Early Flight Control System Overview

Roger Racine

9 Sept. 2002

Topics

- EFCS Task Statement
- Development Approach
- Architecture
- Test Environment
- Relevance to Redesign

EFCS Task Statement

- The SSFPO, in Feb. 1992, asked Draper to develop an Early Flight Control System (EFCS) as a feasibility demonstration of flight critical SDP-level functions essential for controlling the Space Station Freedom for Mission Builds 2-4.
  - Develop and demonstrate a system that could be used to provide schedule relief.
  - Implement simplified (as compared to the "mainline") versions of the essential systems (DMS, GN&C, EPS, C&T, etc.).
  - Replace the truss avionics with an MDM-based system.
  - Follow mainline truss avionics external interface specifications (to the SSCC, the Shuttle, and all lower level MDMs and firmware controllers).
  - Use rapid prototyping.

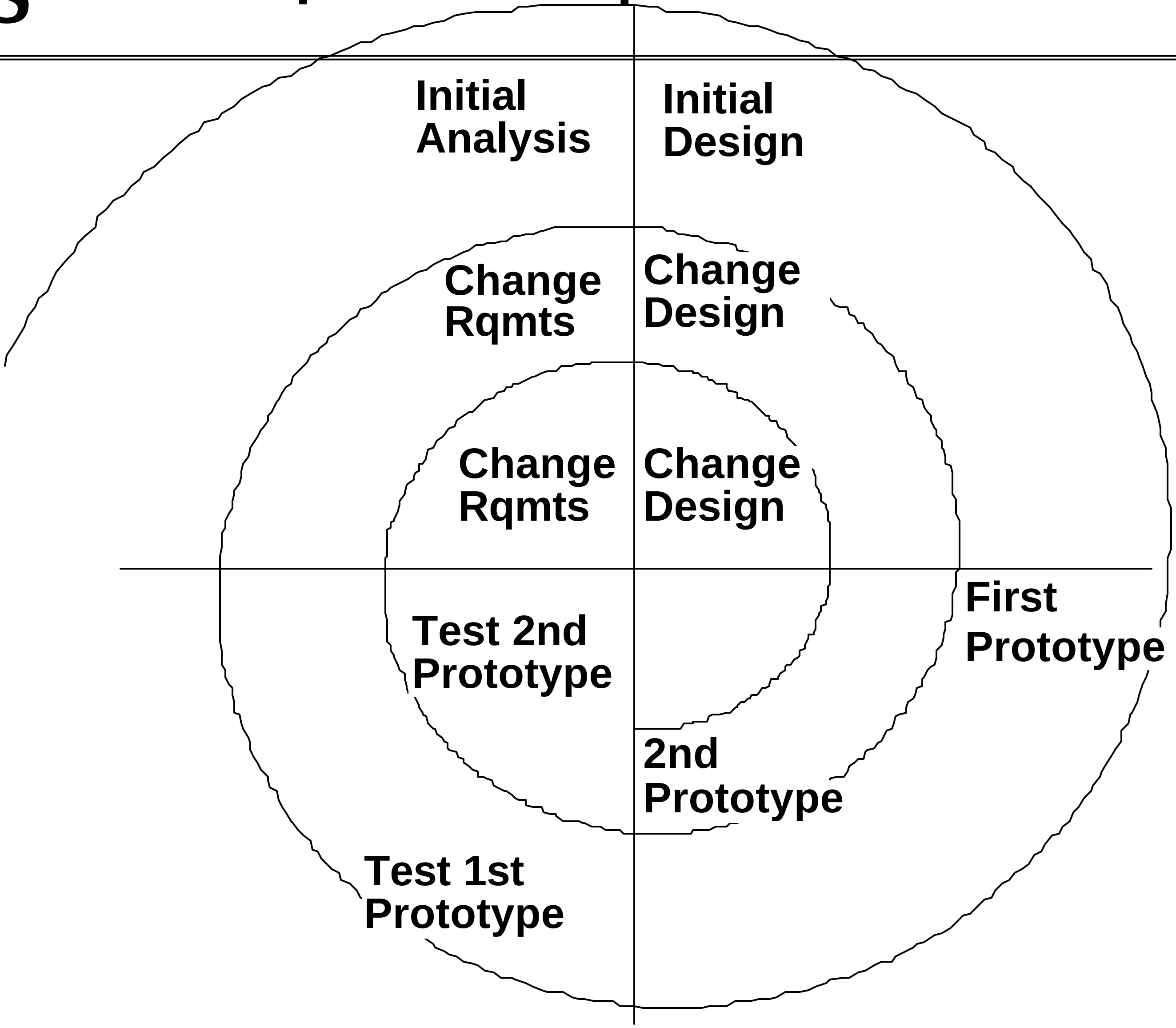
EFCS DEVELOPMENT

- Utilized a small multi-disciplined System Engineering Team.
- Designed an integrated system architecture allowing adding and modifying capabilities as required.
- Includes SSF system requirements for unmanned operations.
- Used the Rapid-prototyping life cycle (Progressive Refinement):
  - — Risk reduction methodology for system development.
  - — Early evaluation of system designs.
  - — Early identification of performance issues.
  - — Minimal early documentation.

EFCS Requirements Development

- Used Mainline Requirements documentation as starting point for requirements.
- Added GPS, for time, position, velocity and attitude.
- Simplified where appropriate.
  - — Eliminated functions to support payloads or manned operation.
- Generated requirements for each subsystem.

Spiral Development Model



Initial Analysis

Initial Design

Change Rqmts

Change Design

Change Rqmts

Change Design

First Prototype

Test 2nd Prototype

2nd Prototype

Test 1st Prototype

ISS

Software Development Phases

- Development process is "progressive refinement".
  - — Four demonstrations were scheduled:
    - First demonstration (Nov. 92) showed basic capability for each system, integrated into a Station-level simulation.
    - Second demonstration (May 93) refined and added further capability to each system.
    - Third demonstration was scheduled to refine the overall system, add further capability, show that Draper on-board software meets external interfaces, and show that GCS meets FSW interfaces.
      - Þ Redesign changed priorities; redesigned Data Management System interface.
    - Fourth demonstration was scheduled to show that the integrated system was essentially complete.

