvent classes hold information that is required in the newly proposed Incident class, so these two classes can serve as a good starting point. The current CAD Client displays all the information in three JTextPanes (header,body, footer). These need to be modified to include scroll bars and dividers.

The following variables found in CAD Client may not be necessary anymore:

- CADScreenNum currentScreenNum - Current CAD screen number.
- TreeMap ¡CADScreenNum, Integer> pageLocationMap - Map of CADScreen numbers and the current page displayed.

- boolean pageLocationSaved - Boolean to designate whether the screen's page location has been saved.

The old CAD Client had different screens, but the new ones can be relocated to a single window, since tabbed panes and scroll bars are now available. Since the command line operations are no longer part of the main cad client view, a dedicated Powerline object should be created, and the following command-line functionality must be transferred over to it:

- CADCommandLineView CADCommandLinePane - The command line(seen as the #1/ "header" at the top of the screenshot in Figure 1).
- CADCommandParser cmdParser - CAD Command line parser.

Overall, there is a lot more information that can be displayed in the new CADClient and the structure much of the data and displays different. There are some similarities, but it would likely be beneficial to build the necessary displays and classes from scratch to avoid building a new system around old data. However, it is also worthwhile to study the current architecture so that if can be used in the future if applicable. For example, all the incident classes have the following similarities:

- The constructor to initialize information.
- The instance variables required to function correctly.
- Private enums for XML_tags.
- The *toXML* method.
- The *fromXML* method.

and changing structures that are still necessary (and will see extremely heavy use) is a waste of time and resources that could be put to better use elsewhere.

5 User Analysis

5.1 The Trainee

Age: 21 and above.

5.1.1 Primary Uses

Education and training

1. Familiarize users with the layout and functions of VisiCAD
    - VisiCAD has build-in redundancies; there are often several ways to carry out the same task, and users will need to use the system for an extended period before they determine which methods suit them best.
2. Allow users to practice techniques expected of CalTrans/CHP dispatchers in a risk-free environment, including:
    - Dispatching police, firefighters, tow trucks, ambulances, road construction crews and other emergency/public services.
    - Coordinating their efforts with other departments.
    - File reports on persons/vehicles/places of interest (missing people, suspects, etc)

5.1.2 Level of Expertise

Little to none.

- These users are either entirely unfamiliar with computer-aided dispatch (CAD) software, or is unfamiliar with VisiCAD specifically. The majority of users will be members of this group.
    - The user may or may not be familiar with modern computing technology, depending on several factors, including age, past job history, and economic status.
        * The previous version of VisiCAD used a text-only display, so it is possible that a trainee may be familiar with the system, but unfamiliar with modern computer interfaces.
- Users in this group are already familiar with emergency dispatch protocols and terminology. The training software assumes this familiarity; trainees are learning to use VisiCAD, not being trained as dispatchers.

5.1.3 User Needs

- The training software must reflect VisiCAD accurately enough that a trainee will not notice any significant difference between the two, other than functionality not necessary for training.
  - VisiCAD features that are not essential to swift emergency response (scheduling vehicle maintenance, adding new vehicle/personnel entries, etc) do not need to be as accurately represented, as the goal of the training program is to train users to quickly and skillfully use the software in high-pressure/time-sensitive situations.

5.2 The Proctor

Age: 35 and above.

5.2.1 Primary Uses

1. Teach other, less experienced users how to properly use VisiCAD.
   - XML coordinates pending dispatch events with video/audio files to simulate real-world issues dispatchers will need to be able to deal with (traffic collisions, hazardous materials being released, suspicious persons, restricted traffic flow, fires, etc). The training software will include all current schema, so expansive knowledge of XML is not necessary for proctors.
   - The training room is outfitted with a multi-screen video wall, dispatch stations, and telephones, simulating all the equipment that an actual dispatcher would have available.
2. Evaluate the ability of students being trained in the use of VisiCAD.
   - Any updates made to an incident are shared between all running instances of the training software, allowing user(s) to watch trainees' ability to respond to incidents in real time while working in tandem.
     - Dispatch stations are situated to discourage trainees from looking at each others' screens, so that trainees can be evaluated individually when necessary.

- Telephones in the training room are routed to the proctor's room, where users can use audio files – as well as their own voices – to 'call in' incidents, allowing them to evaluate a trainee's ability to deal with different types of 911 callers.

5.2.2 Level of Expertise

High.

- This type of user is already familiar with VisiCAD
  - The user is familiar with modern computers, as well as how VisiCAD behaves whilst running on them.
  - Because the user is already familiar with VisiCAD, it is assumed that the user is also familiar with emergency dispatch terminology and protocols.
- This type of user is already familiar with the VisiCAD training program, as well as has the expertise necessary to operate the training room.

5.2.3 User Needs

- The training room must represent a real-world dispatch room closely enough that the operator can provide their trainees with a realistic experience.
- An unrestricted view of what the trainees are doing within the training software (both individually, and as a group) so they can accurately evaluate whether or not they are sufficiently prepared for the real thing.
- Separation between proctor and trainees. In order to get the most accurate evaluations possible, the trainees must be unable to see/hear/interact with the proctor in any meaningful manner that is not a part of the simulation.

6 Tools, Teamwork, and Communication

For a project the size of the TMC Simulator, an ad hoc development process was not structured enough to ensure that everyone involved could work together effectively and efficiently. The people involved included: four developers, who were simultaneously working on different parts of the system (GUI development, back-end Java development, the XML interface to be used by the proctors, and networking); two managers who were also in charge of creating and maintaining the physical hardware and acting as end users (see User Analysis: The Proctor); and the go-betweens at CHP and CalTrans who worked with the developers to help provide the most realistic simulation possible.

In order to keep the development process as organized and streamlined as possible, a variety of different techniques and tools were utilized.

6.1 Teleconferencing

Because of the developers' and managers' geographical separation, regular in-person meetings were not feasible. Due to this, telecommunications had to serve as the primary substitute for physical meetings.

6.1.1 Procedure

1. Begin Conference - At the agreed-upon time, the managers call all of the team members who were necessary (and available) for the conference. Each team member is called in turn and then added to the conference, at which time the team confirms that everyone has been successfully added to the call.

2. Developer Updates - At the beginning of every conference, each developer participating in the conference provides the others with a brief run down of what their experiences on the project in the time since the last conference. This may include things like: completed modules, bugs discovered, challenges/roadblocks, and things that may require additional assistance from another developer.

3. Manager Updates - After all of the developers have gotten everyone else up to speed on what they've been doing, the managers break down what they hope to accomplish in the immediate future (what parts of

the system they hope to have completed, any upcoming meetings or deadlines, etc) and what that means for each individual developer.

4. Scheduling - At the end of a conference call, the last order of business is to schedule the date and time of the next one. The day proceeding the next scheduled conference, a confirmation email is sent out to verify that all involved parties are still available.

6.1.2 What Worked

Due to the fact that all of the developers/managers lived far enough apart from each other that regularly meeting in person was an issue, the ability to still have regular conferences regardless of location was a definite help; one of the developers was still capable of participating in the conferences while visiting family in China. Determining when all the necessary parties were available for a conference call was extremely low cost - all a manager needs to do is send out an email and CC everyone.

The ability to get immediate feedback from a manager or developer was also very helpful. One of the drawbacks of asking a question through email is the response time; it may be multiple days before an email garners a response, and there is still no guarantee that the response will answer the question effectively. Being able to get responses in real time allows one to get immediate clarification, which significantly speeds up the communication process over email.

