The INTERSECT Open Federated Architecture for the Laboratory of the Future

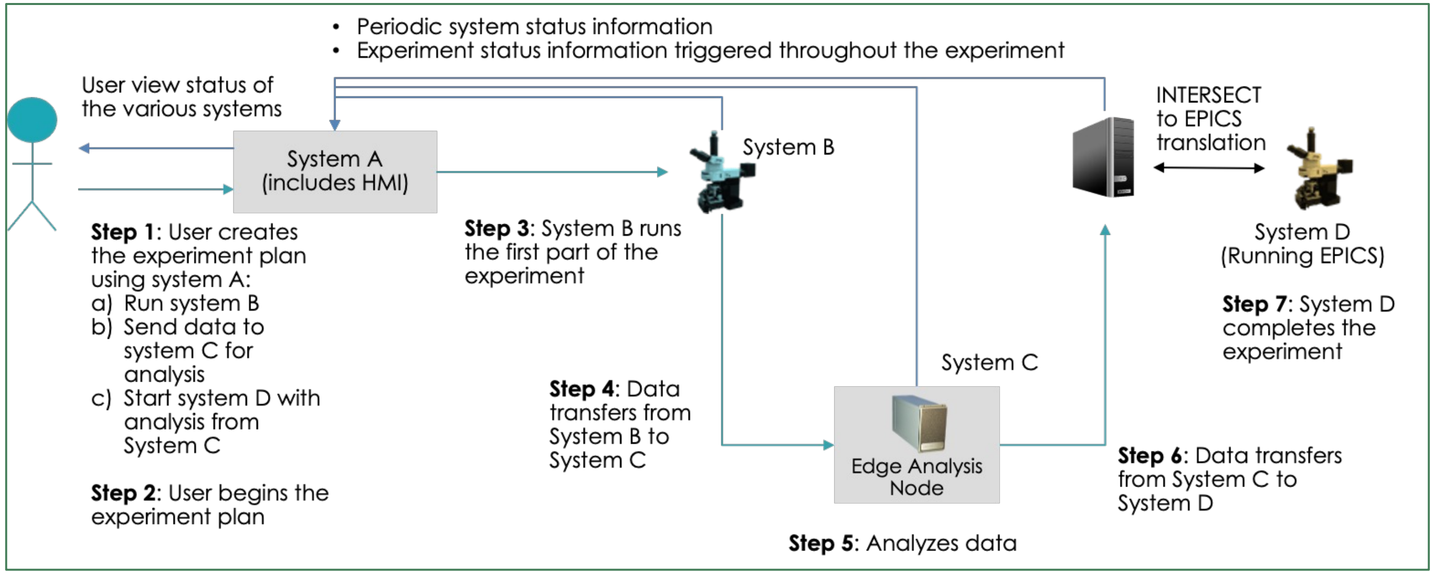
Christian Engelmann, Olga Kuchar, Swen Boehm, Michael J. Brim, Thomas Naughton, Suhas Somnath, Scott Atchley, Jack Lange, Ben Mintz, and Elke Arenholz

Oak Ridge National Laboratory

Contact: Christian Engelmann
engelmannc@ornl.gov

Overarching Goal of INTERSECT LDRD Initiative

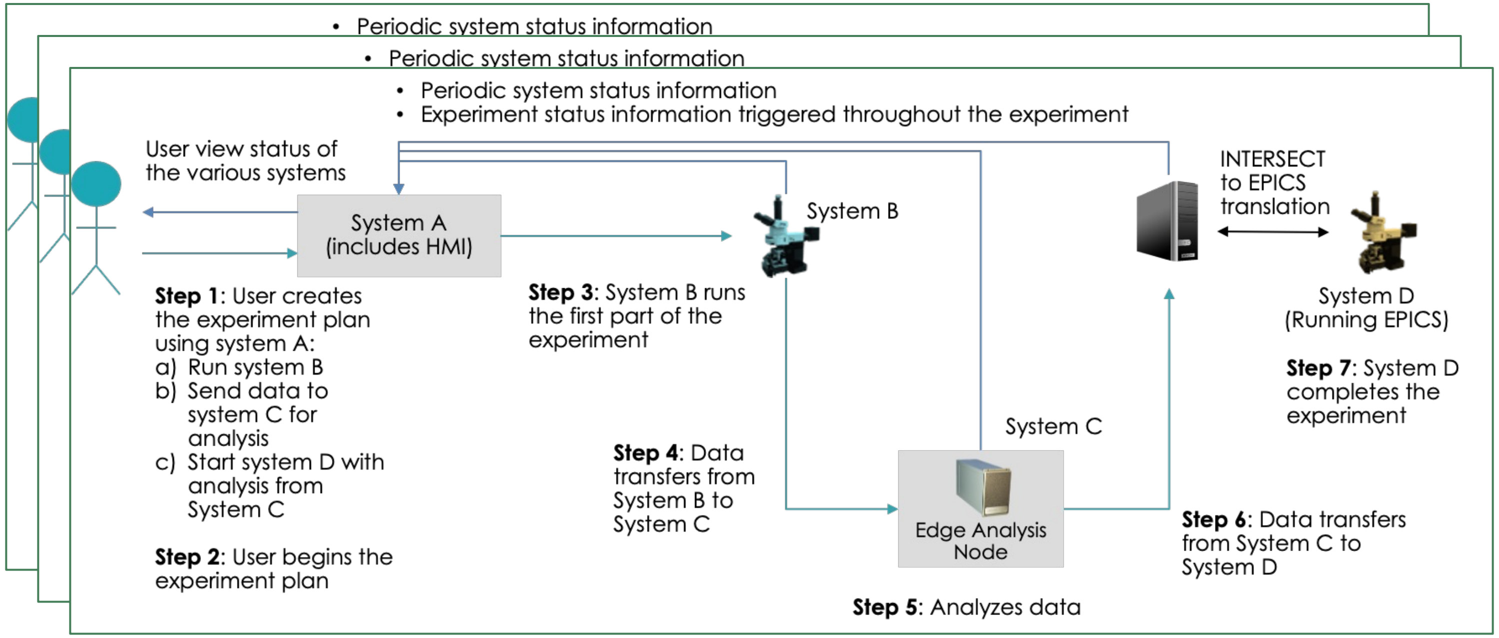
- Enable science breakthroughs using intelligent networked systems, instruments and facilities with
  - Autonomous experiments
  - “Self-driving” laboratories
  - Smart manufacturing
  - AI-driven design, discovery and evaluation



OAK RIDGE National Laboratory

Problem Statement

- A federated instrument-to-edge-to-center hardware/software architecture is needed to autonomously collect, transfer, store, process, curate, and archive scientific data and reduce human-in-the-loop needs with
  - Common interfaces to leverage community and custom software
  - Pluggability & composability to permit adaptable solutions, reuse, and digital twins
  - An open architecture standard to promote adoption by DOE's science facilities



