Declarative Self-Expand Service Access Framework for NASA Mission Users

Rose Pajerski,	**Senior Scientist, Fraunhofer Center for Experimental Software Engineering, University of Maryland, College Park, MD U.S.A.**
Jinghong J. Chen,	**Ph.D., Principal Software Engineer, Honeywell Technology Solution Inc, MD U.S.A.**
David Warren,	**Project Lead, Honeywell Technology Solution Inc, MD U.S.A.**
Keiji Tasaki,	**NASA GSFC, Code 452, Deputy Project Manager, Ground Systems**

Fraunhofer

FRAUNHOFER CENTER FOR EXPERIMENTAL SOFTWARE ENGINEERING

Honeywell

NASA

SPACE NETWORK ACCESS SYSTEM

Ground System Architectures Workshop (GSAW) 2010
"*Innovation on the Ground*"

Topics

- Background, Issues, and Challenges
- Solution Proof of Concept
- Design Patterns & Approaches
- Implementation Approaches
- Operational Scenario
- Lessons Learned
- Future

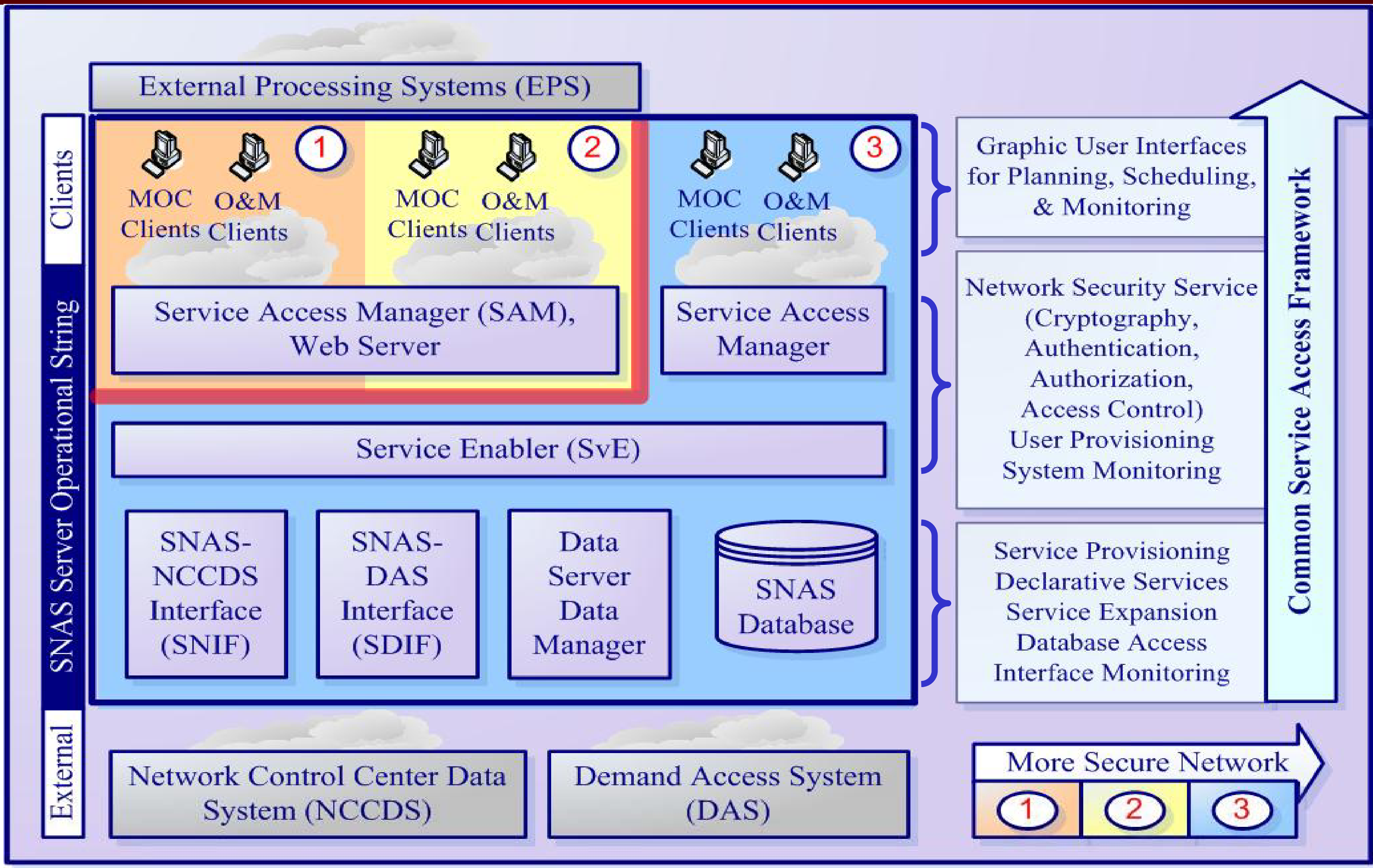
NASA Mission Service Access: Background, Issues, and Challenges

