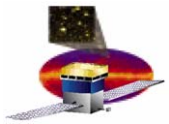
GLAST Large Area Telescope

Instrument Flight Software
EM2 Design Review
29 January 2004

Overview / Management / Schedule

T. L. Schalk
U. C. Santa Cruz

tas@slac.stanford.edu



Overview Roadmap

- Observatory
- FSW team organization
- Design strategy
- Engineering Model 2
- FSW methodology
- Code life cycle(s)
- Metrics
- Schedule
- Drivers

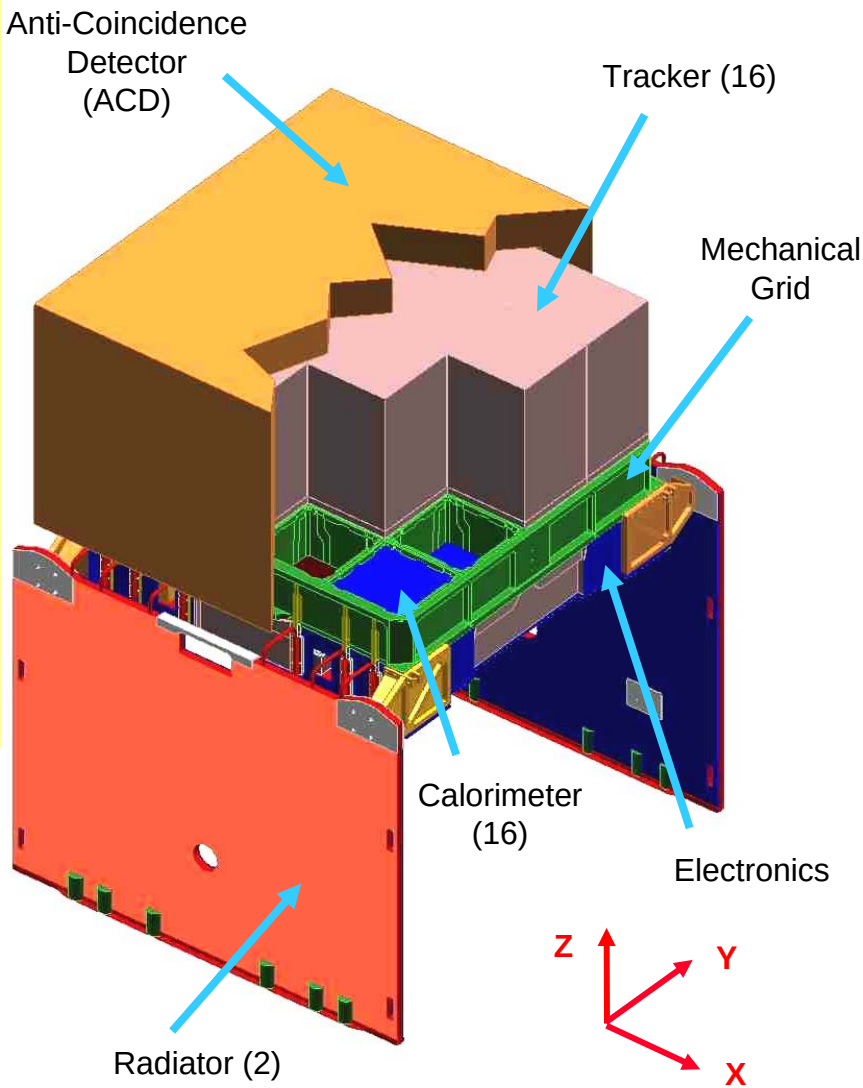
On the road to discovery

Large Area Telescope

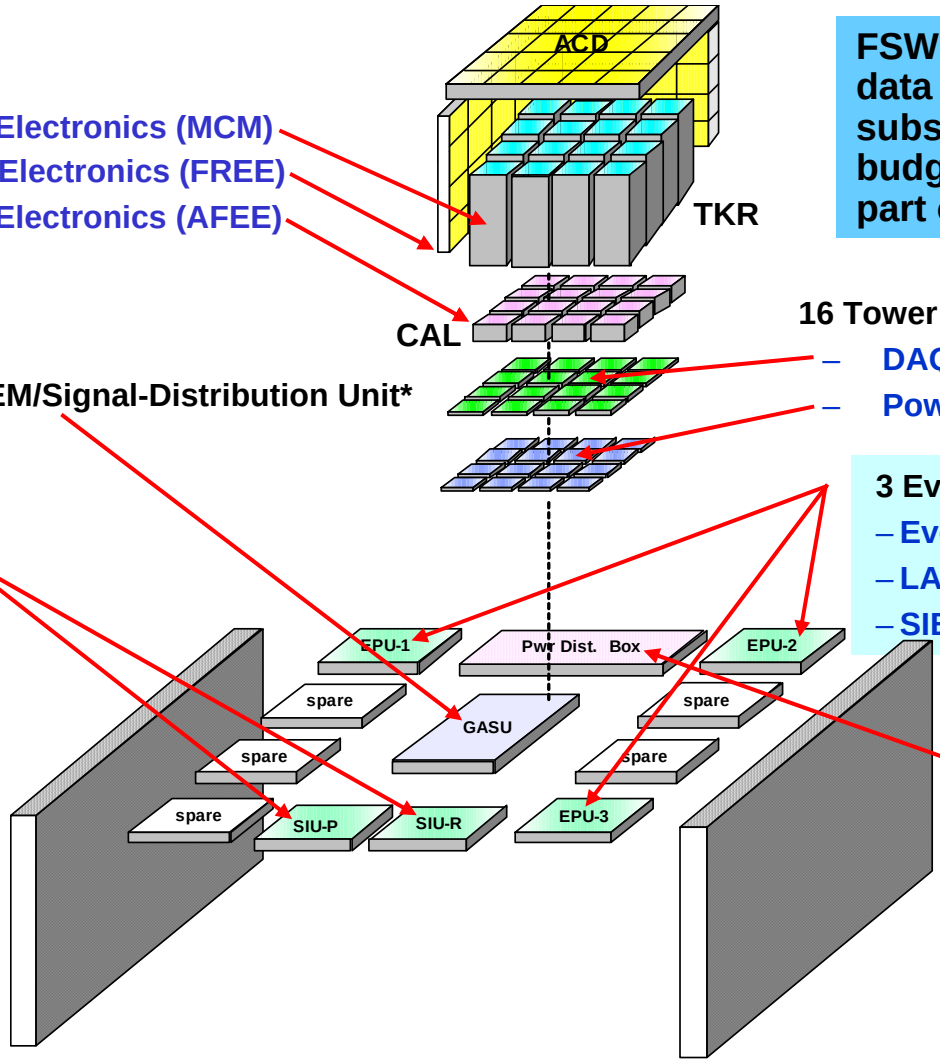
Mass:	**3000 kg**
Instrument Size:	**180 x 180 x 280 cm (X-Y-Z)**
Radiator Area:	**2.83 m^2 each side**
Science FOV:	**2π sr, +Z, above I/F plane**
Mounting:	**4-point mounting I/F at grid**

Key LAT Accommodation Requirements:

- **Pointing Knowledge (10 arcsec, 1σ-r)**
- **Science Data I/F and Peak Rate (40 Mbps LVDS)**
- **Operational Power (650 W OAP)**
- **Clear Radiator FOV on +/- Y faces from LAT I/F Plane (LIP) to L/V Interface Plane (PAF)**
- **Significant Mass Drives Observatory Axial CG**



LAT FSW: executes on SIU and EPU



TKR Front-End Electronics (MCM)
ACD Front-End Electronics (FREE)
CAL Front-End Electronics (AFEE)

Global-Trigger/ACD-EM/Signal-Distribution Unit*

FSW is an integral part of the data acquisition (DAQ) subsystem and is managed, budgeted, and scheduled as part of the DAQ subsystem