ISS

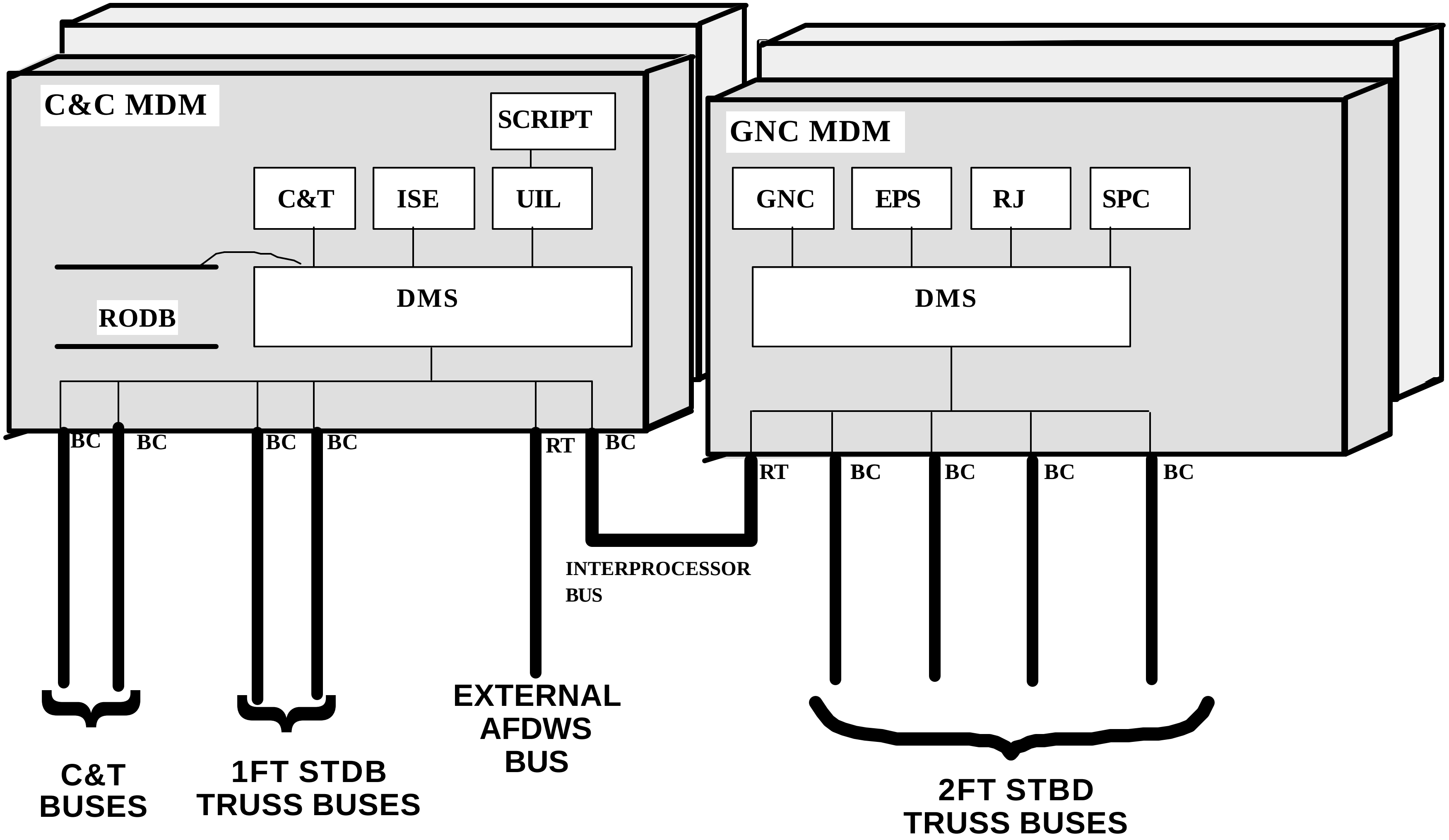
Top Level Functionality

SVCS	GNC	ISE	C&T	RJ	EPS
DMS P	Process Control C	ISE Cont C	ACS Gnd Comm C	Exec Cmds P	Exec & Control P
UIL P	Attitude Control C	SYS Cont C	End-to-End Gnd Comm C	Monitor MDM P	Monitor EPS S
	Nav & Guidance C	S. Pwr Cont C	FDI C	Auto Track S	Ctrl Primary Pwr S
	Attitude Determ P	FR P	ACS St’able Ant P	FDI S	FDI S
	Pointing & Support P	Station Modes S	NonACS C/O S		
	FDI S				

