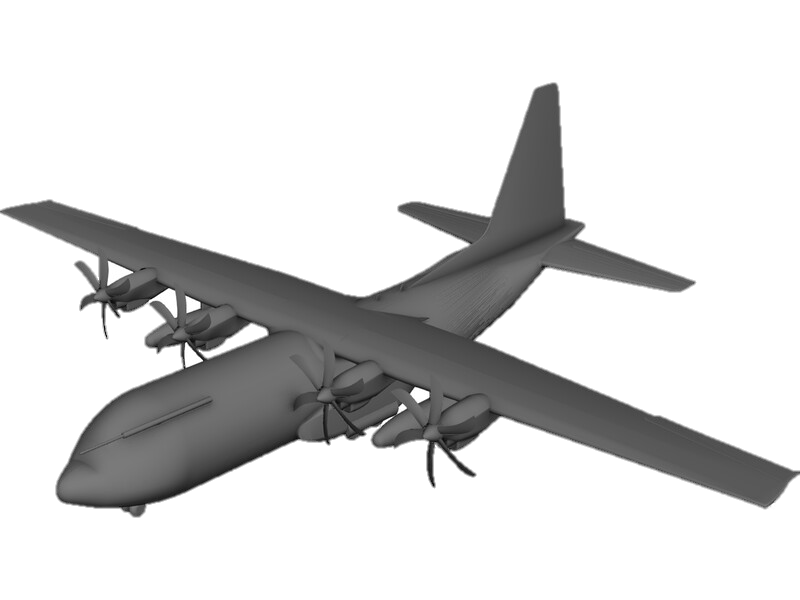
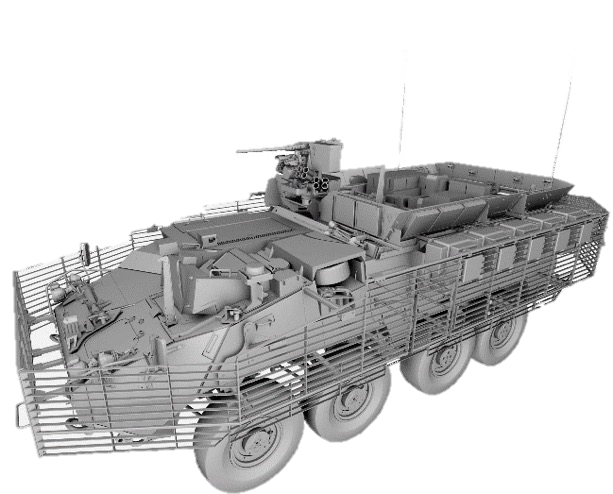
CAD MODEL INVENTORY ManageMENT System

Organizing Large Collections of Military 3D Geometry Models









Christopher Sean Morrison

Senior Computer Scientist

TAble of  
Contents

Software Summary 1

CAD Model Inventory Management System 1

Purpose 2

Audience 2

Introduction 4

Software Description 6

Core functionalities and features 6

Stakeholder Analysis 9

Primary Target Audience 9

Secondary Target Audience 10

Tertiary Stakeholders 10

Market Analysis 12

Resiliency to Market Changes 12

Positioning STrategy 13

Technical Specifications 14

Hardware Requirements 14

Software Requirements 14

Development Infrastructure 15

Development Roadmap 16

Minimum Viable Product (MVP) 16

Timeline 18

Risk Management 19

Risk Assessment 19

Mitigation Strategies 19

Conclusion 21

References 22

Appendix A: Software Architecture 23

Appendix B: GUI WIREFRAME Concept DESIGNS 24

# Software Summary

## CAD Model Inventory Management System

This software overview details an inventory management system (IMS) for computer-aided design (CAD) 3D geometry models and their associated data files in a U.S. federal government analytic modeling and simulation (M&S) context. In particular, the proposed software development considers the niche challenges and restrictions involved creating and deploying a solution compliant with federal regulations, in adherence with military information technology (IT) policies, easily integrated with available IT infrastructure, and compatible with existing production workflows. The software is also envisioned to address unique data requirements of an existing enterprise 3D model repository with thousands of models, hundreds of principal variants, and tailored to existing analysis environment requirements. The proposed IMS product hereafter will be referred to by the project name “**CADventory**”.

Unlike existing product data management (PDM) systems available commercially, this proposed IMS is a domain customization intended to require significantly less administrative, financial, and technical overhead. In order to accelerate acquisition, open source technologies will be leveraged to the greatest extent possible while maintaining high standards for compliance, testing, and software assurance.