Additionally, conferencing over the phone allows developers who aren't as skilled at giving presentations to feel less nervous about the whole ordeal. Something about not needing to physically stand up in front of an audience helped everyone to feel more relaxed, which is important in a field that is rife with introverts and sufferers of social anxiety.

6.1.3 What Did Not Work

The most glaring issue with teleconferences being the primary form of meetings for the project was that they were not taken as seriously as in-person meetings. Professionalism took an obvious downturn when people realized that they could just pull things up on their computers if another member of the team asked about them. Everyone seemed less-prepared than they

would be for an in-person meeting, and several of the developers treated the conferences as more of an annoying chore that needed to be completed than as an important part of the development process.

This belief that a teleconference required less preparation also denied the developers something important - the act of preparing for the presentation. The process of preparing for a presentation often helps the presenter to gain a deeper understanding of the topic they're presenting. Giving a formalized presentation on a specific topic requires the presenter to really consider exactly what they did, how they did it, and what problems they faced while doing it. Denying the developers this chance to better understand their own modules led to a lot of rambling responses and question-dodging during conference calls.

Because the team as a whole took teleconferencing less seriously than they would an in-person meeting, there was also a lot of less-than-stellar communication when scheduling them. Multiple times when a conference was scheduled, one simply did not happen; no call from a manager, and no warning that the conference had been postponed or called off. Conversely, there were also conferences where a developer had already confirmed their availability, but were unreachable at the scheduled time (again, with no update from the team member that they would be unavailable). This breakdown in communication led to multiple meetings where the team was unable to discuss everything that needed to be discussed because the necessary team member was unreachable.

6.2 Project-Dedicated Cloud Storage

For the same reasons stated in Section 5.1 (Teleconferencing), keeping all the necessary project documents (design notes, requirements documentation, screenshots, and other miscellaneous information necessary for development) stored on a local machine - or every member of the team keeping their own locally saved versions - was not feasible or advisable. Instead, all of the documents were saved on a Google Drive made accessible only to team members.

6.2.1 Procedure

1. Save on the cloud - Any documents that a developer feels the rest of the development team should have access to should be saved on the cloud drive.

   NOTE: If a document is edited/updated, make sure that the most recent version is available on the cloud.

6.2.2 What Worked

Everyone involved with the project always having access to all necessary documents was incredibly helpful for speeding up development - the amount of time waiting for someone with important information to share it with the rest of the team was reduced significantly. In addition, there was never any need to search through long chains of emails in order to find old documents, and no worrying as to whether or not everyone else had all of the documents that they needed to continue their work on the project.

6.2.3 What Did Not Work

Just because everyone *technically* had access to all of the documentation they needed, that did not necessarily mean that they it was easy to access said documentation. The Google Drive was extremely unorganized; everyone fell into the trap of thinking that as long as the information was accessible, then everything was alright. However, one of the primary benefits of using a cloud service (to keep all documentation in one place) was nullified by the fact that the drive became so cluttered with superfluous information and outdated versions that finding the desired information became a timely ordeal in and of itself.

While having everything accessible in a single, centralized location was helpful most of the time, it also made the development team rather careless with their documentation. Situations arose more than once where a developer was unable to do the work that they wished to do because they lacked internet access. This made them unable to access the documents they needed because they had grown so used to accessing the information online that they had neglected to save local versions.

6.3 Github

Ensuring that all developers always have access to the most updated version of the code base is always important - particularly when working on large projects - but the inability for the TMC Simulator development team to meet in person for the majority of the development time made it even more important. A github server was utilized in order to deal with this issue.

6.3.1 Procedure

1. git status - Check whether or not there have been updates to the code base.
    (a) git pull - If changes have been made to the code base, update the local version of the code.
2. git add <file or directory> - Stage changes for commit.
3. git commit - Request a commit message (title and body) then save all staged changes along into a"current version" which then must be pushed to the master server before others can see the updated code.
4. git push - Update master server with all committed changes.

6.3.2 What Worked

Using git, rather than subversion, had definite advantages. Because every developer had their own version of the project saved locally, rather than everyone trying to simultaneously edit the same version of the code, there were far fewer instances where the code needed to be refactored because of conflicts. This also allowed much less frequent and frantic committing when compared to subversion; the ability to make changes to the local version of the code allowed the developers to 'shelve' their work if they needed to shift focus or stop working mid-update. The entire code base was not left in limbo while a developer had it checked out, with all of the other developers waiting for them to commit changes in order to continue working.

6.3.3 What Did Not

The only problem that arose while using git was a permissions conflict. While the developers were able to access the repository, the permissions had been

accidentally set incorrectly, which prevented the team from being able to clone, pull, add, commit, or push for the first few weeks of development. This caused development to stall slightly at the outset of the project, but because nearly all of the development work that was going on at the time was GUI prototyping, it had very little impact on the project as a whole.

7 Lessons Learned

7.1 The Importance of Management and Leadership, and the Difference Between the Two

Two major roles that play incredibly important parts in nearly all real-world software development projects are those of management and of leadership. While in an academic setting, students do not receive much experience working with these roles in a traditional sense. Professors take on most of the responsibilities of both roles, and students are unknowingly provided with incredibly refined and organized management as well as exceptionally understanding and knowledgeable leadership.

The assignments and due dates set by the professor serves as both de facto module assignments as well as a development schedule. A professor's previous experience teaching the course sets a precedent for how long a particular assignment should take, as well as the knowledge necessary to complete said assignment in a satisfactory manner. Upon leaving the world of academia, former students no longer have this luxury, and must learn how to estimate the difficulty and duration of any modules they are working on themselves.

7.1.1 The Role of Management

At the simplest level, the role of a manager is to manage a project's development; to ensure that everything goes as smoothly as possible from the first meeting with the customer to delivering the finished product. To ensure that everyone else working on the project is doing what they should be doing, when it needs to be done. In short, a manager's most important function is to deal with everything not directly related to the design/development/testing of a project, so the developers don't need to focus on anything else.

Perhaps one of the most important qualities in a good manager is the ability to 'read' his or her team. A manager needs to always be on the lookout for anything that might adversely affect the effectiveness of the development team. Determining the levels of stress of each individual team member can help to keep a project going smoothly. Developers often focus on their own assignments so intently that they are unable to realize when a teammate is buried beneath a job that is too large for one person. Because ego and introversion are often issues among software developers, by the time a developer asks for help, it may already be too late; a delay in the completion of a single module may have far-reaching ramifications that delay the completion of the entire project. A development team is only as strong as its weakest link, and a manager must be able to identify when a link needs to be reinforced.

The other most important function of a manager is organization and scheduling. It is a manager's job to develop a rough estimate of how much his or her team is capable of completing in a specific time frame, while also ensuring that a project is delivered on time. This can be exceptionally difficult, because a manager must balance the importance of getting things completed when expected against the risk of overworking a team or setting unrealistic deadlines. If a project is delivered late, it reflects poorly on the team and upsets the customer. However, if a project's schedule is too rigorous, it significantly increases the likelihood that the project will be of poor quality, which also reflects poorly on the team and upsets the customer.

7.1.2 The Role of Leadership

Different from management - but equally important - is leadership. While a manager typically is not directly involved with a project's development, the leadership role is filled by someone who is still responsible for some of the designing, coding, etc. - a senior software developer. If the manager were likened to a General - directing the troops from the command center - then the project leader would be a Captain or Lieutenant and actually lead the troops into battle.

While a manager is in charge of everything not related to development, the project lead is in charge of development itself, and serves as a bridge between management and the development team. A senior developer must have previous experience working on software development teams as well as

experience working with management; this allows a senior developer to see the project from both sides and help management understand the needs of the developers and the developers understand the needs of management.

