Module Guide for MCT

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

April 4, 2024

1 Revision History

Date	Version	Notes
Jan. 17, 2024	1.0	Completed MG for MCT application
April 3, 2024	1.1	Updated documentation to reflect current implementation of the application and updated MG based on feedback

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 the team, 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 by Lower Earth Orbiters. 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". 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. *MCT* means the module will be implemented by the MCT 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	Module Numbers
REQ-MCT-001	BackendService	6.1.8
REQ-MCT-002	User	7.2.3
REQ-MCT-003	Authentication, User	7.2.4, 7.2.3
REQ-MCT-004	Schedule	7.2.2
REQ-MCT-005	Schedule	7.2.2
REQ-MCT-006	Satellite	7.2.1
REQ-MCT-007	Satellite, Schedule	7.2.1, 7.2.2
REQ-MCT-008	Satellite, Log	7.2.1, 7.2.6
REQ-MCT-009	Schedule	7.2.2
REQ-MCT-010	Schedule	7.2.2
REQ-MCT-011	Schedule	7.2.2
REQ-MCT-012	Schedule	7.2.2
REQ-MCT-013	Schedule	7.2.2
REQ-MCT-014	Schedule	7.2.2
REQ-MCT-015	User	7.2.3
REQ-MCT-016	User	7.2.3
REQ-MCT-017	User	7.2.3
REQ-MCT-018	User, Schedule	7.2.3 7.2.2
REQ-MCT-019	Schedule	7.2.2
REQ-MCT-020	Schedule	7.2.2
REQ-MCT-021	Schedule	7.2.2
REQ-MCT-022	Satellite	7.2.1
REQ-MCT-023	Satellite	7.2.1
REQ-MCT-024	Satellite	7.2.1
REQ-MCT-025	Satellite	7.2.1
REQ-MCT-026	Satellite	7.2.1
REQ-MCT-027	Satellite, Schedule, SatelliteQueue	7.2.1, 7.2.2
REQ-MCT-028	Satellite, Schedule, SatelliteQueue	7.2.1, 7.2.2
OPT-MCT-001	Schedule	7.2.2
OPT-MCT-002	Schedule	7.2.2
OPT-MCT-003	Schedule	7.2.2
OPT-MCT-004	Schedule	7.2.2
OPT-MCT-005	Schedule	7.2.2
OPT-MCT-006	Schedule	7.2.2
OPT-MCT-007	Schedule	7.2.2
OPT-MCT-008	BackendService	6.1.8

Table 2: Trace Between Requirements and Modules

Anticipated Change	Modules	Module Numbers
AC1	Satellite	7.2.1
AC2	Satellite	7.2.1
AC3	Satellite	7.2.1
AC4	Satellite	7.2.1
AC5	Satellite	7.2.1
AC6	Satellite, Schedule	7.2.1, 7.2.2

Table 3: Trace Between Anticipated Changes and Modules

9 Use Hierarchy Between Modules

The Log module is one of the basic modules in the hierarchy. Both Satellite Queue and Database modules use it for logging. The User module uses the Database module too for managing user-related data. The Backend Service is the orchestrator of the system, and some of the key components employed by the Backend Service to assume its roles include the User and Authentication modules and the Satellite Queue with the Schedule, all to manage the satellite data and operations. This implies a hierarchy situation where every higher-level module depends on the functioning of lower ones, thereby, allowing systematic testing of the system architecture without having circle dependencies.

