OAK RIDGE National Laboratory

The Interconnected Science Ecosystem (INTERSECT) Architecture

Christian Engelmann

Olga Kuchar, Swen Boehm, Michael Brim, Jack Lange, Thomas Naughton, Patrick Widener, and Ben Mintz

Alumni: Rohit Srivastava, Suhas Somnath, Scott Atchley, and Elke Arenholz

Contact: Christian Engelmann
engelmannc@ornl.gov

ORNL is managed by UT-Battelle LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

INTERSECT Initiative: Exascale System to Ecosystem


Ben Mintz, CS Director
Rob Moore, Exp Director

Goal

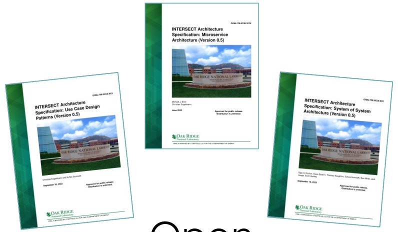
- Develop a scalable, integrated, and interoperable software framework to enable autonomous workflows, experiments, and smart connected ORNL laboratories

Approach

- Develop an open architecture
- Develop and integrate common software frameworks, tools, and services
- Demonstrate use cases to drive and exercise INTERSECT

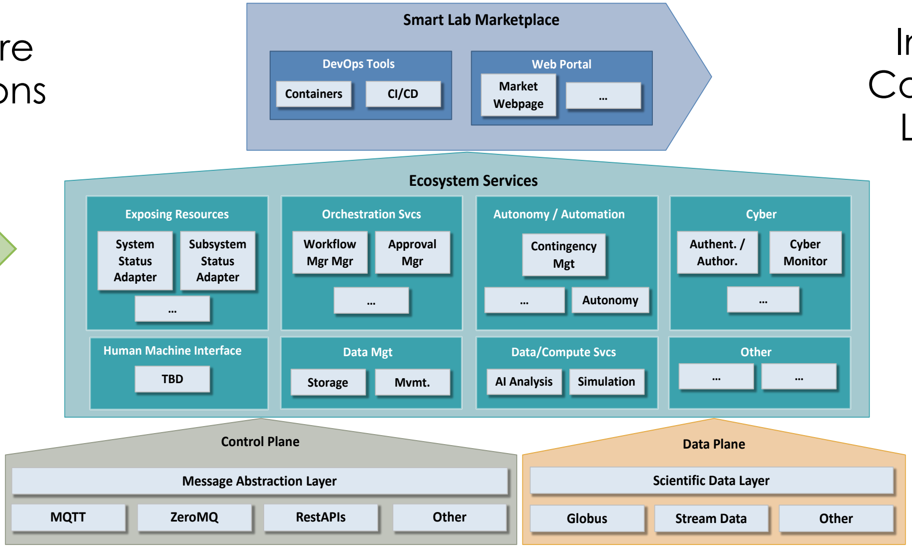
Successes

- 20+ papers submitted and/or accepted
- 10+ software artifacts
- 6 INTERSECT demos including
  - Autonomous Electron Microscope control loop
  - Digital twin for additive manufacturing
  - Automated flow chemistry
- Position ORNL for future DOE ecosystem development