- Current NASA Service Access Environment for Missions Users
  - Multiple NASA and Commercial owned networks
  - Common standard security rule/constraints enforced
  - Various network communications protocols
  - Front-end software developed in different programming languages hosted on heterogeneous hardware/operating systems
- Problems
  - Lack of generic tool re-use between missions and ground systems
  - Redundant development efforts for front-end access systems
  - Inconsistency in network security implementation
  - Can't expand easily with new service, networks and protocols, and hardware platforms
- Desired Solution
  - A generic service access framework to abstract infrastructure complexities of applications, data, and heterogeneous platforms for future reuse and expansions

Solution: Proof of Concept

- NASA Space Network Access System (SNAS)
  - Tasked to replace legacy TDRSS scheduling and planning systems
  - Two Client software types, Five Server software subsystems
  - Web Server, Oracle database
  - Parallel HA server clusters for different operational modes
  - Support concurrent Open/Internet and Closed networks users
- Currently operational (Release 2), serving multiple NASA missions (26)
- Support TDRSS Access Services:
  - SSAF, SSAR, MAF, MAR, KSAF, KSAR, Shuttle, etc
- Declarative and Self-Expand characteristic (rapid service elasticity):
  - Data format structure and bindings for all supported services are declared and specified in the database and XML files. Server software code is generic. No hard-coded decoding/encoding is needed
  - Flexible for adding or deleting service
  - Service activation and deactivation are automated
  - Service change review approval processes are automated