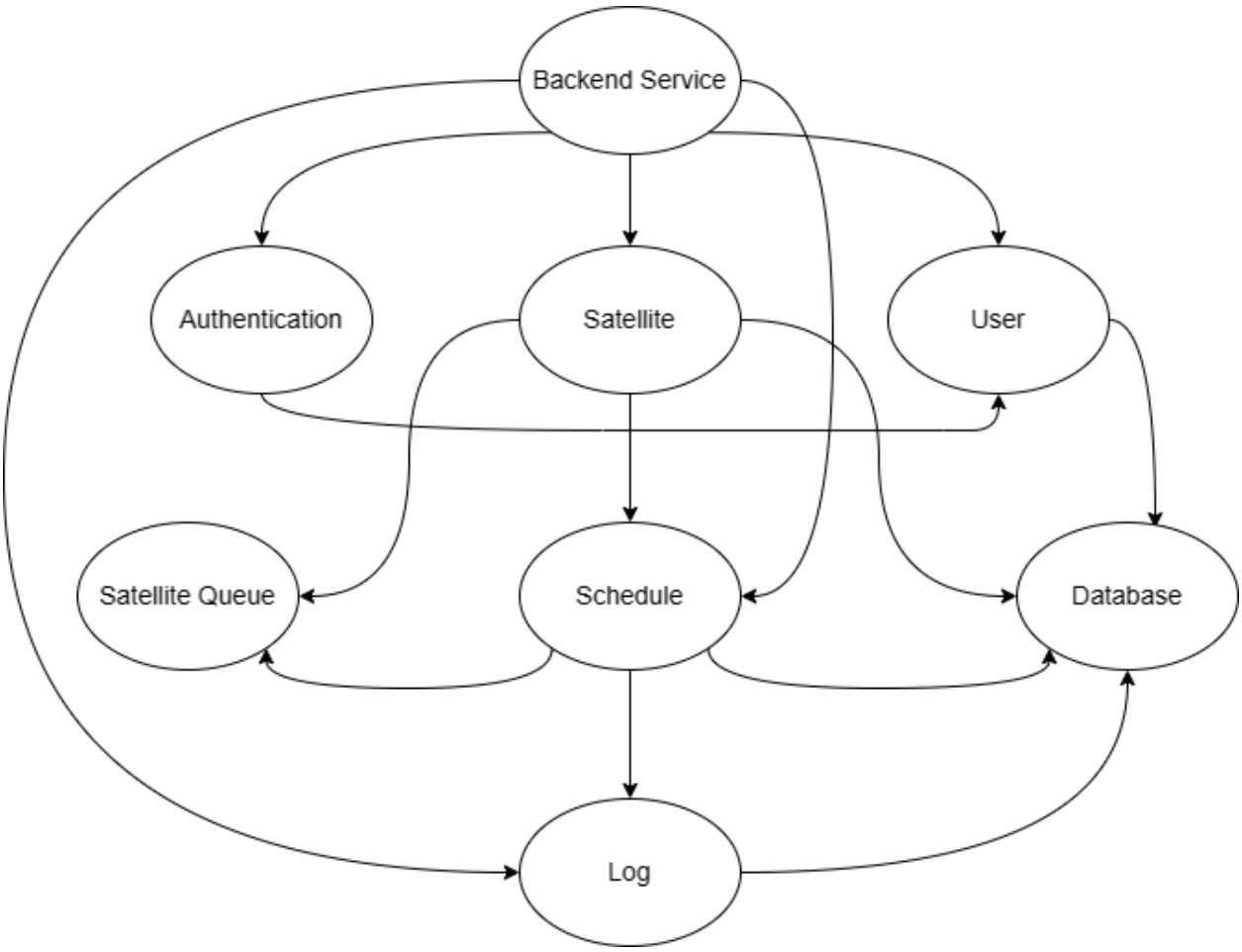

Figure 1: DAG diagram depicting the hierarchy between modules

10 Timeline

The deadlines in our project's timeline are planned to match the main goals we aim to achieve in each stage. We've split the work into parts, each with its own deadline, so we can work on different things at the same time and make sure everything comes together smoothly. For example, we're finishing the User module early because it's important for setting up who can do what in the system. Getting the Authentication module done early is also key because it makes sure that only the right people can access the system while we keep building it. The set deadlines help us keep track of our progress. This way, we can make sure we finish the whole project on time without rushing and making mistakes. In short, our timeline helps us stay organized and work efficiently so that we can complete our project successfully and on time.

Module and Description	**People Responsible**	**Deadline**
Satellite (7.2.1) - Should implement future overpass, satellite information and record updating functionalities. Should add test cases for all the endpoints created	Quinn	Jan 31, 2024
Schedule (7.2.2) - should create the endpoints required for adding commands to schedules and testing them	Quinn, Umang, Rishi	Jan 31, 2024
User (7.2.3) - should create endpoints for creating, updating and permissions for users. Should add test cases for all the endpoints created	Dhruv	Jan 25, 2024
SatelliteQueue (6.1.4) - Should add the functionality to execute schedules in the satellite queue based on the overpass times.	Umang, Rishi	Jan 31, 2024
Log (7.2.6) - should create endpoint to receive logs for all commands and make the frontend page.	Quinn	Jan 20, 2024
Authentication (7.2.4) - should create functionality to authenticate a user using auth0	Diamond	Jan 25, 2024
Database (6.1.7) - should create all the schemas for users, satellites, commands, schedules and logs.	Umang, Rishi	Jan 31, 2024

Table 4: Table for the timeline of the module