## Purpose

### CADventory is a 3D CAD file, categorical tagging, and associated metadata inventory management system.

The primary purpose of CADventory is to directly facilitate the data organization, productivity, collaboration, and file management needs for teams of 3D modelers. These existing teams manage large existing collections of models, often spanning decades of legacy acquisitions. Teams also must manage data files for a variety of sources and at atypical fidelities. This can include management of myriad file types including CAD files, images, hand measurements, scanned documents, miscellaneous data files, imported/exported/converted geometries, and more. Managing this library of CAD models will be greatly facilitated by standardization, automation, and consistent data management (embodied by the tool).

## Audience

The tool will need to incorporate appropriate business logic behaviors specific to government, scientific research, and analysis needs. This means identifying unique application development restrictions and requirements, as well as policy and regulatory mandates. The principal target domain is 3D CAD modeling for military analyses. The primary users of CADventory are known as “target describers” in a military M&S analysis context. Secondary users include respective project managers, team leaders, engineers, and analysts that interface with said 3D modeling teams.

The proposed deployment environment for CADventory includes both unclassified and classified use. The technical implications of running in a non-networked, air-gapped, and/or classified environment are taken into consideration in the proposed architecture, application design, and choice of technologies. All proposed business logic and inventory management features are, however, agnostic to said restrictions and will remain applicable to and usable within other (non-government) environments. As such, all aspects of CADventory software design and development are unclassified and embrace a collaborative open source architecture approach for continued development and maintenance.

# Introduction

The military has developed 3D geometry models for various analyses purposes for decades (Klopcic and Reed, 1999). Such analysis can include survivability, lethality, vulnerability, signature studies, and more. These models are typically custom-tailored with unique fidelity requirements that are not typically encountered in the CAD industry. This includes stringent elements of solid modeling (Rossignac and Turner, 1994) and a need to analyze entire vehicles in-memory, performantly, and at extreme levels of fidelity. While solid modeling is not unique to military assessment, some source domains (e.g., manufacturing) and most target domains (e.g., finite element analyses (FEA)), do not typically involve the same type of preparation work due to specific analysis domain requirements (Deitz, 1984), and an analysis approach centered around the ray tracing technique.

With development beginning in 1979, the varied and complex requirements resulted in the development of BRL-CAD, 3D modeling and rendering software custom-developed to address the niche analysis domain. The enduring implication of such specialization means that existing 3D modelers often must convert geometry models to and from other CAD systems. This adds an additional data management and integration considerations which are discussed below in more detail.

The end result is the existence of several data repositories of 3D CAD models in BRL-CAD and other CAD formats of entire vehicles, vehicle variants, non-vehicle geometries, and associated data files spanning decades. These files are currently and predominantly managed by multiple agencies and companies, entirely ad hoc.

As such, this document details a proposed solution for the ad hoc management of CAD model inventory through the development of a software suite. This software suite aims to be compatible with existing workflows and expectations of users while both organizing and streamlining data management. It's designed to empower existing modeling teams, streamline workflows, and safeguard sensitive information, all while improving production pipeline and data management efficiency. It will be tailored to meet the unique challenges faced by defense researchers and CAD “target describers”.

While comprehensive data management is not believed to be achievable in the proposed 10-week development timeframe, the goal is a functioning prototype that demonstrates efficacy for future continued development. It’s envisioned that future support, updates, and customization will follow in concert with the ever-changing landscape of military R&D needs.

# Software Description

### CADventory is a 3D CAD file, categorical tagging, and associated metadata inventory management system.

CADventory is a 3D geometry inventory management system designed to address the distinct needs of high-level CAD model management, particularly as it pertains to high-performance computing, the organizational needs of an existing library of CAD models, and production analysis environments. CADventory aims to provide an interface for managing those models, tagging them categorically, organizing their associated files, and bringing efficiency to existing workflows.

Due to compliance, infrastructure, and security limitations, the primary system architecture is envisioned as a desktop application that either accesses a locally mounted network file share or other locally accessible filesystem containing existing CAD models and associated files. CADventory aims to facilitate indexing of an existing library of model data files.

## Core functionalities and features

1. Geometry data management:   
     
   CADventory shall provide facilities for organizing CAD data with associated files pertaining to specific (typically vehicle) assets.
2. Existing data integration:   
     
   CADventory shall work with existing hierarchical file repositories where models are groups into folders, with files for related models grouped into subfolders.
3. Metadata management:   
     
   CADventory shall keep track of pertinent CAD model metadata through categorization, tagging, and indexing capabilities.
4. Multiuser access:   
     
   CADventory shall allow multiple users to work simultaneously on the same model repository, accessed over a network file share. As a desktop application, CADventory may rely on existing operating system infrastructure for user authentication, filesystem access control, and logging.
5. Version control:   
     
   CADventory shall maintain a history of data and metadata revisions, ensuring data integrity and facilitating rollback to previous data and/or reconstruction of indices.
6. Visualization:   
     
   CADventory shall provide a visual index of 3D models for ease of navigation, discovery, viewing, and user experience.
7. Scalability:   
     
   CADventory shall be compatible with existing model repositories consisting of millions of files, thousands of folders, and handling large data files efficiently and at scale.
8. Compatibility:   
     
   CADventory shall be compatible with existing infrastructure, tools, and CAD file formats to the extent possible at the filesystem level.
9. Source code:   
     
   CADventory shall be developed in a manner that results in maintainable, performant, and portable source code. CADventory shall be developed using version control.