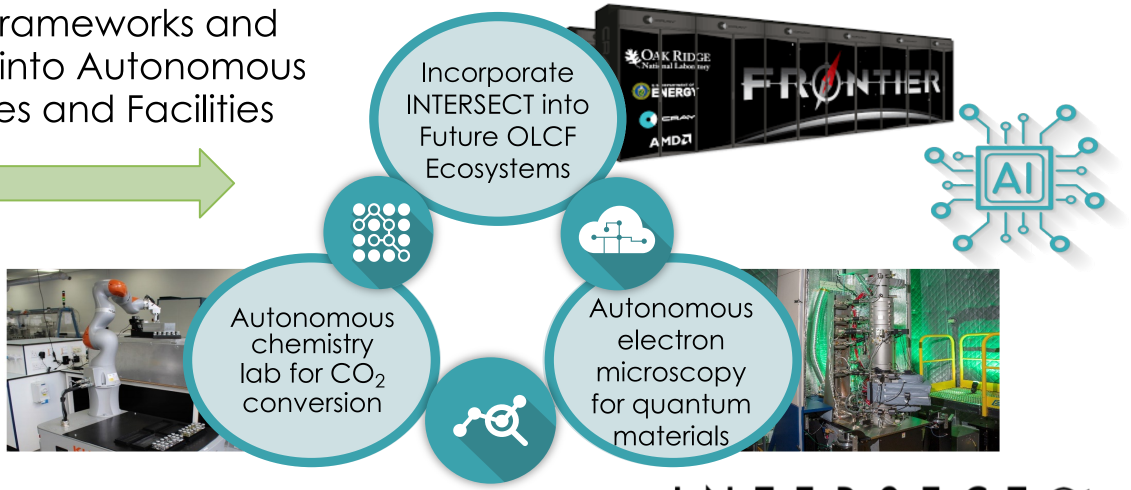


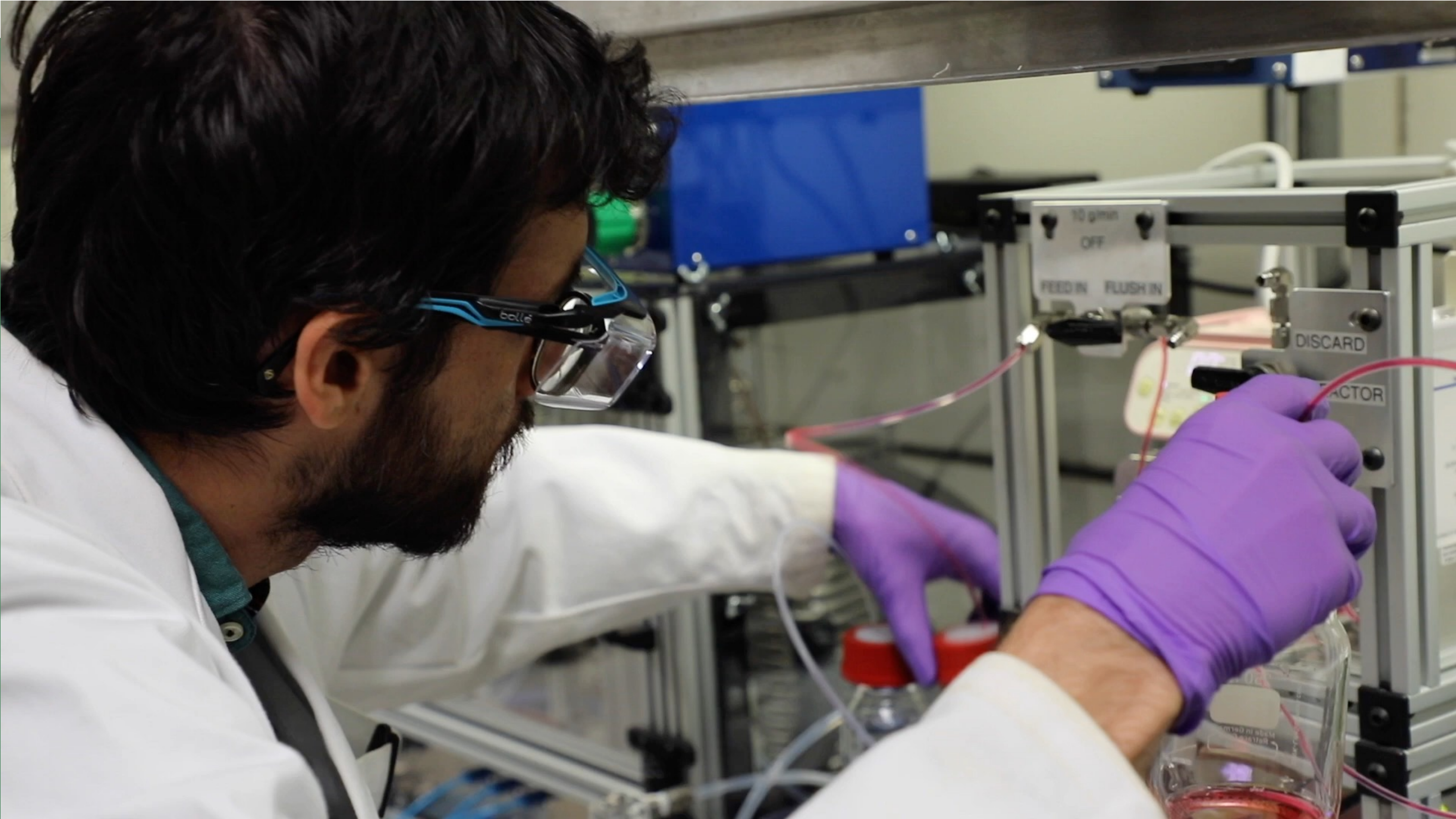
Open Architecture Specifications

Software Development Frameworks and Services

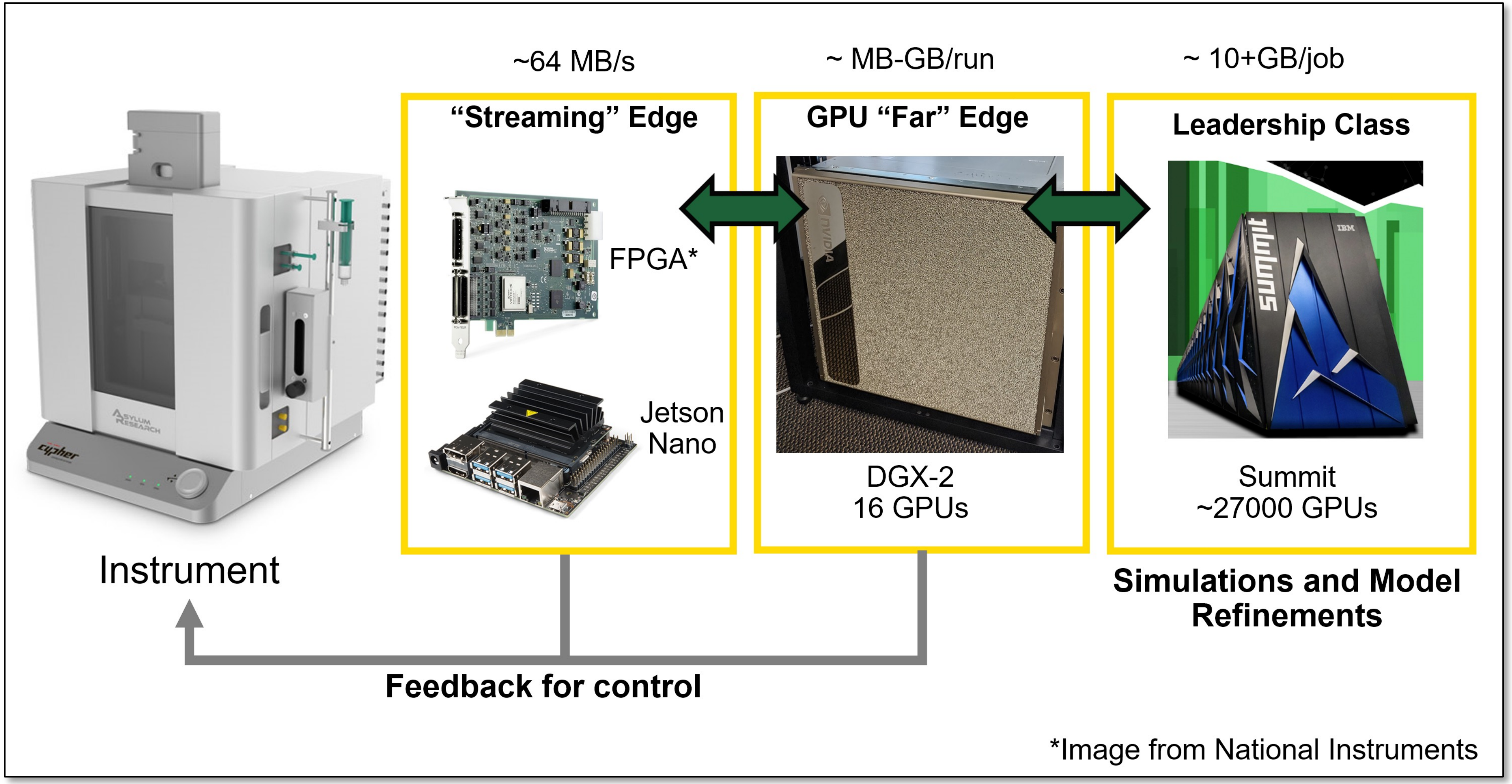


Integrate Frameworks and Capabilities into Autonomous Laboratories and Facilities




10 g/min
OFF
FEED IN
FLUSH IN
DISCARD

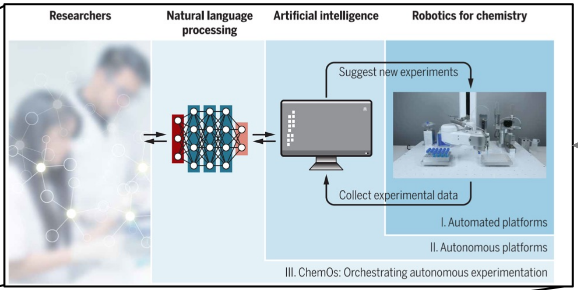