16 Tower Electronics Modules
- DAQ electronics module (DAQ-EM)
- Power-supplies for tower electronics

3 Event-Processor Units (2 + 1 spare)
- Event processing CPU
- LAT Communication Board
- SIB

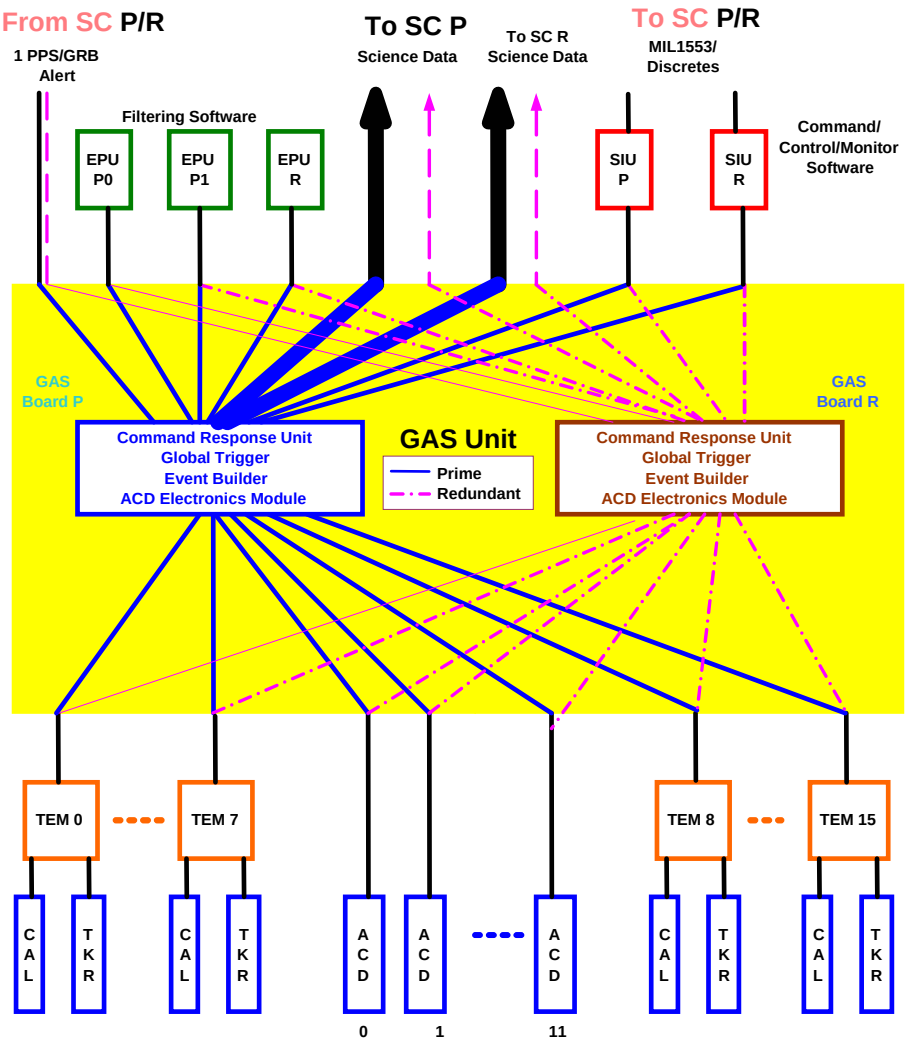
Spacecraft Interface Units
- Storage Interface Board (SIB): Spacecraft interface, control & telemetry
- LAT control CPU
- LAT Communication Board (LCB): LAT command and data interface

Power-Distribution Unit (PDU)*
- Spacecraft interface, power
- LAT power distribution
- LAT health monitoring

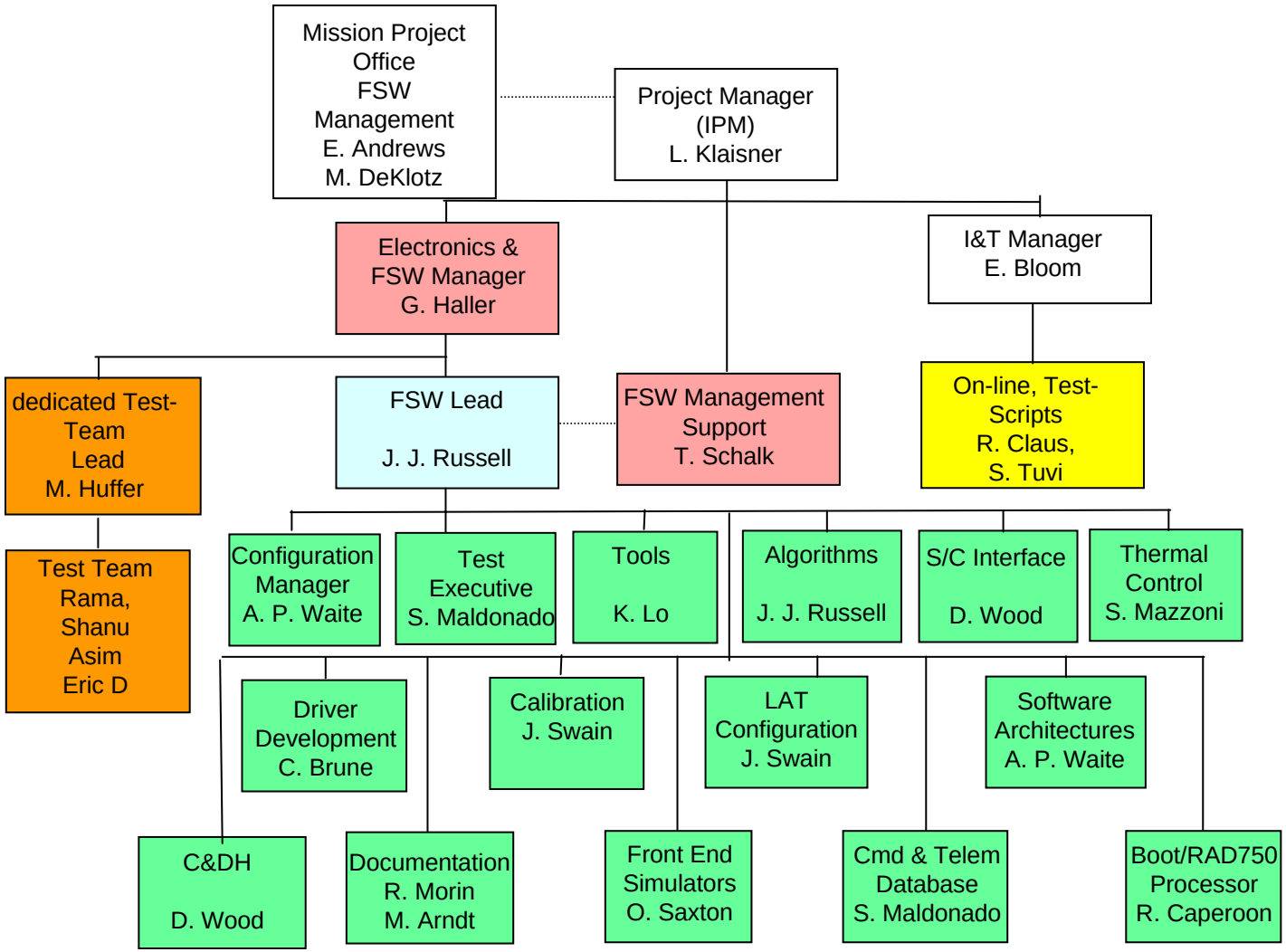
*** Primary & Secondary Units shown in one chassis**

LAT Electronics (Signals)

- TKR: Tracker
- CAL: Calorimeter
- ACD: Anti-Coincidence Detector
- TEM: Tower Electronics Module
- EPU: Event Processor Unit
- SIU: Spacecraft Interface Unit
- GAS Unit: Global Trigger-ACD-Signal Distribution Unit
  - CRU: Command/Response Unit
  - EBM: Event Builder Module
  - GEM: Global trigger Electronics Module
  - AEM: ACD Electronics Module



