Module Guide for MCT

Team #12, Lower Earth Orbiters
Diamond Ahuja
Rishi Vaya
Buu Ha
Dhruv Cheemakurti
Umang Rajkarnikar

January 18, 2024

1 Revision History

Date	Version	Notes
Jan. 17, 2024	1.0	Completed MG for MCT application

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	description
AC	Anticipated Change
DAG	Directed Acyclic Graph
M	Module
MG	Module Guide
OS	Operating System
R	Requirement
SC	Scientific Computing
SRS	Software Requirements Specification
UC	Unlikely Change
FIFO	First In First Out
MCT	Mission Control Terminal

Contents

List of Tables

List of Figures

3 Introduction

Decomposing a system into modules is a commonly accepted approach to developing software. A module is a work assignment for a programming team (**?**). We advocate a decomposition based on the principle of information hiding (**?**). This principle supports design for change, because the "secrets" that each module hides represent likely future changes. Design for change is valuable in SC, where modifications are frequent, especially during initial development as the solution space is explored.

Our design for the MCT application follows the rules layed out by **?**, as follows:

- System details that are likely to change independently should be the secrets of separate modules.
- Each data structure is implemented in only one module.
- Any other program that requires information stored in a module's data structures must obtain it by calling access programs belonging to that module.

After completing the first stage of the design, the Software Requirements Specification (SRS), the Module Guide (MG) is developed (**?**). The MG specifies the modular structure of the system and is intended to allow both designers and maintainers to easily identify the parts of the software. The potential readers of this document are as follows:

- New project members: This document can be a guide for a new project member to easily understand the overall structure and quickly find the relevant modules they are searching for.
- Maintainers: The hierarchical structure of the module guide improves the maintainers' understanding when they need to make changes to the system. It is important for a maintainer to update the relevant sections of the document after changes have been made.
- Designers: Once the module guide has been written, it can be used to check for consistency, feasibility, and flexibility. Designers can verify the system in various ways, such as consistency among modules, feasibility of the decomposition, and flexibility of the design.

The rest of the document is organized as follows. Section 4 lists the anticipated and unlikely changes of the software requirements. Section 5 summarizes the module decomposition that was constructed according to the likely changes. Section 6 specifies the connections between the software requirements and the modules. Section 7 gives a detailed description of the modules. Section 8 includes two traceability matrices. One checks the completeness of the design against the requirements provided in the SRS. The other shows the relation between anticipated changes and the modules. Section 9 describes the use relation between modules.

4 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 4.1, and unlikely changes are listed in Section 4.2.

4.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision. The approach adapted here is called design for change.

AC1: The specific TLE for satellites to track, per user.

AC2: The number of satellites to track, per user

AC3: The type of information requested by stakeholders

AC4: The precision of measurement specification requested by stakeholders

AC5: The ground station coordinate information

AC6: The specific commands to be sent to satellites by operators and application administrators

4.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decision should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

UC1: Server on which the application in hosted

UC2: Format of satellite data input

UC3: Format of TLE data for tracking

UC4: Encoding scheme of command sequences to be sent to a satellite

UC5: Database technology being used to persist command sequences and log outputs

5 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: Hardware-Hiding Module

...

Level 1	**Level 2**
Hardware-Hiding Module	N/A
Behaviour-Hiding Module	Satellite Module Schedule Module User Module Log Module Database Module Authentication Module BackendService Module
Software Decision Module	SatelliteQueue Module

Table 1: Module Hierarchy

6 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 2.

6.1 Modules

6.1.1 Satellite Module

The Satellite module is responsible for all information related to a satellite, including calculating positional data, future overpasses, and generating polar plot data.

6.1.2 Schedule Module

The Schedule module is responsible for all logic pertaining to storing, updating, and executing command sequence(s) for a satellite target.

6.1.3 User Module

The User module is responsible for storing information about users in the MCT application. This includes creating new users and updating information for existing users.

6.1.4 SatelliteQueue Module

The SatelliteQueue module implements the queue data structure to store commands scheduled for execution in the upcoming overpass.

6.1.5 Log Module

The Log module implements logging functionality from the ground server.

6.1.6 Authentication Module

This module will employ the user's email and password for authentication purposes and to fetch their complete information, a prerequisite for accessing the satellite and scheduling data.

6.1.7 Database Module

This module will utilize multiple data structures to store, collect and return data. It will also enable users to view and edit stored data.

6.1.8 BackendService Module

It's responsible for initializing the backend processes of the application.