# Stakeholder Analysis

The primary stakeholders for CADventory are 3D CAD geometry modelers in R&D analysis environments creating, updating, and managing multiple CAD models. The secondary audience are individuals that said CAD modelers interface with regularly including project managers, team leaders, engineers, and analysts. Tertiary stakeholders are those indirectly affected by CADventory development or deployment, individuals that are not consumers or providers of 3D model data.

## Primary Target Audience

Primary users will require a tool that can manage large files including large CAD geometry files but also significant associated data files (e.g., terrain data, point cloud scan data, collections of photography). The tool will need to address the specific workflow needs of those users which typically entails working with models at a vehicle level or scene level (consisting of multiple vehicles or vehicles in situ environments), ***not*** at the assembly or part level as is typical in other CAD modeling domains.

Given the military domain, maintaining accuracy and fidelity are of paramount importance. Also significant are data security requirements, which are to be satisfied by existing IT infrastructure restrictions and policies which place intrinsic technical limitations on application architecture and behavior. For example, it is extremely complicated from a policy perspective to set up web server infrastructure, so the proposed application architecture for users involves working with existing network file system infrastructure.

## Secondary Target Audience

Secondary user interests and requirements center more around oversight and workflow integration. Engineers and analysts in particular will be concerned with the integration of CAD models with other (non-CAD / analysis) tools from primarily a consumer perspective, receiving access to both preliminary and finished 3D models from primary users.

Secondary users have an even greater need for introspective and indexing capabilities, facilitating their ability to find and aggregate indices of available 3D models. This greatly affects their ability to coordinate among different teams and analysis product consumers.

Other secondary stakeholders include project managers and team leaders whose principle responsibility is being apprised of current/recent modeler activity, resource allocations, and overall status of ongoing modeling efforts.

## Tertiary Stakeholders

Given the professional and military development environment, other noteworthy stakeholders including IT support staff that will be concerned with IT infrastructure implications, to include but not limited to filesystem utilization, taxing of server CPU resources, and any added maintenance implications.

Similarly, adherence to federal regulations are intrinsically requisite given the U.S. federal application domain. This includes adherence to the following standards, regulations, and policies:

1. NIST SP 800-37 Rev 2, Risk Management Framework for Information Systems and Organizations: A System Life Cycle Approach for Security and Privacy.
2. NIST SP 800-53, Security and Privacy Controls for Information Systems and Organizations.
3. DoDI 8310.01, Information Technology Standards in the DoD.
4. DoDI 8500.01, Cybersecurity.
5. DoDI 8510.01, Risk Management Framework (RMF) for DoD Systems.
6. Army Regulation AR 25-2, Army Cybersecurity (formerly Army Information Assurance).
7. DISA Application Security and Development Security Technical Implementation Guide (STIG), v5.

# Market Analysis

Across the U.S. federal government are a wide variety of CAD systems and PDM systems in use for myriad purpose. The defense industry and analysis R&D domain in particular is characterized by a demand for typically very high fidelity, security, and distribution of modeling tasks across a team of expert modelers. The features and capabilities of existing PDM systems are highly part-centric and assembly-centric, often focused around massive industry domains such as manufacturing or product design. They cater predominantly to general CAD functionalities.

Moreover, PDM systems are typically highly specialized and integrated with particular commercial CAD products and/or vendors with detrimental compatibility or feature implications. PDM systems also typically entail significant licensing costs, license management, IT infrastructure implications, and significant technical expertise requirements to set up and maintain them. As PDM systems are typically major enterprise investments, this also implies an inherent risk to the organization, beholden to the longevity and licensing of the PDM supplier.

