Research and Development of the scripting engine for the ESTCube-2 Mission Control

Revision 0.1 (2017-08-31)

Proposed by:
Umesh .A Bhat
Team Lead - Mission Control System (ESTCube)
umesh.bhat@estcube.eu, +372 56168039
# Table of Contents

Abstract 3  
Introduction 3  
**Modules** 4  
  Scripting Engine 4  
  Scripting Scheduler 4  
  Scheduling Database 5  
**Tasks** 5  
**Notes** 5  
**Review Process** 5  
**Intellectual Property** 6  
**References** 6  
**Appendix** 7  
  EC-2 MCS Development Workflow 7  
  EC-1 Space Data Chain 7  
  EC-2 MCS Architecture (Snipped) 8  

Proposal: Research and Development of the scripting engine for the ESTCube-2 Mission Control  
Approved for Public Distribution  
© August 2017, Eesti Tudengisatelliidi Sihtasutus (Estonian Student Satellite Foundation)
Abstract

This is a project proposal for the Software Project course [3] for Tartu Ülikool IT students. The objective is to research and develop a scripting engine for the ESTCube-2 Mission Control System (EC-2 MCS) based on set guidelines and architecture. The scripting engine enables the operator to schedule satellite command and control scripts to run on the EC-2 MCS at defined intervals or on-demand. The deliverables include the scripting engine, the scripting engine scheduler, the schedule database and all the accompanying documentation. The developers are expected to deploy the developed solution into the staging area of the EC-2 MCS.

Introduction

The ESTCube-2 Mission Control System (EC-2 MCS), based on the microservices architecture [1] is crucial for the ESTCube-2 mission [2] operations. The EC-2 MCS connects the satellite operator and the satellite via the ground stations. It enables the operators to command and control the entire mission through the web browser. In the background, the EC-2 MCS is comprised of multiple independent applications talking to each other via REST API’s. Each application is hosted in a Docker [8] container and performs one specialized role. This enables easy development, modification, upgrades, replacement and replication of the entire system over time.

The operator communicates with the satellite with the help of predefined commands [9] set forth by the EC-2 OBCS (On-board Computer System) subsystem team. The commands range from simple telemetry operations to complex orbital maneuvers. Hence, it is vital that the system provides a fail-safe (to a certain degree) way of operating the satellite.

The operator schedules commands to be run on-board the satellite with the MCS command terminal [10] [11]. The commands are first processed by the EC-2 OBCS Codec [12], which converts them into AX.25 packets [13]. The packets are then given a unique ID (tracked in a separate database) and assembled according to the ICP (Internal Communication Protocol) [13]. If there are files to upload, the command terminal picks the files from a file repository (Git) and splits them using a packet splitter. The packets are scheduled through the Contact Automation module, which automatically calculates the satellite over-passes and picks the nearest authorized ground station for transmission. The responses from the satellite are archived and then displayed on the command terminal.

The command terminal works good for singular command execution, but for multiple commands, automatic execution and response handling, a better solution is required. Hence, the MCS scripting engine was conceived. The EC-2 MCS scripting engine serves two main purposes: multi-line script execution and intelligent responses. This enables the operator to quickly and easily run pre-defined scripts at scheduled intervals with a high degree of automation and minimal errors. The scripting engine empowers all the EC-2 subsystem teams to run their experiments and collect responses from a centralized and standardized system.
Modules

*More details to emerge after technical discussion with developers*

The modules to be developed are listed here:

Scripting Engine

The EC-2 MCS scripting engine module is an independent REST API application and bundled with a scripting language. The scripting language should ideally be a DSL language and should have the right balance between capabilities and ease-of-use. But feel free to have an open mind, you may also evaluate other solutions to tackle this, for example: an event-driven architecture ([https://www.nginx.com/blog/event-driven-data-management-microservices](https://www.nginx.com/blog/event-driven-data-management-microservices)). The command scripts would be stored in a separate repository. The scripting engine API would update the repository and support basic CRUD operations. A Git repository is chosen over a database to store the scripts, as version control is a better choice for script files and rollbacks. The scripting engine API would refer the EC-2 OBCS codec while executing the command scripts. The scripting engine then pushes the commands to be executed.
Scripting Scheduler

The EC-2 MCS scripting scheduler is an independent REST API application and handles the current and periodic execution of command scripts. The developers are free to chose the underlying technology for reliable scheduling, execution and exception handling.