FSW Organization Chart



LAT FSW Development Overview

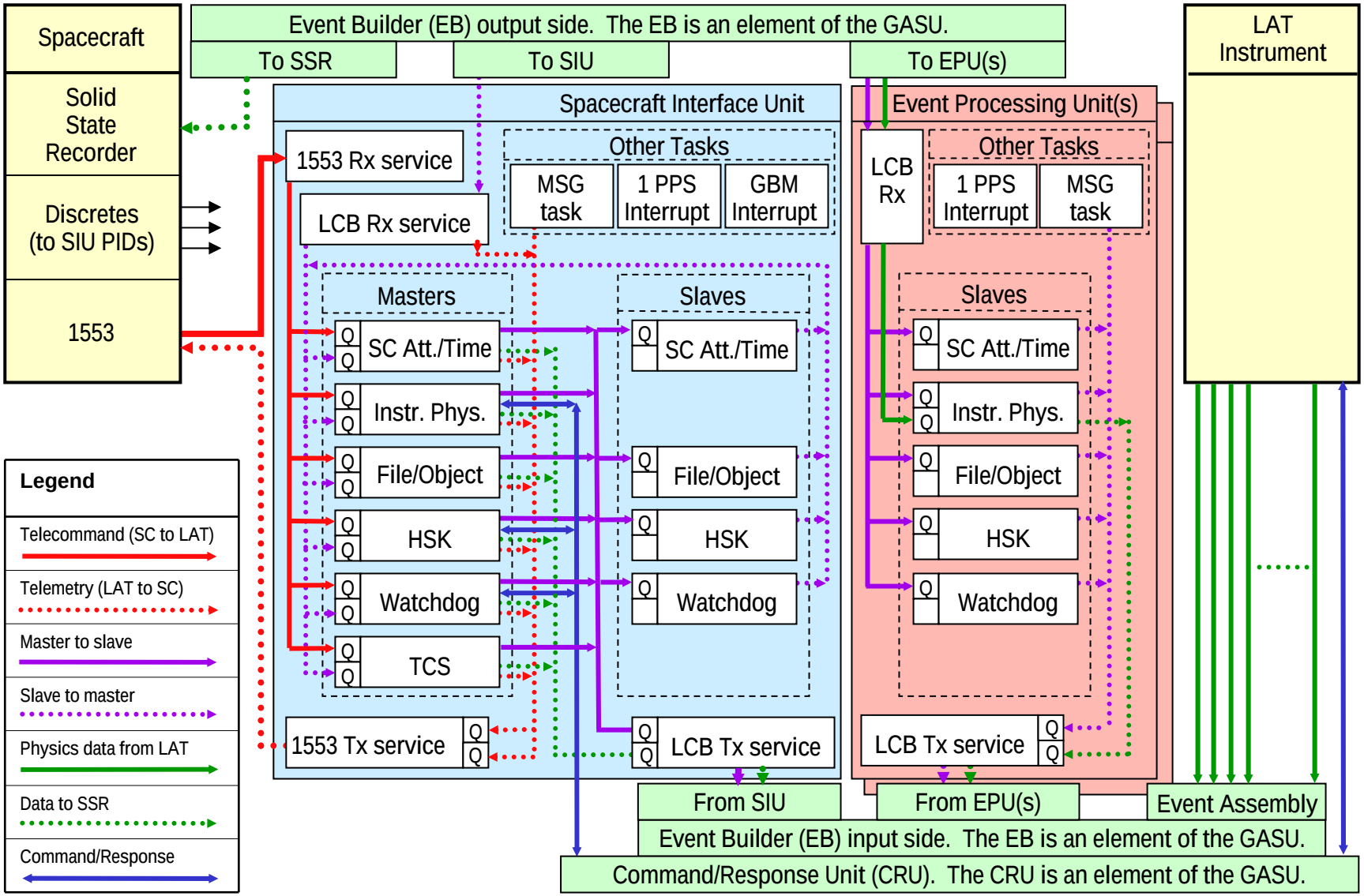
- **FSW is an integral part of the data acquisition (DAQ) subsystem and is managed, budgeted, and scheduled as part of the DAQ subsystem**
  - **> 3 million configuration bits to configure system and to take event data properly**
  - **Majority of software is to configure & setup hardware (boot, TKR, CAL, ACD, DAQ trigger, event-builder, sub-system setup, etc)**
  - **Rest is filtering, data monitoring, health & safety, telemetry**
  - **Once running, most is done in hardware**
- **FSW builds are coordinated with the DAQ hardware builds in both timing and content**

LAT FSW Strategy

- Incremental FSW builds to coincide with the natural hardware builds as follows:
- Major Builds
  - Engineering Model 1
    - Run single tower, single CPU
      - Used for EGSE test-stands for hardware development (built, tested, done)
      - Used for LAT single-tower engineering unit by I&T group (built, tested, done)
  - Engineering Model 2
    - Multiple towers, GASU*, single CPU
      - Used for EGSE test-stands for hardware development (in progress)
      - Used as test-bed for hardware and software development & test (in progress)
- Releases
  - ISIS (Instrument Spacecraft Interface Simulator) Build & Release
      - To be delivered to the Spacecraft vendor (Formal Release)
  - Full LAT Build & Release
    - Complete set of 16 towers, GASU*, full set of CPU's
      - Used as test-bed for hardware and software development & test
      - To be delivered to I&T for full-LAT testing (Formal Release)

* GASU includes LAT Global Trigger, LAT Event-Builder, Command-Response Unit, ACD Electronics Module

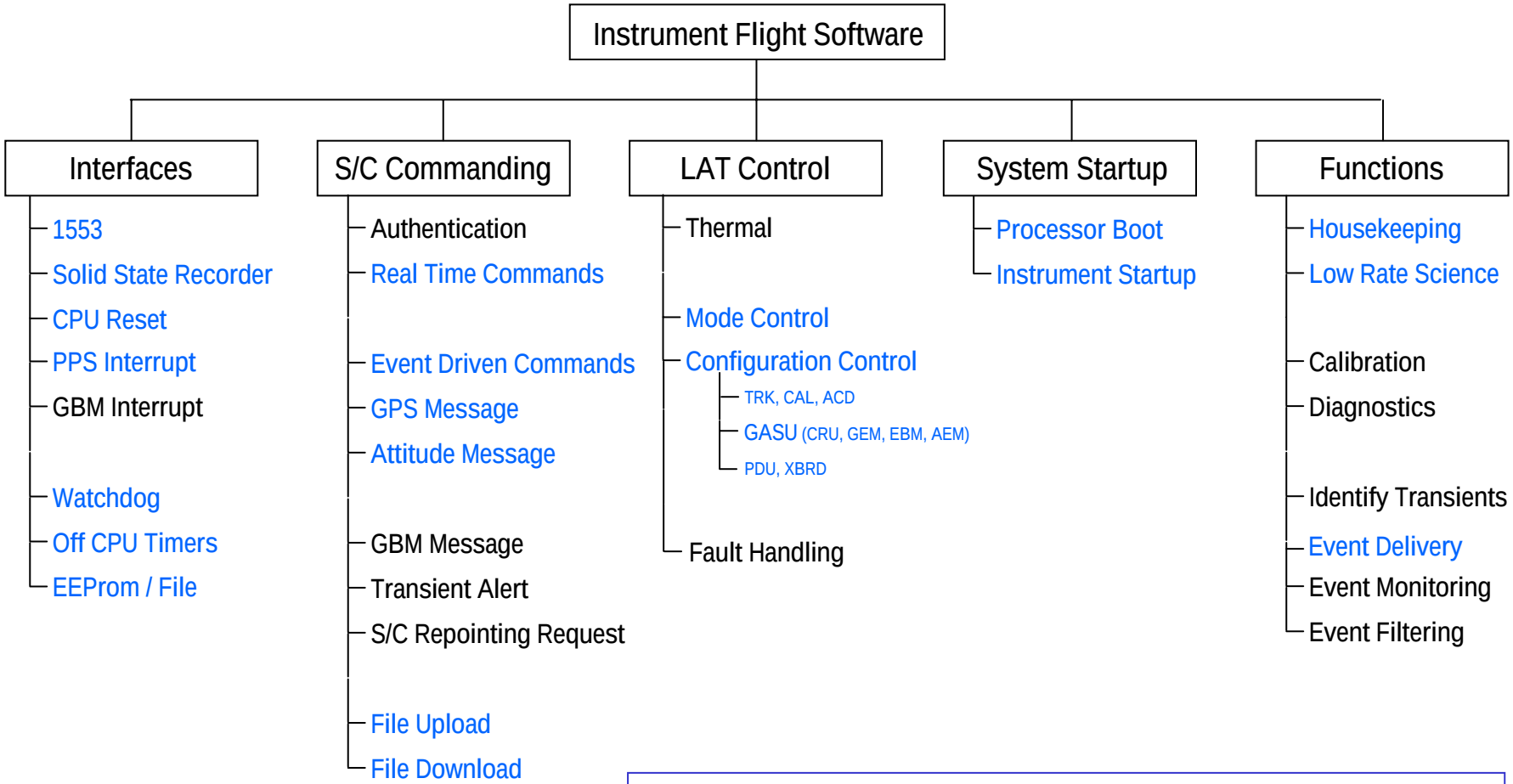
LAT FSW Architecture



EM2 Consumers

- Field test stands for ACD, CAL, TKR, & DAQ
  - Functionality to test subsystems
    - Configuration
    - Event read-out
  - ACD has special hardware needs, being dependent on
    - All elements of the GASU
      - CRU, AEM, GEM, EBM
    - Through CRU/AEM, access to
      - ACD front end electronics (GAFE, GARC)
  - Field maintenance is the responsibility of LAT I&T
    - FSW responsible for providing embedded system software only
- Spectrum Astro
  - Instrument Spacecraft Interface Simulator (ISIS)
- I&T
  - Building up the instrument
- Instrument Flight Software (ourselves)
  - Development of instrument flight software final deliverable
  - Development done on the test-bed
    - A full flight-like DAQ system (except sensors)
    - Sensors emulated by Front End Simulators (FES)

Engineering model 2 elements



Note:

— This is instrument flight software, not spacecraft flight software

Functions to Requirements Mapping

	Test Stands		Spectrum Astro	I&T	FSW Test-bed	FSW Deliverable Requirement
	TKR,CAL	ACD				
Configure TKR and CAL front end electronics	Y			Y	Y	5.3.4.6
Configure ACD front end electronics		Y			Y	5.3.4.6
Configure GASU (CRU, GEM, EBM, AEM)		Y	Y	Y	Y	5.3.4.6
Configure PDU			Y	Y	Y	5.3.4.6
Configure XBRD	Y	Y		Y		N/A
Configure by compressed file			Y	Y	Y	5.3.4.6
Real event delivery (instrument to CPU)	Y	Y		Y	Y	5.2.2.3
Housekeeping data stream	Y	Y	Y	Y	Y	5.3.4.8
RAD750 boot and crate initialization			Y	Y	Y	5.2.2.1, 5.3.4.1
1553 bus communications			Y	Y	Y	5.3.1.1
Telecommand/telemetry database and services			Y	Y	Y	5.3.4.2, 5.3.4.4
Emulated event delivery (to science data interface)			Y			5.3.1.4
CPU internal communications/task frameworks			Y	Y	Y	N/A
Software watchdog			Y	Y	Y	5.2.1.2, 5.3.2.1
Wall clock time services (GPS)			Y	Y	Y	5.3.4.3.4-5

