FSW Workshop 2011

Using Flight Software in a Spacecraft Simulation Tool

Debbie Clancy
JHU/APL
debbie.clancy@jhuapl.edu
443-778-7721

The Johns Hopkins University
APPLIED PHYSICS LABORATORY

Agenda

- **Overview of RBSP and FAST**
- **Technical Challenges "Dropping" FSW into FAST**
- **FAST FSW Modeling Efforts**
- **FAST Software Development Process**

Overview of RBSP and FAST

What is RBSP?

- **RBSP is the Radiation Belt Storm Probes Mission**
- **Two observatories that are spin stabilized ~5 RPM with the objective to provide an understanding, ideally to the point of predictability, of how populations of relativistic electrons and penetrating ions in space form or change in response to variable inputs of energy from the Sun**
- **Launch Readiness Date: Aug 2012**

What is FAST?

- **FAST is the Flight Accelerated Simulation Tool**

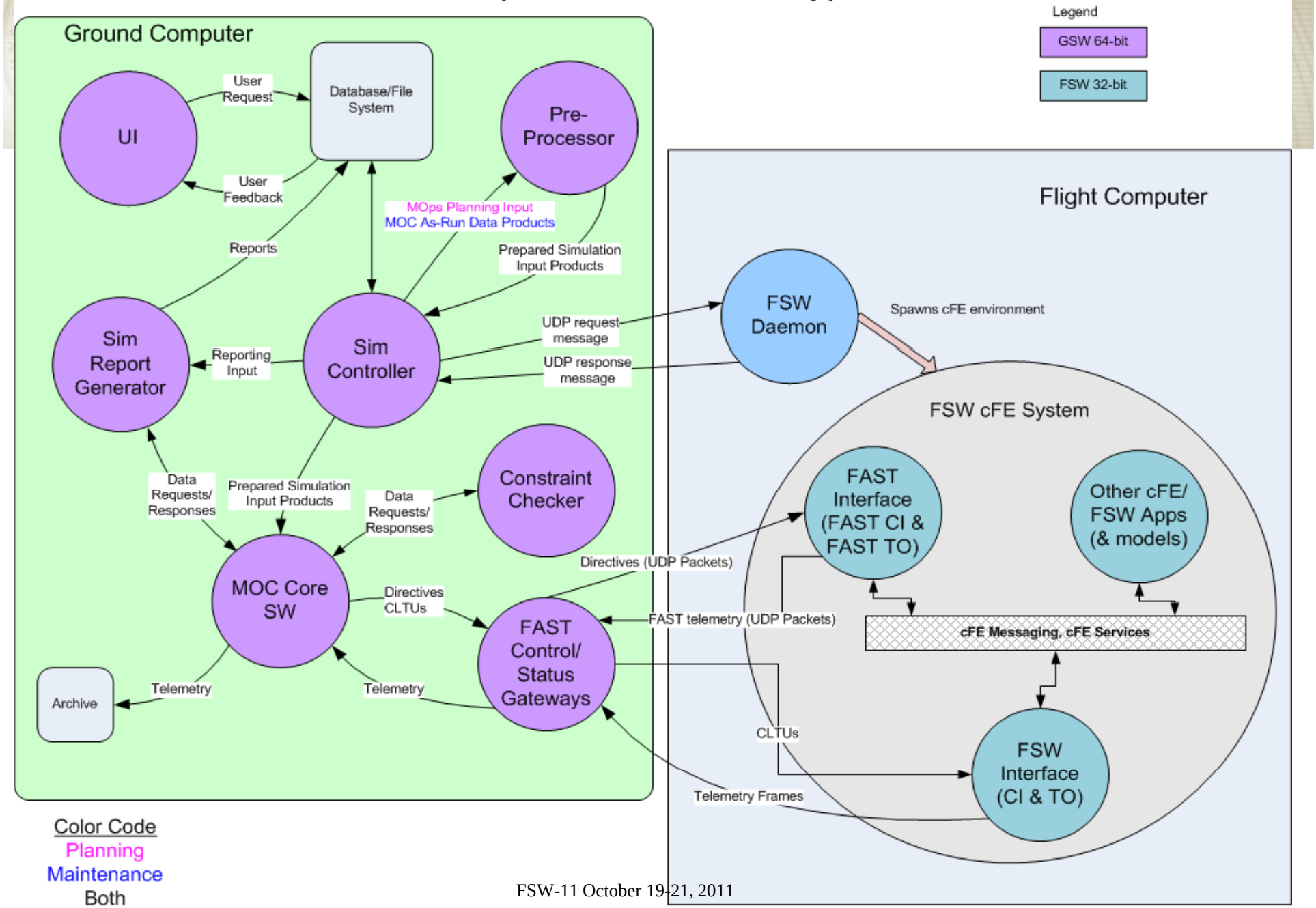
- **Mission Operations tool that simulates the state of the Spacecraft**
  - **Validate spacecraft command sequences (e.g. check for specific constraints) as a result of planning and scheduling activities; dry run the command sequences**
  - **Predict changes in spacecraft state arising from planned command sequences ; detect instances where predicted spacecraft states violate flight constraints**

