UNIVERSITY OF WISCONSIN-LA CROSSE

Graduate Studies

ONREQUEST NET – CAFM 1.0

A Manuscript Style Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Software Engineering

TJ Teegan

College of Science and Health

May 2006
ONREQUEST NET – CAFM 1.0

By TJ Teegan

We recommend acceptance of this manuscript in partial fulfillment of this candidate’s requirements for the degree of Master of Software Engineering in Computer Science. The candidate has completed the oral examination requirement of the capstone project for the degree.

____________________________________  _______________________
Dr. Kenny Hunt      Date
Examination Committee Chairperson

____________________________________  _______________________
Dr. Kasi Periyasamy      Date
Examination Committee Member

____________________________________  _______________________
Dr. Mao Zheng      Date
Examination Committee Member

____________________________________  _______________________
Dr. Vijendra Agarwal      Date
Director, University Graduate Studies
Abstract


This manuscript describes the methodology used to create a network-based application that has the annotative features found in computer aided facility management (CAFM) software. The project was sponsored by Advanced Planning Technologies Inc. to replace OnRequest, their current facilities management software. The OnRequest software is installed on a single client machine. All data used by OnRequest is also stored on the client. In OnRequest sharing data amongst people is cumbersome. To access another user’s files they must first be transmitted through third party software such as email, ftp, or removable disk. OnRequest Net – CAFM 1.0 stores all data in a database server environment where multiple users can have access to the same data simultaneously. In version 1.0 of this software not all of the features currently found in OnRequest are present, just those dealing with managing database data and the annotation of drawings.
Acknowledgement

I would like to express my sincere thanks to my project advisors Dr. Kenny Hunt and Dr. Kasi Periyasamy for their valuable guidance, and the project sponsor Advanced Planning Technologies Inc. for initiating this project and providing the support for this project. I would also like to express my thanks to the Computer Science Department and the University of Wisconsin-La Crosse. Finally, I wish to thank my wife and sons for their patience and encouragement during the tenure of this project.
# Table of Contents

Abstract ........................................................................................................................................ iii
Acknowledgement ..................................................................................................................... iv
Table of Contents ......................................................................................................................... v
List of Figures ............................................................................................................................. vii
Glossary ......................................................................................................................................... viii
1. Introduction ................................................................................................................................. 1
2. Methodology ................................................................................................................................. 4
   2.1 Prototyping Model .................................................................................................................. 4
   2.2 Requirements .......................................................................................................................... 6
       2.2.1 System Requirements ................................................................................................. 10
   2.3 Prototypes ................................................................................................................................ 10
       2.3.1 Prototype Version 1 ...................................................................................................... 11
       2.3.2 Prototype Version 2 ...................................................................................................... 15
       2.3.3 Prototype Version 3 ...................................................................................................... 16
       2.3.4 Prototype Version 4 ...................................................................................................... 18
       2.3.5 Prototype Version 5 ...................................................................................................... 19
       2.3.6 Prototype Version 6 ...................................................................................................... 23
       2.3.7 Prototype Summary ..................................................................................................... 25
   2.4 Testing Methodology ............................................................................................................. 25
   2.5 Requirement and Design Modifications .............................................................................. 26
3. Limitations ................................................................................................................................... 28
4. Continuing Work ......................................................................................................................... 29
5. Bibliography ............................................................................................................................... 30
6. Appendix ..................................................................................................................................... 31
   6.1 UML Class Diagram for Prototype 1 ..................................................................................... 31
   6.2 UML Class Diagram for Prototype 2 ..................................................................................... 32
   6.3 UML Class Diagram for Prototype 3 ..................................................................................... 33
   6.4 UML Class Diagram for Prototype 4 ..................................................................................... 34
6.5 UML Class Diagram for Prototype 5 ................................................................. 35
6.6 UML Class Diagram for Prototype 6 ................................................................. 36
List of Figures

Figure 1: Prototyping Development Model ................................................................. 6
Figure 2: OnRequest Net – CAFM 1.0 User Interface. ............................................. 14
Figure 3: OnRequest Net – CAFM 1.0 Database Records Tool Pane ....................... 20
Figure 4: OnRequest Net – CAFM 1.0 Color Code Editor ....................................... 21
Figure 5: OnRequest Net – CAFM 1.0 “Space Type” Color Code. ........................... 22
Figure 6: OnRequest Net – CAFM 1.0 Report Authoring ....................................... 24
Glossary

Advance Planning Technologies (APT) – Corporate sponsor of the OnRequest Net – CAFM 1.0 project. Please note the project was initially called OnRequest Web – CAFM 1.0 but the name was changed in December of 2004.