Software Design for Safety

- The software safety environment
  - Software cannot damage hardware (hardware protects itself)
  - Reprogrammable on orbit (except for primary boot code)
- The software safety philosophy during development
  - Leverage the fact that software cannot damage hardware
  - Make unexplained conditions “fatal but not serious” and reboot
    - Decreases complexity
    - Increases reliability / robustness
    - Immediate and graceful exit quickly identifies code weaknesses
      - Improves efficiency for producing reliable / robust final code
  - On a case by case basis, develop recovery strategies
    - Not recoverable and CPU compromised: Stay with reboot strategy
    - Not recoverable but CPU integrity good: Report to ground and await intervention
    - Fully recoverable: Perform recovery action, continue operation

Software Fault Handling

- **Fault Recovery**
  - **On basis of fault identified**
    - **Reboot**
      - **Always attempt to save a block of information describing the fault condition in a known fixed memory location so that it can be picked up and sent to ground after the reboot**
    - **Notify ground and await intervention**
    - **Execute recovery procedure and continue operation**

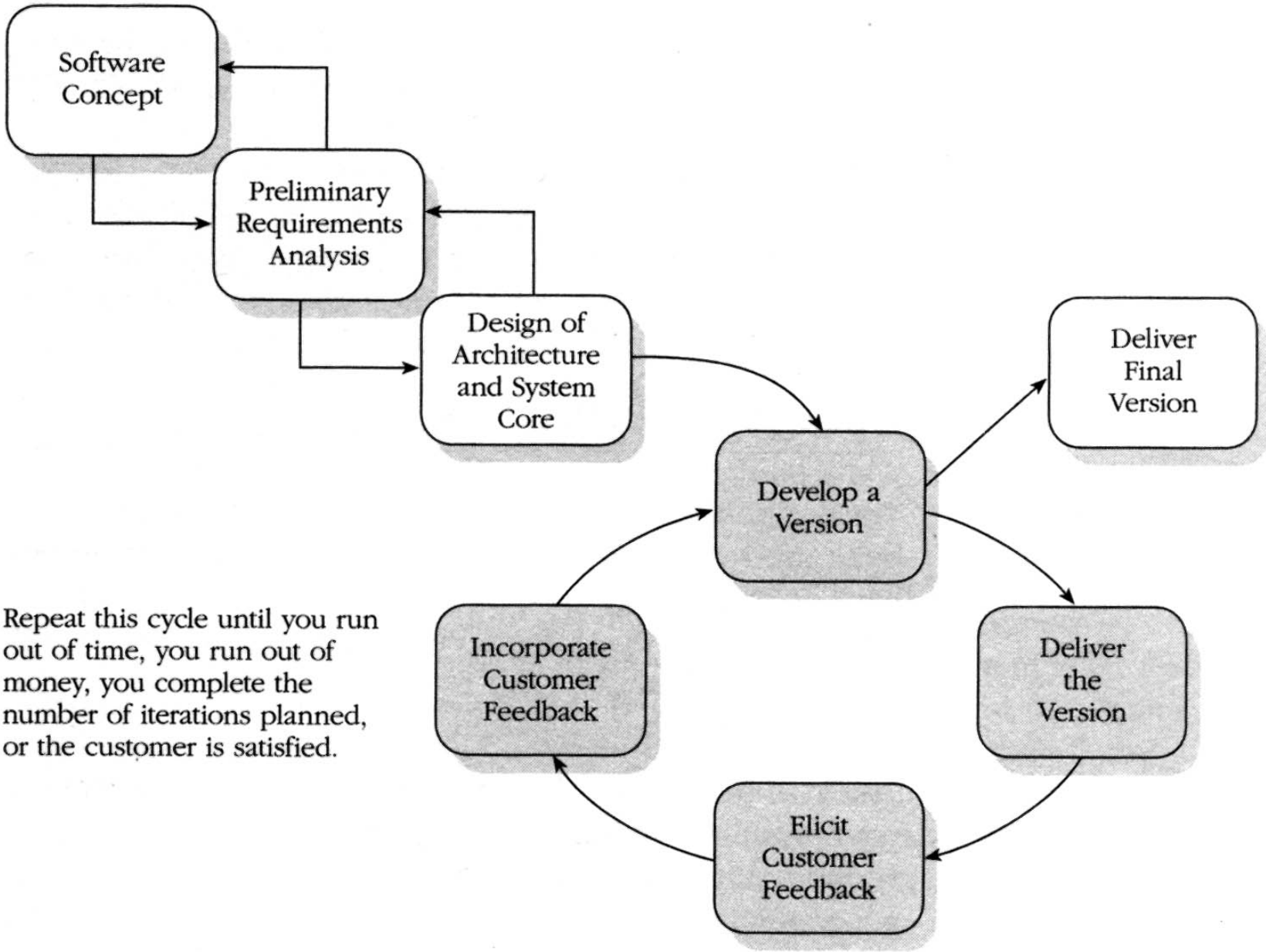
Development Environment

- **Embedded System**
  - Processor / operating system: BAE RAD750 / VxWorks
  - Toolset (Wind River Systems):
    - Language: C
    - Development platform: Sun / Solaris
    - Compiler / linker / binutils: GNU cross compiler suite
    - Debugger: Crosswind
- **Host System**
  - Processor / operating system: Sun / Solaris or Intel / Linux
  - Toolset (host simulation or cooperating processes):
    - Language: C
    - Development platform: Sun / Solaris or Intel / Linux
    - Compiler / linker / binutils: GNU compiler suite
    - Debugger: GDB / DDD
  - Toolset (test executive and scripting): Python / XML / MySQL / Qt / Perl
- **Other Tools**
  - Requirements management: DOORS
  - Code / configuration management: CMX / CMT / CVS
  - Autogeneration of documentation: Doxygen, in-house developed tools
  - Documentation: Microsoft office suite (also Adobe / Framemaker, etc.)

Software Development Approach

- Software lifecycle model
  - Iterative / incremental development model
  - Multiple builds with increased capability with each build
  - Regression testing on each build
- Requirements flowdown, analysis, review
  - Flowdown from program and system specs
  - Peer reviews
- Continuous cycle of development and test
- Code management
  - Formal control through the CMX / CMT / CVS toolchain
- Configuration management
  - Formal control through project management tools
    - LATdocs
- FSW CCB

Evolutionary Delivery model



Repeat this cycle until you run out of time, you run out of money, you complete the number of iterations planned, or the customer is satisfied.

