The European Ground Systems – Common Core (EGS-CC) Initiative

Nestor Peccia, ESA/ESOC

GSAW 2012

Background

- Within Europe, different systems used by different companies/agencies
  - Some of them common to operations and AIT, some specific
- Many of the existing systems have reached or are reaching their end of life:
  - Become excessively complex with time
  - Use old software technologies and hardware platforms
  - Difficult to modernise
- Compatibility/exchange of information:
  - Often multiple systems are used in AIT of a space system by different companies or at different levels (e.g. payload/system) or in different phases
  - Little synergy across missions and mission phases

EGS-CC Initiative Objectives

- The EGS-CC Initiative aims at developing a common infrastructure enabling
  - Seamless transition from spacecraft Assembly, Integration and Testing (AIT) to mission operations
  - Enable overall cost reductions by sharing development, sustaining and maintenance of a single infrastructure
  - Facilitate cost and risk reduction when implementing space projects
  - Enable the modernization of legacy EGSE and MCS systems
  - Enable the exchange of ancillary implementations across organizations

EGS-CC Stakeholders

- EGS-CC is a collaboration of European prime industry and space agencies to develop a common core
  - Astrium Satellites
  - Astrium Space Transportation
  - Thales Alenia (France and Italy)
  - OHB Systems
  - ESA (ESTEC and ESOC)
  - CNES
  - DLR
- Initiative considered strategic by all parties
- Memorandum of Understanding between the EGS-CC partners
  - Development of the EGS-CC in open competition according to ESA contract conditions and processes
  - Adoption of the EGS-CC for institutional missions and then for commercial missions as well