Computer Aided Facility Management (CAFM) – A software system that uses graphical, textual, and numerical data to maintain information about a facility. Typically the graphical data is a building blueprint, and the textual data is entered by users. The numerical data may be either entered by users or derived from the graphical data. The goal of a CAFM system is to maintain information about a facility and to allow significant modifications to its graphical data. For example, a CAFM system could be used to modify the color of any entrance deemed an emergency entrance to red while all other entrances remain black, to color code every carpeted area in a facility to show the age of the carpet, or to highlight fire exit routes for a facility. Typically, CAFM systems track and maintain floor plans, building and property information, furniture, equipment, occupancy data, space characteristics and usage, and safety information.

Data Exchange File (DXF) – A two dimensional vector graphics file format whereby geometric shapes are described as a set of coordinates on the X and Y axis.

Elements – Within the context of this document, an element refers to a geometric shape such as polygon, line, arc, or circle present in a blueprint or vector graphic. Any grouping of elements may also be referred to as a single element.

Icon – Within the context of this document, an icon refers to a drawing which can be imbedded into a SVG drawing. Once inserted an icon be moved, rotated, or scaled.
Prototyping Model – A method of developing software in which requirements are collected and then an incremental design process is implemented. A prototyping process begins by determining a set of functional requirements to implement. The selected requirements are then designed. The design is implemented by reusing any applicable code from the previous prototype, and then adding the newly designed components to create a new prototype of the system. Finally, the prototype is tested. Testing encompasses all new functionality introduced in the current prototype as well as the functionalities implemented in previous prototypes. New prototypes are created until all functional requirements have been implemented.

Scalable Vector Graphics (SVG) – An XML based vector graphics format defined by the World Wide Web Consortium.

World Wide Web Consortium (W3C) – The W3C is an international consortium where member organizations, a full-time staff, and the public work together to develop Web standards. W3C's mission is to lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web.
1. Introduction

The OnRequest Net – CAFM 1.0 project was conceived by Advanced Planning Technologies (APT). The goal of the project was to create a computer aided facility management (CAFM) system to replace the CAFM software sold and supported by the company. APT determined their current CAFM software, called OnRequest, did not provide enough flexibility in sharing information between users. Therefore, APT commissioned this project to replace OnRequest with software that shares data via a network connection, while maintaining or improving upon the features found in OnRequest.

CAFM software extends the functionality of computer aided design (CAD) software to include additional features that provide facility managers with the tools to track and report on facilities information. Typically, CAFM systems track and maintain floor plans, building and property information, furniture, equipment, employee and occupancy data, space characteristics and usage, and safety information.

APT and their clients use CAFM software to perform several tasks such as:

- Generate a report estimating the cost of replacing a building’s roof.
- Determine how much new gravel would be required to re-gravel a baseball diamond.
- Estimate the amount of paint necessary to re-paint a room.
- View all the electrical outlets in a building.
- Keep inventory of furniture and other equipment.
- Print floor plans of buildings showing the emergency exit routes from each of the building’s rooms.
- Track harmful materials in a building such as asbestos.
• Display the age of the carpeting in every room in a building.

The above examples are a brief overview of the types of tasks performed by facilities managers using CAFM. A CAFM system is able to generate meaningful calculations, reports, and diagrams by managing information about the physical traits of a facility.

The market leader in CAFM software is Archibus, INC. They developed ARCHIBUS/FM, which is a suite of integrated applications each focusing on a different aspect of CAFM. The features found in various ARCHIBUS/FM applications allow users to manage property, leases, spaces, furniture, equipment, building operations, emergency preparedness, environmental assessments, fleet management, and room reservations. Archibus customers are expected to purchase only the modules they need. Some of the ARCHIBUS/FM applications require third party software. For example, the “ARCHIBUS/FM Overlay for AutoCAD with Design Management” application requires the AutoCAD software from Autodesk.

OnRequest Net – CAFM 1.0 differs from ARCHIBUS/FM in that it is intended to be a single CAFM solution bundling all of its functionalities into a single software package. Also, the software is being designed with the goal of being easy for novice users to operate with minimal assistance or training from APT, while the ARCHIBUS/FM packages require substantial multi-day training in a classroom environment. APT required that no third party software would need to be purchased by their clients to use the application; however their clients are expected to have their floor plans and other drawings in a digital format before using the software.