Waterfall model with subprojects

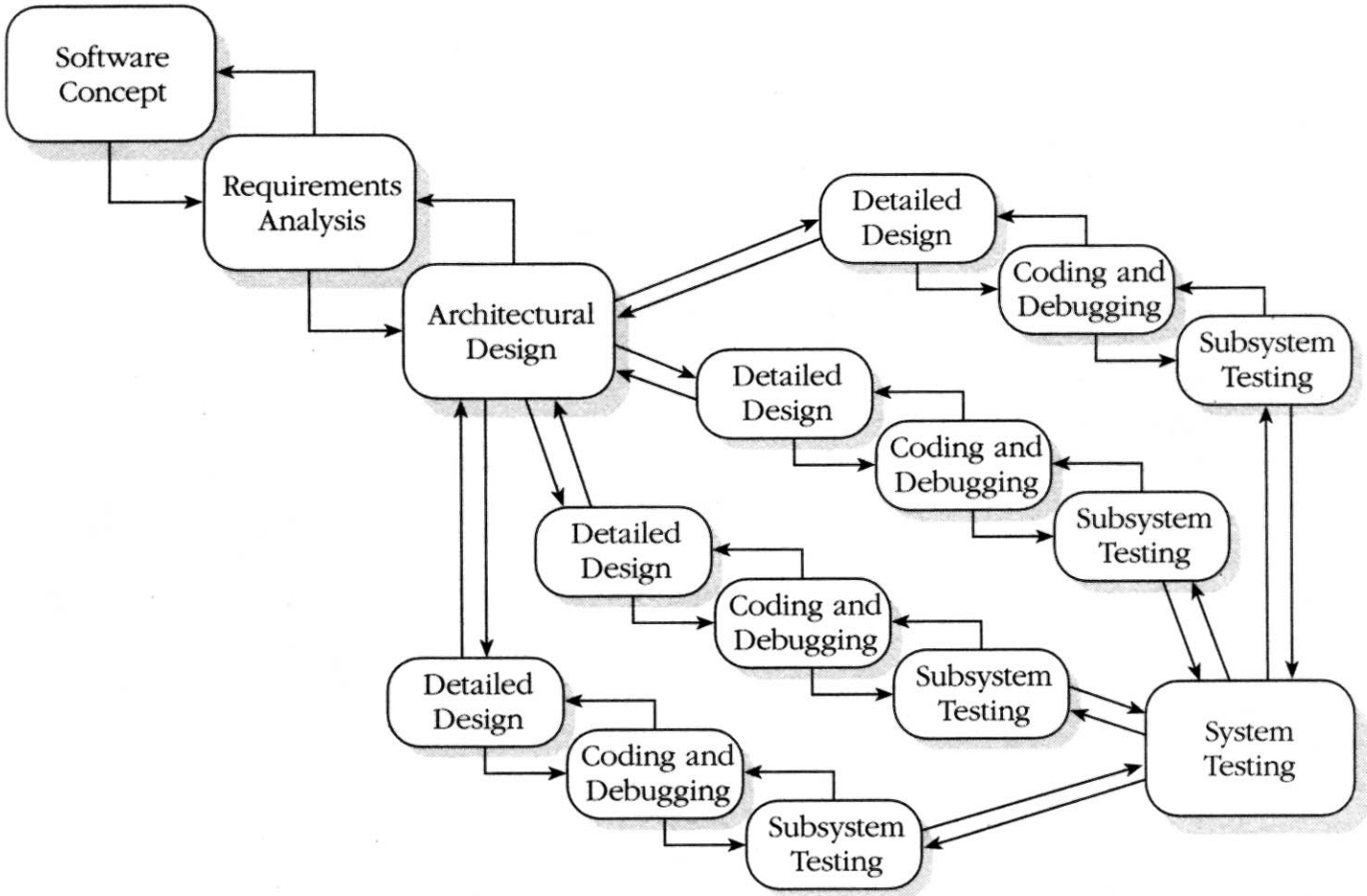


Figure 7-6. *The waterfall model with subprojects. Careful planning can allow you to perform some of the waterfall's tasks in parallel.*

Management & Configuration

- Version numbers follow strict naming conventions indicating:
  - Major: non-backwardly compatible interface changes
  - Minor: backwardly compatible interface changes
  - Patch: bug-fixes and performance enhancements
- A traditional build is a coherent collection of tagged packages
- Changes to builds delivered for formal testing (e.g. ISIS, FU), must be approved through configuration manager and CCB process
- Configuration management
  - Formal control through project management tools
    - LATDocs System
    - Non conformance reporting system

Software Testing Approach

- **Software code and package level test**
  - **Performed on developer's platform or captive embedded system**
  - **Verify algorithm development, debug software logic**
- **Software composite test**
  - **Higher-level functionality tests – combine many packages**
  - **Verify functionality and interfaces**
- **System build tests**
  - **Highest level tests – verify / validate against requirements**
- **Test environment**
  - **Software / hardware integration and test**
    - **Performed on FSW test bed with breadboard / brassboard hardware (COTS and then RAD750)**
    - **Verify software executing on target processors with real-time operating system (VxWorks).**
    - **Verify software interfaces with input/output hardware in loop**
  - **Software / system integration and test**
    - **Performed on flight spacecraft hardware in EGSE environment**
    - **Verify FSW with flight spacecraft hardware**

Code Organization

- FSW partitioned into functional blocks based on the Software Requirements Specification (SRS)
  - Functional blocks are then mapped into packages, the fundamental unit of the code management system
- Package code is (and has been) version/configuration controlled via Code Management System
  - Package structure includes
    - Source code
    - Documentation directory (manuals, developer guides, …)
    - Software Development Folder (a directory)
      - Development notes, version log, running log, to do lists, …
    - Package test directory
      - Pure test code
      - LTX test definitions/scripts
- Build: a collection of version-tagged packages
  - For an example for EM1 build see
    - https://oraweb.slac.stanford.edu:8080/pls/slacquery/bbrdownload/FSW-EM1-release-VDD.pdf?P_FRAME=GLAST&P_DOC_ID=10444

SW Code Management

- **Packages exist at 3 levels**
  - **Test**
    - **“Sand-box” in the developer’s private area**
  - **Development**
    - **Public area used to exercise code in a non-critical environment**
  - **Production**
    - **Public area used for official, tagged versions**
    - **A number of these can exist at a given time**
      - **One is declared the *current* production version**
      - **Back copies exist for comparison and other purposes**
    - **All production versions are tagged in CVS**

Code Metrics

- ***Summary:***
  - **Identify productivity metrics and definition of Source Line Of Code**
- ***Action Plan –***
  - **Action 3A: GPO, SLAC: Develop a mutually understood and agreed upon definition for metrics measurement. Specifically for SLOC and SLOC/day. Use this to put the IRT concerns to bed once and for all. Report on these metrics to Project . Recommend that after we get a good "baseline" estimate, that we ask for updates of 'actual' vs. 'estimated' at the package/unit level at build/release milestones. (See Test Process Action (#4/5) for more on Test Metrics.)**
  - **Determine whether there is adequate validity to providing estimated productivity metrics to build/release milestones. If we use them, evaluate productivity/performance needs to support planned future milestones from the past performance.**
  - **Action 3B: GPO, SLAC: Work a baseline to report out, using that baseline on our overall project estimates.**

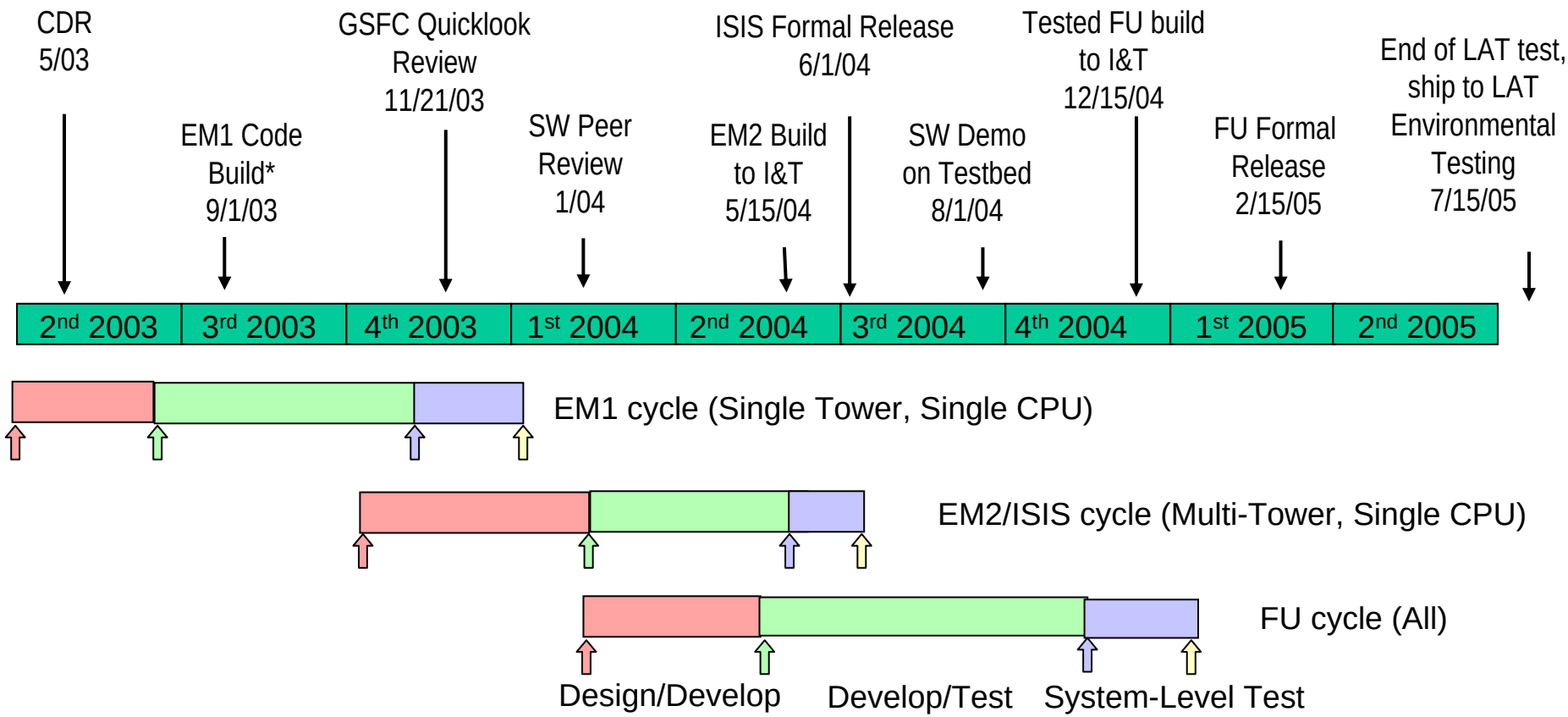
Productivity Definitions

- Implicitly a Source Line Of Code is tested and verified (I guess this sentence makes it explicit! :-)
- Duration of Project Life for which to measure amount of effort associated with a piece of software - From the Requirements Analysis Phase through FQT. Since FQT often reveals problems which need to be corrected, the effort associated with the correction and retest of problems found in the initial FQTs is included. Maintenance/Enhancements following FQT are not considered part of the project duration (and therefore the staffing for these efforts are not included in productivity estimates).
- Due to the duration of the effort on a project which is used to measure productivity, intermediate measurements of SLOC/day are inherently risky metrics to take stock in.
  There may be some value in assessing iterations (Build 1, Build 2, etc) for tracking purposes, but the parameters for the assessment must be made clear and the error bars on this type of intermediate assessment are typically large (eg. 2X).
- Note: In all cases (LAT, GBM, SC), we are utilizing the C programming language (except in the cases where assembly language is required for specific tasks. These are very few.)
- C Source Line Of Code - Semi-colon delimited statement of lexically valid programming language.
- Flight Software. Only lines of code that are compiled into executable images for the flight processor and are placed into memory on the spacecraft/instrument are considered Flight Software. ALL flight software must be tested to the same rigorous standard and held to meet requirements.
- Productivity. Productivity estimates will be greatly influenced by what tasks are included in establishing the amount of effort (measured in staff-months) to perform the FSW tasks. Here at GSFC, and for GLAST, the effort associated with the FSW task includes:
  - (1) FSW Management/Leadership.
  - (2) FSW Dev't & Test Engineering Support.
  - (3) FSW Design: Includes all analysis, design, development and Unit Test.
  - (4) FSW Testing: Beyond the Unit Testing, Through Build and Release Testing.
  - (5) H/W Support.