A project lead also provides the rest of the development team with a single figure to rally around. They must provide the glue that holds the team together; the team needs to have a superior who they can still view as 'one of them' - someone who will go to bat for them if they find themselves in a less-than-perfect situation with management. Additionally, a leader must ensure that morale and enthusiasm for the project stays high. Few things can slow development like a lack of enthusiasm; developers are far more likely to put in their best work if they feel like they are working on something interesting and worthwhile.

However, a project lead must also be on the lookout for when a member of the development team is not pulling their weight and be ready to inform management if necessary. Management is generally too busy to oversee everyone on the development team, and a project lead needs to be able to tell the difference between a stressed or demoralized developer and one who is incapable of doing what is required of them.

7.2 The Importance of Scheduled, Structured Communication

In the majority of software software projects, ensuring that the system works as intended is not the most difficult part of development. With the exception of cutting-edge projects that require the creation of never-before-seen functionality, cursory research can provide significant insight into similar projects and offer plenty of recommendations as to what will work and what will not. A system that is given an appropriate amount of design time is rarely limited by the actual complexity of the modules that need to be completed in order to ensure that it works as planned.

Rather, the difficulty lies in the sheer amount of man-hours required. A project with a small team may be required to stretch development over weeks, months, or even years, due to the limited personnel working on it. Conversely, a project with a large team has to coordinate the efforts of dozens - or even

hundreds - of different people, all simultaneously working on different parts of the project. Coordinating all the different parties responsible for parts of development, and maintaining this coordination throughout the project's life-cycle, marks one of the key difference between software engineering and simply programming. Communication between the different developers, departments, or agencies must be considered as important as the actual coding of the project.

Due to this importance, all communication must be structured and scheduled appropriately, just as a development team must agree on a commenting style and a process for reporting bugs. In an interview about communication in the software development field, Joseph Adzima - the senior developer at the Bend, Oregon branch of Sony Computer Entertainment America - stated that approximately 50% of each workday was spent on workplace communication. [1]

With communication being such an integral part of the software development process, it becomes especially important that said communication is done in an organized, predictable fashion. Conferences, progress reports, and code reviews must be required at regular, scheduled intervals, rather than simply on request. Similarly, spending the time at the beginning of the project's life-cycle to establish formats and processes for each form of communication makes said communication much more effective and efficient in the long run.

8 Major Setbacks

8.1 Communication Weaknesses and Development Hell

During the early stages of development, communication was regular and frequent. Emails updates regarding scheduling and meetings were constantly being sent to developers, and teleconferences were occurring at regular, predictable intervals. As things progressed, however, and focus shifted from implementing the GUI to networking the system and implementing the backend, communication became less and less frequent; less and less predictable.

All of the developers were working on different schedules, and manage-

ment was overseeing more than a single project. This difference in scheduling became far more pronounced once the fall semester at Cal Poly began, since several of the developers (and one of the managers) were involved with university classes at the time. With the majority of the team's schedules significantly fuller than they were during the first months of development, the lag time between emails became more pronounced; an email that would have received a response in a day now took several days, or even a week.

It was during this period when the project's lack of a dedicated means of bug reporting became extremely apparent. Bugs were primarily reported using email, rather than an issue-tracking system. Everyone involved - with the exception of myself - were using their Cal Poly email accounts to communicate with the rest of the team, and after school started once again, it was much easier for an report to become lost among course and department emails. These difficulties effectively reporting issues allowed those issues to go unresolved for long periods of time, forcing any developers who were waiting on a bug to be addressed to find something else to work on in the mean time.

As the projected continued moving closer to completion, communications continued to deteriorate; multiple weeks passed without any contact with management, and the conference calls were happening far less frequently. Without management facilitating things, there was significantly less communication between developers. Thankfully the way the projected had been broken into individual assignments for each developer ensured that so long as everyone continued work on their parts of the project, things would continue to go smoothly.

Then, about a month before the project was scheduled to be delivered, communications resumed. Development was still on track, give or take a week or two, and it appeared as if the dispatcher training would be able to go ahead as scheduled. However, as the delivery date got closer, communications broke down once again; the developers were informed that they would have a few weeks more than originally planned to get everything completed, but they were not told what had caused the delay. The new delivery date came and went, with no communication from management, and the developers began desperately trying to get in contact with them.

When contact was finally established, the developers were informed that there was a dispute between several of the governing bodies (CalTrans and UC Irvine) about the training program, and that the project no longer had a concrete delivery date. They were told that development should continue to completion, but that there was no longer a scheduled beginning to the dispatcher training program.

Development continued - albeit much more slowly than when the team was working to a deadline - until the project was completed. However, when the lack of a deadline was combined with the busy schedules of student-developers, it took several additional months of work to complete what could have been completed in a few weeks' time.

8.2 Incompatibility Between IDEs and Loss of Important GUI Files

At the outset of the GUI's development life cycle, it was suggested by the lead GUI developer that everyone utilize the NetBeans 7 integrated development environment. The reasoning behind the suggestion was that NetBeans 7 provided almost all of the functionality necessary for project development - support for the most commonly-used software versioning/control systems (Subversion, Mercurial, and Git); native compiler support for Java, Java Swing, and XML schema; and a built-in Java debugger.

The most important element to the project, however, was the Java Swing GUI builder. This feature allows developers to work with Swing using a drag-and-drop interface in addition to working with the source code directly. The GUI lead believed that this was an important feature to utilize, because three of the four developers had no previous experience using Swing. The interface allows users to reposition and rename GUI elements, alter the values of class variables, and specify default fonts, text sizes, and window dimensions without needing to search through thousands of lines of GUI code in order to do so.

Any elements designed using the GUI builder are reflected in a .form file that is generated by NetBeans in tandem with the .java source-equivalent. As long as the user has both files (and there are no versioning conflicts be-

tween the two), the user is given a visualization of what the GUI components currently look like. However, if extensive changes are made to the source files outside of NetBeans, conflicts can arise that make the GUI builder unable to visualize the GUI components.

This is where the problem arose. Several of the developers ignored the suggestion to use NetBeans during back-end implementation. Due to a lack of regular, structured communication directly between individual developers, a significant amount of back-end code had been implemented for multiple GUI elements before the mistake came to light. By this point, the .java files had been significantly altered from what the NetBeans .form files expected, so they ceased to function correctly. Research into rectifying the problem proved fruitless; there was no way to reverse-generate the .form files using the source code. While the GUI elements still functioned properly when the system was running, there was no longer any way make changes to them without doing it manually, which made making any changes significantly more time-consuming.

Additionally, the lack of .form files would also make the system significantly more difficult to maintain in future revisions to the project, when the tacit knowledge of the original development team has been lost due to employee turnover. Due to these issues, significant additional time had to be spent between iterations to ensure that the .form files would work correctly once more.

9 Future Improvements

9.1 Communication

In order to ensure that a breakdown in communication similar to one that occurred during the late stages of development of the TMC Simulator does not happen on future projects, the requirements and process of all forms of communication must be made more highly structured. Formalized processes for how each form of communication must be carried out should be developed to ensure that they always occur in an orderly, predictable fashion.

9.1.1 Meetings/Conferences

1. Scheduling - Meetings must take place regularly, at scheduled intervals rather than only when the team feels they are necessary. This will ensure that lines of communication are always open, and that there will never be an extended periods of silence that could allow errors to snowball.

