Software Architecture Document

SimuWare

Software Engineering Course Project

SAD Revision History

Date	Version	Description	Author
2024-03-21	1.0	Significant Use-Cases: the key requirements	SimuWare Architecture Team

Table of Contents

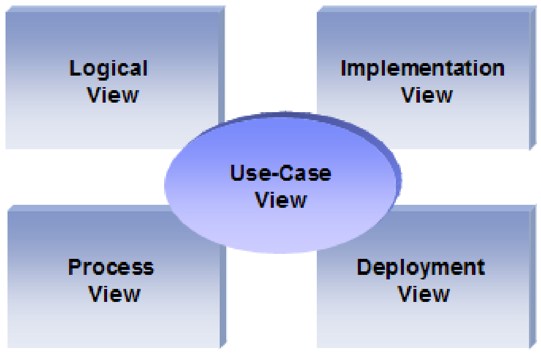
1. Introduction

The purpose of this document is to provide a detailed architecture design of SimuWare, a 3-D simulating environment with Arduino by focusing on four key quality attributes: usability, availability, maintainability, and testability. These attributes were chosen based on their importance in the design and construction of the application. This document will address the background for this project, and the architecturally significant functional requirements. Each of the aforementioned quality attributes will be described through a comprehensive set of scenarios followed by an architectural overview, which includes a bird's eye view and a full description of patterns and tactics that will be used to address the core quality attributes. This will be followed up by a look at a couple views into the system. Finally, acknowledgements, references, and appendices will round out the document. The intention of this document is to help the development team to determine how the system will be structured at the highest level. It is also intended for the project sponsors to sign off on the high level structure before the team shifts into detailed design. Finally, the project coach can use this document to validate that the development team is meeting the agreed upon requirements during his evaluation of the team's efforts.

Architectural activities	Software Architecture Document
Step 1 - Identify and prioritize significant Use-Cases	Section 4
Step 2 - Define the candidate architecture	Section 3, 5.1, 10, 11
Step 3 - Define the initial Deployment Model	Section 7
Step 4 - Identify key abstractions	Section 9
Step 5 - Create an Analysis Model	Section 5
Step 6 - Create the Design Model	Section 5
Step 7 - Document concurrency mechanisms	Section 6, 7
Step 8 - Create the Implementation Model	Section 8

1.1 Purpose

The purpose of SimuWare is to provide a virtual platform that enables users to assemble physical and electrical components in a 3D game-like environment and to simulate their interactions. SimuWare aims to offer an educational and prototyping tool for simulation enthusiasts, students, educators, and researchers.



The “4+1” View Model allows various stakeholders to find what they need in the software architecture.

1.2 Scope

1.2.1 Goals

SimuWare will feature a user-friendly interface for selecting, placing, and connecting components within a virtual environment. Users will also be able to simulate the behavior of assembled components.

1.2.2 Benefits

- Facilitate simulation of prototypes without the need for physical assembly.
- Aid in identifying and rectifying flaws in circuits before building real-life prototypes.

1.3 Definitions, Acronyms and Abbreviations

RUP: Rational Unified Process

UML: Unified Modeling Language

SAD: Software Architecture Document

API: Application Programming Interface

IDE: Integrated Development Environment

1.4 References

[Unreal]:	Unreal Engine: 3D game development platform used for creating the virtual environment. https://www.unrealengine.com/en-US/uses/architecture
[Aurdino]:	Arduino GitHub Repository: Source for Arduino APIs and Libraries. https://github.com/arduino

1.5 Overall description

SimuWare offers a rapid-prototyping environment for circuits and microcontroller-based simulations. Users can test their prototypes within the software, eliminating the need for physical assembly and reducing resource waste.

1.5.1 User Interfaces

SimuWare is a desktop application designed for Windows operating systems.

1.5.2 Software Interfaces

Unreal Engine: Used for rendering the 3D virtual environment.

Unity: Alternative option for 3D environment rendering.

C++: Programming language for backend development.

Arduino APIs: Interfaces with Arduino microcontrollers for component simulation.

1.5.3 Product Functions

- **3D Environment**

  Provides a 3D virtual environment using Unreal Engine or Unity.

  Users navigate the environment using intuitive controls.

- **Part Assembly System**

  Users select from a library of Arduino components.

  They can place selected components within the 3D environment.

  Components snap or attach to each other realistically when placed adjacent to each other.

- **Physics Simulation**

  Incorporates physics simulation to simulate interactions between assembled components.