Current State of the Field: Instrument/Data Science Workflows

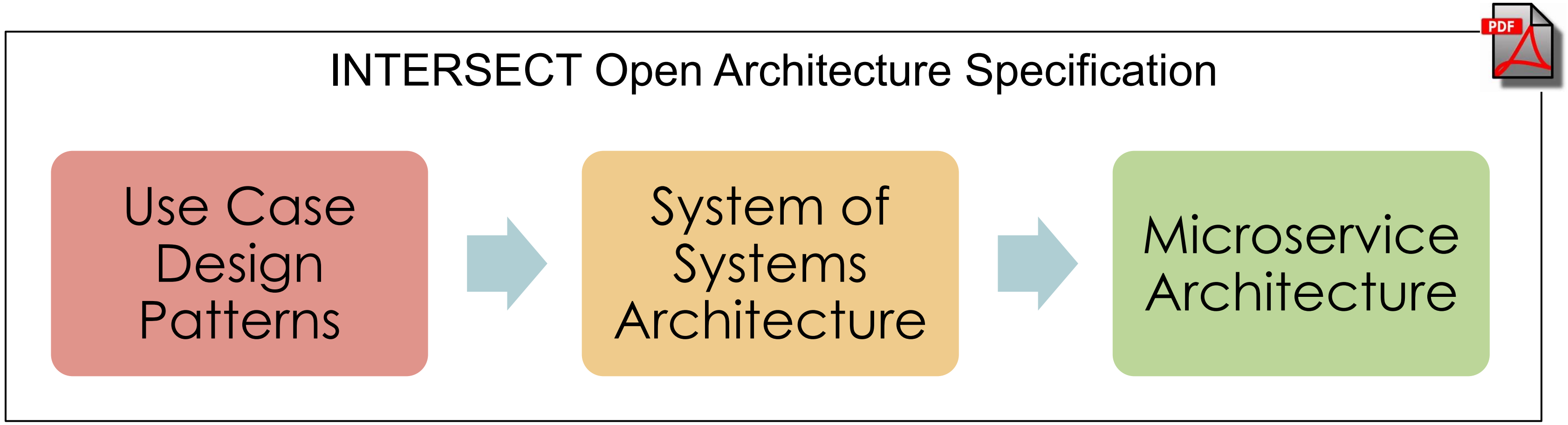
- ***There are ~300 workflow solutions for instrument and data science***
- Only very few are holistic; none offer an architecture standard
  - NERSC Superfacility framework: SLAC LCLS + Esnet + Cori
  - ANL Globus Automate, Gladier, Balsam and funcX: APS + Theta
  - ORNL's FY20/21 Federated Science Instruments LDRD: SNS/HFIR + OLCF
  - Autonomous robot-controlled chemistry laboratory at the University of Liverpool
  - Various other laboratory, university and industry human-in-the-loop solutions

Architecture?

- ***What is an architecture: A set of principal hardware/software design decisions!***
  - What hardware/software components are involved?
  - Why and how do they communicate with each other?
  - What data formats do they use and why?
  - How and when do they synchronize with each other?
  - What are the work, data and control flows?
  - How do they interface with the rest of the world, like humans?
- ***Why an architecture?: Would you buy a house that wasn’t architected?***
  - An architecture is purpose-driven design with forethought!
  - It permits creating flexible solutions that meet different requirements.
  - It prevents reengineering costs.
  - It allows standardization (IEEE/DIN/ISO/…).

Approach

- Create an open federated hardware/software architecture for the laboratory of the future using novel methods, consisting of



INTERSECT Architecture Specification

A written documentation of the INTERSECT Architecture, like a blueprint

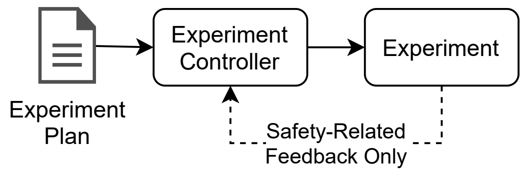
- Science Use Case Design Patterns
  - Abstract descriptions of the involved hardware and software components and their work, data and control flows.
- System of Systems (SoS) Architecture
  - Detailed design decisions about the involved hardware and software components from different points of view (e.g., logical, physical, operational, data, …)
- Microservice Architecture
  - Detailed design decisions about software microservices, including their functionalities, capabilities, compositions, with control, work, and data flows.

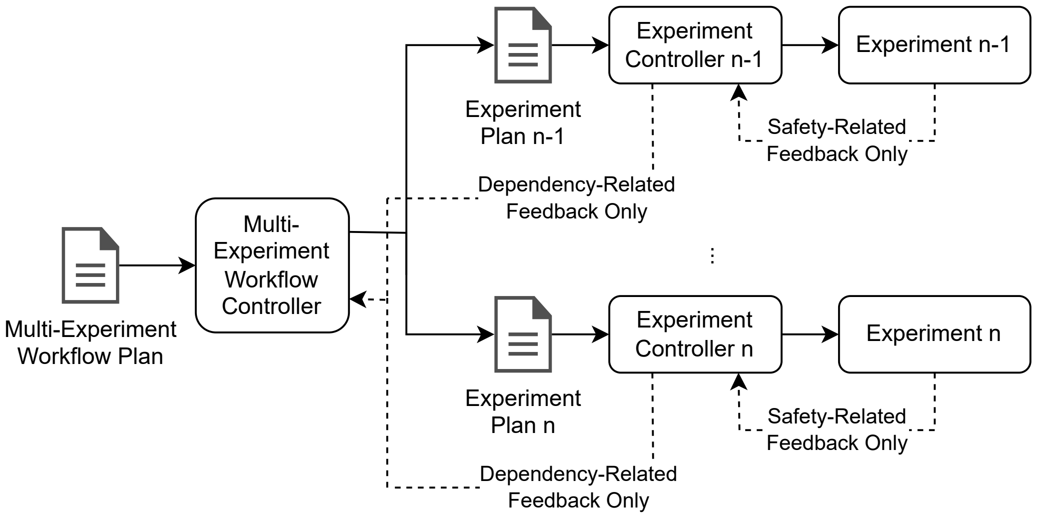
Current State of the Field: HW/SW Architecture

- ***Design patterns*** have been used for several decades to systematize software development using proven software engineering techniques
  - Used in object-oriented programming, distributed systems, data processing/automation, etc.
  - Also: HPC resilience design patterns (Christian Engelmann's 2015 DOE Early Career Award)
- ***System of Systems architectures*** have been used in the defense sector for designing complex systems from smaller and easier to design systems
  - 5 key characteristics: Operational independence, managerial independence, geographical distribution, emergent behavior, and evolutionary development
  - Example: DARPA's System-of-systems Technology Integration Tool Chain for Heterogeneous Electronic Systems (STITCHES)
- ***Microservice architectures*** are the modern approach to decompose complex software
  - Examples: Netflix OSS, Kubernetes, Cray/HPE system management

Science Use Case Design Pattern Anatomy

- ***Basic template is defined in a control problem paradigm***:
  - Open vs. closed loop control
  - Single vs. multiple experiment control
- Universal patterns that describe solutions free of implementation details
- Patterns may exclude each other or may be combined with each other
- Described pattern properties:
  - *Name, Problem, Context, Forces, Solution, Capabilities, Resulting Context, Related Patterns, Examples, and Known Uses*





