MATH 348.00 Software and System Design

Mount Vernon City Water Archival Database

Software Architecture Document

Version 1.0

Revision History

Date	Version	Description
<02/25/2020>	<1.0>	First Draft of the Software Architecture Doc.

Table of Contents

Software Architecture Document

1. Introduction

This introduction provides an overview of the entire *Software Architecture Document* for the Mount Vernon City Water Archival Database. It includes the purpose and scope of the project, relevant definitions, acronyms, abbreviations, and references, and an overview of the system.

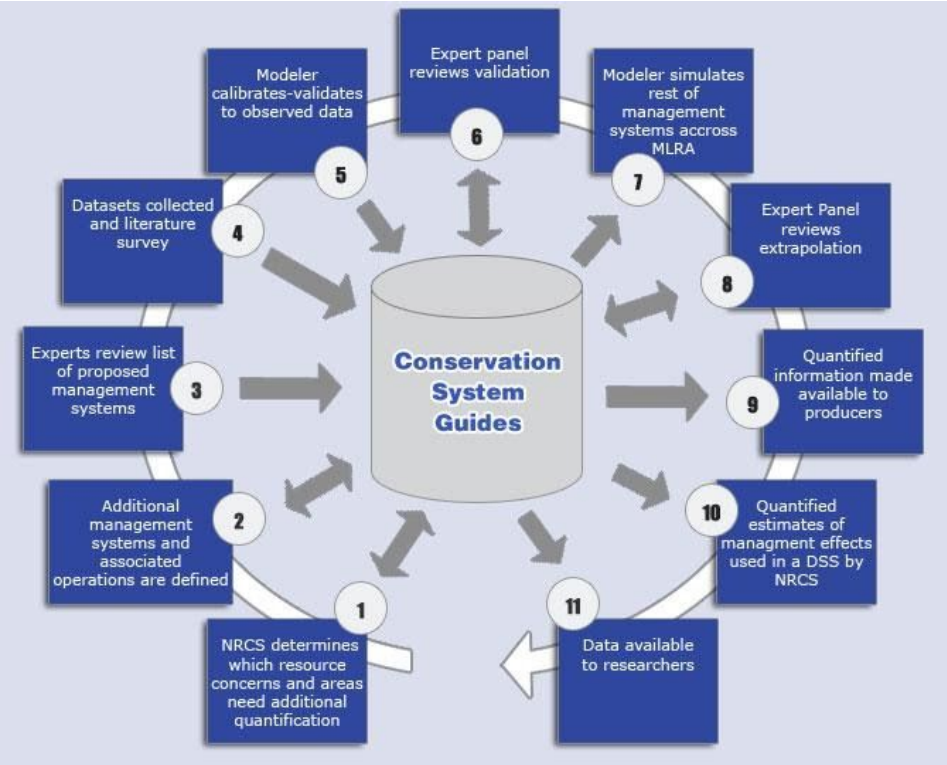
1.1 Purpose

This document provides an architectural overview of the Mount Vernon City Water Archival Database (MVAD).

The primary purpose of the MVAD is to serve as a read-only database, allowing employees of the Mount Vernon Water and Wastewater department to search for and view customer data from before the year 2016.

This document is intended to convey the important architectural decisions made in designing and building this system. The document is intended to help those involved in the project better understand the problems to be solved and how they will be represented with the system.

The following diagram shows the different steps that will be taken to set and get information from the MARIA database effort:



1.2 Scope

The scope of this document is the implementation of the MVAD.

1.3 Definitions, Acronyms, and Abbreviations

See the System Glossary.

1.4 References

The reference book used to model all the architectural diagrams was: *The Unified Modeling Language Reference Manual;* James Rumbaugh, Ivar Jacobson, and Grady Booch.