OnRequest Net – CAFM 1.0 supports all the basic features found in CAFM applications, such as the ability to view drawings, add elements to a drawing, enter information about any element present on the drawing into a database, retrieve information from the database, generate reports, and use data from the database to modify
attributes of the elements in the drawing. Version 1.0 does not include robust support for creating or editing drawings. It is planned that future releases of the application will add graphical editing features. In version 1.0 of the software it is required that drawings will be created outside of the application and imported into the application for annotation and data entry.

In addition to the annotative functions of the software, APT decided that the application should not store any application data on the client computer, but rather all data should be stored in a database server environment. This design allows users to have access to the same data simultaneously. APT also determined that to provide secure access to the system users will need to log into the system with a username and password. Moreover, different users should have access to different functions. For example, a system administrator would have access to more functions than other types of users.

This thesis describes the development of OnRequest Net – CAFM 1.0. It was developed using a series of prototypes as described in subsequent chapters. Future versions of this software are planned and will be developed using the same approach.
2. Methodology

This chapter describes the methodology used to develop OnRequest Net – CAFM 1.0. Two methods of software development were considered for this project, the waterfall model and the prototyping model. The waterfall model is an iterative process used to create a single build of software. The single build is the complete application meeting all requirements. The prototyping model uses two or more increments through an iterative process to create several builds of the application. Each prototype contains more functionalities than its predecessor, and prototypes are iteratively created until all requirements are met.

APT wanted to be able to show early versions of the software to their customers. For this reason a prototyping process was chosen to develop the application, so that at various stages of the product’s development APT would have the ability to evaluate the software and show prototypes to others.

2.1 Prototyping Model

Prototyping is a method of developing software in which requirements are collected and an incremental design process is then implemented. Typically an iteration through the process would begin by determining a set of functional requirements to implement. The selected requirements are then designed. The design is implemented, reusing any applicable code from the previous prototype to create a new prototype of the system. Finally, the prototype is tested. New prototypes are created all functional requirements have been implemented.
The following iterative development process was decided upon to develop each prototype from the functional requirements. The process is depicted Figure 1 and enumerated in the following list.

1. Consult with client, in this case APT, to determine which functional requirements should be implemented in the current iteration.
2. Perform any necessary research into methodology or technology needed to implement the selected functional requirements.
3. Create a new version of the design document which includes the object oriented design for the functional requirements being implemented in the current iteration, and any existing object oriented design from previous iterations.
4. Implement the new version of the design document. Use all applicable code from the last prototype in the new version of the software.
5. Test the prototype using functional testing methodology, including regression testing.
6. Present the prototype to the client for review.
   a. If there is a problem with any of the added functional requirements.
      i. Review the requirements in the requirements document with the client.
      ii. Make the necessary modifications in the requirements.
      iii. Begin the process from step 2 with the corrected functional requirements.
   b. If the client is dissatisfied with the implementation of any of the added functionalities.
      i. Gather user feedback on the implementation.
      ii. Begin the process from step 3. Modify the design based on the user feedback.
2.2. Requirements

APT provided the *OnRequest WEB – Layout Design*[1] document as the basis for specifying the functional requirements for the OnRequest Net – CAFM project. The document detailed what features were expected for two versions of the application. Version 1.0 would provide the framework allowing users to log into the system, to retrieve data from the database server, to annotate drawings, to enter data about drawing elements, and work with data. The design document also described a successor version of the software would add tools to the application allowing the user to edit and create
drawings. A requirements document was created for version 1.0, but not version 1.0’s successor.

Each feature detailed in APT’s documentation was broken down into several functions. For example, the Icon Library feature was divided into arbitrarily named functions such as initialize icon library, display icon library, search icon library, select icon from library, and insert icon into drawing. As each function was identified, it was compared against other previously identified functions. Functions with similar or overlapping functionalities were combined. In the previous example, the insert icon into drawing function would eventually be combined with other similar functions to create the more general insert element into drawing function.

Each of the identified functions was placed into one of the following five categories.

- General Software Requirements – Requirements for basic system operation or functional requirements. These requirements are somewhat generic that they could not be fit into any of the other four categories.
- Graphical Editing and Annotation Software Requirements – Requirements for manipulating graphical data and adding various elements to a drawing.
- Database Software Requirements – Requirements for editing and manipulating data, generating reports from data, and deriving data through user created formulas and functions.
- Account Management Software Requirements – Typical administrative level requirements dealing with managing user accounts and access to the system.
- Graphical User Interface Requirements – Requirements detailing the functionality of various graphical user interface components.

APT was consulted to ensure the designer’s interpretation of the functional requirements were congruent with the collection of requirements detailed in OnRequest WEB – Layout Design. Over the course of several conversations, the requirements were
refined to reflect the original intent of the functionality mentioned in APT’s documentation.