OAK RIDGE National Laboratory

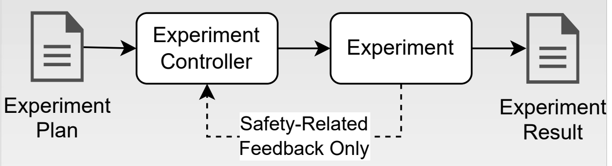
Science Use Case Design Patterns: Classification

- ***Strategy patterns:*** High-level solutions using different control features
- ***Architectural patterns:*** More specific solutions using different hardware/software architectural features

Strategy	Experiment Control		Experiment Steering		Design of Experiments		Multi-Experiment Workflow
Architectural	Local Experiment Control	Remote Experiment Control	Local Experiment Steering	Remote Experiment Steering	Local Design of Experiments	Remote Design of Experiments	

Science Use Case Design Patterns: Strategy Patterns

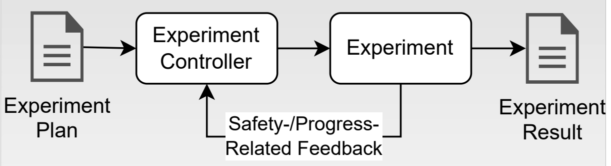
Experiment Control



❖ Executes existing plan

- Open loop control
- Automated operation

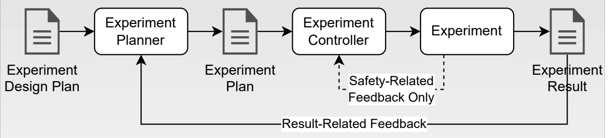
Experiment Steering



❖ Executes existing plan, depending on progress

- Closed loop control
- Autonomous operation
- Extends patterns:
    - *Experiment Control*

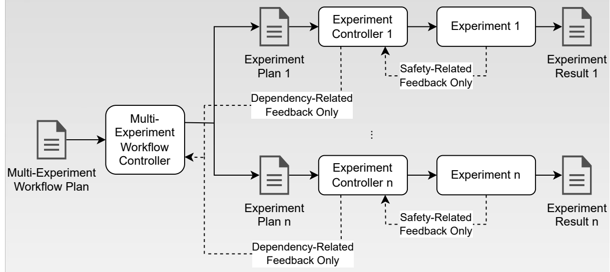
Design of Experiments



❖ Creates/executes plan, based on prior result

- Closed loop control
- Autonomous operation
- Uses patterns:
    - *Experiment Control*
- May use patterns:
    - *Experiment Steering*

Multi-Experiment Workflow



❖ Executes existing plans (workflow/experiments)

- Open loop control
- Automated operation
- Uses patterns:
    - *Experiment Control*
- May use patterns:
    - *Experiment Steering*
    - *Design of Experiments*

Science Use Case Design Patterns: Architectural Patterns

Table 2: Features and relationships of the science use case architectural patterns

Architectural Pattern	Related Strategy Pattern	Remote Components
Local Experiment Control	Experiment Control	None
Remote Experiment Control	Experiment Control	Controller
Local Experiment Steering.	Experiment Steering	None
Remote Experiment Steering	Experiment Steering	Analyzer and Controller (optional)
Local Design of Experiments	Design of Experiments	None
Remote Design of Experiments	Design of Experiments	Analyzer and Planner (optional)

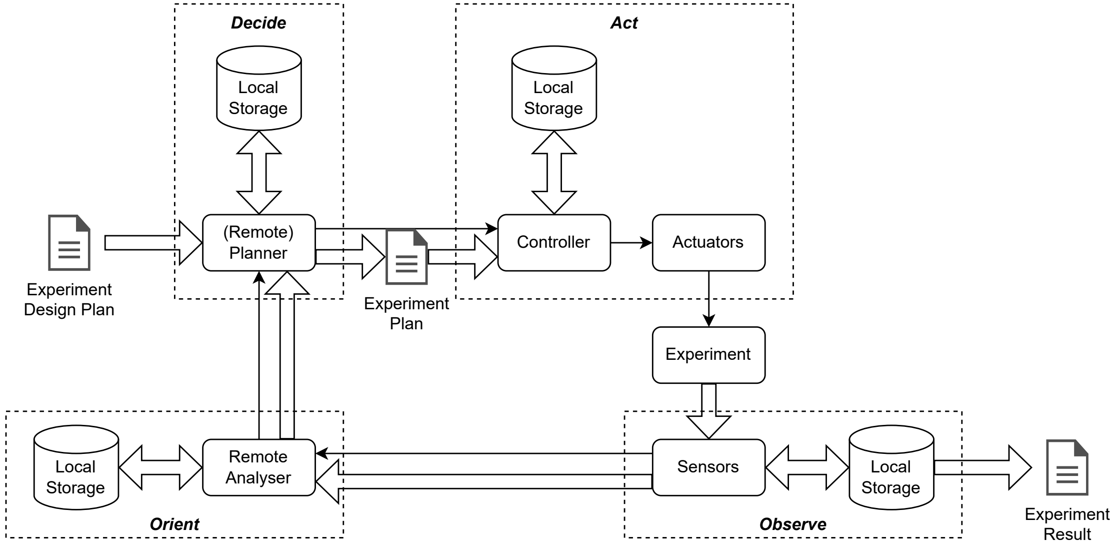
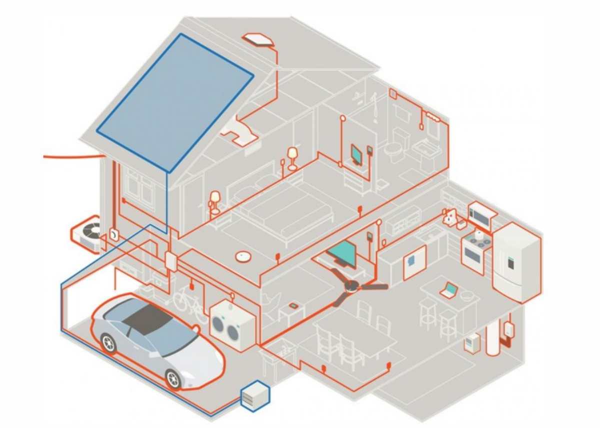


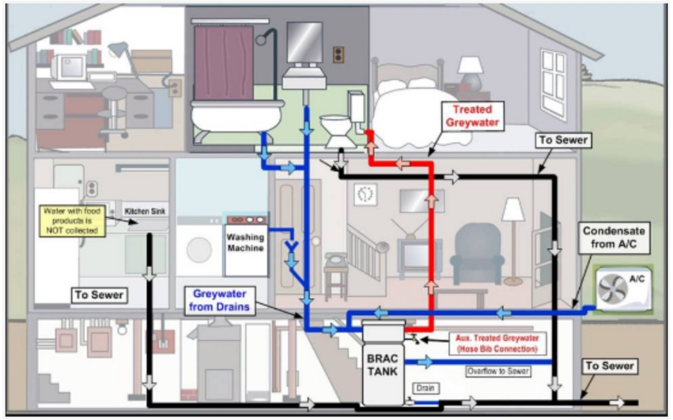
Fig. 3: Remote Design of Experiments architectural pattern