7 Module Decomposition

Modules are decomposed according to the principle of "information hiding" proposed by **?**. The *Secrets* field in a module decomposition is a brief statement of the design decision hidden by the module. The *Services* field specifies *what* the module will do without documenting *how* to do it. For each module, a suggestion for the implementing software is given under the *Implemented By* title. If the entry is *OS*, this means that the module is provided by the operating system or by standard programming language libraries. *Software Engineering* means the module will be implemented by the Software Engineering software.

Only the leaf modules in the hierarchy have to be implemented. If a dash (–) is shown, this means that the module is not a leaf and will not have to be implemented.

7.1 Hardware Hiding Modules (M1)

N/A

7.2 Behaviour-Hiding Module

Secrets: The contents of the required behaviours.

Services: Includes programs that provide externally visible behaviour of the system as specified in the software requirements specification (SRS) documents. This module serves as a communication layer between the hardware-hiding module and the software decision module. The programs in this module will need to change if there are changes in the SRS.

Implemented By: –

7.2.1 Satellite Module

Secrets: Overpass dates, satellite name and intl code.

Services: Fetch future overpasses, satellite telemetry information and manage (add, update, delete) records of satellite systems.

Implemented By: MCT

Type of Module: Abstract Object

7.2.2 Schedule Module

Secrets: Command sequences to execute for a satellite's scheduled overpass.

Services: Manage (add, update, delete) records of scheduled command sequences for a satellite system.

Implemented By: MCT

Type of Module: Abstract Object

7.2.3 User Module

Secrets: User email and role.

Services: Manages (reads, creates, updates) user records.

Implemented By: MCT

Type of Module: Abstract Object

7.2.4 Authentication Module

Secrets: User email and password.

Services: Registers and signs users to the MCT application.

Implemented By: Auth0

Type of Module: Library

7.2.5 Database Module

Secrets: Records to store, update and remove from the database.

Services: Stores data and provides read and write access to the data store.

Implemented By: MongoDB

Type of Module: Library

7.2.6 Log Module

Secrets: SatelliteId to find logs for a particular satellite.

Services: Gets logs for all executed commands for a specific satellite.

Implemented By: MCT

Type of Module: Abstract Object

7.3 Software Decision Module

7.3.1 SatelliteQueue Module

Secrets: An array to manage scheduled command sequences for the next overpass in a FIFO order. This ensures that the queued commands will always be up to date.

Services: Queue data structure to store commands scheduled for execution in the upcoming overpass.

Implemented By: SatelliteQueue Module

8 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Req.	Modules
REQ-MCT-001	BackendService
REQ-MCT-002	User
REQ-MCT-003	Authentication, User
REQ-MCT-004	Schedule
REQ-MCT-005	Schedule
REQ-MCT-006	Satellite
REQ-MCT-007	Satellite, Schedule
REQ-MCT-008	Satellite, Log
REQ-MCT-009	Schedule
REQ-MCT-010	Schedule
REQ-MCT-011	Schedule
REQ-MCT-012	Schedule
REQ-MCT-013	Schedule
REQ-MCT-014	Schedule
REQ-MCT-015	User
REQ-MCT-016	User
REQ-MCT-017	User
REQ-MCT-018	User, Schedule
REQ-MCT-019	Schedule
REQ-MCT-020	Schedule
REQ-MCT-021	Schedule
REQ-MCT-022	Satellite
REQ-MCT-023	Satellite
REQ-MCT-024	Satellite
REQ-MCT-025	Satellite
REQ-MCT-026	Satellite
REQ-MCT-027	Satellite, Schedule, SatelliteQueue
REQ-MCT-028	Satellite, Schedule, SatelliteQueue
OPT-MCT-001	Schedule
OPT-MCT-002	Schedule
OPT-MCT-003	Schedule
OPT-MCT-004	Schedule
OPT-MCT-005	Schedule
OPT-MCT-006	Schedule
OPT-MCT-007	Schedule
OPT-MCT-008	BackendService

Table 2: Trace Between Requirements and Modules

Anticipated Change	Modules
AC1	Satellite
AC2	Satellite
AC3	Satellite
AC4	Satellite
AC5	Satellite
AC6	Satellite, Schedule

Table 3: Trace Between Anticipated Changes and Modules

9 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. **?** said of two programs A and B that A *uses* B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A *uses* B if there exist situations in which the correct functioning of A depends upon the availability of a correct implementation of B. Figure 9 illustrates the use relation between the modules. It can be seen that the graph is a directed acyclic graph (DAG). Each level of the hierarchy offers a testable and usable subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