1.5 Overview

This document consists of 7 sections, which are described below:

- Section 1 is simply an introduction to the software architecture of the MVAD.
- Section 2 addresses the goals and constraints of the system's architecture
- Section 3 describes the architectural representation of the system.
- Section 4 describes the five views in which the system documentation is divided by following the Rational Unified Process (RUP).

- Section 5 of this document talks about other system considerations such as size and performance of the system.
- Section 6 describes some system quality issues.
- Section 7 is a bibliography of the references used to create this document.

2. Architectural Goals and Constraints

The archive system architecture has been designed with the following objectives in mind:

1. To enable easy access to historical archive information that has been collected for the last ~25 years on a per-account basis.
2. To allow users to easily search for large numbers of accounts at once, given only an address OR an account number.
3. Possibly to enable easy generation of reports, which may contain any manner of information, in an easily accessible, human-readable, and machine accessible format.

The major design and implementation constraints for the system are:

1. Flexibility
2. Adaptability

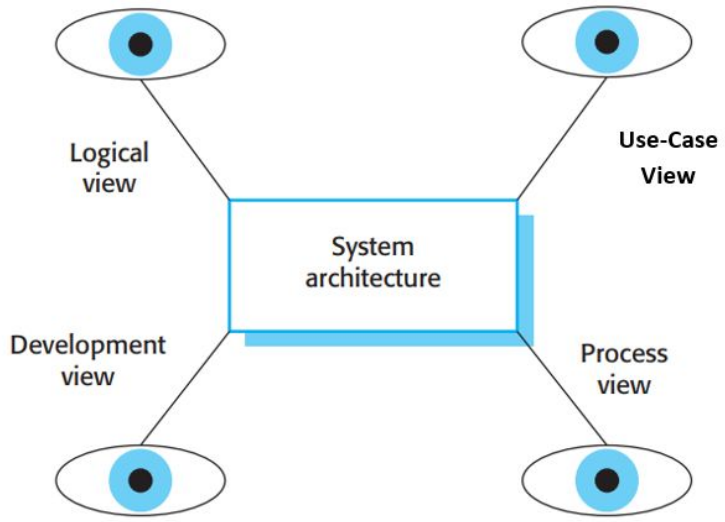
A full list of the system requirements can be found in the system requirements document.

3. Architectural Representation

3.1 Architectural Views

Architectural models of a software system are used to highlight details of the software requirements and design. Due to the difficulty of representing all of the relevant information about a system's architecture in a single diagram, system architecture is often broken down into different views or perspectives. The number of these views can be different relative to the software goal, however it is commonly accepted that there are four fundamental architectural views.

The following is a brief description for each of the views:



Adapted from Sommerville

- Use case view: Describes the set of scenarios and/or use cases that represent some significant, central functionality of the system.
- Logical view: Shows key abstractions in the system as objects or other classes including relationships between the entities within the system requirements
- Process View: Non functional requirements: describes the design's concurrency and synchronization aspects. This view will display the processes that form the systems' mechanism. These will be represented as collaboration, sequence, and activity diagrams.
- Deployment View: Describes the mapping of the software onto the hardware and shows the system's distributed aspects.

3.2 Architectural Design Patterns

The design pattern used to create the MVAD is in line with the MVC (Model View Controller)

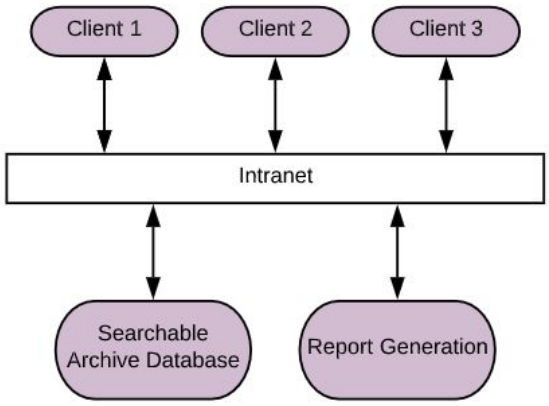
The design pattern used to create the CPSS is the MVC (Model View Controller) design pattern. The MVC design patter clearly separated the web application's behavior, presentation and control. The modularity of this design pattern allows for easier code reuse, more centralized control, bugs easier to track down and code easier to modify. The presentation, or view, of the CPSS has been implemented keeping in mind the Model 2 usage pattern, which makes use of servlets as front controllers and maps incoming request to specific operation and selects views based on the model and session state. In some steps of the application the Model 1 pattern, in which a servlets is not used as the front controller, is also used. Some of the sections of the CPSS where Model 1 architecture is used are the DB Input For and the CPPE tool.