## Resiliency to Market Changes

The military analysis domain has been around for more than 50 years, surviving many dominant CAD and PDM systems that ultimately failed or were overcome by a competitor. Over decades, there have been numerous CAD industry leaders bought out or killed by competition or other market forces. As such, it’s in the U.S. Governement’s long-term interest from a risk-management perspective to remain agnostic and resilient to changes in market dominance. Development of CADventory satisfies a need for independence by providing a solution that can be continuously developed, maintained, and customized as needed. This of course comes with costs, but this cost has been demonstrably proved beneficial by the 40+ year old development of BRL-CAD, an open source solid modeling system customized for military CAD needs in order to remain resilient to CAD industry change. CADventory, with its specialized focus on 3D geometry management and integration in adherence with federal/military standards, will continue to fill a unique niche in a similar manner.

## Positioning STrategy

CADventory’s unique selling proposition (USP) lies in its ability manage large collections of 3D models at a high filesystem level, with a user-friendly interface that integrates well for managing existing CAD model repository data, and all while adhering to stringent IT infrastructure restrictions. With the primary target audience being current professionals working in a military analysis context, the needs of an audience with niche requirements will be met better than existing commercial solutions that target broader industry needs.

# Technical Specifications

## Hardware Requirements

The software shall work on modern (as of 2024) personal computing hardware such as a typical desktop or laptop computer released within the past 2 years.

The software shall not require specific compute hardware such as specific GPUs or CPU types (e.g., RISC/CISC, ARM/M1/M2, etc).

The software shall work on systems with at least 8GB of memory available. The software will work optimally on systems with 64GB or more of available memory.

The software may assume adequately available system memory and filesystem storage for any intermediate processing needs.

The software shall work across a local network. Software performance may be subject to network performance implications accordingly.

## Software Requirements

See Stakeholder Analysis above for a list of regulatory requirements and technical specifications.

The software shall work cross-platform, minimally running on **Mac OS 13.6+**, **Linux**, or **Windows 11+** for prototyping, testing, and demonstration purposes.

The software must be implemented using a technology stack compatible with existing in use and/or compatible with current IT policy without involving special permission or exemption requests. As such, the software may utilize **Java w/ Swing, Java w/ NetBeans Platform, or C++ w/ Qt**.

The software may utilize hidden (dot) folders, text files, and/or SQLite database files for per-model metadata storage.

The software shall recognize and support indexing of a variety of CAD file formats, minimally including **BRL-CAD**, Creo, NX, CATIA, Rhino3D, and SpaceClaim.

The software shall recognize the availability of related CAD data including images (photography), documents (txt, pdf, and docx), and other data files identified in phase I as critical information.

The software shall utilize checksums and/or digital signatures to ensure data integrity and detect change.

The software may be reviewed for STIG category I compliance, but will only need to document any outstanding issues identified.

## Development Infrastructure

The project may use **Gitlab** for revision control, development logging, and continuous integration (CI) testing.

The project shall demonstrate CI testing for at least one platform.

The project shall incorporate some code coverage testing.

The project shall incorporate some code quality testing.

The project shall set up backups of all project data.

# Development Roadmap

With only 10 weeks to complete conceptualization, coding, and testing, software development of CADventory is comprised of three principle phases accordingly. The first phase, conceptualization, covers project setup, design, and initial architecture scaffolding over two weeks. The second phase, coding, entails six weekly spiral sprints working towards the MVP. The remaining third phase, testing, centers around refinement and compliance auditing for the remaining two weeks.

## Minimum Viable Product (MVP)

The user-accessible features of the software are described in more detail in the Software Description section prior. MVP installation deployment demonstration may be limited to a single operating system platform, either Windows, Mac OS, or Linux.

The minimum features necessary for MVP include the ability to:

1. manage CAD inventory (create/add, read, update, and delete),
2. populate the inventory using GUI or automation,
3. generate an inventory report for models matching user-specified criteria (e.g., land vehicles tagged as “wheeled”).

Primary trackable inventory items relevant for this military analysis market domain and MVP development include the following configuration items:

1. Model short name (e.g., ICV)
2. Model long name (e.g., M1126 Infantry Carrier Vehicle)
3. Model alias(es) (e.g., Stryker)
4. Model “type” (e.g., wheeled)
5. Organizational owner (e.g., U.S. Army)
6. Model Source (e.g., National Ground Intelligence Center)
7. Filesystem location (e.g., /h/tlib/Strkyer/ICV)
8. Model creation date
9. Last modification date
10. Modeler(s)
11. File owner(s)
12. Model suitability (e.g., VL-analysis ready)
13. Classification (e.g., Unclassified)
14. Conversions (e.g., STL, 3DM, STEP)
15. Exists Photography (e.g., yes/no)
16. Exists Drawings (e.g., yes/no)
17. Exists Measurements (e.g., yes/no)
18. Listing of unaccounted files
19. Listing of accounted files
20. 1-page summary sheet generation per model

CADventory is not to modify any existing CAD library file(s), only index and track files in-situ with model-associated metadata stored accordingly alongside CAD models (e.g., in hidden folders or SQLite).