Scheduling Database

The scheduled commands are stored in a relational database. The developers are free to utilize a NoSQL database, provided there are valid arguments. Also, refer http://sdtimes.com/choose-right-database-microservices.

Tasks

- Research and figure out the best scripting language for the scripting engine.
- Divide, assign and follow the tasks in JIRA as a team.
- Document the scripting engine API in Apiary.
- Develop the required modules in Gitlab using TDD (Test-driven development).
- Deploy using continuous integration to Unicorn.
- Document the deliverables in detail in Confluence.

Notes

- Need a state machine for performing sequences of operations, depending on some specific telemetry parameters.
- You can use the EC-1 Scripting Engine [4][15][16] for reference.
- During the course of development, the developers would be interfacing with the on-ground version of ESTCube-2/ satellite simulator.

Review Process

- The Tartu Ülikool supervisors would be provided with ESTCube-2 accounts providing them access to:
  - JIRA - Tasks and Issues
  - Confluence - Documentation
  - Bitbucket - Code Repository
  - Gliffy - Architecture/Workflow Illustration
  - Google Drive - Real-time Collaboration
  - Gitlab - Collaborative Development and Continuous Integration
  - Apiary - API Specification
  - Moqups - GUI Mockups (if required)
  - Unicorn - The EC-2 MCS Development Server (Ubuntu + Docker) over SSH
● The student team would be working in collaboration with the EC-2 MCS team and using Fleep (Fleep.io) and Skype as their primary communication channels.
● In-person meetings would happen at the Institute of Computer Science meeting rooms (https://www.cs.ut.ee/en/studying/facilities-equipment) or at Tartu Observatory.
● All design and work decisions have to be documented in the official channel.

Intellectual Property

The resulting intellectual properties and outcomes would be licensed to the Estonian Student Satellite Foundation under separate licence agreements with the developers.

References

[1] EC-2 MCS Architecture
https://icubesat.org/papers/2017-2/2017-b-3-4-microservices-architecture-for-cubesat-mission-control-systems/
[4] EC-1 Scripting Engine
https://confluence.tudengisatelliit.ut.ee:8433/display/MCS/Scripting+Engine
https://confluence.tudengisatelliit.ut.ee:8433/download/attachments/12648821/MD_Gatis_Stevenbergs_FINAL.pdf?version=1&modificationDate=1405344020000&api=v2
[9] EC-2 Commands
https://docs.google.com/spreadsheets/d/1-KZvg9i6nXI0vCUJVRvbMS3qiiDKowmq9Sa1BSe4ZM
[11] EC-2 Command Terminal -
https://confluence.tudengisatelliit.ut.ee:8433/display/EC2MCS/Command+Terminal
[12] EC-2 OBCS Codec -
https://confluence.tudengisatelliit.ut.ee:8433/display/EC2OBCS/MCS+codec
[15] Sample task - Separate Scripting engine from the old MCS -
https://kratt.physic.ut.ee:8443/browse/MC-9
[16] EC-1 Space Data Chain
https://confluence.tudengisatelliit.ut.ee:8433/display/EC2MCS/Space+Data+Chain

The confluence.estcube.eu/tudengisatelliit.ut.ee links require an ESTCube account
Appendix

EC-2 MCS Development Workflow

EC-1 Space Data Chain

Proposal: Research and Development of the scripting engine for the ESTCube-2 Mission Control

Approved for Public Distribution
© August 2017, Eesti Tudengisatelliidi Sihtasutus (Estonian Student Satellite Foundation)
Proposal: Research and Development of the scripting engine for the ESTCube-2 Mission Control

Approved for Public Distribution

© August 2017, Eesti Tudengisatelliidi Sihtasutus (Estonian Student Satellite Foundation)