All identified functional requirements are specified in *Software Requirements for OnRequest Net – CAFM 1.0*[2]. The following list summarizes the most important functional requirements.

- The system should be secured such that only users with a valid username and password will have access.
- The application should connect to a remote database where all data is stored, including drawings.
- Drawings must be able to be downloaded to a client machine from the database where they should be able to be viewed and manipulated.
- A user should be able to save changes to a drawing.
- Each drawing must be comprised of several layers. Individual layers should be able to be hidden, revealed, or displayed at reduced opacity.
- Each individual element in the drawing should be selectable by the user.
- Each element in the drawing should be able to be locked or unlocked. A locked element is still selectable, but cannot be modified.
- An entire drawing and layers within a drawing should also be able to be locked or un-locked.
- A user must be able to zoom into or out of a location on a drawing.
- A user must be able to pan up, down, left, or right on a drawing to view a particular aspect of the drawing.
- New elements should be able to be inserted into a drawing by a user. The user must be able to modify the newly inserted elements.
- Elements in a drawing should be able to be moved to different locations within the drawing.
- Elements should be able to be rotated, made larger, or made smaller.
• The order of elements should be able to be changed such that one element overlaps on top of another.
• Elements in a drawing must be able to be moved to specific coordinates on the drawing or on other elements.
• A library of graphical icons should be provided, where an icon is an individual element.
• The icon library should be searchable by keywords.
• Drawings must be able to be annotated with text elements.
• A user must be able to designate the color of individual elements, whereby the color is determined by criteria supplied by the user. These are referred to as color schemes.
• Color schemes should be able to be created by the user and saved for use again in the future. Color schemes must also be able to be deleted.
• Drawings in DXF format should be able to be imported into the system.
• Records in the database tables should contain information about elements in a drawing.
• A user must be able to identify the elements that have records in the database.
• A user must be able to edit the data that references an element.
• Reports should be able to be generated from data derived from the database.
• Once a report is authored, the underlying queries and formatting used to generate the report must be able to be saved for future use. Reports should also be able to be deleted.

Before the initial prototype was designed and implemented the requirements document was submitted to the client, APT, for review and was accepted.
2.2.1. System Requirements

It was required that the software must be able to run on a PC under Windows 98/2000/XP. The PC must have at least a 1GHz processor speed, 256 MB of RAM, and a network connection for connecting to a remote database. The system requirements were purposely set at a low level to accommodate laptop computers and older desktop machines. The software was being tested on a PC running Windows 2000 and matching the minimum requirements.

Initially, it was decided that APT would be responsible for hosting and maintaining the database server. Once the software was established, APT planned to offer hosting as an option for their clients, but also allow them to host their own database server on site.

2.3. Prototypes

An object oriented design document was created during each iteration. The final design is specified in Design Document for OnRequest Net – CAFM 1.0[3]. Listed below are some of the decisions made during the designing of the application.

- The application would be written in Java.
- Related Java classes would be grouped into Java Packages.
- UML class diagrams would be developed for each prototype.
- The native format for all graphics in the application would be Salable Vector Graphics (SVG).
- All viewing and editing of SVG graphics would be accomplished using the Batik SVG Toolkit[4].
- A MySQL database server would be used during the development of the software.

The following sections detail the six prototypes deployed during this project. A UML class diagram for each prototype can be found in the appendix of this document.
2.3.1. Prototype Version 1

The basic framework of the system was implemented in the first prototype. For example, a user could log into the system, open drawings, zoom in on a drawing, and view different parts of a drawing by panning up, down, left, or right. No database server was integrated at this point, and no networking protocols were implemented. Temporary flat files were used to mimic the type of relationship between the database server and the application.

It was decided that Scalable Vector Graphics (SVG) would be used to render drawings. SVG was chosen because it is well defined by the World Wide Web Consortium, and tools are readily available for working with SVG, such as the Batik SVG Toolkit from the Apache XML Project[5]. Also, SVG drawings can be expressed as simple text which is easy to compress for quick transfer over a network.

The Batik SVG Toolkit contains graphical user interface components which are used to display and work with SVG. These components were integrated into the design of the first prototype and became a prominent part of the user interface for the duration of the project.

The prototyping model was particularly useful while designing the graphical user interface. Allowing the client to see and use the application facilitated discussion between the client and developer to design an interface that met their goals.