EATCS and OMCS are not included

Complete	Partial	Stub Only

ISS EFCS Functional Block Diagram



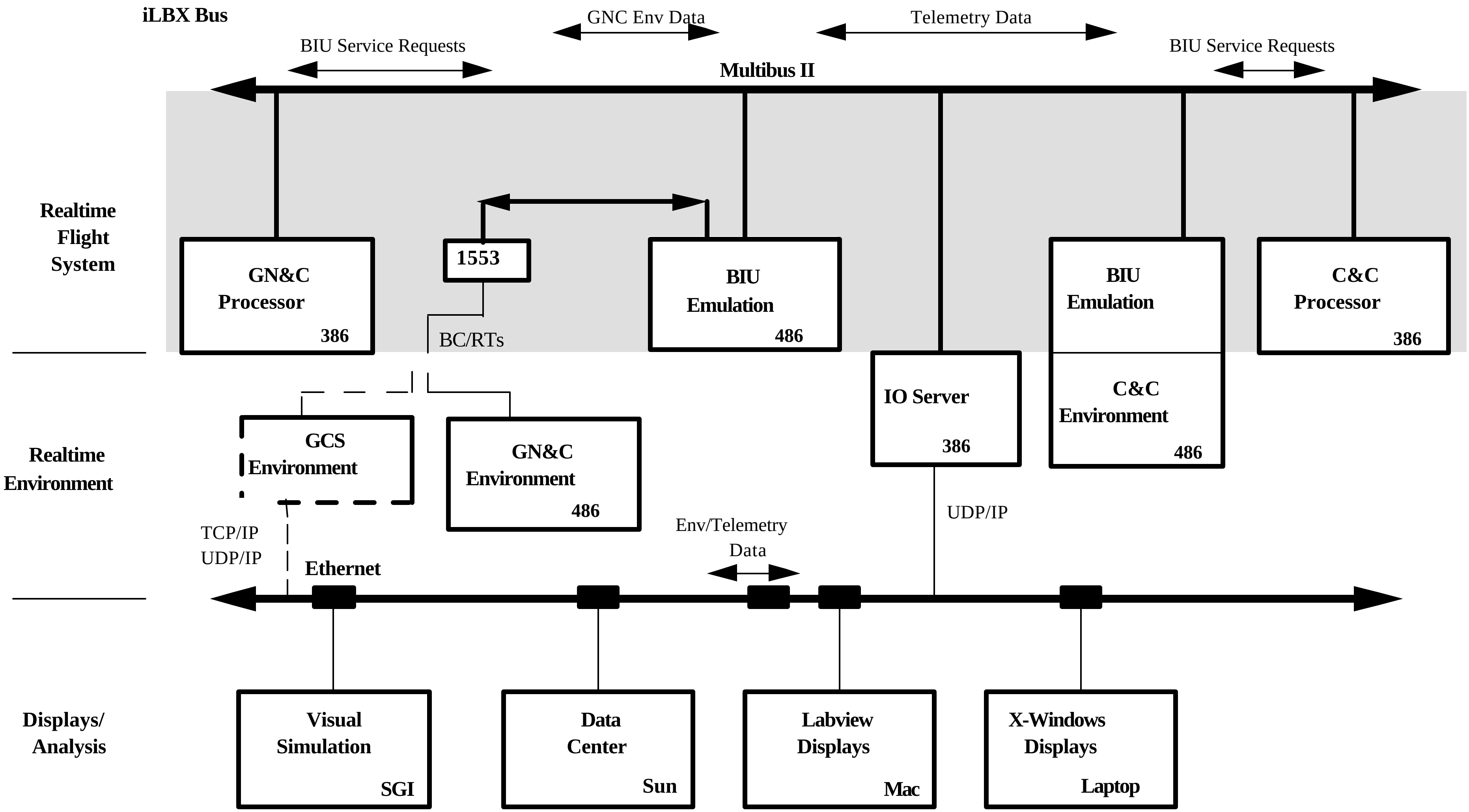
Development Environment

- Development utilizes the following process:
  - Integrated system development and testing is performed on a non-realtime host based configuration.
  - This integrated system is then moved to the Realtime Testbed.
- This two-phased process permits:
  - System development, system integration and integrated testing is performed without complication of realtime operations.
  - Realtime-specific modifications to integrated system are made as required when the integrated system is ported to the realtime testbed.

Host-Based Configuration

- Initial integration is performed, non-real-time, on host computer.
- Host and real-time testbed are running identical software except for machine-dependent routines.
- All systems and all environment modules, are linked together into one Ada program (real-time environment uses multiple Ada programs).
  - — Application interfaces remain the same.
- Unique within Space Station Program.
- Benefits
  - — Instrumenting software for debugging does not affect timing.
  - — It is possible to stop a simulation, look at data, and then continue.
  - — Many simulations can run at the same time.