2. Minutes - During each meeting, there will be one attendee who is designated as the minutes-taker. A minutes-taker will be responsible for

    - Taking notes that provide an accurate record of decisions and discussions made during the meeting.
    - Writing up a summary of responsibilities of each member of the team (based on the decisions/discussions that took place).
    - Copying and distributing the minutes to all participants.
    - Ensuring that the minutes are appropriately filed for future reference.

3. Previous Minutes - At the beginning of each meeting, attendees must briefly review the minutes of the previous meeting. This is to ensure that all participants have any and all topics from the previous meeting fresh in their mind.

4. Final Review - At the end of each meeting, attendees will review decisions/discussions that took place over the course of the meeting and ensure that everyone knows the what they are responsible for before the next meeting takes place.

9.1.2 Emailing

- Project-dedicated Email Address - At the outset of the project, all participants must create (or will be assigned, if appropriate) an email address exclusively for use on the project. This will ensure that all emails about the project are collected in a single area, and that they will not be mixed in with emails irrelevant to the project.

- Email Group - An emailing group must be established for the project in order to ensure that everyone involved with the project can see all

communication that takes place. It must be guaranteed that no one ever misses an email because of a mistyped email address or a CC omission.

- Scheduled Progress Reports - Each time a team member works on the project, they must record a brief summary of what they worked on and any difficulties they encountered.

9.1.3 Bug Reporting

Rather than reporting bugs via email, an issue tracking system (such as Bugzilla, Mantis, or Trac) should be used.

1. When a bug is detected - before reporting the issue - the reporting party must ensure that:
   (a) They are working with the most current version of the system.
   (b) That the issue detected is indeed a bug.
   (c) That the bug has not already been reported.
2. When reporting a bug, the reporting party must:
   (a) Provide a description of the situation that caused the bug to arise.
   (b) Provide as accurate a description as possible as to what exactly causes the bug to occur.
   (c) Rank the severity of the bug encountered (cosmetic, trivial, minor, major, critical).

9.2 Leadership and Division of Duties

While the various parts of the TMC Simulator system were effectively divided among the development team in a manner that played to each individual member's strengths. However, the team lacked clearly defined development roles; every team member was simply considered a developer. There were no distinctions between developers other than the code segments that they had been assigned. While everyone was considered in charge of completing their own parts of the project, the team lacked any kind of hierarchy or power structure, which led to some difficulties.

The team lacked someone with the authority to direct the others. There was management to ensure that development continued to progress, but the lack of a lead programmer was sorely felt.

- Single Leader - A single member of the development team must be designated the team leader. The selection of the team lead should be based not only on which developer has the most experience overall, but on which one is the most familiar with the project domain. The project leader has final say in all decisions regarding development, and acts as a sort of developer/manager hybrid. The project lead must also work to generate enthusiasm for the project, and to make sure that the morale of his or her team stays high.
- Clearly Defined Roles - Each member of the team will be required to rank their own abilities in each of the domains deemed essential to project development by the project lead (obviously requirements, design, implementation, integration, and testing, but also more project-specific things such as GUI-building, networking, databases, etc). Based on these rankings, the team leader will assign lesser leadership roles to each other member of the team (documentation lead, testing lead, GUI lead, integration lead). The team lead should defer to these other leaders regarding their domains of expertise unless he or she has a very good reason for doing otherwise.

10 Conclusion & Final Thoughts

At the project's outset, the end goal was defined as delivering a software system that 'represents the real-world version of VisiCAD Inform well enough that the proctor is able to judge whether or not the trainees are skilled enough in its use to be ethically permitted to do so in situations where mistakes could result in the loss of life.' In that regard, the project could be considered a success. The proctors - both of whom have enough experience with the real VisiCAD Inform that they are able to determine how faithfully the simulator replicates its behavior - determined that the simulator was accurate enough to accept for use in the training program.

However, the processes and techniques used during the development life cycle left plenty to be desired. Lax standards of communication allowed the project to drag on for far longer than necessary, and almost prevented its completion entirely; poor division of authority led to mistakes that could have been easily preventable with a more clearly defined power structure; and difficulties getting in contact with involved parties (CHP, CalTrans, and UC Irvine) made the chosen software development model (the iterative model) less effective than originally planned.

The project lacked any clearly defined processes for all of the forms of communication utilized during the project. Emails were often sloppy or incoherent, and more than once a member of the team missed a conference call because someone failed to CC them on the scheduling email. Meetings themselves were ad hoc and unstructured, accomplishing very little other than allowing everyone to provide status updates. Significantly more could have been accomplished in time necessary if meetings had been sufficiently organized and planned out.

Similarly, the development team believed that everything could be successfully carried out without need for a well-established power structure - that management would be enough. This mistake clearly demonstrated why the majority of businesses operate using such power structures; occasionally there *needs* to be someone who is capable of 'puling rank' in order to ensure that everything gets done *properly*, rather than just gets done.

The primary lessons learned while working on this project are that things do not always go according to plan, and that the benefits of well-established structure is often worth the necessary time cost. The larger and more complex a process becomes, the more time must be spent following it, and to inexperienced software developers, time not spent programming is time wasted. However, those costs are never imposed needlessly. When a team needs to know exactly what they were working on three and a half weeks ago, they will be glad that they put in the time to take careful minutes at all meetings and filled out regular progress reports. Really, the two primary takeaways from the project go hand-in-hand; things do not always go according to plan, and when they don't, you will be glad that you took the time to ensure that there are already fail-safes in place.

11 Development Journal

11.1 June

11.1.1 June 9, 2013

- Got an email from one of my potential bosses today, asking me to come in for an interview. Honestly, I'm surprised that they had room for more people on the project – there've been fliers up around campus looking for Java developers for at least a month or two. It took me a while to realize that the job in question was the same one, or I would've contacted them beforehand. When my OS project partner said he'd talk to his bosses about it, I thought he was just being nice/polite.

- The 'design documents' they sent me are extremely confusing, and not particularly helpful. Either they were done as a formality by someone who didn't work on the project very much, or they were done by people who weren't experienced in making design documents. All I could really glean from it is that there will be some networking involved (and from what Derek told me, that's why they hired him – thank heavens for CPE majors and their ability to enjoy super-low-level development, I suppose), as well as a Java GUI. And it appears that some data will need to be translated from XML into Java class data, and then back. Makes me think that the people who will be operating it aren't exactly software developers themselves.

11.1.2 June 16, 2013

- Got the job! Finally! While it isn't something that I'm extremely interested in, at least it's some out-of-school experience. Decent pay, too; nothing like I'd get working at a full-time software-development job, but much better pay than my last one. And I get to use Java instead of Flash, so that's a good thing. It's still GUI development, like my last job, but at least now I won't have to feel like a Graphic Design major pretending to program.

- It's for CalTrans, which surprised me. I had been half-expecting it to be a job working for Dr. Staley (a somewhat intimidating thought, in my mind), considering Derek was the one who told me about it.

  - It's rather intimidating working for a government agency, now that I think about it, but being able to put on my resume that I worked on a CalTrans project is pretty cool, too.

- Derek admitted that he suggested me for the project because I know a lot more about Java than he does. That's fine with me! He knew things about C that I had no idea were possible (There're some courses that CPE majors have to take that involve manually swapping bits and playing with registers...ugh) when we were working on OS stuff, so I guess I'll have to show him exactly why I prefer Java.

  - Another intimidating prospect – I have the most Java experience of anyone working on the project, and I think even more than either of my bosses.

    * Maybe this is a good thing. If I'm trying to keep my code simple/elegant enough for everyone else to understand, it may also keep me from trying convoluted, hacky solutions if I get frustrated trying to make something work. However, since I'm responsible for everything the users will end up seeing/interacting with, I doubt I'll be the one rushing at the end. Also, if I make a Java suggestion, it may hold more weight with everyone else working on the project.