- **Software only, hardware independent platform that simulates spacecraft states and functions at a rate faster than real-time**

Architecture Building Blocks

- **Architecture has three major components**
  - **User Interface - Web browser**
    - **Adobe Flex**
  - **Simulation Controller (”Ground Computer”)**
    - **Pre-processing activity**
    - **Simulation session control**
    - **Reporting**
    - **Constraint checking**
    - **JAVA, C++, XML, MySQL Database**
  - **Simulation Engine (“Flight Computer”)**
    - **Built on RBSP FSW which uses GSFC’s cFE (core Flight Executive)**
    - **Models/simulated components**
    - **C**

FAST Closed-Loop Design
(MOC Core in the Loop)



Technical Challenges
“Dropping” FSW into FAST

Three Technical Challenges for FAST/FSW

- **Endian Nature of the Processor**
- **Time Management**
- **Persisting and Initializing State**

#1 Endian Nature of Processor (1 of 4)

- **Endian 101**
  - **Issue is how bytes are stored in memory**
  - **Big Endian: Most significant byte in lowest address and least in high address**
    - **PowerPC (e.g. RAD750), SPARC (e.g. Leon)**
  - **Little Endian: Least significant byte in lowest address and most in high address**
    - **Intel (e.g. PCs, Pentium)**

#1 Endian Nature of Processor (2 of 4)

- **CCSDS communications protocol specifies bit and byte ordering**
  - **Similar to Internet Protocol which defines Big Endian is the standard network byte order. CCSDS is Big Endian.**
- **Although APL FSW is compliant with CCSDS communication protocols, it is built for and runs on Big Endian processors**
- **FAST Challenge:**

  From FSW executing on a 33-Mhz Rad-750 32-bit running VxWorks (Big Endian)

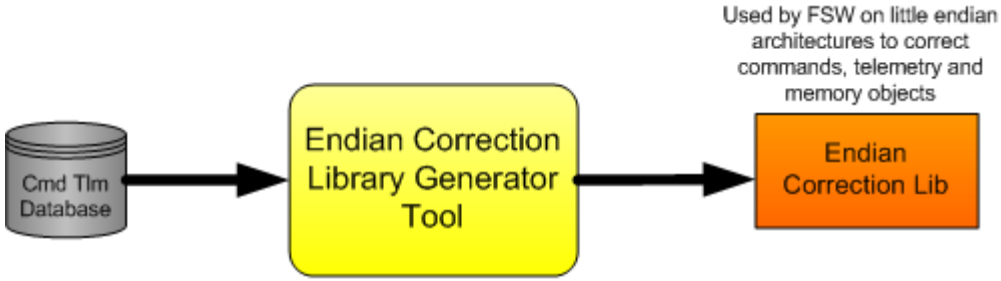
  To FSW executing on a multi-core 3.06-Ghz x86 64-bit running Real-Time Linux (Little Endian)
  - **Command argument processing and telemetry generation are Endian sensitive**

#1 Endian Nature of Processor (3 of 4)