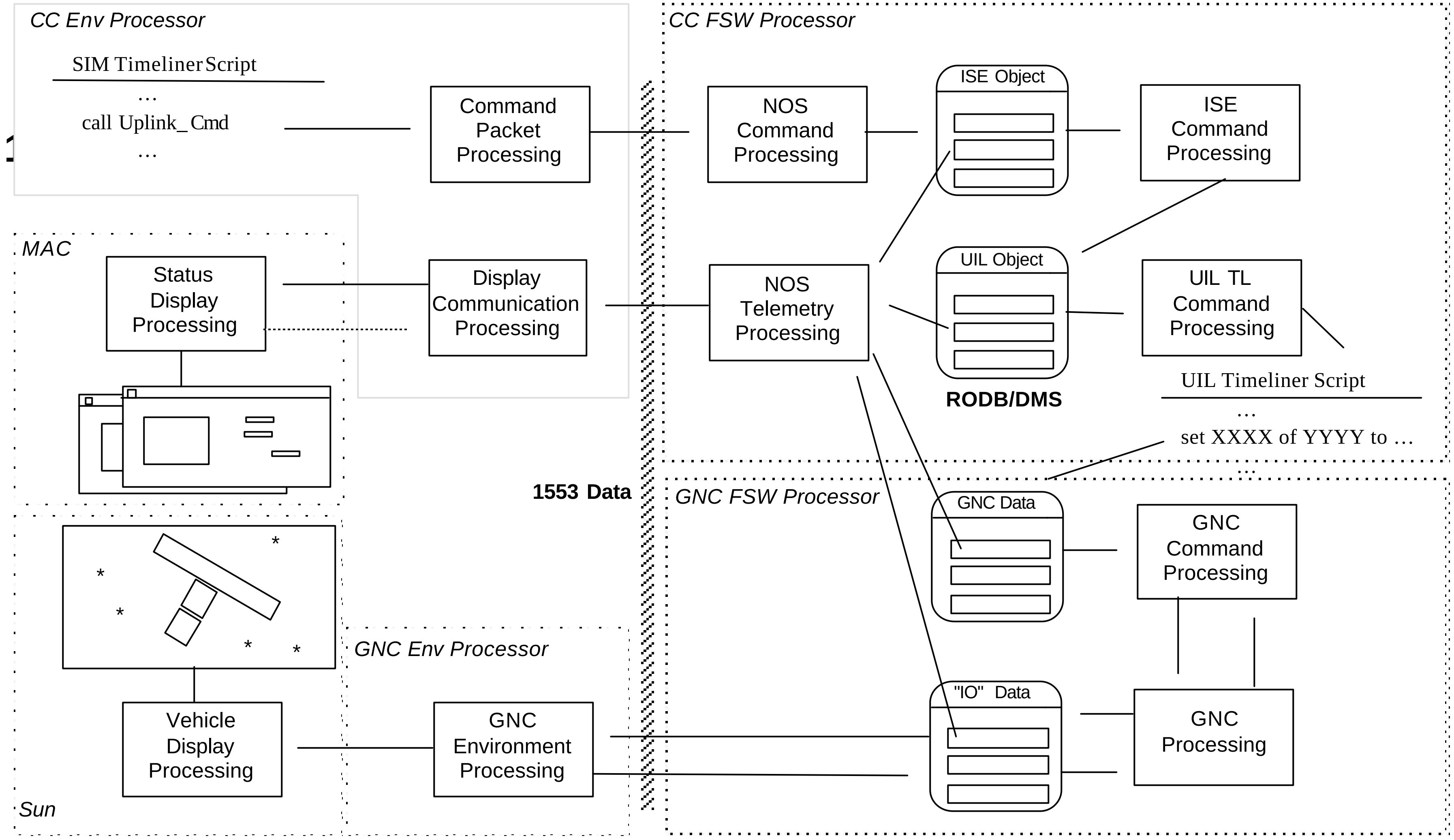
EFCS Test Bed Configuration



Real-Time Testbed

- Currently running flight software on the SDP emulation.
  - 80386DX (about twice as fast as MDM 80386SX).
  - Multibus II backplane bus.
  - No EEPROM; all RAM.
- Real-time environment models, along with a model of a Bus Interface Unit, run on 80486.
- Ethernet card is used by the Environment processors to send simulation data to “outside world”.
- Data Center collects and logs data, sends data to displays or analysis programs.

Demo Data Flow



Roles Needed

- For the Control System software, the following roles need to be partitioned among the available personnel:
  - — Overall leader
    - » Responsible for creating the Software Development Plan, maintaining the schedule, creating status reports, etc.
  - — Requirements Analyst
    - » Responsible for writing the Software Requirements Specification (SRS)
  - — Control algorithm developer
    - » Responsible for the design of the control systems
      - Generates at least the Top-Level Design documentation for the Control system
  - — Software architect
    - » Responsible for the high-level software design
      - Creates at least the Top-Level Design documentation laying out the structure of the software

Roles (Continued)

- — Control software coder
  - » Writes the Control software
- — Design documenter
  - » Writes the Detailed Design document
- — Test Lead
  - » Writes the Software Test Plan
- — Test SW algorithm developer
- — Test SW coder
- — Version Control person
  - » Responsible for dealing with the version control system
- — Integration lead
  - » The problem solver. Responsible for integrating the Control software with the other software in the ISS, and getting it to work
- — Display developer
  - » Takes telemetry data and displays it

Guidelines

- Expect requirements changes
  - Trying to stay ahead of the main developers means NASA or the contractors might change something
- The customer wants demonstrations. Part of the job is making sure the demonstrations are professional
  - Look good
  - Provide enough information to show the system working well
- All the software was developed quickly. There is no guarantee that problems are all due to new software