Architecture Framework and Views

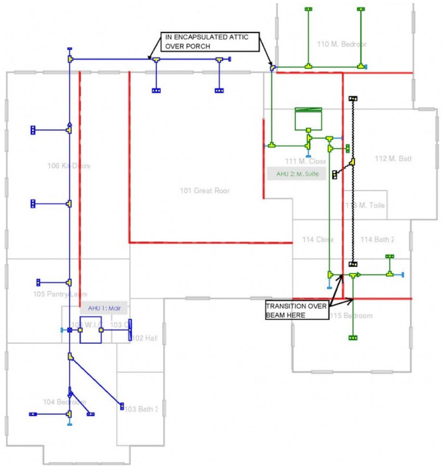


Electrician's view

Buyer's view



Plumber's view



HVAC prof's view

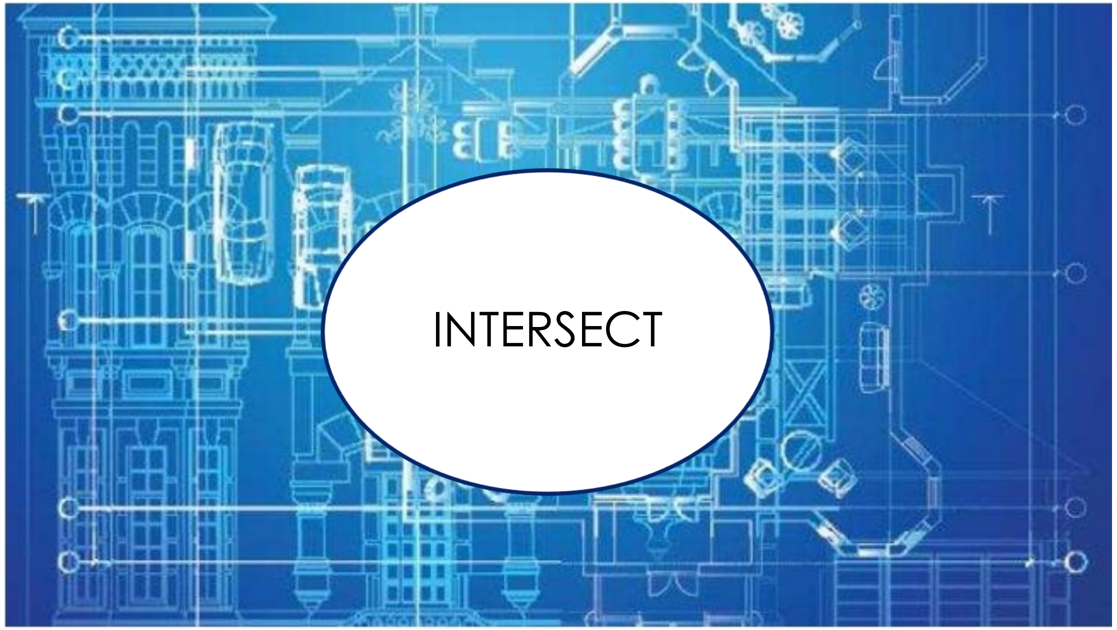
Intersection of multiple constraint spaces

INTERSECT System of Systems Architecture Views

User View

Data View

Operational View



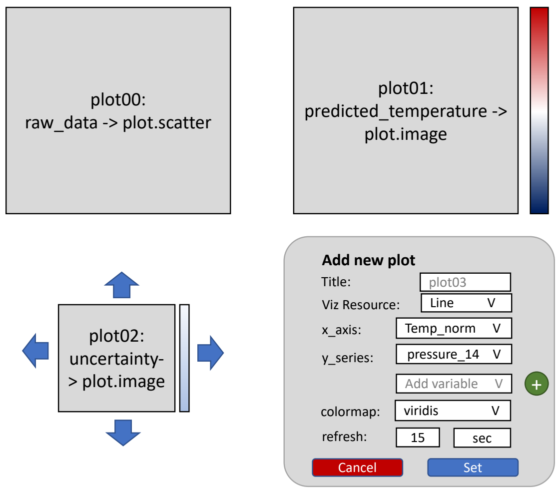
Logical View

Physical View

Standards View

System of Systems Architecture: User View

- ***Highlights user-facing functionality***
- Does not include system-internal interactions
- Described activities:
  - Logging into dashboard
  - Experiment creation
  - Start experiment
  - Steer experiment
  - Experiment end



Resource Monitor

Resource	Status	Performance
Asylum Research Cypher (P.I.M.ARC1)	Normal	0.25 locations/sec
NVIDIA DGX 2 (P.C.E.DGX2CNMS1)	Normal	8 sec/batch turnaround time
OLCF Summit (P.C.H.OLCFSummit)	Degraded	12 min/simulation
Edge Storage (P.D.E.BlackPearlCNMS)	Down	0 MB/sec
Add Resource or component of		