  Components interact realistically with each other and the environment (e.g., gravity, collisions).

- **User Interface**

  SimuWare provides a user-friendly interface for selecting, placing, and manipulating components.

  The interface allows users to access tools for assembly, simulation, and data visualization.

- **User Characteristics**

SimuWare caters to simulation enthusiasts, students, educators, and researchers. Users can range from beginners to advanced users with knowledge of microcontrollers and basic circuit design.

1.5.4 Constraints

Performance may vary on different hardware configurations based on factors like the presence of a graphics card and available RAM.

1.5.5 Assumptions and Dependencies

- The platform assumes basic ideal-physics assumptions, and results may differ from real-world simulations.
- Users should have basic knowledge of Arduino and familiarity with basic circuit design principles.

2. Architectural Representation

The views used to document the SimuWare. application are:

Logical view

Audience: Designers.
Area: Functional Requirements
Related Artifacts: Design model
In the logical view, SimuWare's architecture provides a comprehensive depiction of the system's object model and highlights the critical use-case realizations. This view offers a high-level abstraction of the system's functionality, emphasizing the logical arrangement of components and their interactions. It serves as a foundation for understanding the system's behavior and guiding the design process.

Process view

Audience: Integrators.
Area: Non-functional requirements
Related Artifacts: N/A
The process view of SimuWare's architecture describes the design's concurrency and synchronization aspects. It focuses on how different components of the system interact with each other in terms of processes, threads, and communication mechanisms.

Implementation view

Audience: Programmers.
Area: Software components
Related Artifacts: Implementation model, components
The implementation view of SimuWare's architecture describes the layers and subsystems of the application. It provides guidance to programmers on how to organize and structure the codebase, including details on software components, their dependencies, and interactions.

Deployment view

Audience: Deployment managers.
Area: Topology.
Related Artifacts: Deployment Model.
The deployment view of SimuWare's architecture describes the mapping of the software onto the hardware and shows the system's distributed aspects. It outlines how the different software components are deployed across various hardware resources, including servers, networks, and other infrastructure elements.

Use Case view

Audience:All stakeholders, including end-users
Area:Scenarios and Use Cases
Related Artifacts : Use-Case Model, Use-Case documents
The use case view of SimuWare's architecture describes the set of scenarios and/or use cases that represent significant, central functionality of the system. It outlines how users interact with the system and the various tasks they can perform

3. Architectural Goals and Constraints

This section describes the software requirements and objectives that have some significant impact on the architecture

3.1 Technical Platform

SimuWare will be deployed on the Unreal Engine framework, leveraging its robust infrastructure and scalability features. The choice of Unreal Engine prioritizes compatibility with the development stack and ensures support for the required functionalities, including 3D rendering, physics simulation, and user interaction.

3.2 Transaction

SimuWare's transactions, although not traditional in the sense of database transactions, will be managed efficiently within the simulation engine. The architecture will ensure that interactions between components within the virtual environment are executed reliably and atomically, maintaining the integrity of the simulation. Transaction-like behaviors will be implemented to handle component interactions and state changes seamlessly.

3.3 Security

SimuWare's architecture will prioritize security measures to protect user data and prevent unauthorized access. This includes implementing secure communication protocols, access controls, and encryption mechanisms to safeguard sensitive information within the application.

3.4 Cross-Platform Compatibility

SimuWare will be designed to run across multiple platforms, including desktop and potentially mobile devices. The architecture will ensure cross-platform compatibility by utilizing platform-agnostic technologies and frameworks, enabling users to access the application from various devices and operating systems.

3.5 Reliability/Availability (failover)

The architecture of SimuWare will prioritize delivering an intuitive and immersive user experience. This includes designing user interfaces that are easy to navigate, providing real-time feedback and visual cues, and optimizing interactions for smooth and natural manipulation of components within the virtual environment.

3.6 Performance

Performance optimization will be a key consideration in SimuWare's architecture. The system will be designed to deliver fast response times and smooth simulation experiences, even when handling complex scenarios and interactions. This will involve efficient resource utilization, parallel processing, and caching strategies to minimize latency and maximize throughput.

4. Use-Case View

This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system. The only use-case with a significant impact on the online catering architecture is the one related to online orders. It includes a search feature as well as a call to external services

4.1 Simulating Physical Objects

Description:A user accesses the SimuWare application and navigates to the physical object simulation feature. The user selects objects from a library of 3D models representing physical objects and assembles them within the virtual environment. The user then initiates the simulation process to observe the behavior of the assembled objects under various conditions, such as gravity, collisions, and external forces.