One of the early decisions made through the prototyping process was to eliminate the necessity for the software to be run within a web browser. It was initially assumed that the software would be a true web based application perhaps housed within a java applet, or similar executable, and embedded in an HTML file. APT was displeased with the early user interfaces created in this fashion and identified three major problems.
• First, in an application that deals primarily with graphics, users like to have plenty of screen space. By running the software within a web browser, approximately 10% of the usable screen space was lost to the web browser’s graphical user interface, giving the user less screen space to work with their drawings. It should be noted that currently most web browsers allow the web browser’s controls to be hidden from the user via scripting embedded within HTML. Making the web browser’s control unavailable to the user would alleviate this concern. However, there was no guarantee that this functionality would always be available in the future. Therefore, an assumption was made that this particular problem would be faced at sometime during the application’s life cycle.

• Second, the user has to deal with two separate sets of controls. One set of controls for the web browser, and another for the application’s controls. APT expected this to cause confusion amongst users. For example, users might try to use the web browser’s “Back” function at inappropriate times. This may lead to lost work as the web browser goes back to a previous web page and terminates the application prematurely.

• Lastly, any changes or updates to the user’s web browsing software could potentially cause the application to function poorly or no longer function at all. For example, the user could install a version of the Java Virtual Machine that is not supported by the application. Any changes to the user’s web browsing environment could lead to support issues, frequent software updates, and in some cases an unacceptable amount of system down time while software patches are developed and tested.

For these reasons it was decided that OnRequest Net – CAFM 1.0 would be a self contained executable with the ability to connect remotely to a database server.
APT had strong opinions about how complex the interface should be. Specifically, APT did not want the user to be overwhelmed with controls at any given time. They had noticed that other CAFM systems, including their current system, typically allowed several tool screens to be open at any time. This had caused software support issues for the company in the past. It was therefore decided to limit the amount of controls available to the user at a given time. To accomplish this goal, a single tool pane was designed. The controls available on the tool pane are contextually determined by the choices the user has selected from the controls at the top of the screen. In this way, a user can quickly switch between tool sets within the system, but is limited to working with a single set of tools at a time.

Another issue concerned displaying multiple drawings to the user. Rather than designing a traditional multi-window interface, a tab based system was implemented. As a drawing is opened, a tab with the drawing’s name is displayed at the bottom of the screen. A user can switch between multiple drawings by clicking a desired drawing’s tab. The final user interface is pictured in Figure 2.
Figure 2: OnRequest Net – CAFM 1.0 User Interface.

The tool pane is oriented at the left-hand side of the screen, toolbar at the top, and the drawing tab filling the remainder.
2.3.2. Prototype Version 2

The second prototype focused on ensuring that the SVG file format would be compatible with the other software components. Because the drawings APT uses were in DXF file format, it became evident that the software must provide a way to convert the existing DXF files into SVG files. Initially it was hoped that third party software could facilitate the file conversion. However, investigations into third party software packages did not yield any suitable results. One problem encountered was due to the way curves were expressed in the DXF files APT currently had for all of their clients’ drawings. The third party software was unable to convert these curves properly. This led to some graphical elements being lost during the conversion process. Another problem was with the methodology used by most vendors to generate an SVG file from a DXF file. Third party software tended to merge all of the individual line elements in the DXF file into a single path element in the SVG file. Therefore the ability to identify and work with any single line element within the SVG could be lost. This was in direct conflict with other requirements of the software. For example, it was specifically required that each element in the drawing must be able to be selected by the user, and also be available to have information about the individual element entered into the database. It was eventually determined that the ability to convert DXF files into SVG files would be developed in house.

It was also decided that the graphical elements that could be inserted into a drawing by the user, referred to as icons, would be housed in individual flat files and then inserted into the SVG file as a referenced object. Initially it was thought that each icon would be inserted into the SVG drawing as a group of elements. However, there were advantages to using the ability to reference other graphics files within SVG. APT would then have the ability to modify one of the icons in its library and have the modification instantly realized through all of their clients’ drawings with no additional work for the end users. In APT’s current system if an icon was modified, the old occurrences of the icon would have to be deleted and replaced with the new icon manually.
Another advantage would be decreased SVG file size. File size was a major concern because the user’s drawings were to be stored in a remote location on a database server. APT was interested in decreasing the amount of time it would take for a user to retrieve a drawing from the remote server. If every icon was inserted into a drawing as a new group of elements, then each time an icon was inserted all of the elements comprising the icon would be recreated within the drawing. Each icon may have anywhere from 1 to over a 1000 elements, and typically each drawing would have hundreds of icons added to it. This could cause the SVG files to become excessively large, and to difficulties uploading and downloading drawings in a timely fashion. By using the object referencing feature of SVG, it was guaranteed that each time an icon was inserted into a drawing only, one new element would be added, thus greatly reducing file sizes and transmission time.

