Version: 1.1

SF-Agricolture Architecture/Design Document

Table of Contents

Introduction 3

Design Goals 4

System Behavior 4

Logical View 5

High-Level Design (Architecture) 5

Detailed Class Design 6

Development View 7

Physical View 7

Change History

Version: 1.1

Modifier: Vidrascu Adrian

Date: 04/10/2020

Description of Change: Final adjustments

Version: 1.0

Modifier: Vidrascu Adrian

Date: 04/14/2020

Description of Change: Initial drafts

Introduction

This document describes the architecture and design for the SF-Agricolture application being developed for the ITI G.Marconi – Verona, Italy. SF-Agricolture is a web application that calculate the yelds of a field. SF-Agricolture has a simply UI that requires zero computer skills. Web visitors simply introduce their data and the application will do the rest.

The purpose of this document is to describe the architecture and design of the SF-Agricolture application in a

way that addresses the interests and concerns of all major stakeholders. For this application the major stakeholders are:

- Users and the customer – they want assurances that the architecture will provide for system functionality and exhibit desirable non-functional quality requirements such as usability, reliability, etc.
- Developers – they want an architecture that will minimize complexity and development effort.
- Project Manager – the project manager is responsible for assigning tasks and coordinating development work. He or she wants an architecture that divides the system into components of roughly equal size and complexity that can be developed simultaneously with minimal dependencies. For this to happen, the modules need well-defined interfaces. Also, because most individuals specialize in a particular skill or technology, modules should be designed around specific expertise. For example, all UI logic might be encapsulated in one module. Another might have all logic related to GPS coordinates.
- Maintenance Programmers – they want assurance that the system will be easy to evolve and maintain on into the future.

The architecture and design for a software system is complex and individual stakeholders often have specialized interests. There is no one diagram or model that can easily express a system's architecture and design. For this reason, software architecture and design is often presented in terms of multiple views or perspectives [IEEE Std. 1471]. Here the architecture of the SF-Agricolture application is described from 4 different perspectives [1995 Krutchen]:

1. Logical View – major components, their attributes and operations. This view also includes relationships between components and their interactions. When doing OO design, class diagrams and sequence diagrams are often used to express the logical view.
2. Process View – the threads of control and processes used to execute the operations identified in the logical view.
3. Development View – how system modules map to development organization.
4. Use Case View – the use case view is used to both motivate and validate design activity. At the start of design the requirements define the functional objectives for the design. Use cases are also used to validate suggested designs. It should be possible to walk through a use case scenario and follow the interaction between high-level components. The components should have all the necessary behavior to conceptually execute a use case.

Design Goals

There is no absolute measure for distinguishing between good and bad design. The value of a design depends on stakeholder priorities. For example, depending on the circumstances, an efficient design might be better than a maintainable one, or vise versa. Therefore, before presenting a design it is good practice to state the design priorities. The design that is offered will be judged according to how well it satisfies the stated priorities.

The priorities for the design that follows are:

- The design should minimize complexity and development effort.
- The design should take into account the development environment which is 4 small teams with complementary skills that work across time and space (teams are not co-located). Ideally the design

should result in 4 loosely coupled components of equal size and complexity. If the components have well-defined interfaces each team can work independently coding to the interfaces of the other components. The concerns of each component should be narrow so that each team can specialize on a particular technology or skill.

System Behavior

The use case view is used to both drive the design phase and validate the output of the design phase. The architecture description presented here starts with a review of the expect system behavior in order to set the stage for the architecture description that follows. For a more detailed account of software requirements, see the requirements document.

Logical View

The logical view describes the main functional components of the system. This includes modules, the static relationships between modules, and their dynamic patterns of interaction.

In this section the modules of the system are first expressed in terms of high level components (architecture) and progressively refined into more detailed components and eventually classes with specific attributes and operations.

High-Level Design (Architecture)

The high-level view or architecture consists of 4 major components:

Figure 2 System Architecture

- The **Model** is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application
- The **Database** is a central repository for data on buildings, their locations and associated audio segments.
- The **View** consists of any representation of information such as a chart, diagram or table. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.
- The **Controller** accepts input and converts it to commands for the model or view.

Detailed Class Design

Development View

Physical View

Physical view depens of visitor's device. For any device the website has a different View.