- **FAST Solution:**
  - **RBSP did not take on the challenge of making FSW purely 'endian neutral'**
  - **Solving the problem is more complicated than blind byte swapping of words**
    - **Data format must be known in order to properly interpret commands or generate telemetry**

#1 Endian Nature of Processor (4 of 4)

- **FAST Solution (continued):**
  - **RBSP inserted endian corrections in strategic places in FSW (enabled for FAST tool compilation only)**
    - **Perl script handles the code generation for byte swapping of all bit-packed fields (i.e. the endian correction library)**
      - **Solves majority of the cases**



All of the information to perform Endian correction on commands and telemetry can be found in the command and telemetry database.

#2 Time Management

- **FAST Challenge:**
  - **Run FSW faster-than-real-time while keeping GSW in sync**

- **FAST Solution:**
  - **GSW and FSW communicate control via directives**
    - **Run Faster than Real Time**
    - **Run Real Time**
  - **FSW component uses >1 processor core**
  - **FSW component uses some real-time Linux processes**
  - **FAST utilizes 2 machines to support one instance of the tool**
  - **Time Correlation**
    - **GSW syncs its time to FSW MET**

#3 Persisting State and Initializing State

- **FAST Challenge:**
  - **Ability to capture FSW (i.e. spacecraft) state data during and after simulation**
  - **Ability to load state data before simulation and save at the end of a run**

- **FAST Solution:**
  - **Hear details in another FSW-11 presentation by Chris Monaco, "RBSP Mission Ops Flight Software Simulator – Saving and Restoring Sessions"**

FAST FSW Modeling Efforts

Modeling C&DH FSW

- **Using *"the"* FSW in FAST gives you the 'real' functionality (e.g):**
  - **Command management**
  - **Memory management**
  - **Load and execute time-tags**
  - **Autonomy engine**
  - **Storage variables***
  - **Computed telemetry***
  - **Rules***
  - **Macros**
  - **Data Collection Buffer**
- **Closed loop system provides all the events**

***** ***"Real" execution results dependent on fidelity of data***

FAST "Models"

- **Models fill-in for hardware not present (e.g. PDU subsystem, SSR)**
- **Models support constraint checking and reports**
- **Can be simple (set state) or complicated (emulate some aspect of the hardware)**
- **For RBSP, these have been simple…**

Modeling Spacecraft Subsystems

- **Initially a simple model of the Integrated Electronics Module (IEM) interfaces**
  - **Accept subsystem commands and produce static (i.e. 'canned') telemetry with the possibility of changing (simple) telemetry based on command**
- **Build upon model as understand constraint needs**
- **Model control via FAST directives**

Modeling SSR: CFDP and File System

- **CFDP (CCSDS File Delivery Protocol)**
  - **A protocol to support a reliable and efficient way to downlink files**
- **CFDP will not be closed loop (acknowledged mode)**
  - **CFDP software still runs but assumes all transactions complete successfully**
  - **Some CFDP telemetry can be tracked (e.g. closed transactions)**

- **File System**
  - **FAST emulates the file system**
  - **Accounting of file sizes and locations based on recording setup**
  - **Dependent upon instrument models producing correct loading (average data rates)**

Modeling Instrument Interface(s)

- **Minimum model to support SSR needs**
  - **Provide average telemetry rates**

FAST Automated Test Agent

- **Permits a FAST simulation run to be executed without user intervention (i.e. bypassing the user interface)**
- **Driven by an input file or by a request coming in over a socket that triggers a simulation run**
- **Test agent polls the database, much like the UI, to determine when the simulation run has completed**
- **Resulting log files and artifacts are collected for post-test analysis**