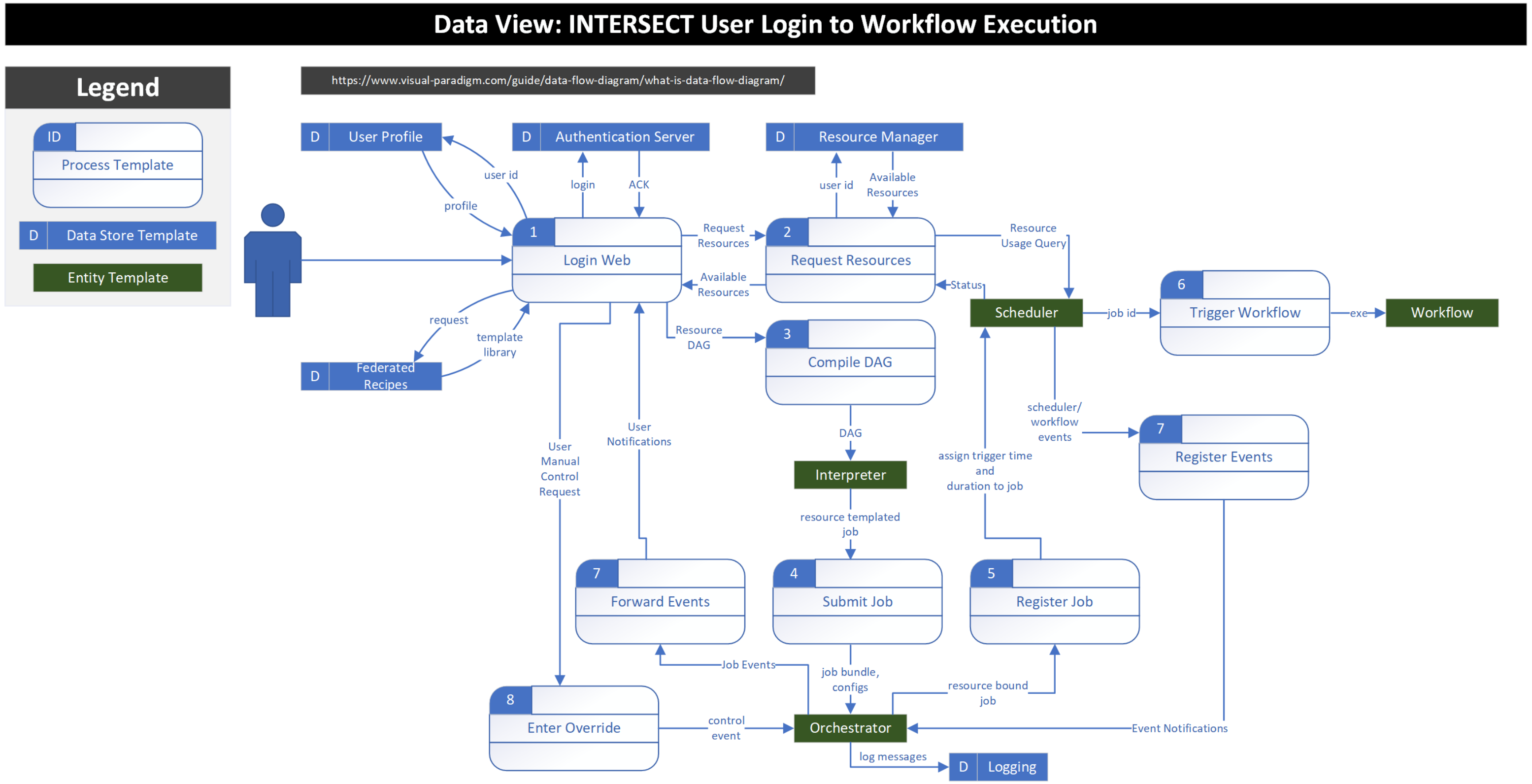
Campaign progress

Time: Elapsed: 01 hr 12 min Est. remaining: 45 min 13 min

Progress: 68%

System of Systems Architecture: Data View

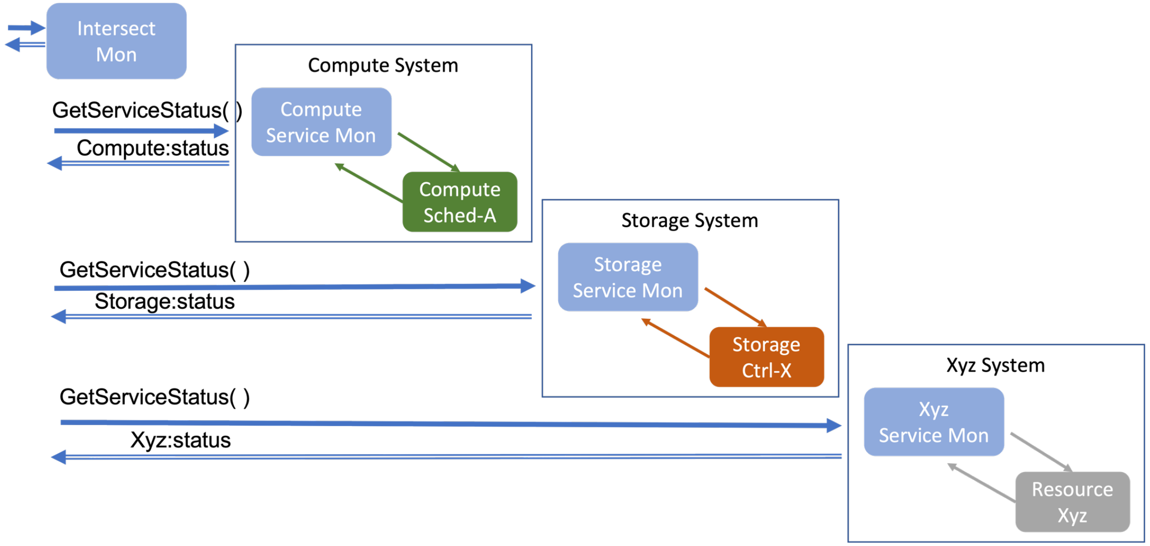
- ***Highlights the system's data needs and framework***
- Includes conceptual, logical, and physical data models
- Does not include specifications for scientific, instrument, experiment data



Initial Data Flow Diagram

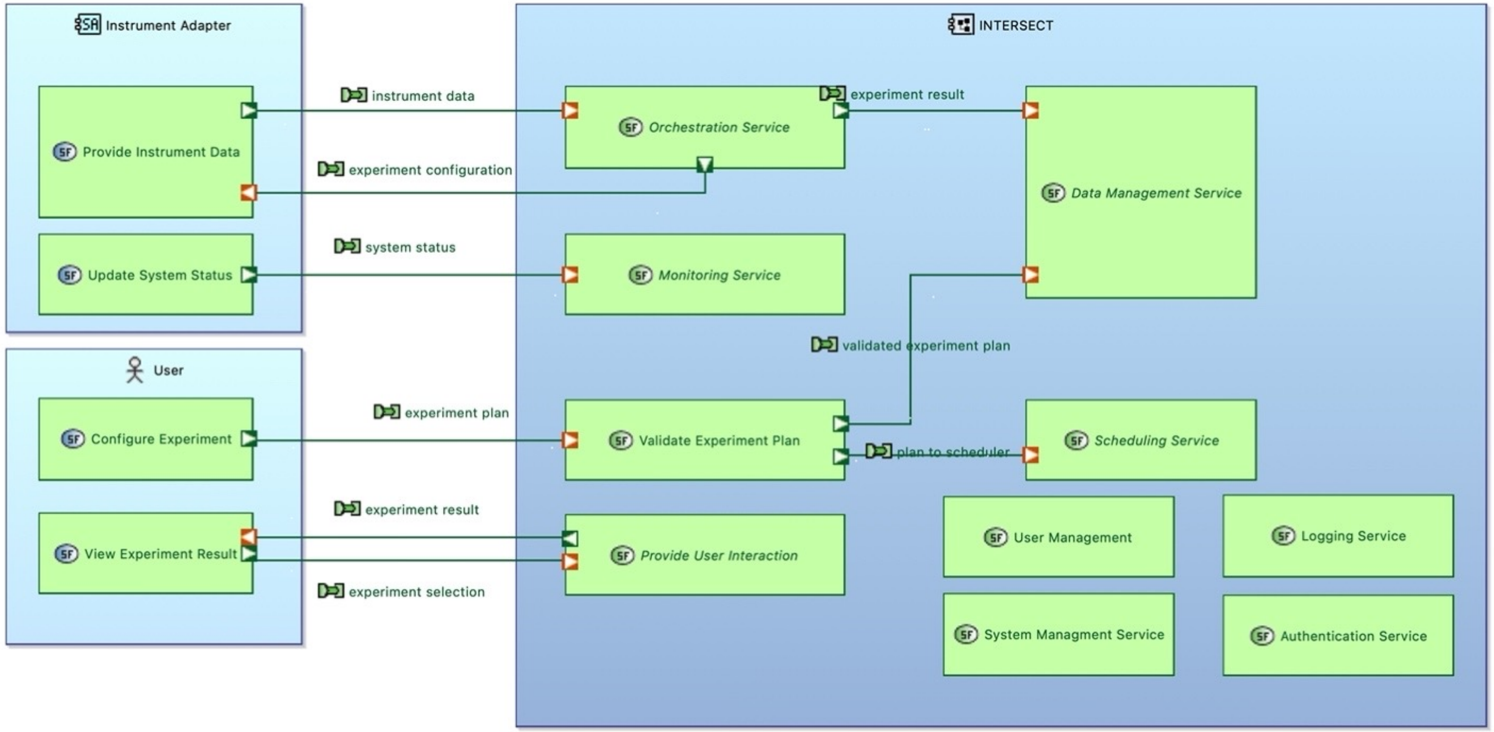
System of Systems Architecture: Operational View