In addition to adding the capability to convert DXF files into SVG files, the second prototype introduced the icon library that would allow the user to search for an icon by keyword and category, and see a preview of all of the icons returned by their search.

2.3.3. Prototype Version 3

This version of the prototype focused on expanding the icon library’s functionality to include the ability to insert icons into a drawing and manipulate them. The decision to use referenced objects as icons dictated that a single SVG element referencing the icon must be placed in the SVG document. An icon is a graphic file that can be referenced multiple times in any SVG document.

The Batik SVG Toolkit provided the tools necessary to create the new element and insert it into the SVG document. The Batik SVG Toolkit also facilitated the modification of an element’s affine transform to perform basic functions such as translate, rotate, and scale.
APT asked that the user be able to select an icon from the icon library’s preview area, and then drag the icon onto the drawing and place it where desired. Users should be able to select both icons and regular SVG elements in the drawing, such as polygons and lines. It should be possible to select multiple objects; however only icons should be allowed to be moved, scaled, rotated, and deleted. When an element in the SVG document is selected, there must be a graphical cue to the user to let them know which elements are in the current selection.

It was decided when a user selects a polygon or line that the formatting of the element would be temporarily changed. All lines and the fill of the selected element would be set to red. Also, the lines would be thickened to make the selection more pronounced. Once the element was de-selected, the original line and fill formatting would be restored. However, this solution would not work for icon selection. Because icons were implemented as independent pictures being inserted into the SVG document, it was not possible to directly modify the referenced picture’s lines and polygons. A solution was implemented to create a temporary semi-transparent red rectangle with the same width and height of the selected icon and then insert the rectangle into the SVG document directly on top of the selected icon. Once the icon was de-selected, this temporary rectangle could be discarded.

During testing it was realized that selection was difficult to use in certain circumstances. For example, selecting an individual line on the drawing was impossible unless the user was using the zoom function and viewing the line at a higher scale. Also, an individual line might be located beneath one of the lines comprising a polygon making the selection of the individual line unworkable. The solution to this problem required considerable re-designing of the selection feature.
Initially selection was performed by using tools within the Batik SVG Toolkit to determine which element was underneath the mouse’s pointer when the left mouse button was pressed. It had to be modified so that elements near the mouse’s pointer that were very small would also be recognized. The solution was to examine all elements within a specified distance from the mouse’s pointer, determine which element was closest, and then select the closest item. While this solution worked in several situations, it did not solve all of the problems. It failed when two elements overlapped and the user wanted to select the bottom-most element. The selection feature was further expanded to address this issue. In addition to determining what elements were close to the mouse’s pointer, the ability to cycle through all elements near the pointer was added. For example, if elements A, B, C, and D are all near the mouse, and they are assumed to be sorted by distance from the mouse’s pointer. At the time of selection, the user can cycle through those elements by holding down the <Shift> key and continually pressing the mouse’s left button. These additional changes resolved the above problems with the selection feature.

2.3.4. Prototype Version 4

In the third prototype the ability to select an icon in a SVG document and move it to a different location using the computer’s mouse was introduced. The current implementation did not meet all of the functional requirements; it lacked the ability to move an icon or group of icons to a predetermined point on a line or polygon via the computer’s mouse. Internally this feature was referred to as “sticky walls”.

The goal of the fourth prototype was to enhance the selection of icons. In this prototype, when an icon is selected, any point on that icon could be chosen by the user. As the icon is moved over a line or polygon, the endpoints, vertexes, and midpoints on the line or polygon would be highlighted. If the icon is close to any of the highlighted points on the line or polygon, the icon would be placed in such a way that the point the user chose on the icon would match the highlighted endpoint, vertex, or midpoint. A
A typical example of this functionality being icons representing doors placed into open doorways on a blueprint. In this example, when the door icon is selected the user may chose one of its corners. Then, when placing the icon in a doorway, the user could match the corner of the door icon to the corner of the open doorway. This would give the user a way to ensure precision when placing icons on a drawing. While this functionality was well understood the implementation was difficult. The Batik SVG Toolkit lacked some of the tools required to implement this feature. Several designs were tried, and each one encountered significant problems. Fortunately, a new version of the Batik SVG Toolkit was released while version four of the prototype was being developed. The updated toolkit contained the tools required to accomplish the design goals. Specifically, it implemented a feature that allowed the application to recognize the distance of the icon from any particular point on any SVG element. Once the new tools were integrated into the prototype, the “sticky walls” feature was completed.