**Auditing & Backups**

All accesses by CADventory users must be tracked in a way that allows auditing of system usage and any indexing or metadata changes. Information will minimally include username, timestamp, and type of access.

CADventory should provide administrative functions for backing up cached inventory information and checking the integrity of existing inventory data.

## Timeline

|  |  |  |  |
| --- | --- | --- | --- |
|  | Week | Goal | Milestone |
| Phase 1: Conceptualization | 1 | Architecture review, Dev log setup, Automated backups | Software Overview Doc |
| 2 | Data management design, Metadata storage setup, Initial GUI scaffold | Empty GUI Display w/ CI |
| Phase 2:  Coding Sprints | 3 | CRUD interface for indexing CAD vehicle model folders |  |
| 4 | Presentation of CAD inventory data items | GUI Input & Editing Prototype |
| 5 | Automated indexing of existing data files |  |
| 6 | Automated CAD model 1-page report generation  (GIST integration) |  |
| 7 | Summary reporting and presentation of CAD inventory |  |
| 8 | Portability resilience, Multiuser resilience | MVP |
| Phase 3:  Compliance | 9 | Unit testing, performance testing, coverage testing | Test results |
| 10 | STIG Audit, Usability Refinement, Summary | Final Report |

# Risk Management

## Risk Assessment

There are a number of notable risks in CADventory’s proposed development schedule and scope.

Given the strict 10-week development window, there is no room for schedule slippage. As such, this primarily puts feature development and capability (e.g., milestones achieved) at risk. This will naturally put phase 3 at the greatest risk if phase 2 development gets behind schedule

There is also (always) potential for scope creep and implementation issues. For example, development may identify additional critical inventory items that need to be tracked. A single GUI bug could put feature developmemt at risk.

There is also potential technical risk ensuring robust multiuser access. For example, there is minimal schedule flexibility for ensuring good network file system behavior under concurrent multiuser access to an SQLite or file data store.

## Mitigation Strategies

As there is little room for schedule modification in light of issues encountered, the primary mitigation strategies available are to ensure CI testing infrastructure is up-and-running early. Test-driven development early-on in the CI pipeline can help minimize scope changes and reduce rework effort.

As for actually reducing scope, the principle feature that may be pulled from the MVP if necessary may include automation features that help automatically populate the inventory based on existing files. Alternatively, we may need to solely rely on manual user entry of inventory information one model at a time if delays are encountered. Another potential complexity reduction possible with minimal impact to the MVP is a reduction in the number of trackable inventory items including deferring the inclusion of information about supplier, analysis status, and 1-pager summary information. This will consequently also result in simpler summary inventory reporting.

If there are infrastructure limitations that prevent concurrent access inconsistency or if portability issues are encountered, multiuser access and/or portability may be deferred in order to prioritize summary the central focus of the product, i.e., CAD inventory management and summary reporting.

# Conclusion

In summary, this software overview proposes the development of a novel CAD model inventory management system that will allow for the indexing and summary reporting of existing collections of military 3D geometry models. It provides a desktop interface that will be usable by a team of “target describers” to help index, organize, and manage models previously and actively being developed. The proposed CADventory desktop application allows for tagging and recording of CAD models and CAD model metadata via a technical approach consistent with military computing environment requirements and regulatory restrictions. This interface will provide near-term immediate benefit to existing stakeholders in a niche production analysis environment.

A detailed accounting of core features and functions are provided with consideration of stakeholder impact, market impact, strategic investment, technical restrictions, and development risks. A multi-phased development roadmap details primary features along the critical path and a plan for addressing software development within a 10-week window.

# References

* *Klopcic, J. and Reed, H. (1999).* Historical Perspectives on Vulnerability/Lethality Analysis*. AD-a361 816. Army Research Laboratory. Aberdeen Proving Ground, MD.*
* *Rossignac, L., and Turner, J., "*Solid modeling*," in IEEE Computer Graphics and Applications, vol. 14, no. 2, pp. 13-, March 1994, doi: 10.1109/38.267466.*
* *Deitz, P.H. (1984).* Solid Modeling at the US Army Ballistic Research Laboratory*. ADA147491. U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD.*

# Appendix A: Software Architecture

Here is a general simplified depiction of the application software architecture, focusing on key components and their primary associations. The UI presents information from the filesystem for a selected model library, stores and tracks per-library information via a storage manager, and generates library summary reports. Report generation and file handling are built on top of existing libraries provided for accessing geometry data and metadata as well as related images, documents, conversions, and other files directly associated with a given model.