These tools are very simple as far as navigation and a front controller is not needed since the controller is implied through the higher level container (the CPSS).

3.3 Architectural Style

We went with a Client-server architecture style where the system is presented as a set of services with each being delivered by a server. The clients will use this server to retrieve their archived data.

A brief description of the model and how we wish to implement it can be found here:



3.4 Architectural Process

MVAD follows the Rational Unified Process (RUP), whose goal is to enable the production of the highest quality software that meets end user needs within a predictable schedule and budget. The RUP is an iterative process divided into four phases: inception (the establishment of the projects business case), elaboration (establishment of project plan and system architecture), construction (the system implementation), and transition (system deployment).

During the inception phase, the business case and vision documents will be defined for the MVAD. The business case includes the criteria needed for the successful development of the system, and a schedule of the major milestones. The requirements document defines the project's problem to be solved, the project's purpose and the project's major stakeholders. During this phase, a PowerPoint presentation for the entire system will be created which will serve as a proof-of-concept for the system.

During the elaboration phase, the MVAD will have an easy to understand software architectural foundation and an implemented project plan. As previously described, the architectural foundation is composed of a set of UML diagrams that entirely describes the system's functionality.

During the construction phase, the MVAD will be incrementally and iteratively developed and tested. The development implies the complete coverage of the software requirements by following the different UML diagrams defined in the previous phase. At the end of this phase, the MVAD should be a completely

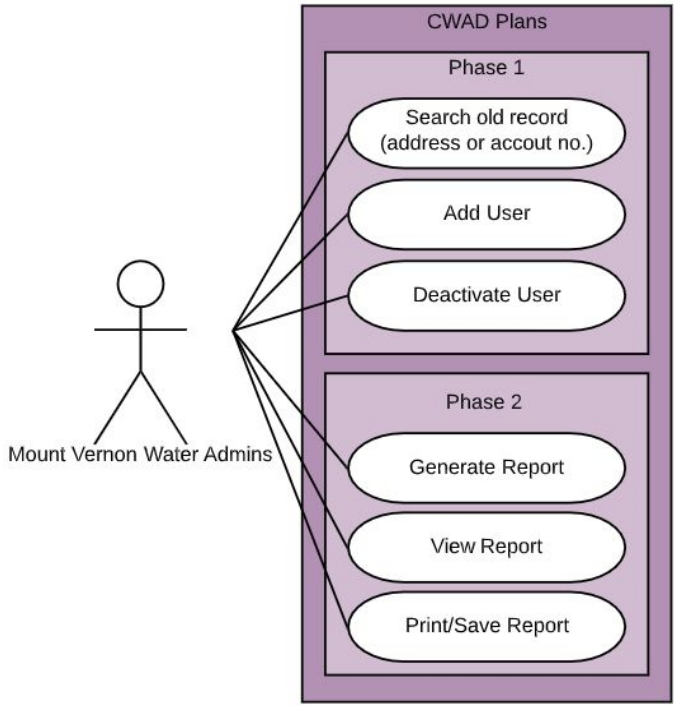
designed, implemented, and tested system that is ready to be deployed for use.

Deployment of the MVAD will occur during the transition phase where the system will be placed on a server for the use of database administrators.