      · I should suggest developing in NetBeans. I was a little angry when Dr. Dalbey forced us to use it, but it's visual representations of javax.swing GUI components makes GUI-building much easier to understand. The rest of the team seems afraid of using swing, because of its reputation for being clunky and time-consuming to use.

11.1.3 June 23, 2013

- I finally got to meet Nick and Vincent, the other two people working on the project with us, but only by phone. We used a 6-way conference call, which was a little strange for me. Everyone's phones and voices all seemed calibrated to different levels, some people's voices were fuzzier than others. It made communicating much harder for me. I never knew when someone was going to start talking, so I was interrupting someone as well as being interrupted, and more than once someone

launched into a lengthy explanation only to realize that someone else had been talking simultaneously.

  - It seems like conference calls are something I'll have to get used to. We have a room in bldg 007, but there's a distinct lack of computers in it, and it seems more like a storeroom than anything else.
    * Having only a desktop computer is going to make working together a bit of an issue for me. Thankfully Neil and Jeff are working on getting a GIT server set up.
      · Never used GIT before... Only SVN. But Derek made it sound like people have stopped using SVN in favor of GIT, since it's much easier to work with. Talk about something you learned a year ago already being obsolete... Damn...
  - The lease is going to be up on my apartment soon, so I probably won't be a convenient, 5 minute walk away (I probably live the closest of anyone on the project, so expecting people to come in to work probably isn't the best idea).

- Didn't get to work as much this week as I would have liked to. Everyone was busy with end of the year/graduation stuff, and my parents came to visit, so I ended up spending a lot of time with them while I had the chance.

11.1.4 June 30, 2013

- Haven't heard anything from Jeff or Neil this week, but Jeff said that he was going on vacation with his family for the early part of July.

- Nick is getting to work on translating XML schema to Java classes.

  - The simulator has a series of video screens, computers, and phones. The proctor uses the XML schema to time video/audio events from the control room.
    * This allows them to coordinate video/audio with new incidents appearing, so the trainees can react to them.

- Neil took some screenshots of the real VisiCAD's UI so I could get an early start on the one for the simulator. It's kind of a strange feeling,

knowing what it's going to look like before we actually get to work on the back end; usually I'd do things the other way around. Neil and Jeff said they're having trouble scheduling a time for us to go in and actually work with VisiCAD itself, but having an idea about what they want everything to look like is something, anyway.

- We seem to already be having bad luck in the client area, but we have a visit to CHP to see the real deal in the works. I'm curious as to what the hold-up is... They didn't give me a non-disclosure agreement when I was hired, so I don't think it has to do with confidentiality, but I could be wrong... Though it could just as easily just be that the CHP is busier than I imagined.
  - I've started getting reacquainted with swing – my previous experience with it was in CSC 308/309, but I that was mostly back-end development. Thankfully, Sun's documentation for it is just as good as the rest of the javadocs.
    - NetBeans has been really helpful with this so far. The drag and drop interface makes it really easily to tinker around and see just how different swing objects interact with each other. NOTE: the drag and drop section of the GUI builder automatically names any objects added to the stage, and it names them very badly. jButton1, jTextPane2, etc. Remember to rename any variables added this way, or the GUI will quickly become an impossible-to-read labyrinth of Swing code.

11.2 July

11.2.1 July 6, 2013

- Jeff finally got in touch with us again – I was right, he had already left to go on vacation with his family. We officially have a date where we're going to visit CHP to get a chance to play around with the real VisiCAD. We aren't allowed to take a camera in, though, so I'm not entirely sure how much information we'll be allowed to take with us when we leave. Need to remember to pay as much attention as I can and take good mental notes... Who knows when we'll get another chance to use it freely. Neither Jeff nor Neil seemed to want to confirm

that there was an issue with confidentiality, or if we just need to go through the right channels to prove that we need to be there.

  - Jeff and Neil also brought up the idea of road-tripping down to UC Irvine (where the training rooms are located) to see everything in action. We'll also get to visit Orange County's dispatch center and get to interview dispatchers about how they use the software. Sometime in the beginning of August?

- I've started rapid prototyping some of the more basic GUI forms. Been using a combination of the drag and drop interface in NetBeans and custom code. The NetBeans GUI builder is proving a good choice, as I can see how things will behave without needing to code anything out. However, the more I use Swing, the more I begin to see how overbuilt it is. There are a lot of useful parts of it, but there's just as many parts that are practically useless. jInternalFrames seem to behave exactly the same way as jFrames, and there's generally a lot of unnecessary overlap.

  - Need to look into how to make a dropdown menu that is editable – jComboBox?
  - Swing doesn't seem to have a built in date picker object, may need to look into an extension or developing our own from scratch.
  - Many of the forms are missing information. I don't know what certain tabs of jTabbedPanes look like, since only one can be on screen at a time for taking screenshots.
    * Search form - Entire advanced tab is unviewable.
    * Search form, basic tab - All tabs in the bottom-most tabbed pane are unviewable.
    * Cardfile form - Do the different category tabs simply provide different ways of sorting the same information, or is the information actually different?
    * IncidentViewer form - Unknown functions of 'person' button, 'vehicle' button, 'location' button, and 'information' button.
    * - IncidentViewer form - All tabs in the bottom-most tabbed pane are unviewable.

- The Powerline pane seems to simply be a limited-functionality command line. Different commands (between 1 and 5 letters, some of which have other parameters) open/bring focus to other screens within the system.

  - The previous simulator used a text-based interface which used the same letter combinations. Easy way to test the functionality of the new Powerline – route the input from the Powerline into the old and new versions, and compare the outputs.

11.2.2 July 18, 2013

- Got to do our CHP visit today. Of course we had to prove who we were and clear everything at the front desk before we were allowed into the majority of the building – I wasn't really surprised, and we had an escort everywhere we went inside. More alarm bells dealing with confidentiality, but again, they didn't make us sign NDAs or anything of that nature. We took a ton of screenshots of all the different screens/tabs/autofills/etc, and they told us that they'd review everything and send us what we were 'allowed to see.' I asked about confidentiality and they said we'd hear from them if anything we took shots of is in fact confidential, but they seemed pretty confident that it wouldn't be. I guess they're just making sure to follow protocol – covering their own asses – which is fine, considering I was doing pretty much the same thing from the opposite perspective.

  - Jeff, Neil, Vincent, and Derek (Nick couldn't be there with us) all seemed really casual about the whole thing. I can't tell if I'm being paranoid, or if I'm being diligent. Derek is CPE, and Vincent and Nick are both second years, so they wouldn't have taken CSC 300 regardless of their majors (though I think Vincent is CPE and Nick is CSC). Maybe I'm the only one who's worrying because I'm the only one who has thought about it.
  - CAD stands for computer-aided dispatch, and the dispatchers all seemed to know what to do should an issue occur with VisiCAD; there are other, more archaic ways to deal with dispatching. Is VisiCAD safety-critical software? A careless dispatcher could assign multiple units to the same incident... What if two ambulances

showed up to an accident when only one was required? Or what if a saving error keeps a suspect entry from being committed to memory? It definitely seems like if it isn't safety-critical, it's at least safety-critical adjacent.