Productivity based on definitions

Person	Months		Person	Months	
JJ	43		Curt B	40	
Tony	43		Sergio	30	
GXH	22		Kim Lo	18	
Erik A	21		James S	22	
Eric Hansen	15		Owen S	22	
Amadeo P.	18		Dan W	43	
Huffer/TTL	22		Ray C	20	
Rama	18		Brian D	6	
Shanu	16		Steve M.	16	
Eric D/Asim	15		Mike D	15	
Mark A.	15		R Morin	15	
Ric & Selim	18		TBD SE	15	
Total Man Months On Project:				528	
Total Man DAYS on Project				10560	
Estimated SLOC on LAT:				100000	
Estimated Productivity (SLOC/day):				**9.47**	

Breakdown of Development Cycles



Design/Develop: Start design, code small prototypes, no hardware available, only descriptions
Develop/Test: Code and test against real hardware, take snap-shot at end (i.e. define build)
System-Level test: Build Test

*: EM1 was built/tested/delivered, group is now integrating/testing another recently available piece of hardware for EM1, the LCB.

Software Key Milestones for I&T/ISIS

- August 03
  - Tested Software to control/read-out single tower with Calorimeter and Tracker sub-system (done, EM1), delivered to I&T
- Jan 04
  - SW peer review
- June 04
  - I&T requires software to control/readout multi-tower (i.e. GASU) configuration (EM2)
- June 04 (TBD)
  - Instrument Spacecraft Interface Simulator (ISIS) to Spectrum Astro
- August 04
  - Demo to LAT system engineering on fully instrumented test-bed
- December 04
  - I&T requires tested software to control/readout full LAT
  - FU SW build to I&T
- February 05
  - I&T requires FU software release to operate/test (whole LAT is integrated)
  - Start of system testing
- May 05
  - End of system test

- FUNCTIONAL DEMONSTRATIONS every month
  - with Lehman review / ISIS / mid FU having larger audience

EM 2

- **EM2 is an iterative step on the way to flight code that supports general LAT (& spacecraft) development which develops needed core functionality.**

- **Major Builds**
  - **Engineering Model 1**
    - **Run single tower, single CPU**
  - **Engineering Model 2**
    - **Multiple towers, GASU*, single CPU**
- **Releases**
  - **ISIS Build & Release**
  - **Full LAT Build & Release**
    - **Complete set of 16 towers, GASU*, full set of CPU's**

Extra Slides…

LAT FSW Team Heritage

- Very experienced people
  - HEP
  - NRL
- Successful track record
  - Architected, designed, implemented, tested, and commissioned major large experiments
- Leads are developers
- Leads are scientists

Experiment	# CPUs	Man-years	Time
SLD	> 500	12	4 years
Babar	> 200	15	3 years
LAT	3	20	In progress

SLD: SLC Large Detector

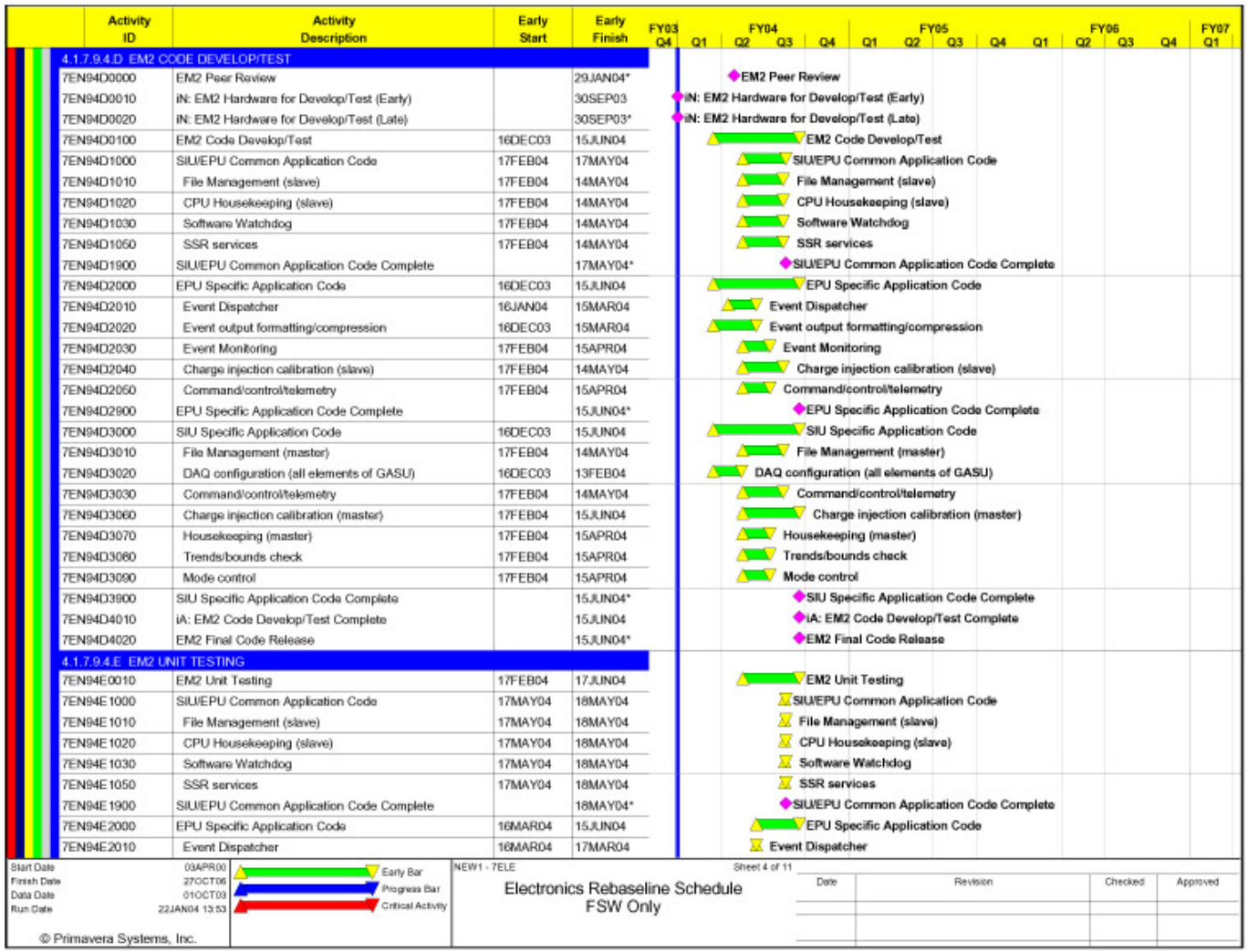
BaBar: B-B Detector (Matter-Antimatter Exp.)