- ***Highlights tasks, activities, procedures, information exchanges/flows from the perspective of operations stakeholders***
- Does not include formats for data exchanges or details of user applications



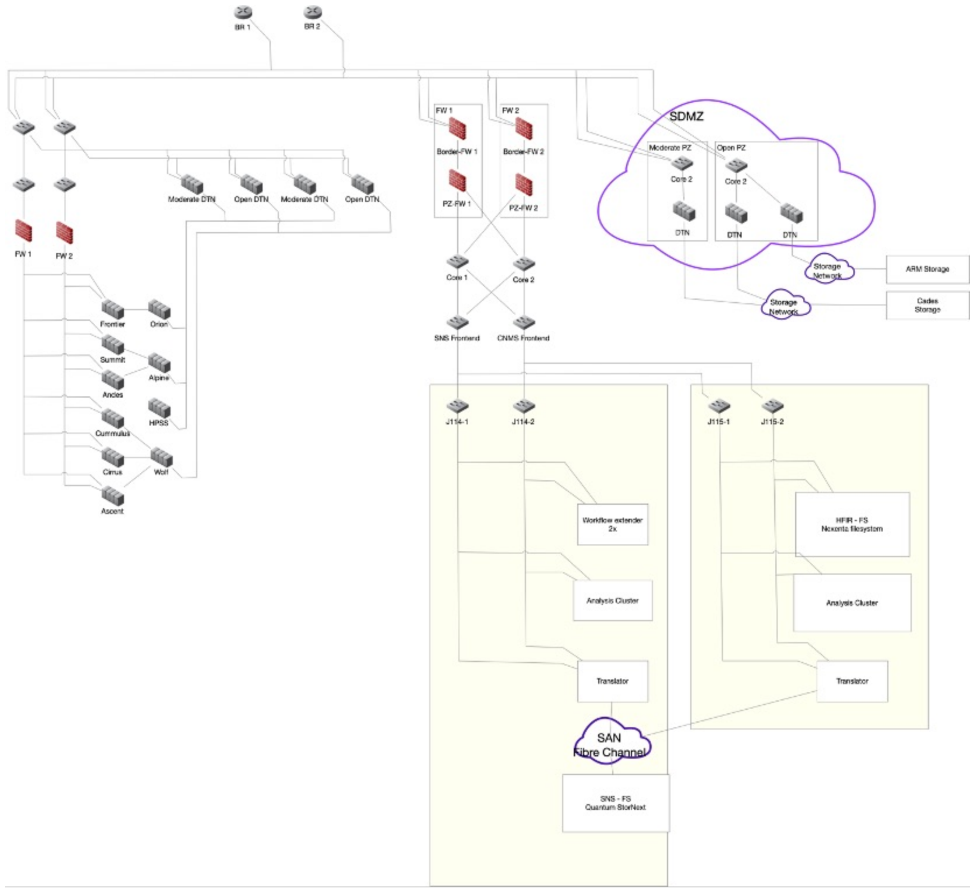
System of Systems Architecture: Logical View

- ***Highlights the logical composition and interactions between the different systems***
- Includes:
  - Definition of system concepts
  - Definition of system options
  - System resource flow requirements capture
  - Capability integration planning
  - System integration management
  - Operational planning



System of Systems Architecture: Physical View

- ***Highlights the underlying system components from the perspective of resource managers/owners, system administrators, network engineers, and facility space managers***
- Includes descriptions and definitions of physical systems, networks, connectivity and organizational boundaries
- Does not include specifications for instruments, resources, experiments and data



System of Systems Architecture: Standards View

- ***Highlights various standards including instruments specific standards, messaging standards, and other external standards***
- Provides a table of supported standards and other views or architecture elements that are impacted by each standard
- Provides a block diagram to illustrate exactly where each standard impacts a given system

Table 3: Example of messaging standards maintained in the standards view

Name	Version	Affected Views	Affected Elements
INTERSECT Core Messages	1.0	Data, Logical, Operational	Microservice Capabilities: All
Compute Allocation Capability	1.0	Data, Logical	Microservice Capabilities: Application Execution, Container Execution, Host Command Execution
Compute Queue Capability	1.0	Data, Logical	Microservice Capabilities: Compute Queue Reservation
NION Swift API	0.16.3	Logical, Operational	Systems: Electron Microscopes
Robot Operating System (ROS)	2.rolling	Logical, Operational	Systems: Additive Manufacturing