SNAS System Architecture



Development Goals

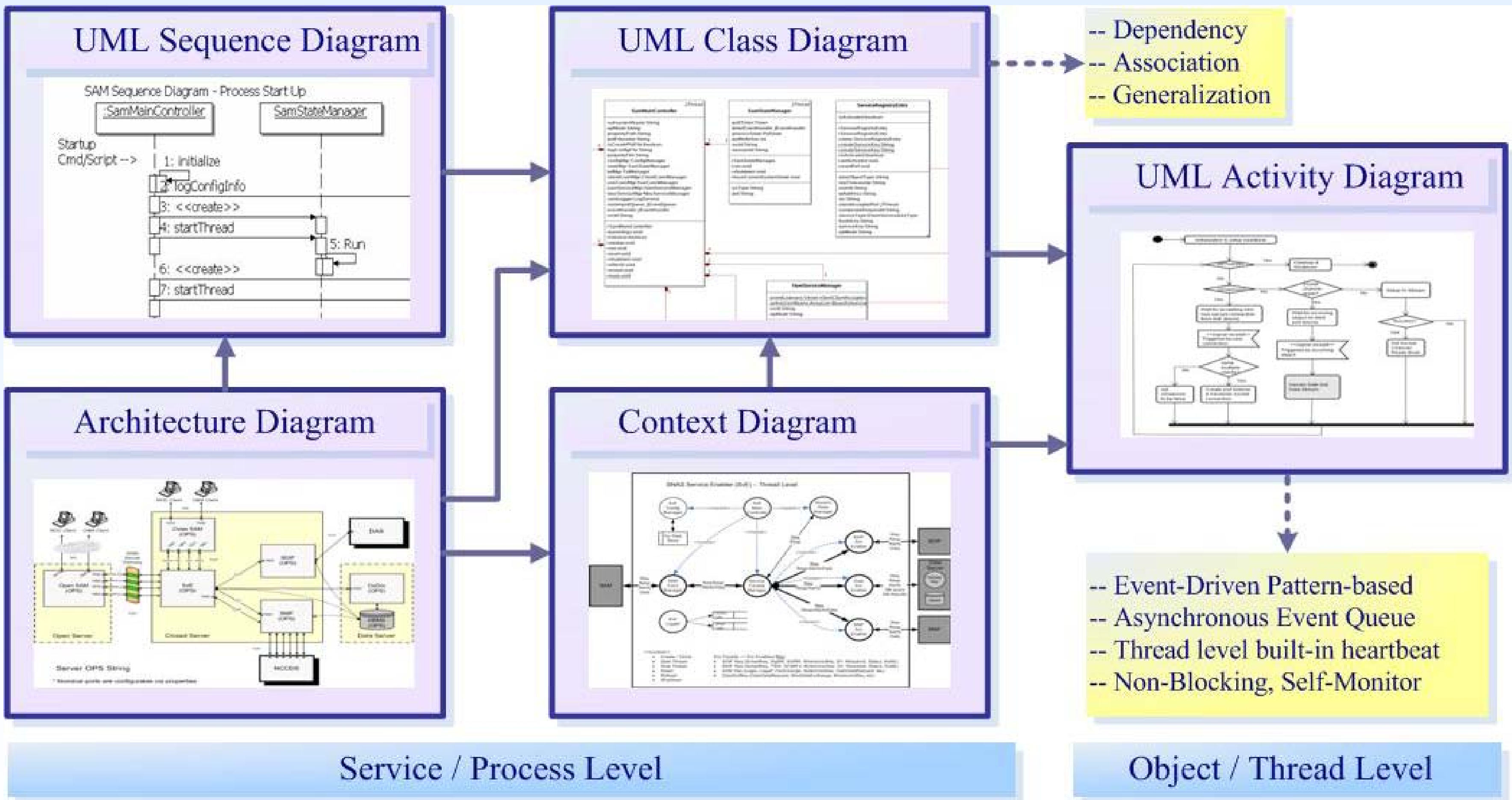
- ☑**Scalability** - scale up or down with workload demands (elastic). e.g. concurrent user service sessions, selectable mission groups & services
- ☑**Availability** – provides high availability, automatic cluster fail-over
- ☑**Reliability** – automatic self-monitor and recovery (built-in thread level heartbeat), cause no disruption
- ☑**Security** – authentication, authorization, privacy, integrity, non-repudiation
- ☑**Flexibility and Agility** - reusable building blocks (COTS, GOTS frameworks) to speed up development cycles
- ☑**Serviceability** - system's underlying infrastructure components can be updated or replaced without disrupting system's characteristics including availability and security
- ☑**Efficiency** – software can be deployed quickly and easily

Although meeting all these goals for a new infrastructure in a short time can be challenging due to technology immaturity, resource and budget constraints, they were achieved with proven software design and implementation approaches.