**┌──────────────────────────────────────────────────────────────────────┐**

**│ CADventory │**

**│ │**

**│ ┌──────────────────────────────────────────────────────────────┐ │**

**│ │ User Interface (Qt) │ │**

**│ └─────────────┬───────────────────────┬─────────────────┬──────┘ │**

**│ │ │ │ │**

**│ ┌─────────────▼───────────┐ ┌────────▼─────────┐ ┌────▼──────┐ │**

**│ │ Filesystem Processor │ │ SQLite or JSON │◄─│ Report │ │**

**│ └─────────────┬───────────┘ │ Storage Manager │ │ Generator │ │**

**│ │ └──────────────────┘ └────┬──────┘ │**

**│ │ │ │**

**│ ┌─────────────▼───────────┐ │ │**

**│ │ Geometry/Image/Document │ │ │**

**│ │ Handler │ │ │**

**│ └─────────────┬───────────┘ │ │**

**│ │ │ │**

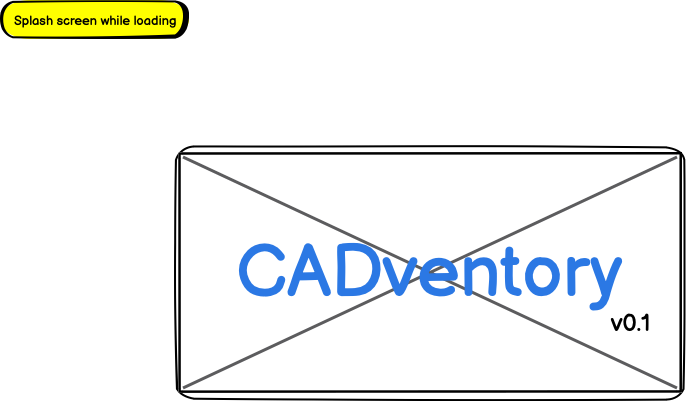
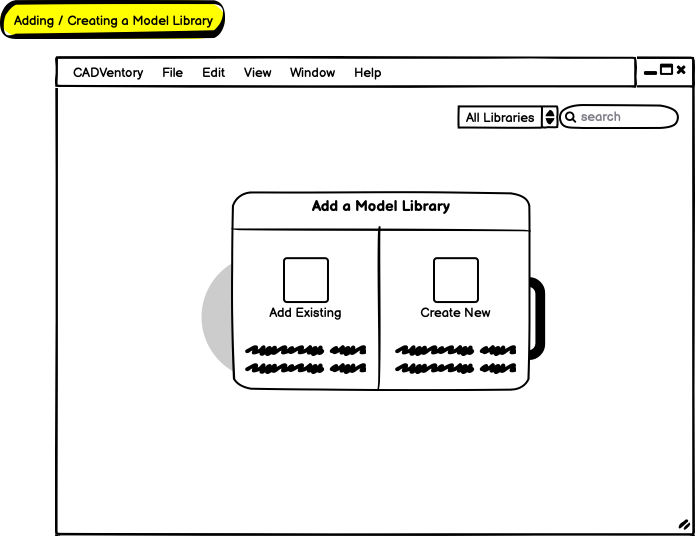
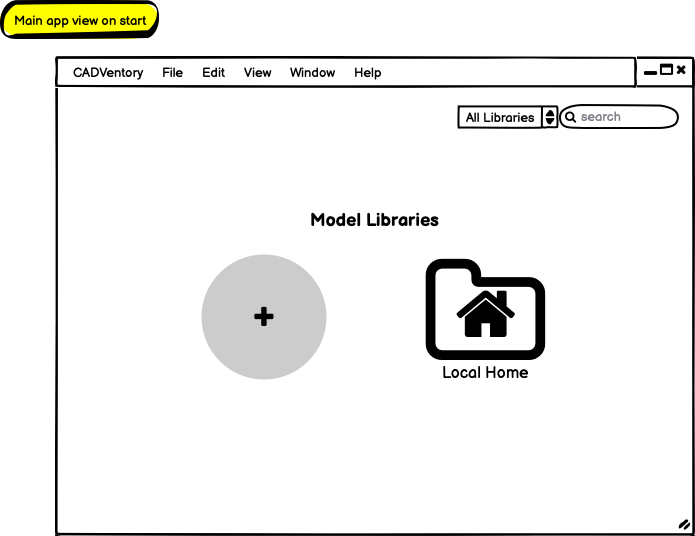
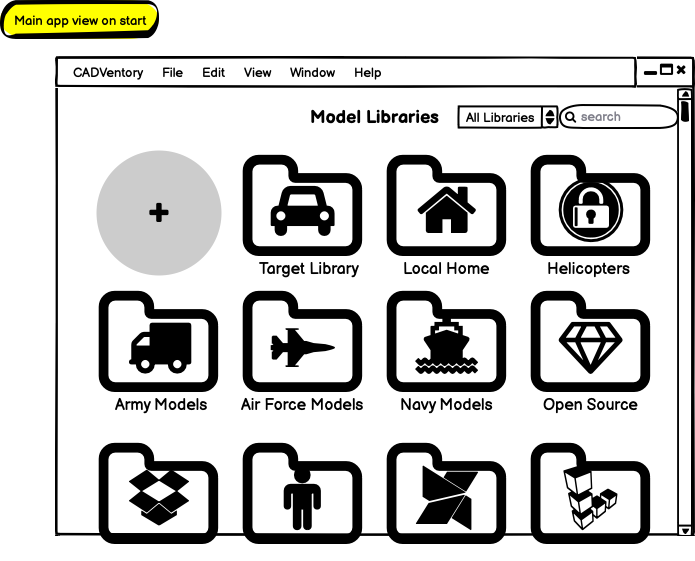
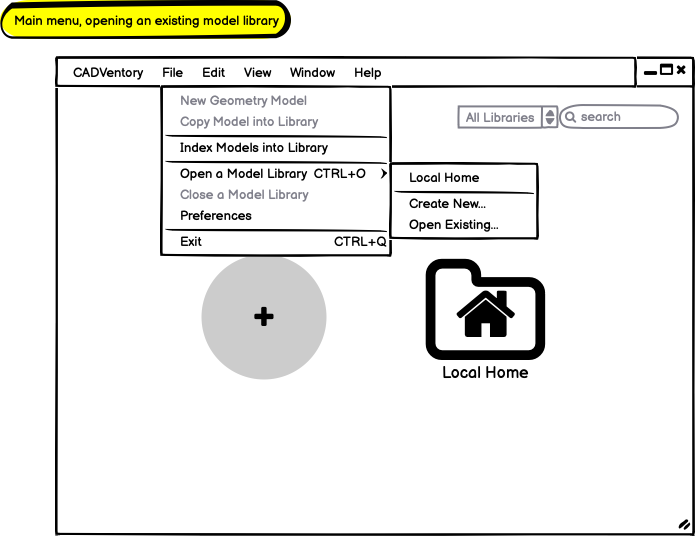
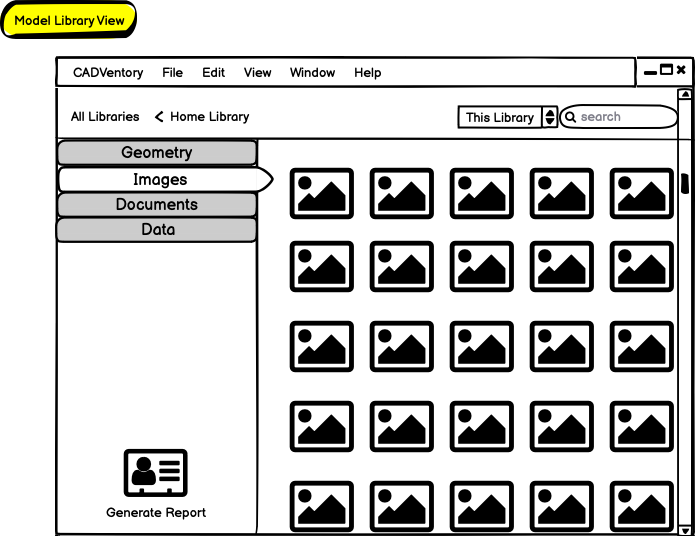
**│ ┌─────────────▼────────┐ │ │**