### 2.3.5. Prototype Version 5

This was the appropriate time in the application’s development to implement a database server, and to design some of the data related functional requirements. At this point, APT felt enough of the graphical features were present to demonstrate the software. However, before the software could be shown to clients, it was necessary to be able to demonstrate the data functions of the application. Specifically, it was determined that the ability to enter information about an element in a drawing, and the ability to modify elements in the drawing by querying the database were required for this prototype.

A third party database server package from MySQL[6] was chosen. MySQL was selected because it is available freely, and met all of the requirements. It was determined that features specific only to the MySQL database server package would not be used. This was to prevent any compatibility issues if the data was migrated to a database server from a different vendor in the future. Tools available from MySQL facilitated
communication between the application and the database server. APT defined the tables that were to be used.

It was required that the user be able to select an element in a drawing and be able to view or modify data about that element in database tables specified by APT. In APT’s current system, when a user views an element’s data all of the data is viewed at once on a single screen. It was preferred to design and implement a different method for viewing and editing data. As a result, a tree based design was implemented in which the data for each table could be expanded or collapsed by the user and data is editable by selecting a node and then making modifications. This functionality is depicted in Figure 3.

![Figure 3: OnRequest Net – CAFM 1.0 Database Records Tool Pane.](image)

*The tool pane is orientated at the left-hand side, and is displaying data for the selected element in the drawing tab.*
Once data was entered for an element, it was required that the data be available for queries written in Structured Query Language (SQL). Elements returned by these queries could then have their color temporarily modified to convey information about a drawing. For example, if the elements that denote an emergency exit are specified in a query, then the database is expected to return the unique identification of those elements and their color can be temporarily modified according to user specification. This feature is referred to as “color coding”.

A user interface was developed for color coding that would assist the user in constructing SQL statements and then allow them to assign a color to the results of each query.

![Figure 4: OnRequest Net – CAFM 1.0 Color Code Editor.](image)

*The name of the color code is entered at the top. Listed in the center are SQL statements created using the controls located at the bottom.*

Once a color code is created, it can be applied to a drawing. Figure 5 illustrates the result of executing the color code pictured in Figure 4 on a drawing containing elements
that meet the criteria. In this example, any element classified as a classroom is colored pink, those classified as a gym are colored yellow, and those classified as IMC are colored blue. The user can stop execution of a color code at any time returning the drawing to its default state.

Figure 5: OnRequest Net – CAFM 1.0 “Space Type” Color Code.

The tool pane displaying the color code menu is orientated at the left-hand side, and a drawing which is being modified by the color code depicted in Figure 4 fills the remainder of the screen.
2.3.6. Prototype Version 6

The data features of the application were further developed in version 6 of the prototype. The ability to author reports from data in the database was added. It was initially thought that the report feature would be fully integrated into OnRequest Net – CAFM 1.0. However, it was suggested that users should have the ability to either generate reports within the application, or be able to execute the report components as a separate executable. The reason for this change is that while many users do work with the graphical and data entry aspect of APT’s current CAFM system, some users may only access the report writing features. As a result, the report writing functional requirements were implemented in such a way that the report writing component could either be accessed from within the application or launched separately. When executed as a stand alone application, the report writer would require the user to log into the system before any data would be accessible.

The report authoring package DataVision[7] was integrated into the system. DataVision was chosen because of the drag and drop nature of its user interface, the flexibility it offered for custom modifications, and its robust set of features including the ability to create functions and formulas. Report authoring is shown in Figure 6.
Figure 6: OnRequest Net – CAFM 1.0 Report Authoring

Report authoring tools are displayed in the foreground. Data fields are chosen from the window orientated on the right-hand side, and then dragged using the mouse onto the report layout in the center of the screen.

After a report is authored it can be saved or printed. The report’s specification can be saved to the database for reuse.
2.3.7. Prototype Summary

The following tracks the progression of the OnRequest Net – CAFM 1.0 project based on each of the six prototypes discussed in the previous sections.

<table>
<thead>
<tr>
<th>Prototype#</th>
<th>#Classes</th>
<th>Lines of Code</th>
<th>Requirements Met</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17</td>
<td>~6000</td>
<td>36</td>
</tr>
<tr>
<td>2</td>
<td>21</td>
<td>~8000</td>
<td>43</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>~8500</td>
<td>50</td>
</tr>
<tr>
<td>4</td>
<td>23</td>
<td>~9300</td>
<td>62</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>~10500</td>
<td>76</td>
</tr>
<tr>
<td>6</td>
<td>31</td>
<td>~12500</td>
<td>82</td>
</tr>
</tbody>
</table>

Please note that classes dealing with the user interface are not counted in the above table. However, the lines of code from the user interface classes are factored in. Also, classes and code extended or derived from third party sources, such as the Batik SVG Toolkit, are not entered in these tabulations.