Service-based Design Patterns & Approaches

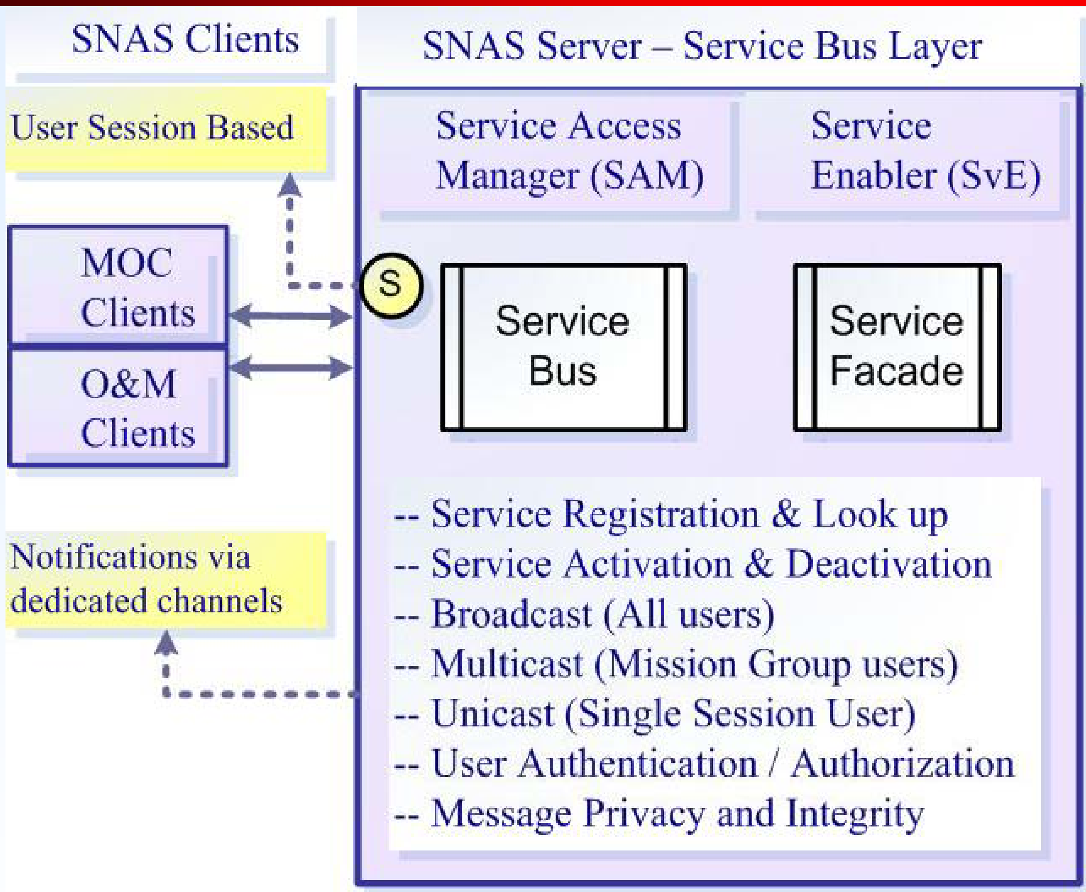
- **Service-based design with UML**
  - Leverage Object-Oriented Design (internal fine-grained) and Service Oriented Architecture approach (external coarse-grained)
  - UML diagrams (strictly selected)
  - Pattern-based design approach enables rapid system development within resource and budget constraints
  - Loosely coupled services with self-monitoring, self-recovery, and self-expanding as built-in features
- **Lightweight common service framework** provides building blocks:
  - Concurrent Service, Network Security Service (e.g. SSL)
  - Communications Service (standard network protocols)
  - Data Access Object (DAO) Service, Transfer Data Service, Logging Service, Time Service, and Generic Utilities.
- Design with focus on extensibilities in the areas of **User Access**, **Communication Network Access**, and **Mission Service Access.**

Service-based Design using UML



Effective in communicating the desired system architecture and behavior within limited time constraint

Implementation: User Access



MOC Client: provides user interface for scheduling, real-time service monitoring and control

O&M Client: provides user interface for SNAS users and subsystems monitoring and maintenance

- **Heterogeneous Clients and hosting platforms**:
  - Co-exist heterogeneous client types with multi-purposed data processing cleanly separated within server subsystems
  - Broadcast and multicast are accomplished with the use of subscribe and publish patterned service bus element embedded in the SAM instances
  - User session based request and response
  - Pure Java applications ensure portability across different operating system platforms.