**│ │ CAD Libraries │◄───────────────────────────────┘ │**

**│ └──────────────────────┘ │**

**└──────────────────────────────────────────────────────────────────────┘**

# Appendix B: GUI WIREFRAME Concept DESIGNS

**  
  
  
  
  
  
  
  
  
  
  
  
  
**

## Infrastructure SETUP

Christopher Sean Morrison  
RSEG127 Software Engineering Studio

There was much debate, but I assign myself to be team lead!

There were no issues setting up a new project on the AWS-hosted Gitlab installation with SSH key authentication. The submitted “Morrison, Demonstration of Repo Interaction.mov” screen recording shows a brief interaction showing a cloning of the repo, an edit in my local IDE (Emacs), and a pushed change.

Emacs is my preferred development environment for the project.

Going forward, I propose developing on Mac, CI testing on Linux, and making UAT deployment builds on Windows. If it’s not too complicated, once I have a GUI building after sprint 1 I’m planning to add Windows CI/CD runners.

A “CADventory” project is set up on Gitlab with initial Linux CI testing in place via Gitlab CI using a default DevOps yml template combined with a basic single-file hello-world compilation setup demonstration.

Gitlab Issues are enabled for the project and have preliminary high-level tasks added. It’s of course anticipated that additional tasks will be added as development progresses, technical objectives are refined, and backlog expands.

### Versioning

For versioning, the plan is to use a similar variant to semantic versioinng as was described in the course notes. I’ve chosen a three-digit versioning scheme that is familiar to my DoD customers which adopts a three-digit convention for identifying and tracking future releases, using the following convention:

**{MAJOR\_VERSION}.{MINOR\_VERSION}.{PATCH\_VERSION}[-{AMENDMENT\_COUNT}]**

Major only increments when there is significant major changes and/or incompatibility. Minor updates more frequently and serves a dual role indicating in-development and release versions via odd and even numbers respectively. Minor updates typically include significant backwards-compatible development activity. Minor reset to zero on any change to major. Patch version numbers may be updated as frequently as necessary, but reset back to 0 on any change to minor. An ammendment count is strictly used to update a posted release.

Example baseline MVP build: 1.0.0

Example pre-MVP minor dev build (5th iteration): 0.1.4

Example minor release build: 1.2.0

Example patch release (13th iteration): 1.2.13

Example dev build (1st iteration): 1.3.0

Example updated patch release: 1.2.13-2

### Issue Management

The plan for managing issues is to use issues like a sprint burndown chart. That is, I plan to define all high-level development objectives that are described in the project overview MVP as issues in Gitlab. Issues are to be marked accordingly per the iteration schedule with an in-progress status (e.g., started, needs\_review) or completion status (i.e., deferred, changed, or complete). As development proceeds, issues will get updated so the set of closed issues gives an indication of progress made, and open issues an indication of known progress planned and remaining.

### Architecture Selection

Considerable time was invested the past week carefully considering the three proposed architectures. A rationale summarization for using C++ with Qt is included for your review, extracted from the project overview document where I’ve been adding additional architecture and software design details.

# Appendix C: Architecture assessment

Here we evaluate primary tradeoff considerations between selecting Java with Swing, Java with NetBeans Platform, and C++ with Qt as the technical architecture for CADventory. To compare the technology stacks, we evaluate considerations such as development complexity, performance, cross-platform compatibility, UI sophistication, policy compliance, and enterprise integration. We use a 1 to 5 scale where 1 indicates lowest cost-benefit and 5 indicates highest.

## Cost-Benefit Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| Consideration | Java w/ Swing | Java w/ NP | C++ w/ Qt |
| Development Complexity | 3 | 2 | 3 |
| Performance | 4 | 4 | 5 |
| Cross-Platform Compatibility | 5 | 5 | 4 |
| UI Sophistication | 3 | 4 | 5 |
| Policy Compliance | 4 | 4 | 5 |
| Enterprise Integration | 3 | 4 | 5 |
| **Total Scoring** | **22** | **23** | **27** |
| **Favorability** | **73.3%** | **76.7%** | **90%** |

## Explanation of scoring

### Development Complexity

The overall ease of development and learning curve is very similar for all three options with Netbeans Platform being (relatively) the most complex due to a far greater lack of supporting resources (e.g., tutorials, guides, articles, example code, etc).

### Performance

All three options are capable of highly-performance architecture design and are scalable to requirements, but the lower-level large file management and efficient filesystem traversal capabilities of C++ give it a particular advantage for working with a large existing library of CAD files and associated data files. Java applications can be less performant (at least without necessitating additional infrastructure) compared to native applications, particularly during high-memory utilization and application startup.

### Cross-Platform Compatibility

While Qt is a cross-platform API, Java-based solutions are intrinsically cross-platform giving them a clear advantage for portability.

### UI Sophistication

Netbeans Platform, while not as modern in presentation, provides many modern and advanced features such as automatic file browsing, logging, filetype associations, document handling, runtime management, extension development, customizability, and more that would otherwise require additional effort as compared with Qt or Swing. Qt is an API framework specifically for creating sophisticated modern GUIs.

### Policy Compliance

All three options can be made compliant with aforementioned IT policy and infrastructure restrictions. Java-based solutions, however, do have additional Security Technical Implementation Guideline (STIG) requirements specific to operation in a military environment as compared with a C++ Qt-based application. This gives C++ with Qt a lesser compliance and ongoing maintenance cost.

### Enterprise Integration

The stakeholders’ R&D enterprise environment, particularly as tailored to existing M&S application development and analysis capabilities results in a favoring towards C++ with Qt, followed closely by Java with Netbeans Platform, as both are in production use by flagship products and targeted stakeholders.