  * Where does that put a training simulator? I can guarantee that if the simulator fails, no one will die, no significant amount of money would be lost, etc (barring a complete failure of the equipment supporting the software, which we aren't responsible for). However, what about if our simulator doesn't properly prepare for someone to use the real thing? All of the trainees are already familiar with dispatch... As I've said before, the purpose of the simulator is to train dispatchers to use VisiCAD, not train people to be dispatchers. But, what if the simulator doesn't adequately represent the real-world system? A dispatcher could make the incorrect move in a critical situation because of a bad habit they picked up training on our simulator. The line between dispatcher incompetence and an insufficient simulator could get pretty blurry here... Then again, the people who will be proctoring the training sessions have experience with the real VisiCAD, as well as with training dispatchers to use it. If the trainees are unable to use the system to adequately respond to the training situations, they can always refuse to clear them to work with the real thing.
  * Considering that the project made it through Cal Poly Corp's legal division, I can assume they've at least considered it. All of the primary developers on the project are students, and I know for a fact that none of them have taken any kind of test to prove that they're software engineers rather than just software developers. Is it safe to assume that CPC wouldn't assign under-qualified workers to a project, or is that giving them too much credit?

11.2.3 July 20, 2013

- Had a brief teleconference with everyone today, mostly to discuss our visit to CHP. We're still waiting on getting our screenshots from them, but Neil and Jeff say that they've been in contact with them, and that

they're on their way. Hopefully we'll get access to everything we looked at – some of the forms are truly labyrinthine.

- Nick is going to be coming back to SLO for the rest of the summer, so another pair of hands working on things should be helpful. No one else has the understanding of the XML side of things that he does, so Vincent and I will need to work with him when we get further into back-end development.

- Jeff and Neil approved of the prototypes I've done so far, messy as some of them are. They seemed impressed by how much of them I've finished, which is always a good thing.

  - Either everyone else believes what I've been working on is much harder than it actually is, or much more time-consuming, and it appears that I'll be designing the entire front-end by myself.

- I was looking through some of the work that Nick has done so far on the XML schema. It turns out that the proctor is responsible for grading how well each student responds to each of the incidents that occur in a simulation. Assuming that the proctor is acting ethically (and knows how to operate the simulator and run the dispatch training, obviously) a situation where a dispatcher who is not ready to use VisiCAD in real world dispatching situations won't ever occur. **I don't think we need to worry about the system being designed to standards that safety-critical software would be; if the system does not accurately represent what it's like to use VisiCAD, the proctors will tell us/refuse to sign off on the simulator.**

11.2.4 July 24, 2013

- Got the screenshots and notes we took while we were at CHP. I'm not 100% – a there were a LOT of screenshots – but it looks like everything is there. Seems I was correct in assuming that they were just following protocol, plus, government agencies are all about covering their asses in terms of legality.

  - Now that I have more to work with, I've been able to finish several incomplete prototypes that I couldn't finish before, due to lack of information.

- I told Neil about the datepickers presenting a potential problem, since NetBeans doesn't have a datepicker object, and they aren't exactly the simplest things to hand-develop. He told me that, since all the incidents will be 'coming down the pipeline', so to speak, there's actually not a lot of need for them. According to him, trainees won't need to be able to access any backlogged incidents.
    * We may need to add them at some point in the future, but for now, he said to worry about getting to work on the most essential parts of the system before working on the periphery.

11.2.5 July 31, 2013

- We're heading down to Orange County to check out the status of the training rooms next Friday.
    - Leaving at 5pm Friday, Aug 11 - Why does it have to be so late?
    - REMINDER - Check with Jeff or Neil about what I should do about logging hours for the trip. Do I count all time I spend on the road trip, even driving? CPC is paying for a rental car for the drive, because it's cheaper than paying one of us to use our car to drive (which pays something like 50 cents per mile driven.. Kinda wished they would let me drive mine after I learned that)

11.3 August

11.3.1 August 13, 2013

- The visit to UC Irvine was only somewhat successful. The machine running Microsoft server – the only computer that is absolutely necessary to run the training rooms - didn't seem to be working. First, it wouldn't turn on at all, so we took out the power supply and tried another. We got power – fans spinning, LEDs lighting up, etc – but no response from the BIOS, and no post beep. We didn't exactly have the time (or the parts) necessary to try to repair the computer, and it was such an old machine that Neil and Jeff seemed to think that it might just be better to build a new one. As it is, Neil took the computer home with him to see if he could fix things.

  - However, we did get to see how everything is set up - how the proctors will interact with the trainees, how the trainees will interact with each other, etc.
  - In order to be safe, we checked the rest of the equipment. Everything else seemed to be working correctly, but I'm really disappointed that we didn't get to see things in action, even if it was the old version that we're working to replace.

- Our visit to the Irvine CHP dispatch center was far more successful. We got to watch dispatchers actually using VisiCAD, as opposed to our previous visit to CHP in SLO, where they seemed apprehensive to tell us about how it actually worked.

  - The Powerline and the Incident Editor/Viewer seem to be the most important parts of the GUI. A lot of the older, more experienced dispatchers seemed to be using almost exclusively the Powerline... Probably for the same reason older programmers love VIM so much; they want to do everything without ever needing to take their hands off of the keyboard.
    * Great! The Powerline is one of the simplest windows to test, provided that there's at least something for each command to pull up or shift focus to. Thankfully the prototypes I've been working on can also function as fakes for this purpose.
      · There is no fundamental difference between the Powerline and the command line that was used in the previous, text-based version of VisiCAD, and there are a series of commands that open up other forms for data entry/retrieval.
      · Potential hurdle – The Powerline needs to auto-complete commands. It won't be a huge deal if we can't get this working (the dispatchers have lists of all of the available commands), but I'd feel better if we minimized the amount of reference material necessary.
  - We might get access to the manual! The people at CHP have to get it cleared first, but they seemed pretty confident that they'd be allowed to give us a copy. It'd be incredibly helpful for pushing things through (assuming the manual is decently written).

    * VisiCAD itself was given to CHP as an executable. They don't get access to the source code, and if there is an update, they perform a system-wide restart to apply it. Of course, having the manual isn't as helpful as having the source code itself, but if we had access to the source code itself, we'd probably be finished already (or CPC wouldn't've even needed to hire us in the first place). But, I'm hopeful either way – having the manual will make it so that we don't have to visit CHP whenever we have a question about how some part of the system is supposed to work.

11.3.2 August 15, 2013

- Another phone meeting with Jeff and Neil... A pretty uneventful one; just debriefing after our trip to Irvine.
  - Neil brought the server machine back to SLO with us, so he could troubleshoot it. He doesn't seem to be able to get things working. It sounds like a classic case of an old/overused computer slowly inching its way to the grave. Or perhaps a power surge destroyed the motherboard/processor/etc. Either way, it looks like we're going to need a whole new machine, which Neil is going to build.
    * I'm not sure exactly how the budget for the project works... Hopefully we'll be able to get the money we need to replace the old computer without having any trouble.
  - We don't have access to the manual yet, but it sounds like everything is in the pipeline – the OC call center seems much more willing to work with us than CHP and SLO PD. They're making sure everything's cleared with the right people, and it sounds like we should have access to the manual soon.

11.3.3 August 25, 2013

- We have our manual! Development on the back end has picked up as a result – Vincent and Nick both seem to be moving forward with what needs to be done on their end, but we haven't heard any new developments from Derek in a while.

- I'm nervous... Derek is our networking guy. None of the rest of us have any experience with them, so we're relying on him to show us the way. We have the network protocols from the previous version, so we can always get started from there and reverse engineer some semblance of functionality, but I'd feel better knowing at least one of us is already comfortable with that type of development.
    * Derek has come out and told us that he hasn't been doing much work on the project, but that's understandable. My workload was rather front loaded – I can make the GUI look like it's supposed to without there being any back end to make it actually do something, but Derek can't really network something if that something doesn't exist yet. I understand that his primary work will start once we have things closer to completion, but still...
    * Right now I'm more focused on getting everything working on one computer and not worrying about using multiple terminals.