BFEM: GLAST LAT Balloon-Flight (proof-of principle for GLAST LAT detectors and DAQ)

“PMCS” schedule for EM2

Activity ID	Activity Description	Early Start	Early Finish
7EN93B1010	Narrowband Telemetry	02DEC03	15JAN04
7EN93B1040	DAQ configuration	02DEC03	15JAN04
7EN93B1050	TKR configuration	02DEC03	15JAN04
7EN93B1060	CAL configuration	02DEC03	15JAN04
7EN93B1070	Event Filtering (EM1 level)	02DEC03	15JAN04
7EN93B3100	iA: EM1 Build Testing Complete		15JAN04
7EN93B3110	EM1 Build Testing Complete		15JAN04*
4.1.7.9.3.C EM1 DOCUMENTATION			
7EN93C0000	EM1 Documentation Start	01OCT03*	
7EN93C1000	EM1 Documentation	01OCT03	15JAN04
7EN93C1010	Version Description Document	01OCT03	15DEC03
7EN93C1020	EM1 Test Report	01OCT03	15JAN04
7EN93C1030	Design Documents	01OCT03	15JAN04
7EN93C3100	iA: EM1 Documentation Complete		15JAN04
7EN93C3110	EM1 Documentation Complete		15JAN04*
4.1.7.9.4 ENGINEERING MODEL 2			
4.1.7.9.4.C EM2 CODE DESIGN/DEVELOP			
7EN94C0000	iN: EM2 Hardware Specification		30SEP03
7EN94C1000	SIU/EPU Common Application Code	01OCT03	15MAR04
7EN94C1010	File Management (slave)	01OCT03	13FEB04
7EN94C1020	CPU Housekeeping (slave)	01OCT03	13FEB04
7EN94C1030	Software Watchdog	01OCT03	13FEB04
7EN94C1050	SSR services	01OCT03	13FEB04
7EN94C2000	EPU Specific Application Code	01OCT03	15MAR04
7EN94C2010	Event Dispatcher	01OCT03	15JAN04
7EN94C2020	Event output formatting/compression	01OCT03	15DEC03
7EN94C2030	Event Monitoring	01OCT03	13FEB04
7EN94C2040	Charge injection calibration (slave)	01OCT03	13FEB04
7EN94C2050	Command/control/telemetry	01OCT03	13FEB04
7EN94C3000	SIU Specific Application Code	01OCT03	15MAR04
7EN94C3010	File Management (master)	01OCT03	13FEB04
7EN94C3020	DAQ configuration (all elements of GASU)	04NOV03	15DEC03
7EN94C3030	Command/control/telemetry	07NOV03	13FEB04
7EN94C3060	Charge injection calibration (master)	01OCT03	13FEB04
7EN94C3070	Housekeeping (master)	02OCT03	13FEB04
7EN94C3080	Trends/bounds check	01OCT03	13FEB04
7EN94C3090	Mode control	01OCT03	13FEB04
7EN94C4000	iA: EM2 Code Design/Develop Complete		13FEB04
7EN94C4010	iN: EM2 Code Design/Develop Complete		15MAR04*

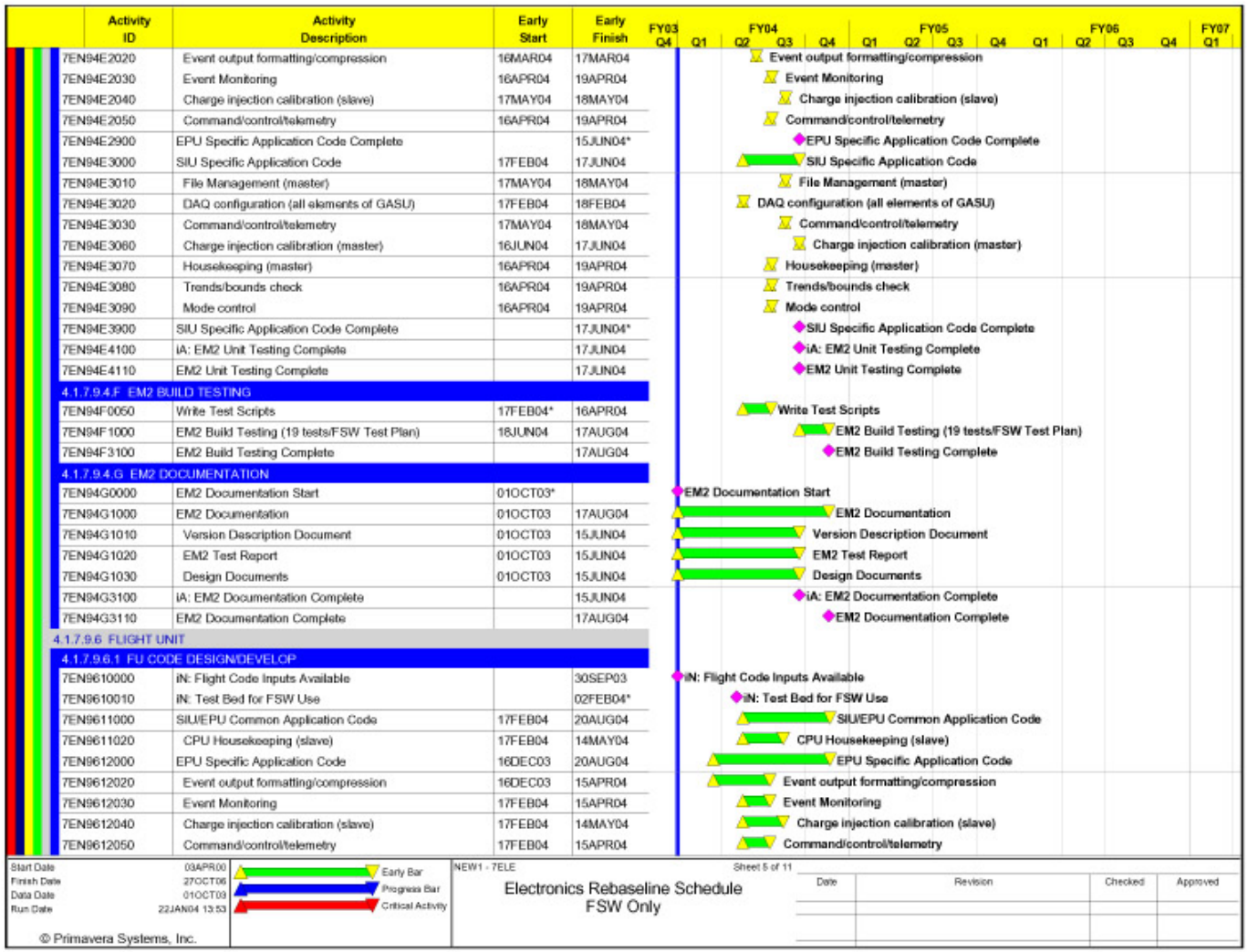
“PMCS” schedule for EM2



Activity ID	Activity Description	Early Start	Early Finish
4.1.7.9.4.D	EM2 CODE DEVELOP/TEST		
7EN94D0000	EM2 Peer Review		29JAN04*
7EN94D0010	iN: EM2 Hardware for Develop/Test (Early)		30SEP03
7EN94D0020	iN: EM2 Hardware for Develop/Test (Late)		30SEP03*
7EN94D0100	EM2 Code Develop/Test	16DEC03	15JUN04
7EN94D1000	SIU/EPU Common Application Code	17FEB04	17MAY04
7EN94D1010	File Management (slave)	17FEB04	14MAY04
7EN94D1020	CPU Housekeeping (slave)	17FEB04	14MAY04
7EN94D1030	Software Watchdog	17FEB04	14MAY04
7EN94D1050	SSR services	17FEB04	14MAY04
7EN94D1900	SIU/EPU Common Application Code Complete		17MAY04*
7EN94D2000	EPU Specific Application Code	16DEC03	15JUN04
7EN94D2010	Event Dispatcher	16JAN04	15MAR04
7EN94D2020	Event output formatting/compression	16DEC03	15MAR04
7EN94D2030	Event Monitoring	17FEB04	15APR04
7EN94D2040	Charge injection calibration (slave)	17FEB04	14MAY04
7EN94D2050	Command/control/telemetry	17FEB04	15APR04
7EN94D2900	EPU Specific Application Code Complete		15JUN04*
7EN94D3000	SIU Specific Application Code	16DEC03	15JUN04
7EN94D3010	File Management (master)	17FEB04	14MAY04
7EN94D3020	DAQ configuration (all elements of GASU)	16DEC03	13FEB04
7EN94D3030	Command/control/telemetry	17FEB04	14MAY04
7EN94D3050	Charge injection calibration (master)	17FEB04	15JUN04
7EN94D3070	Housekeeping (master)	17FEB04	15APR04
7EN94D3080	Trends/bounds check	17FEB04	15APR04
7EN94D3090	Mode control	17FEB04	15APR04
7EN94D3900	SIU Specific Application Code Complete		15JUN04*
7EN94D4010	iA: EM2 Code Develop/Test Complete		15JUN04
7EN94D4020	EM2 Final Code Release		15JUN04*
4.1.7.9.4.E	EM2 UNIT TESTING		
7EN94E0010	EM2 Unit Testing	17FEB04	17JUN04
7EN94E1000	SIU/EPU Common Application Code	17MAY04	18MAY04
7EN94E1010	File Management (slave)	17MAY04	18MAY04
7EN94E1020	CPU Housekeeping (slave)	17MAY04	18MAY04
7EN94E1030	Software Watchdog	17MAY04	18MAY04
7EN94E1050	SSR services	17MAY04	18MAY04
7EN94E1900	SIU/EPU Common Application Code Complete		18MAY04*
7EN94E2000	EPU Specific Application Code	16MAR04	15JUN04
7EN94E2010	Event Dispatcher	16MAR04	17MAR04