Actors:

- User

Preconditions:

- The user must have access to the SimuWare application.
- The user must be logged in to access the simulation feature.
- The library of 3D models representing physical objects must be available for selection.

Flow of Events:

- User accesses the SimuWare application.
- User navigates to the physical object simulation feature.
- User selects objects from the inventory of 3D models representing physical objects.
- User assembles the selected objects within the virtual environment.
- User initiates the simulation process.
- System simulates the behavior of the assembled objects under various conditions.
- User observes the simulation results.
- User adjusts object configurations if necessary and repeats the simulation process.

4.2 Simulating Cars

Description: After completing the simulation of physical objects, the user decides to simulate cars within the virtual environment. The user selects car models from a library of 3D models representing different types of vehicles and assembles them within the virtual environment. The user then initiates the simulation process to observe the behavior of the assembled cars, including acceleration, braking, and steering.

Actors:

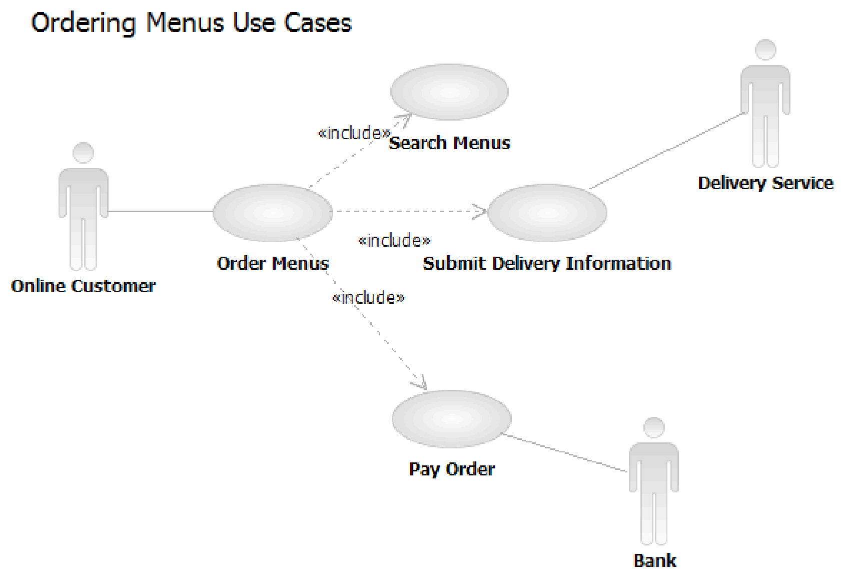
- User

Preconditions:

- The user must have completed the simulation of physical objects within the SimuWare application.
- The user must have access to the car simulation feature within the application.
- The library of 3D models representing cars must be available for selection.

Flow of Events:

- User completes the simulation of physical objects within the SimuWare application.
- User navigates to the car simulation feature.
- User selects car models from the library of 3D models representing vehicles.
- User assembles the selected cars within the virtual environment.
- User initiates the simulation process.
- System simulates the behavior of the assembled cars, including acceleration, braking, and steering.
- User observes the simulation results.
- User adjusts car configurations if necessary and repeats the simulation process.



5. Logical View

5.1 Overview

The logical view of SimuWare encompasses three main components: User Interaction, Simulation Engine, and Component Library.

Module Responsibilities:

- ***User Interaction:*** *This component handles all interactions with the end-user, including input processing, user interface elements, and feedback mechanisms.*

- ***Simulation Engine****: The simulation engine is responsible for simulating the behavior of components within the virtual environment. It incorporates physics simulation, interaction modeling, and real-time rendering to provide a realistic experience.*

- ***Component Library:*** *The component library provides a collection of predefined components that users can select, place, and connect within the 3D environment. These components include various electrical and mechanical parts, each with predefined behaviors and properties.*

The SimuWare application version 1.0 is relatively simple, featuring two basic functionalities: circuit simulation and object manipulation.

5.2 Architecturally Significant Design Packages

5.2.1 Simulating Circuits

- **Description:** This package is responsible for all logic related to simulating circuits. It provides simulation features and necessary components to interact with external services.

  Analysis Model:

  - ***Participants:***
    - User
  - ***Basic Flow***:
    - User accesses simulation features.
    - User selects components and initiates simulation.
    - System simulates circuit behavior.
    - User interacts with simulation results.