4. Architectural View Decomposition

4.1 Use-Case View

The general functionality of the MVAD can be seen in the following diagram:



The MVAD will implement the architecturally-significant subset of the use cases specified above, which include the items in Phase II if there is time.

4.1.1 Use Case details

Detiails on each use case can be found in the ***Software Requirements Specification***.

4.1.1.1 Architecturally Significant Use Cases

The client-server architecture enables user-interface

4.2 Design View

This section describes the logical structure of the system as well as any definitions or acronyms. It starts from the overview of the architecture and then presents its key structural and behavioral elements such as usage and dependency. The documents that make up the logical view for the software system are the system glossary and the static data-defined model.

4.2.1 System Glossary

Term	Definition
Active actors	Types of people interacting with the final software system
Active entry	The account displayed after a search is made and a record is requested
Cooperating System	The working system
Database	Collection of all information monitored by the system
Docker®	A containerization technology that allows specialized applications to be run inside of a dedicated environment, regardless of the host
Field	A cell within a form
Host	The server or computer that the software runs on
CMI® UtyX©	The existing records database system, to be retired
Entry	An account number along with the information associated with it
Software Requirements Specification	A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document
Stakeholder	Any person with an interest in the project who is not a developer
User	Any person using the system to view entries

4.2.2 Static Data Defined Model

4.3 Process View

The process view will describe the system's decomposition as well as the forms of communication between processes, like message passing, activity between components, and message sequencing.

The documents that are incorporated into this view are:

- Collaboration Diagram.doc
- Sequence Diagram.doc
- Activity Diagram.doc

4.4 Component View

The component view will describe the overall component and subsystem organization of the CPSS. This document will reside in the Component View folder.
The documents included in this view are the following:

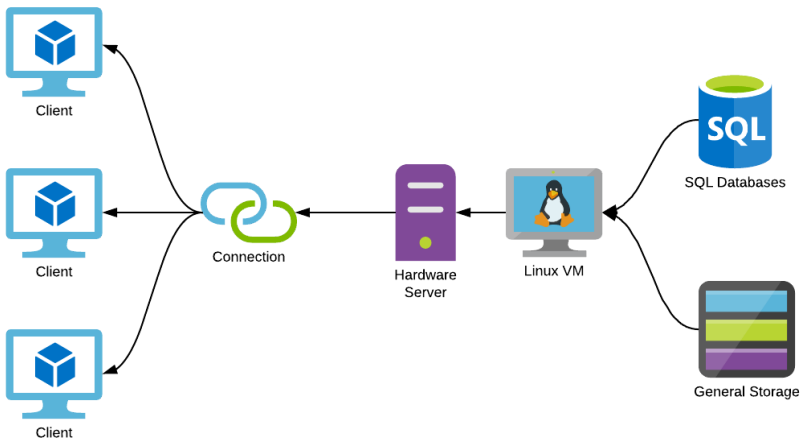
- User Interface Specifications.doc
- System Interface Specifications.doc
- Software Architecture Documentation.doc (this document)

4.4.1 Overview

This subsection will include the various software layers that will exist in the system as well as the boundaries between these layers.

4.5 Deployment View

The deployment view of a system shows the physical nodes on which it executes. The system is comprised of three physical nodes: the browser client, the application server, and the database server. The simplicity of the physical view can be seen in the diagram below:



5. Size and Performance

The Mount Vernon City Water Archival Database not yet finished, but the size of the system can be pre-summarized as follows:

- Number of use cases implemented: 5
- Number of CGI searches: 2
- Number of VMs: 1
- Required PHP extensions: 4
- Disk space requirements: 10 GB
- RAM requirements: 2 GiB
- vCPU cores required: 2 vcores

6. Quality

System quality issues or concerns will go here.

7. Bibliography

James Rumbaugh, Ivan Jacobson, Grady Booch. 2000. The Unified Modeling Language Reference Manual. Massachusetts. 550 pages.