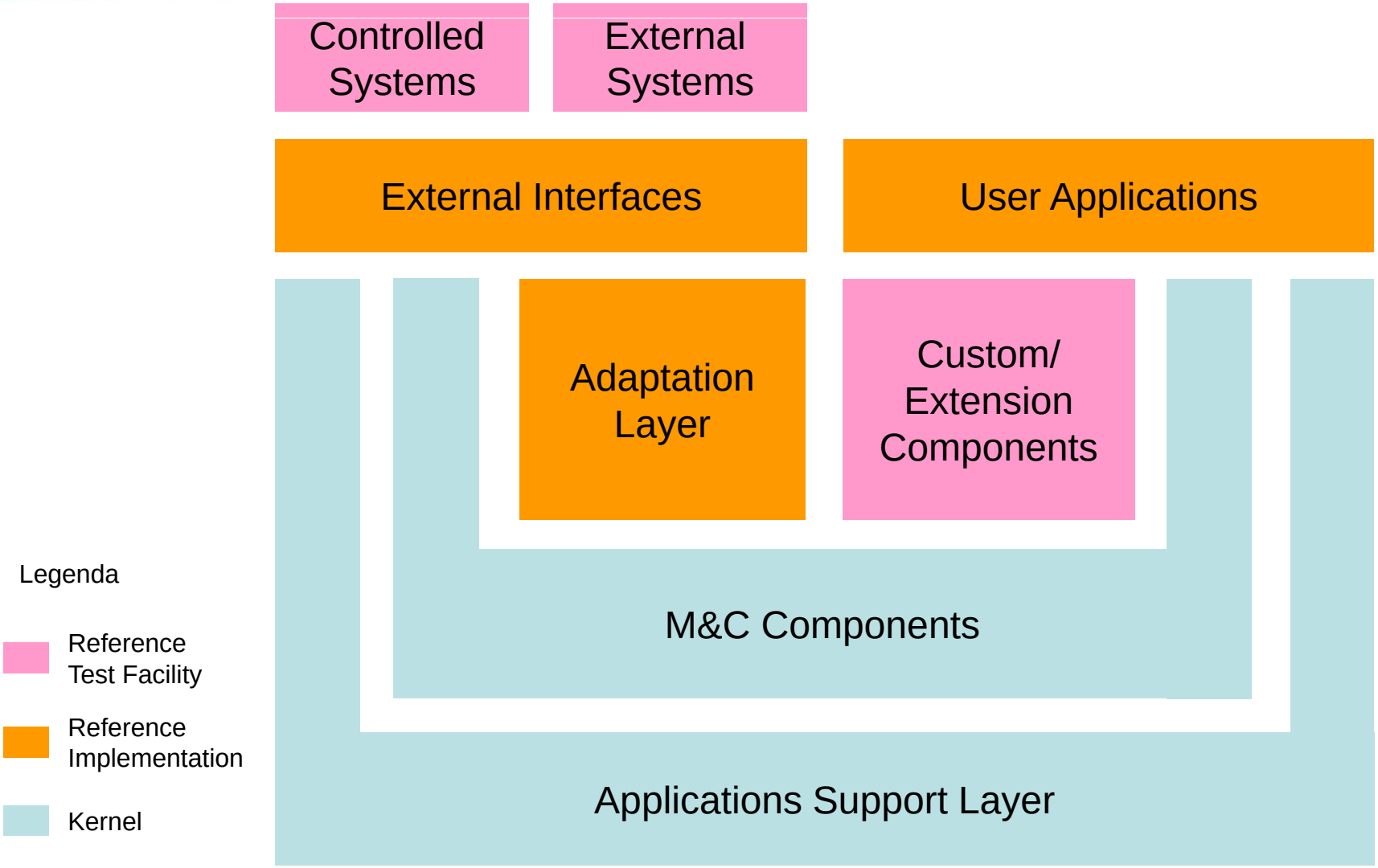
Main EGS-CC System Features

- Scope of the EGS-CC system features is very ambitious
  - Support of all mission types and phases
  - Open, component based, service oriented architecture
  - Generic and extensible functionality
  - Binary compatibility
  - Layered implementation
  - Clear separation between generic M&C functions (kernel) and specific features of the controlled system (adaptation layer)
  - Configurable level of operations abstraction
  - Standardised interfaces (as far as possible…)
  - Technology isolation (as far as possible…)
  - Long term maintainability
  - High Performance and Scalability

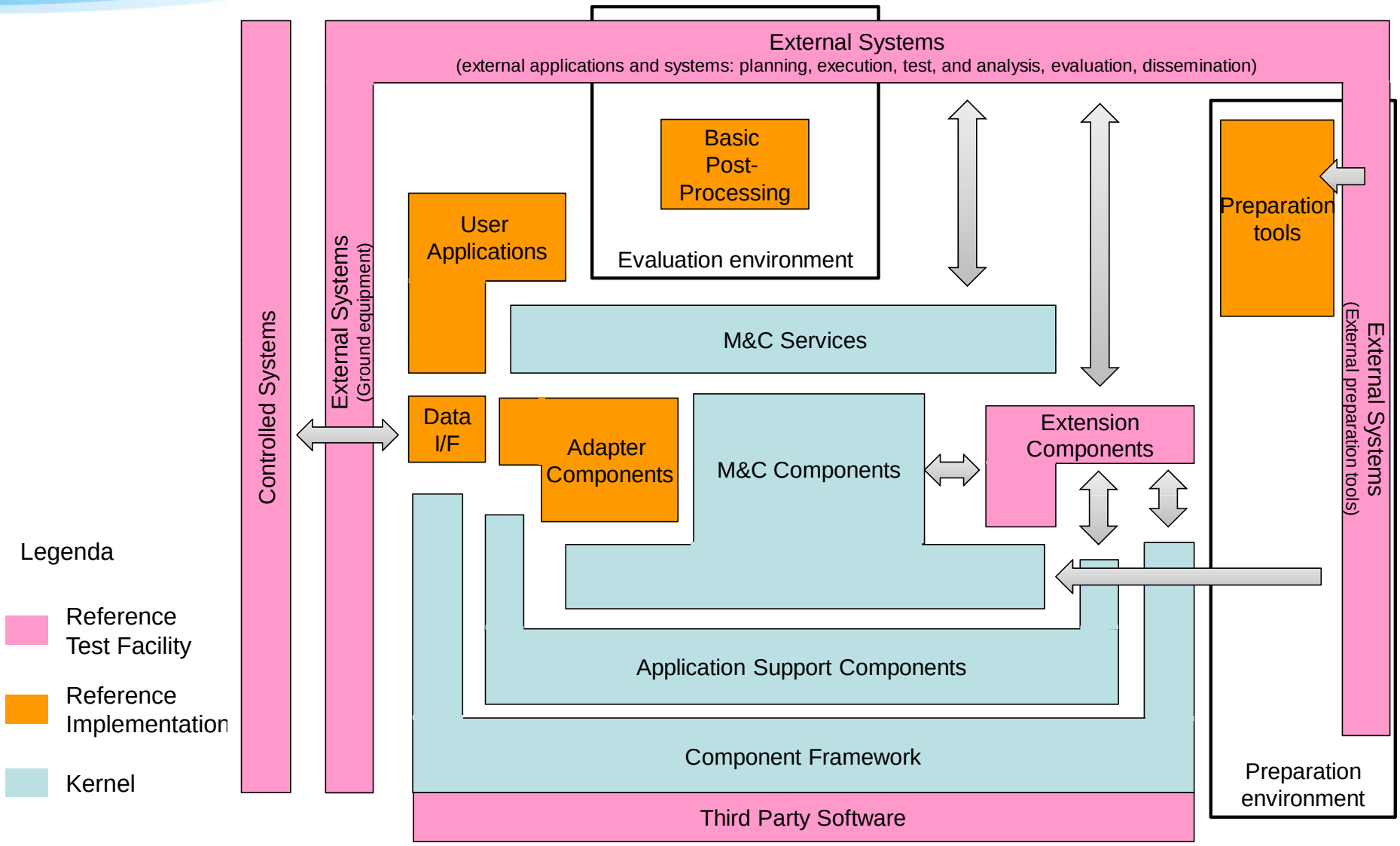
Main System Decomposition (1/2)

- EGS-CC will consist of:
  - Reference Architecture: system decomposition, internal and external interfaces
  - Kernel components: system backbone, mandatory use, non-replaceable, only extensible
  - Reference implementations: system periphery components, replaceable by user's own implementation
  - Reference test facilities: external components used to validate the full system

Main System Decomposition (2/2)