11.4 September

11.4.1 September 1, 2013

- I've been getting a lot more experience using GIT, finally. We've reached the point where everyone is starting to need the ability to access each other's code, so things are beginning to come together. Having access to the manual has really helped Vincent's back end development (and mine, when he's needed my help with things), and we've made quite a lot of progress toward a working version that we can show to Jeff and Neil.

    - This has also meant a lot more "Hey, Stuart, I was working on the back end and noticed...." type of emails. It's irritating sometimes, especially if I think I have some free time and I end up needing to tinker with GUI forms and try to make sure that something trivial is working correctly, like making sure nothing overlaps/blocks anything else, or that everything lines up correctly... Though on the bight side, I'm getting my GUI tested on multiple different

computers, resolutions, and operating systems, so I'm getting a better idea of what works across all platforms and what doesn't.

  * The default Mac look and feel doesn't quite agree with some of the GUI forms. Thankfully we don't need everything to be perfect on Mac, since the system that will be running the simulator is all-Windows, but it'd be nice if it worked on both. That way I (or anyone else, for that matter) wouldn't have any issue switching the system from Windows to Mac. Though I'm not entirely certain what that would mean on the server side. I'll have to ask Derek.

11.4.2 September 4, 2013

- Not a lot to mention, but Neil and Jeff told us that we're in the last month of development... Approximately. Vincent and I have a single-machine demo that Neil sounded pretty excited about, but Derek is still dodging questions about progress being made on the network side. He seems to know that the pressure is on now, but I know that he's working on a few other projects at different jobs.

- I've been doing a bit more of back end development now – I'm trying to help take a bit of the pressure off of Vincent, as he's going to be in CSC 357 this quarter, which is an insanely time-consuming class. If we can manage to get everything working bug-free before the quarter starts, I'll be surprised, but I'd rather get as much of it done with him now, before classes start.

11.4.3 September 11, 2013

- It's been a frustrating few days, to say the least. Vincent has done so much back end development in Eclipse that merging my NetBeans GUI code and his back end code has proved rather frustrating. The .form files that are generated by the NetBeans GUI builder makes it easy to click on any object on the stage and get information about it – variable names, what instance variables default to, any HTML/XML tags used to format text, etc. Eclipse can't interpret those .form files, so Vincent is essentially trying to program functionality into a GUI that he can't even see.

- Thankfully, once Vincent recognized the problem, he let me know as soon as he could. It was a simple matter for me to go through all of the forms and make sure that the names of all of the variables follow the naming convention that we established. He was ecstatic when he saw the renamed files. I think he was worried that he would have to go through, figure out which variables were which objects on screen, then rename them all.

  - Still, we could've avoided the entire problem if he had just listened when I told everyone to use NetBeans. Anyway, no harm done. If he really wants to do his work in Eclipse, I don't see the issue, so long as we can solve any cross-environment issues.

    * Good lesson here – There are people who won't take your advice at face value. Make sure coworkers/subordinates know the reasons behind why you tell them to do something in a particular way. Say them loud and often.

11.4.4 September 18, 2013

- Classes have started for everyone else, so I've definitely been the most available to work on the project, but everyone's assignments seem to be coming together nicely. Nick has decoded the XML and figured out how to translate information into it's respective Java equivalent, and Vincent and I are slowly adding some less-important features to the demo. We seem to be on track for getting everything done in time, though I'm still nervous about getting the server set up. We haven't heard anything from Jeff/Neil in a while, so we made sure to send them emails checking on the status of the replacement server Neil was building.

11.5 Early October, 2013

- We had a teleconference today. Finally... We've been going for longer and longer stretches without hearing from Jeff and Neil. I'm not sure how much they're actually supposed to be supervising us, but at the outstart of the project there were regular emails back and forth between them and us and weekly (or close to weekly) conference calls. Now we're

lucky if the turnaround on an email to them is a week. The calls have stopped completely, it seems.

- The VisiCAD training was supposed to start on October 7th, but it seems things have been stalled or pushed back for some reason. We couldn't get a straight answer out of Neil and Jeff, so it would seem that they're as uncertain as we are. As things stand right now, they told us we could shift to a lower gear – Vincent, Derek, and I were all working in 'sprint to the finish' mode – and that we'll have at least an extra month of time to work on the simulator.

  - From what I understand, there's some kind of disagreement/tension between the group that runs the training (the one who contracted CPC to work on the simulator) and UC Irvine, where their offices are located. I'm not sure if CHP or CalTrans are involved, or what exactly is trying to be accomplished, but Jeff made it sound like someone is making a bid for more funding, though he was unclear who.

11.6 Early November, 2013

- Not much of an update on things – I think we've officially been put on hold. Haven't heard anything from Neil and Jeff in almost a month, and emails about our status aren't getting any responses... I'd at least like to know if they're busy with other projects, if there isn't any more funding for the project... Something, at least.

11.7 Late November, 2013

- We're almost to Cal Poly's winter break – which starts a week or so into December. Jeff and Neil got in contact with us about things we should be working on. Since my work on the GUI is essentially complete (other than resolving cosmetic bugs if/when we encounter them) and Nick is finished with the XML schema, they put us together working on a basic 'developers' user manual.' Basically notes for us (and for those who may have to maintain the system in the future) to help each other understand the parts of the system that they didn't personally work on

- Got some more details about the delays. We confirmed that it is about funding, and there's a disagreement between... Either the group who runs the TMC Simulator (who we as developers have essentially had no contact with, other than the one day we visited them) is trying to get more funding from CalTrans, or UCI wants more money for housing the simulator... Anyway, a lack of funds from somewhere is keeping the simulator training from proceeding as Neil thought that it would (he's the one who runs the training). We're still being paid for our hours, but it sounds like we don't have any deadline in the immediate future.

12 XML Schema

12.1 <CAD_INCIDENT>

- Desc - A short description that will appear on the *Simulation Manager.*
- LogNum - A unique number to identify the incident.
- StartTime - H:MM:SS formatted time indicating when during a simulation the incident will occur relative to the beginning of the simulation (0:00:00).

12.2 <INCIDENT_HEADER>

- LogNum - A unique number to identify the incident.
- Type - The CHP code for the type of incident.
- Beat - The beat within which the incident takes place.
- TruncLoc - Truncated street location of the incident.
- FullLoc - Full location (city/county) of the incident.

12.3 <PARAMICS_LOCATION>

- Direction - Northbound, Southbound, Eastbound, or Westbound (NB, SB, EB, WB)
- Location_Type - Mainline, On-Ramp, or Off-Ramp (ML, OR, FR)

12.4 <INCIDENT_EVENT>

- StartTime - H:MM:SS formatted time indicating when during a simulation the incident will occur relative to the beginning of the simulation (0:00:00).
- ORIG_CMD_LINE - Contains a real CAD log entry UI. ([Log#].D/[Details])
- AUDIO - Audio file to play as part of the incident. (optional)
- WITNESS - Witness information if applicable to the incident. (optional)
- TOW - Towing information if applicable to the incident. (optional)
- UNIT - Information about units assigned to the incident. May be used to assign or recommand units, and to mark whether the are primary units, or active/inactive. (optional)
- SERVICE - Service information, may be used to display assigned services. (optional)

12.5 <PARAMICS>

- *This section is optional*
- Status - Status of the road incident (NEW, CHANGED, or CLEARED).
- Incident_type - Type of road incident (LANE_BREAKDOWN (accident) or LANE_OBSTRUCTION (debris, etc))
- Lane_number - The lane number(s) which is directly affected by the incident (lanes numbered beginning from 1, left to right).