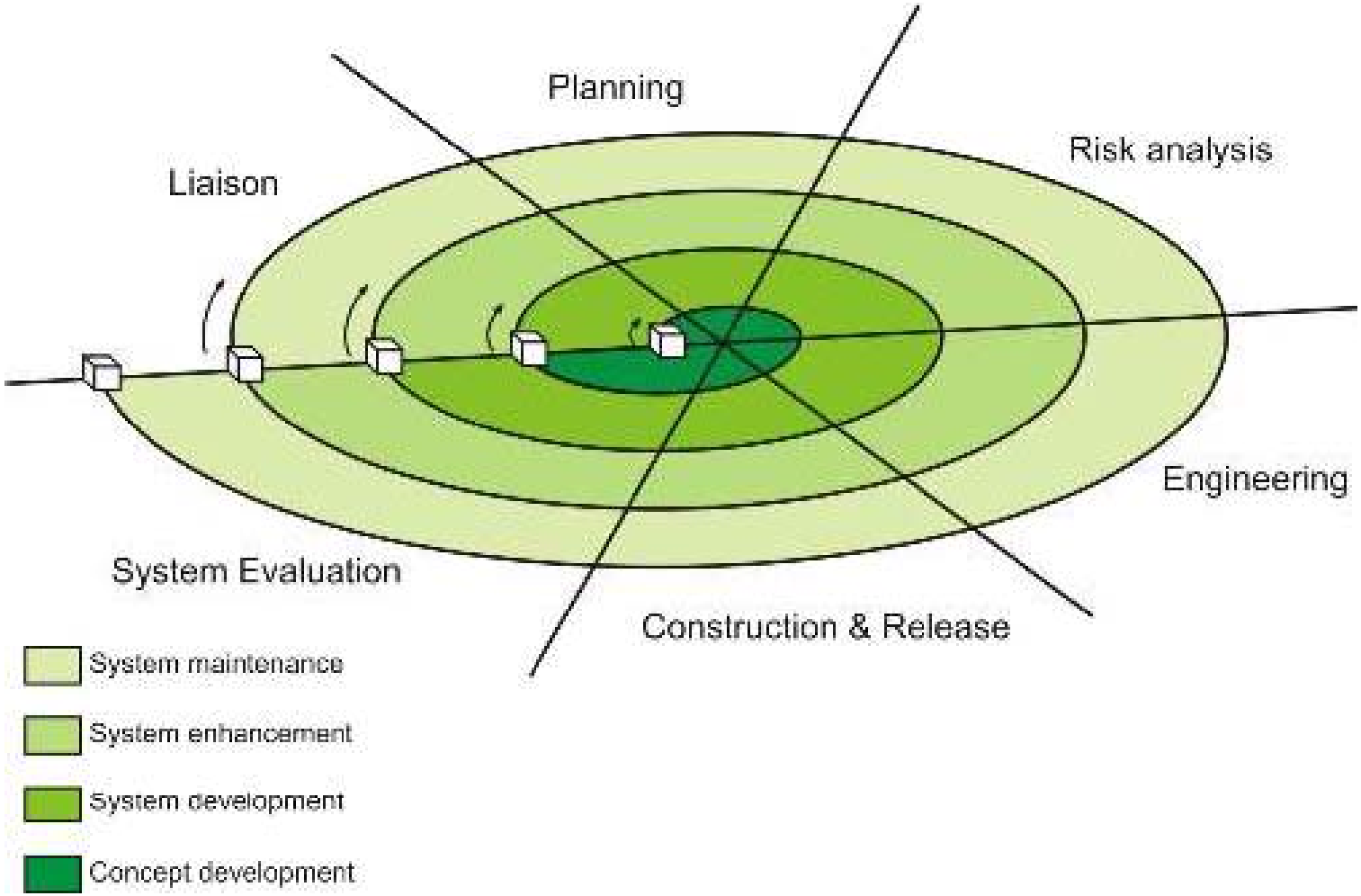
- **Excellent method for repetitive runs, 'weekend' runs**

FAST Software Development Process

Software Development Process for FAST

- **Requirements were not fully understood early on**
- **System design was not obvious, multiple system designs to be considered, each with advantages and disadvantages**
- **Initial approach forward had potential for significant technical unknowns (ref. technical challenges)**
- **Final answer – SPIRAL methodology applied**

Software Development Process - Spiral



The FAST Spiral Approach

- **Three development spirals identified:**
  - **Spiral 1: Control flow**
  - **Spiral 2: Data flow**
  - **Spiral 3: Completion**

Thank You!

Questions/Comments…..