Standardised Ground Data Systems
Implementation: A Dream?

Y. Doat, C. R. Haddow, M. Pecchioli and N. Peccia
ESA/ESOC, Robert Bosch Straße 5, 64293 Darmstadt, Germany
Ground Data Systems at ESA/ESOC: The current approach

- Based on re-used infrastructure which is customised/tailored for the different missions
- The existing ground data systems infrastructure covers most of the domains typically required by ESA/ESOC missions
- This approach has proven to be effective in the past in terms of:
  - Reduced cost of ground segment developments for new missions
  - Increased cross-fertilisation between missions
  - Enabling the development of mission dedicated ground segments within a reasonable time-frame
  - Ensuring maturity and operational suitability of mission systems.
Ground Data Systems at ESA/ESOC: Lack of standardisation

• The current approach has enabled a high degree of standardisation across missions i.e. all missions are based on similar implementations of the ground segment (as they all share the same infrastructure systems)
• However, no standardisation/harmonisation has been achieved between the different ground data systems, not even between infrastructure systems produced by the same divisions
• This has lead to a proliferation of solutions that imply very high maintenance costs
Rational for More standardisation:
If it’s working why fix it?

• Lack of common approach on hardware platforms, operating system and 3rd party products → Proliferation of platform baselines
• Lack of technology harmonisation → Proliferation of dependencies, Heterogeneous HCI (Human Computer Interface) look and feel, Overhead in maintaining the required expertise
• Lack of common design → Proliferation of different solutions for the same problem…
• Lack of consistency across data systems → Increased familiarisation, validation and maintenance efforts
• Lack of synergy across developments → Increased risk and development time to reach mature implementations.
What is the “ESA Ground Operations S/W (EGOS)”?

• Target implementation of ESA ground segment infrastructure systems
  – Aims at standardising and harmonising existing systems
  – Improve interoperability
  – Reduce overall costs

• Constraint: Evolutionary approach required due to size/maturity of existing code base
Standardised Ground Data Systems
Implementation: A Dream?

ESA/ESOC Ground Data Systems
Context

OPS – Operations Preparation System
MPS – Mission Planning System
MCS – Mission Control System
FDS – Flight Dynamics System
NIS – Network Interface System
MATIS – Mission Automation System
SMF – Service Management Framework
STC – Station Computer
G/S – Ground Station

G/S Schedules (GRSS)+
Service Instance Configuration Files (SICF)

S/C Timeline Schedule (SCTS)
S/C Pass Schedule (SCPS)
Mission Automation Planned Schedule (MAPS)

Control System and G/S Link Monitoring and Control Data + SCTS, SCPS and MAPS + Status Messages

Procedure Definitions

Control System Monitoring and Control Data + SCTS and SCPS

G/S Link Monitoring and Control Data

TM and TC Data

Mission Planning Data

Predicts and Radiometric data
EGOS Ground Data Systems Scope

• Spacecraft Control Systems
• Simulators
• Ground Station Equipment
  – SLE Service Provider
  – Ground Station Monitoring and Control
• Mission Planning Systems
EGOS Architecture: Approach

- Move from a ‘vertical’ implementation of the different systems to the ‘horizontal design of common layers’
- Provide development and run-time frameworks for the application layer
- Minimise investment in low-level (middleware) components by re-using suitable 3rd party platforms (e.g. CCM, RCP)
- Maximise re-use of the existing implementation of the application layer (evolutionary approach).
EGOS Architecture Overview

Standardised Ground Data Systems Implementation: A Dream?
EGOS Component Overview

Standardised Ground Data Systems Implementation: A Dream?

EGOS Development Framework

EGOS Deployment and Configuration Framework

EGOS Component Run-time Framework
Service Management Framework

• Exposes services to external users/systems in standardised manner
• Enables ‘transparent’ access to the services provided by a system (interoperability)
• Controls access to exposed services
• Interfaces to internal services via drivers that handle required protocol conversion
Standardised Ground Data Systems Implementation: A Dream?

Service Management Framework Overview

SSM Logical Model

Service Management Framework

- Mission Automation System
- Internal Applications
- External Client (i.e. PI)
- External Applications

System element

- Reporting data
- Event
- Activity

SCOS Application

ORATOS-NG Application

MPS Application

SIMSAT Application

STC Application

EGOS & Non-EGOS Applications

AOCS

Power Subsystem

Star Tracker

Sun Sensor

Space Segment

Thermal Subsystem

MCS

FDS

MPS

Simulator

Ground Station

TC Processor

TM Processor

iSmfDriver

iSmfDriver

iSmfDriver

iSmfDriver

iSmfDriver

GSAW 2007
EGOS Technologies

- **Target platform:**
  - PC Linux (SLES)

- **Languages:**
  - C++
  - JAVA

- **Middleware**
  - Corba Component Model

- **Adaptive Communication Environment (ACE)**

- **Eclipse/SWT, Rich Client Platform (RCP)**
Infrastructure Migration towards EGOS: the MCS example

EGOS based MCS Framework

TM/TC Processing Components

GUIs

Ancillary Systems

SMF

Driver

MATIS

EDDS

Others

Driver

TM Servers

TC Servers

NIS

Client applicat.

Others

Driver

Driver

Driver

Driver

Driver
Current Status

- EGOS High Level architecture is being finalised
- Service Management Framework implemented and accepted
- New systems being developed adopt EGOS concepts and wherever possible already existing components
- EGOS Target Architecture taken into account in the evolution of the existing infrastructure systems
- Design of the EGOS Framework (Development and Run-time) has been finalised.
- Implementation for Core Components expected to start in Q3/2007
- User Desktop design currently being finalised. Implementation will start in Q2/2007
- Identification and redesign of Common Components will start in Q2/2007
Can a Dream become a Nightmare?

• High-level components lifecycle/maintenance
  → Avoid massive re-use of low-level components until they are fully validated and robust

• Technology Lifecycle is too short
  → Identify mechanisms to ‘slow down’ e.g. platform independent technologies, isolate the proprietary implementation, use of virtual machines?

• Avoid to re-implement what already exists
  → ‘Sandwich’ re-engineering i.e. replace middleware and visualisation layers but not the ‘core’ implementation of existing applications (business logic)
Thank you for your attention.
Questions?