  Design Model:

  - ***Process***: Circuit Simulation

Participants:

Basic Flow:

Design Model

Process Delivery

5.2.2 Manipulating Objects

- **Description:This package handles logic related to manipulating physical objects within the virtual environment. It provides features for selecting, placing, and interacting with objects.**

  Analysis Model:

- ***Participants:***
    - User
- ***Basic Flow***:
    - User accesses object manipulation features.
    - User selects objects and places them within the virtual environment.
    - User interacts with objects, adjusting configurations as needed.

Design Model:

- ***Process***: Object Manipulation

5.2.3 Payment Service

Contains all the logic related to the online payment and credit card validation.
The Payment Service is an external subsystem documented in its own Software Architecture Document.

6. Process View

There's only one process to take into account. The J2EE model automatically handles threads which are instances of this process.

7. Deployment View

Global Overview

The deployment view of SimuWare highlights the strategies and processes involved in distributing the software to end-users. Built on the Unreal Engine framework, SimuWare is primarily targeted for desktop deployment on Windows operating systems. The deployment package includes the executable file along with all necessary resources, ensuring a seamless installation experience for users.

Leveraging the capabilities of the Unreal Engine, SimuWare is packaged according to the engine's guidelines, ensuring compatibility and optimized performance. The packaging process involves compiling the project, bundling assets, and generating executable files suitable for distribution.

SimuWare can be distributed through various channels, including online platforms such as Steam, Epic Games Store, or the developer's website. This allows for widespread accessibility and ease of download for users.

Moreover, the deployment view considers the process of updating and patching SimuWare post-deployment. Unreal Engine provides mechanisms for deploying updates and patches seamlessly, ensuring that users have access to the latest features and bug fixes.

Overall, the deployment view underscores the importance of compatibility, efficiency, and accessibility in delivering SimuWare to end-users across different platforms.

Detailed deployment model with clustering

- **Desktop Deployment**: Draw a computer icon representing the desktop platform, with an arrow pointing towards it indicating the deployment of SimuWare as a desktop application for Windows.
- **Unreal Engine Compatibility**: Include the Unreal Engine logo or icon to signify the framework's compatibility, possibly connected to the deployment process with a labeled arrow.
- **Packaging for Distribution:** Create a diagram box representing the packaging process within the Unreal Engine editor, with arrows indicating the steps involved, such as compiling, bundling assets, and generating executable files.
- **Distribution Channels:** Draw icons representing online platforms like Steam, Epic Games Store, and a website, with arrows indicating the distribution channels through which users can access and download SimuWare.
- **Updates and Patching:** Show a looped arrow indicating the continuous process of updating and patching SimuWare, possibly connected to the distribution channels or packaging process.

8. Implementation View

8.1 Overview

SimuWare's implementation is structured into modular components, each encapsulating specific functionality and gameplay elements.

8.2 Modules

8.2.1 Simulation Module:

Responsible for simulating physical and electrical components, managing simulation processes, and handling collision detection.

8.2.2 UserInterface Module:

Manages user interface elements, including main menu widgets and simulation heads-up displays (HUDs), facilitating user interaction.

8.2.3 ControlSystem Module:

Implements control algorithms, feedback systems, and control logic for interacting with simulated objects within the environment.

Each module follows a clear organization, allowing for independent development and seamless integration. Standardized naming conventions and macro utilization ensure compatibility with Unreal Engine's framework and ease of development. The modular architecture enhances code clarity, facilitates collaboration, and promotes maintainability throughout the development process.

9. Data View

The key data elements related to the online catering system are:

10. Size and Performance

Volumes:

- Estimated online orders : 100 a day, with peaks in the evening
- Yummy Inc registered individual customer : about 150
- Yummy Inc corporate customers : about 100

Performance:

- Time to process and online payment (credit card validation + confirmation) : less that 10 seconds required

11. Quality

As far as the online catering application is concerned, the following quality goals have been identified:

Scalability:

- **Description** : System's reaction when user demands increase
- **Solution** : J2EE application servers support several workload management techniques

Reliability, **Availability**:

- **Description** : Transparent failover mechanism, mean-time-between-failure
- **Solution :** : J2EE application server supports load balancing through clusters

Portability:

- **Description** : Ability to be reused in another environment
- **Solution :** The system me be fully J2EE compliant and thus can be deploy onto any J2EE application server

Security:

- **Description** : Authentication and authorization mechanisms
- **Solution :** J2EE native security mechanisms will be reused