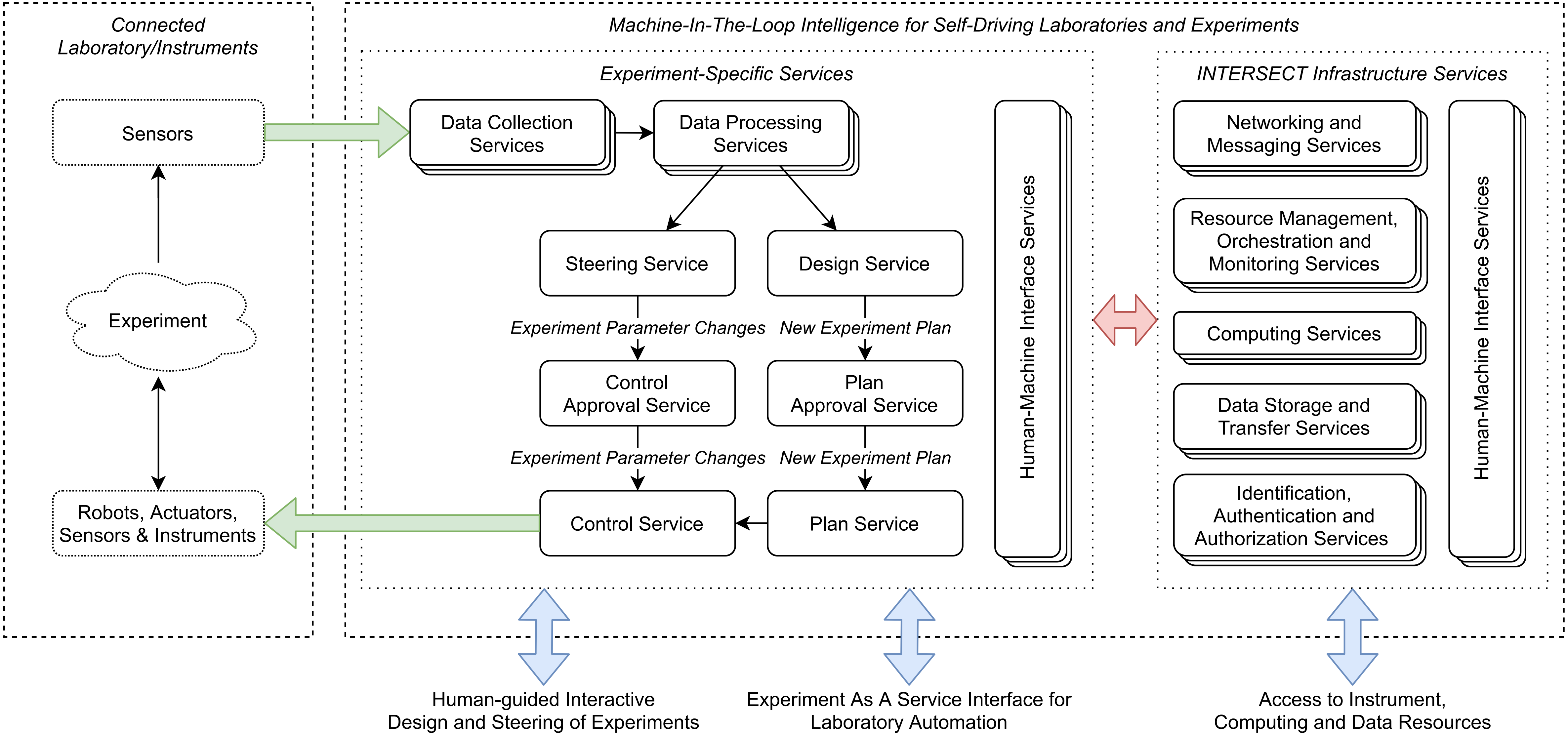
Microservice Architecture: Overview

- A design methodology for structuring an application as a set of loosely-coupled, independently developed, managed, and operated microservices that communicate over the network
  - A flexible alternative to the more prescriptive and tightly-coupled Service-oriented Architecture

- What makes a service a "microservice"?
  - Despite the moniker, has nothing to do with relative size of service code or its resource utilization
  - Definition: A service that provides a well-defined set of functionalities
    - typically scoped using domain-driven design, and defined using APIs and/or message structures

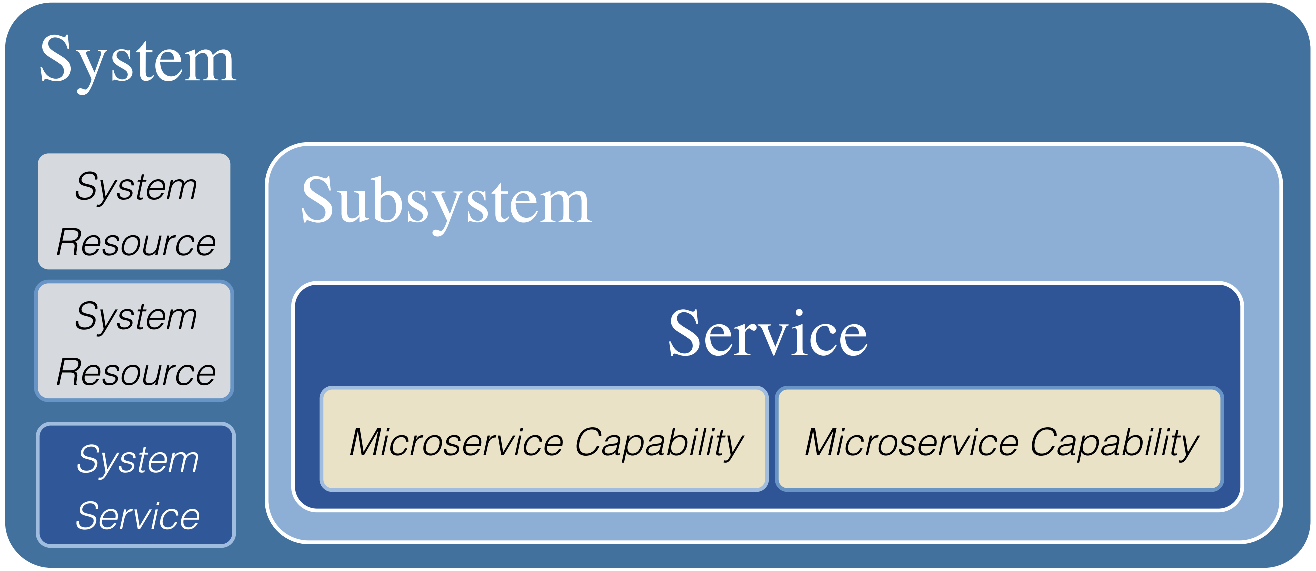
- There may be several interchangeable implementations of a microservice
  - Each implementation may use different underlying technology or algorithms, or may be tailored for use in a particular execution environment (e.g., edge vs. cloud vs. HPC)

Microservice Architecture: Classification & Catalog



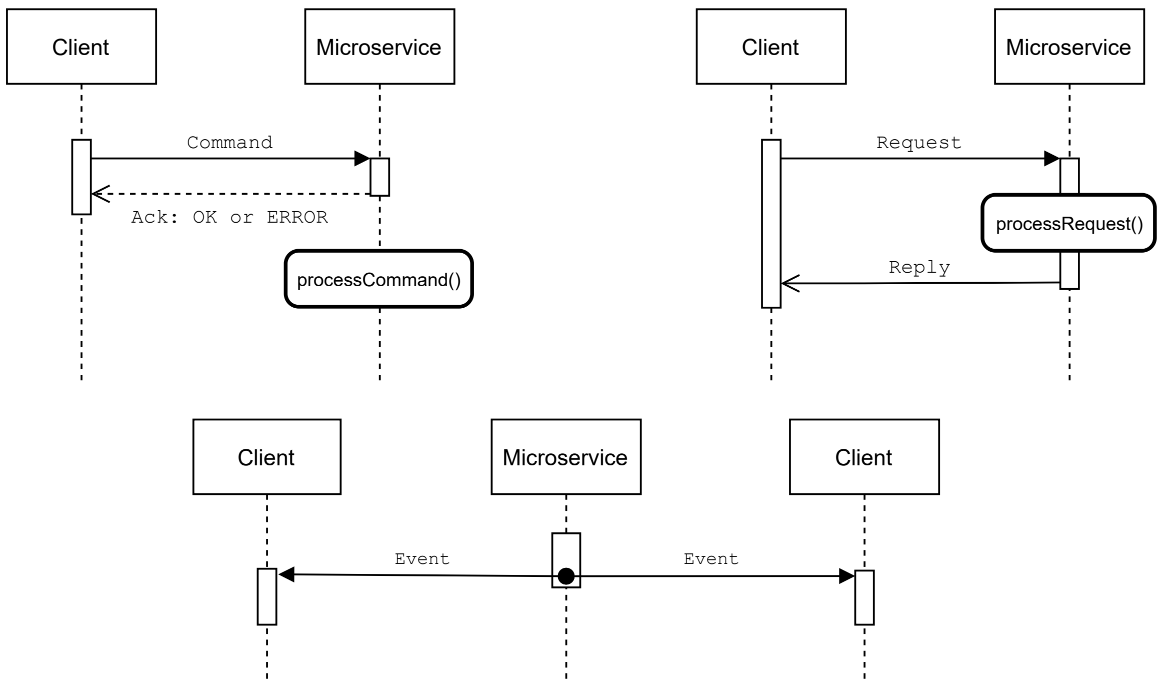
Microservice Architecture: Systems, Subsystems, Services and Microservice Capabilities