“PMCS” schedule for EM2

Activity ID	Activity Description	Early Start	Early Finish
7EN94E2020	Event output formatting/compression	16MAR04	17MAR04
7EN94E2030	Event Monitoring	16APR04	19APR04
7EN94E2040	Charge injection calibration (slave)	17MAY04	18MAY04
7EN94E2050	Command/control/telemetry	16APR04	19APR04
7EN94E2900	EPU Specific Application Code Complete		15JUN04*
7EN94E3000	SIU Specific Application Code	17FEB04	17JUN04
7EN94E3010	File Management (master)	17MAY04	18MAY04
7EN94E3020	DAQ configuration (all elements of GASU)	17FEB04	18FEB04
7EN94E3030	Command/control/telemetry	17MAY04	18MAY04
7EN94E3060	Charge injection calibration (master)	16JUN04	17JUN04
7EN94E3070	Housekeeping (master)	16APR04	19APR04
7EN94E3080	Trends/bounds check	16APR04	19APR04
7EN94E3090	Mode control	16APR04	19APR04
7EN94E3900	SIU Specific Application Code Complete		17JUN04*
7EN94E4100	iA: EM2 Unit Testing Complete		17JUN04
7EN94E4110	EM2 Unit Testing Complete		17JUN04
4.1.7.9.4.F	EM2 BUILD TESTING		
7EN94F0050	Write Test Scripts	17FEB04*	16APR04
7EN94F1000	EM2 Build Testing (19 tests/FSW Test Plan)	18JUN04	17AUG04
7EN94F3100	EM2 Build Testing Complete		17AUG04
4.1.7.9.4.G	EM2 DOCUMENTATION		
7EN94G0000	EM2 Documentation Start	01OCT03*	
7EN94G1000	EM2 Documentation	01OCT03	17AUG04
7EN94G1010	Version Description Document	01OCT03	15JUN04
7EN94G1020	EM2 Test Report	01OCT03	15JUN04
7EN94G1030	Design Documents	01OCT03	15JUN04
7EN94G3100	iA: EM2 Documentation Complete		15JUN04
7EN94G3110	EM2 Documentation Complete		17AUG04
4.1.7.9.6	FLIGHT UNIT		
4.1.7.9.6.1	FU CODE DESIGN/DEVELOP		
7EN9610000	iN: Flight Code Inputs Available		30SEP03
7EN9610010	iN: Test Bed for FSW Use		02FEB04*
7EN9611000	SIU/EPU Common Application Code	17FEB04	20AUG04
7EN9611020	CPU Housekeeping (slave)	17FEB04	14MAY04
7EN9612000	EPU Specific Application Code	16DEC03	20AUG04
7EN9612020	Event output formatting/compression	16DEC03	15APR04
7EN9612030	Event Monitoring	17FEB04	15APR04
7EN9612040	Charge injection calibration (slave)	17FEB04	14MAY04
7EN9612050	Command/control/telemetry	17FEB04	15APR04



FSW LATDocs Document Index

SS - System Specification (Design To Specification)

Doc. No.	V#	DCS	Ext.	S	Primary Author / Document Title
SS-00053		4IW			Russell, J J. Balloon Flight Data Format
SS-00169	??	4SO	PDF		Johnson, Robert P. Tracker Front-end Readout ASIC Specifications
SS-00170	6	4SO	PDF		Johnson, W. Neil Conceptual Design of the GLAST Tracker Readout Controller Electronics ASIC (GTRC)
SS-00208	4	4IW	PDF		Johnson, W. Neil Calorimeter Readout Control, ASIC - Conceptual Design
SS-00286	0	4IW	PDF		Russell, J J. GLAST LAT - Conceptual Design of the Global Trigger
SS-00363	5	4SO	PDF		Bielawski, Rich LAT Dataflow Subsystem Specification - ACD-AEM Interface
SS-00399	2	4SO	PDF		Waite, Anthony P. LAT Flight Software Level IV Specification

FSW LATDocs Document Index, cont.

SS-00424		4IW		Johnson, W. Neil Calorimeter Front End ASIC - GCFE V9 Design Description
SS-00443	??	AIW	PDF	Wallace, James L Conceptual Design of the Instrument Simulator
SS-00461	1	4IW	PDF	Huffer, Michael E. LAT TEM-GASU to CPU Data Formats
SS-00912	0	4IW	PDF	Brune, Curtis LAT Dataflow Simulation
SS-01596	0	4IW	PDF	Huffer, Michael E. The GLAST Global Trigger - Design Specification
SS-01597	2	3IW	PDF	Russell, J.J. Configuration Data: Storage and Transmission
SS-01792	0	4IW	PDF	Russell, J.J. SIB - The GLAST/LAT Spacecraft Interface Board, Hardware Specification

FSW LATDocs Document Index, cont.

TD - Technical Document

Doc. No.	V#	DCS	Ext.	S	Primary Author / Document Title
TD-00331		4IW			Russell, J.J. LAT Flight Software Preliminary Design Report
TD-00560	1	3SO	DOC		Huffer, Michael E. LAT Global Trigger and ACD Hit Maps
TD-00593	4	4IW	PDF		Brune, Curtis LAT Comm I/O Board -- Response FIFO
TD-00597	4	4IW	PDF		Brune, Curtis LAT Comm I/O Board -- Triggering
TD-00605	2	4SO	PDF		Huffer, Michael E. The Tower Electronics Module (TEM) - A Primer
TD-00606	1	4IW	PDF		Huffer, Michael E. LAT Inter-module Communications
TD-00639	1	3SO	PDF		Haller, Gunther The ACD Electronics Module (AEM)
TD-00712	1	3SO	DOC		Russell, J.J. LAT Auxiliary Data Survey
TD-00786	2	4SO	PDF		LAT Flight Software Test Plan

FSW LATDocs Document Index, cont.

TD-00861	1	4IW	PDF	Huffer, Michael E. Test-stand Architecture Redux
TD-00862	1	4IW	PDF	Huffer, Michael E. Test-stand Update
TD-00863	1	4IW	PDF	Huffer, Michael E. LAT Custom Processor Specification
TD-01121	1	3SO	DOC	Russell, J.J. Trigger and Dataflow Resource Usage
TD-01199	10	4IW	PDF	Brune, Curtis LAT Test Stand Communications Interface
TD-01380	3	4IW	PDF	Brune, Curtis LAT Communication Board Driver
TD-01536	1	3SO	DOC	Russell, J.J. SC / LAT ICD for Startup Procedures
TD-01543	1	3SO	PDF	Huffer, Michael E. The Power Distribution Unit Programming ICD Specification
TD-01545	1	3SO	PDF	Huffer, Michael E. The GLT Electronics Module Programming ICD Specification
TD-01546	1	3SO	PDF	Huffer, Michael E. The Event Builder Module Programming ICD Specification
TD-01547	1	3SO	PDF	Huffer, Michael E. The Command/Response Unit Programming ICD Specification

FSW LATDocs Document Index, cont.

TD-01553		3IW		Russell, J.J. Instrument Damage Protection against LAT Processor Hardware or Software Malfunction
TD-01781	1	3SO	DOC	Russell, J.J. LAT Flight Software Package Descriptions and LOC Basis of Estimate
TD-01806	5	3IW	PDF	Russell, J.J. Primary Boot Code
TD-02067	1	4IW	PDF	Waite, Anthony P. Glossary of Flight Software Terms
TD-02150	2	4IW	PDF	Waite, Anthony P. LAT Flight Software Secondary Boot Code
TD-02620	1	AIW	PDF	Swain, James E. LAT Instrument Startup
TD-02627		AIW		Swain, James E. Trigger Test Plan
TD-02634	??	4IW	PDF	Waite, Anthony P. Em1 Code Release Version Description
TD-02659	1	AIW	PDF	Wood, Daniel Lee LAT Flight Software Telecommand and Telemetry Formats

FSW LATDocs Document Index, cont.

XR - Change Record

Doc. No.	V#	DCS	Ext.	S	Primary Author / Document Title
XR-02038	1	3IW	PDF		Russell, J.J. Dcn for LAT Daq Trigger and Dataflow Resource Usage
XR-02619		AIW			Swain, James E. Dcn for LAT Instrument Startup

Legend

Heading	Description
Doc. No.	LATDocs document number
V#	current version number
DCS	Document Control & Status (abridged)
Ext	File Extension
S	Status (? indicates new document)
Author / Title	First author listed by LATDocs, Title

This page last modified on January 22, 2004 at 14:16 PST.
(Included material may have been updated at a different time.)
Contact fsw-web for help requests, comments, bug reports, etc.

Software Fault Detection Methods

- State of health (SOH)
  - Bridge chip built-in test
- Software watchdog
  - All registered tasks must regularly report progress in order for SWD to reset hardware watchdog
- Data validity
  - Checksums
  - Parity bits
  - LCB timeouts, error records
- Data reasonableness checks
  - HSK queries software/hardware regularly
    - Memory correction reports from bridge chip
    - Thermal conditions
    - Power conditions
    - …
  - All instrument configurations read out at beginning and end of all data collection runs (must agree)
  - Event monitoring code on EPUs checks event data for correct format, completeness
  - Event filtering code checks event data for physics consistency

GLAST MISSION ELEMENTS