Top Level Architecture



Kernel Functional Scope

- Application support layer
  - System management and administration
  - Access control (security)
  - Development and run-time framework
  - Time synchronisation
  - Data archiving (raw and engineering)
- Monitoring and Control kernel
  - Space System Model
  - Control activities validation, execution and verification
  - Commanding
  - Procedures execution
  - Reporting data processing
  - Event processing
  - Live, playback, retrieval and replay processing
  - M&C services provision

Reference Implementation Functional Scope

- Adaptation layer
  - Commanding adapter
  - Reporting and event data reception and decoding/extraction
  - PUS services modelling
  - Time correlation
  - TM/TC I/F handler
- User applications
  - User desktop
  - User defined displays (plots, synoptic, alphanumeric)
  - M&C applications (control stacks, history logs, alarm summary display)
  - System displays
  - M&C Model browser (navigation tree)
- Preparation tools
  - Data model definition tool
  - System manager
- External interfaces
  - Tailoring data import
  - Archiving data export

Technology Selection

- Technologies are currently being considered
- Operating System
  - Linux is the selected Operating System for all implementations
  - Compatibility with Windows is required for the user interface and preparation environment
  - Operating system independence should be achieved where possible
- Programming Languages
  - Java is the preferred language for all EGS-CC components
  - Does not rule out the integration into EGS-CC of software implemented in other languages

Run-time Environment Technology Candidates (1/2)

Technology Domain	Candidates
Component Framework	OSGi, Spring, EJB3, CCM, SCA
Service Integration Platform	ESBs (OpenESB, ServiceMix, Mule, JBoss), CCSDS MAL+COM
Communication and Data Distribution	JMS (AciveMQ, Joran), AMQO, OMG Notification Service, OMG Data Distribution Service
System Run-Time Management	JMX
Logging and Tracing	Log4j, OSGi Log Service, OMG Logging Service
File Management	Revision Control: SVN, GIT, Content Repository API for Java (JCR)
Security	JAAS, LDAP

Run-time Environment Technology Candidates (2/2)

Technology Domain	Candidates
Data Persistence	Service Data Objects, Java Data Objects, Java Persistence Api, Java DataBase Connectivity, Hibernate
Data Archiving	RDBS, HDF5, noSQL (Cassandra),
Data Modelling and Editing	UML, Eclipse EMF
Post Processing and Reporting	Eclipse BIRT, JasperReports
Scripting and Automation Languages	Javascipt, Python, Groovy, PLUTO, ..
User Interface	Thick Clients (Eclipe RCP, Swing, QT), Thin Clients (Ajax, RAP,…), Flex
User Defined Displays	SVG, X3D, ..

SDE Technology Candidates (1/2)

Tool Category	Candidates
Software Configuration Management	Subversion, Git, Perforce
Requirements Management	DOORS
UML Modelling	Enterprise Architect, Magic Draw
Integrated Development Environment (IDE)	Eclipse, Netbeans, IntelliJ IDEA
Repository Browser	Atlassian FishEye, ViewVC, WebSVN, gitweb
Build System	Maven, Ant
Compiler / Debugger / Profiler	Javac (OpenJDK, Oracle) / IDE / VisualVM, JProfiler, Eclipse Memory Analyzer
Acceptance and Integration Test Management	TestNG, Jubula
Unit Test Framework	TestNG, JUnit

SDE Technology Candidates (2/2)

Tool Category	Candidates
Issue Tracking	Atlassian JIRA
Source Code Analysis and Metrics	Checkstyle, Cobertura, FindBugs, PMD, Sonar
Continuous Integration	Jenkins
Wiki	Atlassian Confluence
Project Management	Atlassian Confluence
Artifact Manager / Repository	Nexus
Source Code Documentation	Javadoc
Online Code Reviews	Atlassian Crucible
Mailing Lists	Mailman, Majordomo
User Management	Atlassian Crowd
Documentation / Manuals	Wiki, Docbook XML, Microsoft Office, OpenOffice
Installation and Deployment	RPM
Project Web Site	Atlassian Confluence

EGS-CC Project Phases

- Phase A (User Requirements & System Concept) currently in progress
- Phase B (Software Requirements Engineering and Architectural Design) intended to be carried out by industrial consortium (including MCS and EGSE developers) under monitoring of SB and SET
- Phase C/D (Detailed Design and Coding) will include the end-to-end validation based on the development of a full EGSE/MCS system using EGS-CC
- Phase E will cover maintenance and evolution
- Ideally an end-to-end program (covering also support to deployment/exploitation) is going to be established following successful completion of Phase B

Potential Risks

- Difficulty to reach technical consensus
- System over-complexity
- Funding and organisation for later phases
- Synchronisation with stakeholder's own roadmaps
  - Parallel development of systems that will use/interface with EGS-CC
  - Critical schedule
- Risk to lose momentum and commitment

Conclusions

- The EGS-CC initiative is in its early phases
- First time such a level of agreement is reached among so many parties
- Strong stakeholders commitment
- Phase A aiming at defining the programmatic and technical approach
- Objectives are very ambitious but feasible
- The expected benefits justify the commitment and associated investments