Implementation: Network Access

- **Pluggable Communications Networks and Protocols:**
  - Multiple instances of server-side secure SAM are created, tailored, and deployed for different networks
  - SAM instances isolate external interfaces from backend SNAS server subsystems
  - NASA specific security constraints are implemented across different security access boundaries and layers
  - Framework's communication and security services handle communication network protocols and security requirements
  - Future advanced security technology can be easily adopted within the common service framework without impacting business domain data processing

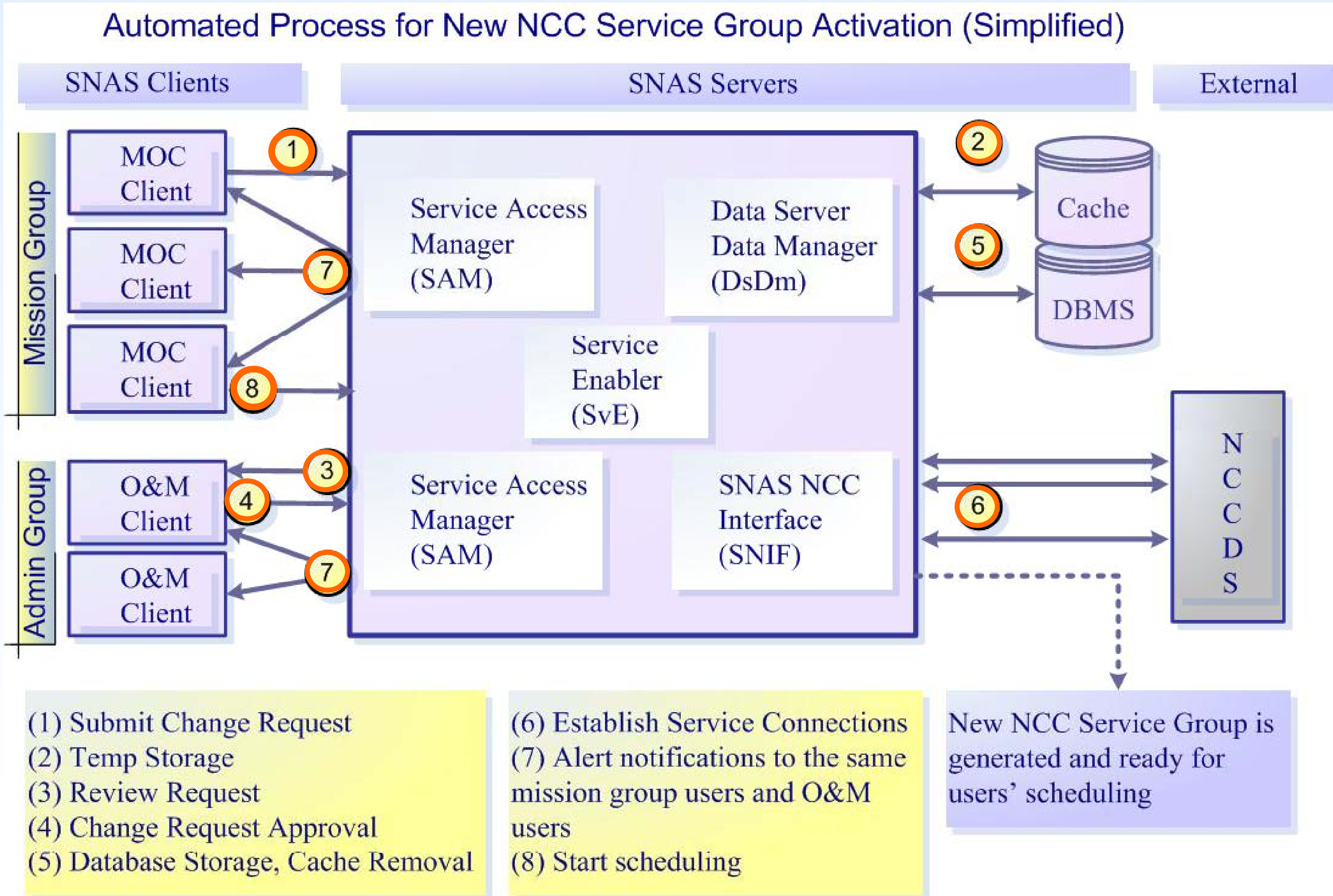
Implementation: Mission Service Access

- **Customer Need**: pluggable mission service capability is the most desired feature
- **Declarative Self-Expand Mission Services**:
  - SN service specifications are defined in the database including service types and parameters
  - No dependencies between different mission services
  - Mission service can be constructed by operator without the need of software change
  - On demand mission service scale-up or scale-down can be initiated by either O&M or MOC (triggers automatic review process)
  - O&M controls the dynamic mission service activation and deactivation remotely from different network access
  - Framework's original concurrent service API enables SNAS server subsystems to scale vertically
  - Timely responsiveness to concurrent MOC user service access sessions are achieved without compromising security

Operational Scenario: Service Change

- Mission service change scope:
  - Operations: create/setup, enable and disable, update, and delete
  - Automated Process: data storage and update, approval or rejection, broadcast or multicast, activation or deactivation, system configuration refresh
- Characteristics:
  - Data exchange between MOC and O&M clients with server subsystems as the broker and decision maker
  - Service type and parameters are protected with locking and synchronization while O&M has the higher privilege for overriding
  - Automatic self-expand physical service connections (TCP/IP)
  - On-demand mission service invocation causes no system shutdown or interruption of other ongoing real time mission services
  - Thread based service group self-cloning technique provides OS scheduling fairness and load balancing