2.4. Testing Methodology

Testing was performed during each iteration of the prototyping process. Initial testing was performed during the research phase to determine if an idea for implementation would be feasible. It is important to note that in the research phase tests were not focused on functionality. The testing during the research phase was used to determine if a certain methodology or technology would be useful in meeting a functional requirement. For example, extensive testing was completed before determining whether icons should be implemented as referenced objects in an SVG document rather than as a group of graphical elements. APT had desired to implement icons as a grouping of elements with the hope that future versions of the software would include tools to modify the graphical
elements of an individual icon in a drawing. However, testing revealed that there were more advantages to using referenced objects. While the ability to modify an individual icon would be lost, referencing resulted in smaller SVG file size, and the ability to change an icon throughout the entire system by modifying the referenced icon’s drawing. Through this testing, APT determined that smaller file size and improved system wide icon management outweighed losing the ability to modify an individual icon embedded in a drawing.

Before the completion of each prototype functional testing was performed first by the developer and later by APT. Functional tests were designed to determine whether a component of a prototype met a specific requirement. Typically, tests were evaluated as either passing or failing to meet a requirement. For example, one test was to determine whether or not the application could successfully meet the requirement of opening a drawing. The test was performed by attempting to open an SVG file within the application. The test was passed once it was determined a selected drawing was being rendered within the application.

Typically, new functionalities were tested most thoroughly with regression testing being performed by APT during their evaluation. Occasionally APT’s feedback during testing led to a design change. For example, the decision to create a stand alone application rather than an executable run within a web browser was the result of APT evaluating the software and suggesting a change to the design.

2.5. Requirement and Design Modifications

Initially it was thought that a customized system would need to be developed to create mathematical functions to perform calculations against data in the database. Research into the problem led to the discovery of open source database reporting components that could be used to fulfill this requirement. As a result, Software Requirements for OnRequest Net – CAFM 1.0 was modified to show that there would no longer be a
customized system developed, and the functional requirements dealing with database calculations were merged with the requirements detailing the database reporting requirements.

*Software Requirements for OnRequest Net – CAFM 1.0* contains an Appendix section detailing all changes in requirements made during the tenure of the project. The original design called for a database reporting utility to be developed. However, there were several open source solutions for database reporting. It was decided to use one of these open source solutions that met all of the functional requirements rather than develop a new report writing system. The *Design Document for OnRequest Net – CAFM 1.0* was reworked to remove the customized system and include *DataVision – The Open Source Report Writer*.

*Design Document for OnRequest Net – CAFM 1.0* contains a Document History section which details the modifications made to the document during each phase of the prototyping process.
3. Limitations

This section enumerates some of the limitations of the current implementation OnRequest Net – CAFM 1.0. All functionalities referred to in this section are described in *Software Requirements for OnRequest Net – CAFM 1.0.*

- The user interface for Layer Sets has not yet been implemented. However, the functionality does currently exist in the design document and the code is present in the final prototype.
- The user interface for Zoom in and Zoom out has not yet been implemented. However, the functionality does currently exist and is accessible using the keyboard.
- The user interface for Panning has not yet been implemented. However, the functionality does currently exist and is accessible using the keyboard.
- The Search functionality remains to be implemented.
- The user interface components for moving, rotating, and scaling elements have been partially implemented. However, the functionality does currently exist.
- The user interface does not limit access based on user type. Currently, anyone who logs into the system has access to all of the functionalities including administrative functions regardless of user type.
- The ability to scale objects in a drawing remains to be implemented.
- The ability to measure objects in a drawing remains to be implemented.
- The ability to change units of measurement in the application remains to be implemented.
4. Continuing Work

The application continues to be developed by APT. APT plans to conduct a beta test with one of their current clients. During this testing, network traffic to the database server will be monitored to determine what type of network resources will be required when the software is released to all of their clients. Also, feedback will be collected from the beta testing to determine if any modifications to the requirements or design are necessary.

A successor version of the software is still in the planning stages, but the basic functionality is described in *OnRequest WEB – Layout Design*. 
5. Bibliography

6. Appendix

6.1 UML Class Diagram for Prototype 1
6.2 UML Class Diagram for Prototype 2
6.4 UML Class Diagram for Prototype 4
6.5 UML Class Diagram for Prototype 5
6.6 UML Class Diagram for Prototype 6