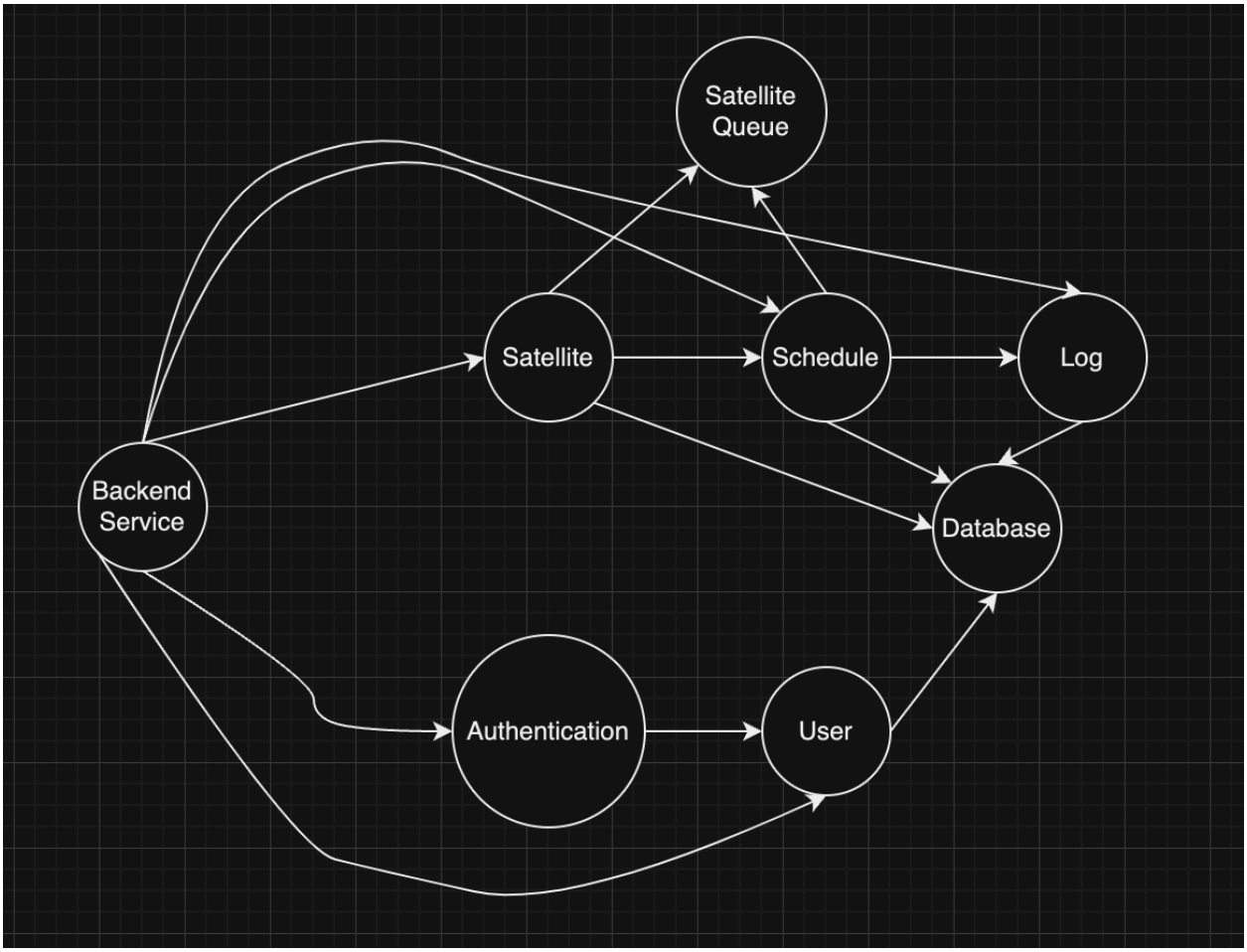

Figure 1: Enter Caption

10 Timeline

Module	**People Responsible**	**Deadline**
Satellite	Quinn	Jan 31, 2024
Schedule	Quinn, Umang, Rishi	Jan 31, 2024
User	Dhruv	Jan 25, 2024
SatelliteQueue	Umang, Rishi	Jan 31, 2024
Log	Quinn	Jan 20, 2024
Authentication	Diamond	Jan 25, 2024
Database	Umang, Rishi	Jan 31, 2024
BackendService	Diamond, Dhruv	Jan 31, 2024

Table 4: Table for the timeline of the module