Operational Scenario: New Service Activation

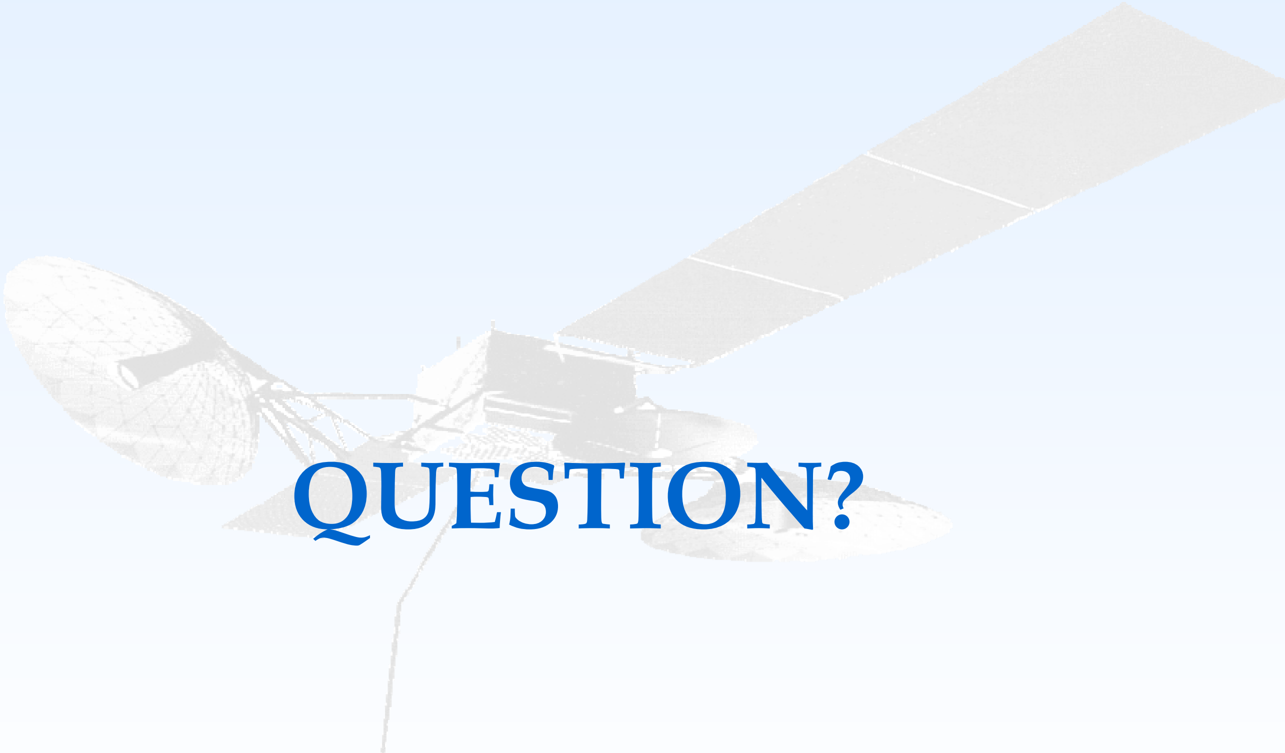


Lessons Learned

- Incremental replacements with innovative techniques for the legacy systems are evolutional processes
- Major lessons learned during development are mostly associated with the reuse of legacy software, COTS, and GOTS
- Systematic reuse approaches are realized by:
  - Thorough legacy system reuse analysis at the early design phase
  - Maximize reuse of proven legacy processing logic in the business domain only
  - Maximize open source software usage to avoid vendor lock-in
  - Eliminate tight coupling among services and service components
  - Disciplined code refactoring to avoid re-work due to incompatible requirements between new and legacy systems

Future

- Service framework as generic tool used between missions and ground systems:
  - Automatic horizontal self-service, and self-expanding
    - Process-based, load-driven self-scaling
    - High availability within server farm
  - On demand component composition and service workflow
  - Open standard interfaces targeting high-throughput and low latency data communications
    - Flexible interface format
    - Infrastructure as a service (IaaS) concept applied between operation facilities
  - Extend security protocols supporting asynchronous messaging model
    - Available security implementations are based on synchronous messaging model
    - Security needed for cross-facilities service offerings via asynchronous messaging



QUESTION?

Backup Slide: Technical Terms

- Definitions
  - ***Cloud Computing***: a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction.