- System consists of
  - Subsystems
  - System resources
  - System services
- Subsystem consists of
  - Services
  - (Subsystem resources)
- Service consists of
  - Microservice capabilities



Microservice Architecture: Interaction Patterns

- Command / Acknowledgement
  - Responds immediately
- Request / Reply
  - Responds after fulfilling the request
- Asynchronous Event
  - Status update or event information
- Can be mapped to asynchronous and RESTful client-server communication implementations



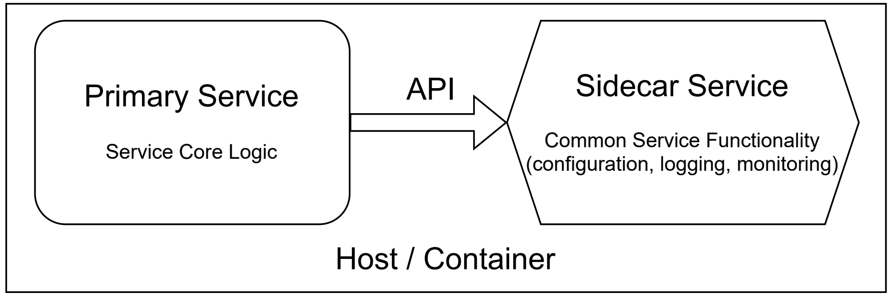
Microservice Architecture: Catalog Example - ***Data Management Services***

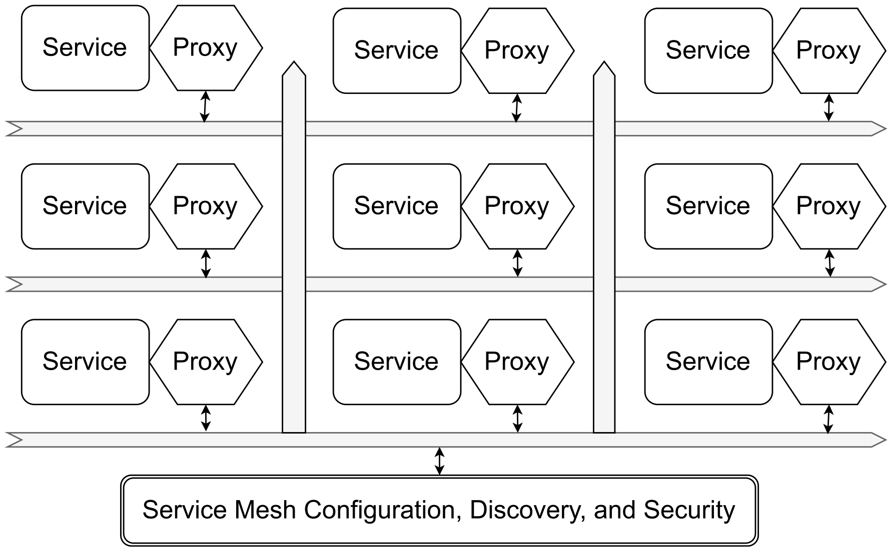
- Data Storage Services
  - File System Storage
  - Key-value Storage
  - Object Storage
  - Relational Database
  - Non-relational Database

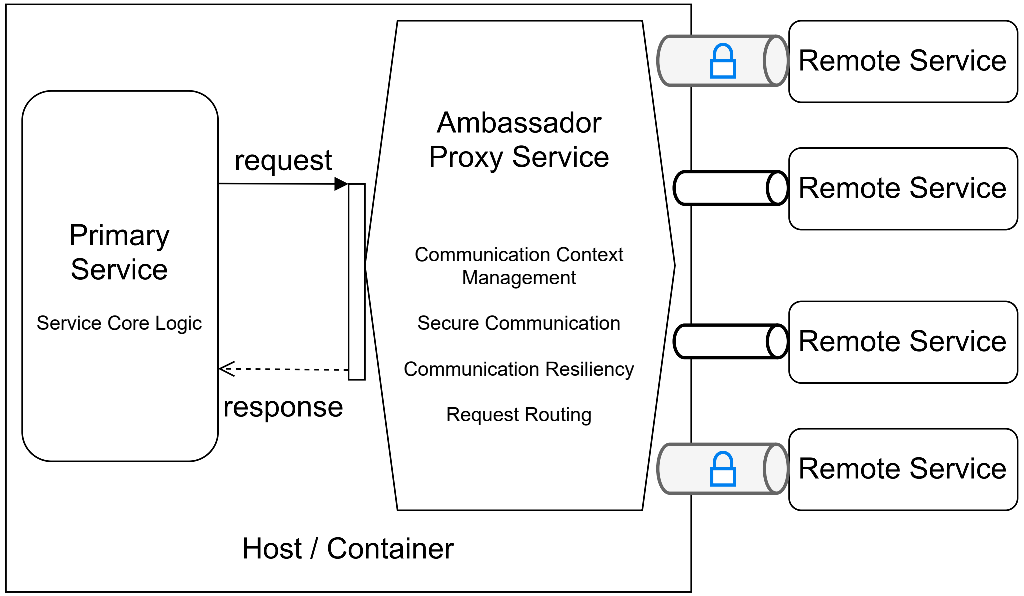
- Data Transfer Services
  - File Transfers
  - Block Data Transfers
  - Streaming Data Transfers
  - Multi-party Data Transfers

Microservice Architecture: Orchestration and Deployment

- Orchestration
  - Asynchronous messaging vs. RESTful services
  - Conductor vs. choreography
- Deployment
  - Sidecar pattern
  - Ambassador Proxy pattern
  - Service Mesh pattern







Status & Future Work

- Refining the INTERSECT Open Architecture and applying it to six initial science use cases at ORNL
    1. automation for electric grid interconnected-laboratory emulation
    2. autonomous additive manufacturing
    3. autonomous continuous flow reactor synthesis
    4. autonomous STEM microscopy
    5. an autonomous robotic chemistry laboratory
    6. an ion trap quantum computing resource
- Publication of the initial INTERSECT Open Architecture specification in the next few months

Questions?

Acknowledgments:

Research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U. S. Department of Energy.