



# Developing a Mission Control System for a Small LEO Satellite

Dr. Jacobus van Zyl  
Sun Space & Information Systems

# Agenda

- ▶ Background
- ▶ Development Approach
- ▶ Build Environment
- ▶ Testing Approach
- ▶ Issue Tracking

# Background

- ▶ What is an MCS?
  - A comprehensive system for monitoring and controlling a satellite
  - It should allow save, efficient, and traceable execution of all satellite operations
- ▶ What is involved?
  - Ground Segment
    - Radios, antennas, satellite dishes, etc.
  - Space Segment
    - On board electronics, various on board computers

# Mission Control System (MCS)

- ▶ Characteristics of our MCS
  - Distributed
    - multiple operators performing different tasks
    - various hardware, possibly remote
  - Real-time operations
  - Multi language
  - Multi platform

# Development Approach

- ▶ Follow an iterative software development approach
  - MCS is developed iteratively (sprints)
  - During each sprint a subset of prioritised features from the backlog item list are added to the product
  - A sprint lasts 5 weeks, has 3 phases
    - Sprint planning and design review (2 to 4 days)
    - Development and testing: 20 to 22 days
    - Final release: 1 day
  - Each release is, although incomplete, of production quality
  - Use VersionOne to manage process

# Sprint Planning

- ▶ Backlog items list loaded from URS
- ▶ Product owner selects and prioritize items to be incorporated during a sprint
- ▶ Team members
  - Picks backlog items in the order of priority
  - Ensure specification is understood
  - Draw up technical specification document
  - Perform detail work estimate
  - Review with team leader and technical lead
  - Continue until capacity is filled

# Sprint Development

- ▶ Complete backlog items in order of priority
- ▶ Backlog item acceptance criteria
  - All backlog item tasks complete
  - Code adhere to coding standard
  - Unit tests complete
  - Release notes updated
  - Feature / Fix was demonstrated to technical lead or team leader

# Sprint Release

- ▶ Integration is a continuous effort
- ▶ Release day used to make a release
- ▶ Release candidate is tested
  - Defects discovered classified:
    - Fix / Feature remove / Document (known issue)
- ▶ A release meeting is held during the following week

# SCRUM meeting

- ▶ Platform for communication between all team members during the *sprint* development phase
- ▶ Main purpose:
  - update team members on progress (particularly the timeliness) with current backlog items,
  - communicate any difficulties that impede development and require attention from the team, and
  - communicate immediate plans.
- ▶ Not a technical session– working meetings scheduled from issues raised

# Build environment

- ▶ Accurev as source version control
- ▶ Luntbuild as build system
- ▶ Three types of builds
  - Nightly
    - Runs 3am each morning
    - Clean build
    - Performs automated unit tests and ATPs
  - Adhoc
    - Triggered manually
    - Perform clean build
  - Continuous integration
    - Occurs every 30 minutes (if code was checked in)
    - Incremental build
- ▶ Developers that checked in code automatically notified if build failed

# Unit Testing

- ▶ Unit tests written using NUnit, or own test frameworks
- ▶ Rhino mocks used to implement interface classes
- ▶ CAB used to provide modularity and decoupling
- ▶ Test results checked by unit test overseer, weekly report to entire team
- ▶ Each test has an author identification tag – if a test fails the author ensures that it gets fixed
- ▶ Unit tests with unidentified authors are assigned authors by the unit test overseer.
- ▶ Code coverage used to check effectiveness of unit test (NCover, NCover Explorer)

# Unit Testing

- ▶ Libraries
  - All public functions must have unit tests covering all lines of execution
- ▶ GUIs
  - Must be implemented using the MVP pattern
  - A humble view approach is preferred
  - No unit tests for the view is required
  - Unit tests for presenter functions with logic (e.g. if-then branches) are required
- ▶ Servers
  - Functions with logic must have unit tests
- ▶ Unit tests are not written by author of code
  - Familiarise other team members with other code sections
  - Serves as a form of code review
  - Motivate the original author to write readable code

# Integration Testing

- ▶ Two approaches:
  - manual
  - automated
- ▶ Manual
  - AIT (Assembly, Integration & Testing)
  - Performed against real system
  - Performed by developer during development
- ▶ Automated
  - Runs nightly
  - Install, test uninstall
  - WIP...

# Issue Tracking

- ▶ Bugs detected in released code
  - noted as a “Known Issues”
  - Version One defect entry describing the bug
  - The defect ID should be noted in the release notes
  - After resolving, the defect ID and bug description is entered in the release notes as a “Resolved Issue.”
- ▶ Bugs found in non-released code
  - not entered in Version One or the release notes
  - author notified