  - ***Rapid elasticity:*** refers to one of the five cloud computing essential characteristic that capabilities can be rapidly and elastically provisioned to quickly scale up and rapidly released to quickly scale down [1].
  - ***Declarative service technique:*** refers to Service-Oriented Architecture (SOA) software implementation technique where service descriptions, and/or processing workflow bindings are defined externally (e.g. database, XML files, or external configuration elements) so that rapid service elasticity can easily be achieved.

[1] NIST Definition of Cloud Computing, http://csrc.nist.gov/groups/SNS/cloud-computing/index.html

Abbreviation & Acronym

API	Application Programming Interface
ANCC	Auxiliary Network Control Center
COTS	Commercial Off The Shelf
DAO	Data Access Object
DAS	Demand Access System
DSDM	Data Server Data Manager
EPS	External Processing System
GOTS	Government Off The Shelf
GSFC	Goddard Space Flight Center
HA	High Availability
HTSI	Honeywell Tech. Solutions Inc
IaaS	Infrastructure as a service
KSAF/R	K-band Single Access Forward / Return
MAF/R	Multiple Access Forward / Return
MOC	Mission Operations Center
NCCDS	Network Control Center Data System
NENS	Near Earth Network Services
O&M	Operations and Maintenance
OS	Operating System
SAM	Service Access Manager
SDIF	SNAS-DAS Interface
SN	Space Network
SNAS	Space Network Access System
SNIF	SNAS-NCC Interface
SSAF/R	S-band Single Access Forward / Return
SSL	Secure Sockets Layer
SvE	Service Enabler
TCP/IP	Transmission Control Protocol/Internet Protocol
TDRSS	Tracking and Data Relay Satellite System
UML	Unified Modeling Language