12.6 <SCRIPT_EVENT>

- TIME_INDEX - H:MM:SS format indicating the amount of time since the beginning of a specific incident.
- INCIDENT - Incident number and a brief explanation of the incident.

NOTE - A script event will have one or more of the following types: General Info, ATMS, CAD, Radio Maintenance, RADIO TMT, Activity Log, Paging & Faxing, Facilitator, Telephone, Intercom, Radio, TMC Evaluation, or Note.

- GENERAL_INFO
  - TITLE - "Incident Description" or other.
  - TEXT - Written description of the incident.
- ATMS, CAD, Radio Maintenance, RADIO TMT, Activity Log, Paging & Faxing, Facilitator
  - NOTE - Further information regarding what is going on at the current time. (optional)
  - INSTRUCTOR - The action to be performed by the instructor. (optional)
  - EXPECTED_ACTION - The action(s) the students are expected to perform in response to an event.
- TELEPHONE
  - NOTE - Further information regarding what is going on at the current time. (optional)
  - INSTRUCTOR - Role attribute for the instructor, including...
    - TV_Reporter_1
    - TV_Reporter_2
    - TV_Reporter_3
    - Radio_Reporter_1
    - Radio_Reporter_2
    - Radio_Reporter_3
    - Public_to_CHP_1
    - Public_to_CHP_2
    - Public_to_CHP_3
    - Public_to_Caltrans_1
    - Public_to_Caltrans_2

        * Public_to_Caltrans_3
    - EXPECTED_ACTION - The action(s) the students are expected to perform in response to an event.

- INTERCOM
    - NOTE - Further information regarding what is going on at the current time. (optional)
    - INSTRUCTOR - Role attribute for the instructor, including...
        * TMT_Dispatcher_1
        * TMT_Dispatcher_2
        * TMT_Dispatcher_3
    - EXPECTED_ACTION - The action(s) the students are expected to perform in response to an event.

- RADIO
    - NOTE - Further information regarding what is going on at the current time. (optional)
    - INSTRUCTOR - Role attribute for the instructor, including...
        * Maintenance_Dispatcher_1
        * Maintenance_Dispatcher_2
        * Maintenance_Dispatcher_3
    - EXPECTED_ACTION - The action(s) the students are expected to perform in response to an event.

- TMC_EVALUATION
    - EXPECTED_ACTION - Action(s) expected by student(s). (optional)
    - EVALUATE - List of students to be evaluated, space delimited.
        * EvalType - Evaluation code ('None' - no evaluation, 'Completed' - Student completed task (Yes/No), 'Scored' - Student given a graded score (3, 2, 1, 0)).

- Note

- TEXT - Text body of the note.

- CCTV
  - TEXT - Indicates any closed circuit cameras available to view the incident.

12.7 XML Example

<SCRIPT_EVENT>
<TIME_INDEX> 0:00:00 </TIME_INDEX>
<INCIDENT> 187 - Stalled DOT/Traffic Collision </INCIDENT>

<GENERAL_INFO>
<TITLE> Incident Description </TITLE>
<TEXT>

This is a stalled DOT truck on SB 55 overpass at 405. The truck is hit by a vehicle, the vehicle is vaulted over the railing, and a 6 vehicle collision occurs below in the NB 405 lanes. The #2 and #3 lanes are blocked on SB 55 and the #1, #2, and #3 lanes are blocked on NB 405. The driver of the truck is uninjured, although there are 4 fatals, 2 major injured, and 4 minor injured in the collision below. MAIT is called to the scene to investigate the collision and a Sig Alert is declared.

</TEXT>
</GENERAL_INFO>

<MAINTENANCE_RADIO>
<EXPECTED_ACTION>

Expect a call from TMC to Maintenance unit sent but saying that the DOT vehicle stalled (or hit by vehicle) on SB 55 was hit. Send a maintenance unit to check it out.

</EXPECTED_ACTION>
</MAINTENANCE_RADIO>

<CHP_RADIO RadioFile="27401.wav">
<DIALOG>

Dispatch: 14-14, Santa Lucia

Field: Santa Lucia, 14-14. Go ahead.

Dispatch: 14-14, Santa Lucia. 11-25 stalled DOT vehicle southbound 55 overpass at 405.

Field: Santa Lucia, 14-14. Enroute from 55 at Dyer Drive.

Dispatch: 14-14, Santa Lucia copies enroute from 55 at Dyer Drive.

</DIALOG>
</CHP_RADIO>

<TMC_EVALUATION>
<EXPECTED_ACTION EvalType="Scored">

CAD - TMC should try to verify incident using CAD, except unable to until minute 1.

2-WAY - Attempt to verify incident via Caltrans unit in area, beat 3.

</EXPECTED_ACTION>
</TMC_EVALUATION>

<NOTE>
<TEXT> Potential: Stalled DOT SB 55 OVERPASS AT 405 </TEXT>
</NOTE>

<CCTV>
<TEXT> NO CCTV's in the vicinity of the incident. </TEXT>
</CCTV>

<SCRIPT_EVENT>
<TIME_INDEX> 0:03:00 </TIME_INDEX>
<INCIDENT> 187 - Stalled DOT/Traffic Collision </INCIDENT>

<CHP_RADIO RadioFile="27402.wav">
<DIALOG>

Dispatch: 14-14, Santa Lucia.

Field: Santa Lucia, 14-14. Go ahead.

Dispatch: 14-14, Santa Lucia. Cellular 911 caller reports stalled vehicle on the 55/405 overpass was hit and a vehicle was catapulted over the railing. Can you verify?

Field: Santa Lucia, negative. I'm one minute away.

Dispatch: 14-14, Santa Lucia. 10-4. One minute away.

</DIALOG>
</CHP_RADIO>

<NOTE>
<TEXT>

CELL CALL REPORTS DOT TRUCK HIT (CAD UPDATE)

</TEXT>
</NOTE>
</SCRIPT_EVENT>

References

[1] Adzima, Jospeh, *Workplace Communication*, Telephone Interview, April 2011

[2] *Bug Reporting Guidelines*, OpenBravo, November 30 2010, May 2014, http://wiki.openbravo.com/wiki/Bug_Reporting_Guidelines

[3] Dalbey, John, *Software Process Models*, September 2011, Web, June 2013, http://users.csc.calpoly.edu/ jdalbey/308/Lectures/SoftwareProcessModels.html

[4] Heater, S. D., *Is Speed Ever Safe? An Ethical Analysis of Agile Practices in Developing Safety-Critical Software*, California Polytechnic State University, March 2012, June 2013

[5] Horn, D. W., *An Integrated Public-Safety Computer-Aided Dispatch System*, Denver, Colorado, 2005, In-press Master's Thesis Project, Regis University, May 2014

[6] Martin, R. C., *Clean Code; A Handbook of Agile Software Craftsmanship*, Prentice Hall, 2009

[7] Soltis, Andy & Sutherland, Amber *Human error to blame for ambulance delay in death of girl, 4*, New York Post, December 19, 2013, Web, May 2014, http://nypost.com/2013/12/19/human-error-to-blame-for-ambulance-delay-in-death-of-girl-4/

[8] Tritech Software Systems, *VisiCAD Inform.* San Diego, California, 2014, Web, May 2014, http://www.tritech.com/products/inform

[9] *What is the Incremental Model, and when to use it*, ISTQB Exam Certification, Online study materials for ISTQB preparation, June 2013, http://istqbexamcertification.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/