Autonomous Experiments Today



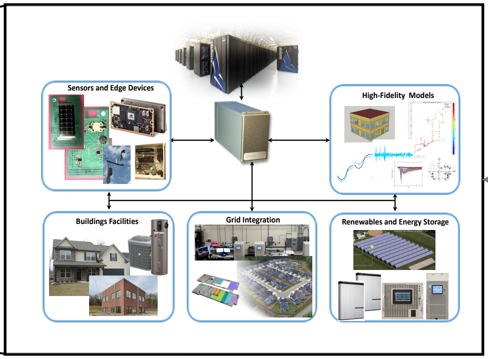
Single Independent Smart Labs

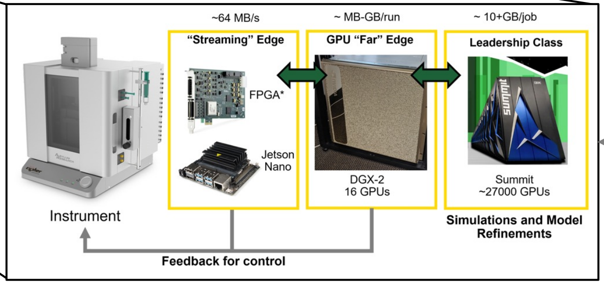
Autonomous Labs of the Future

Autonomous Materials Lab



Autonomous Electric Grid Lab





Autonomous Microscopy Lab

Interconnected Smart Labs

Interoperable Ecosystem is Required

Scientist Really Want ...

Hey Google

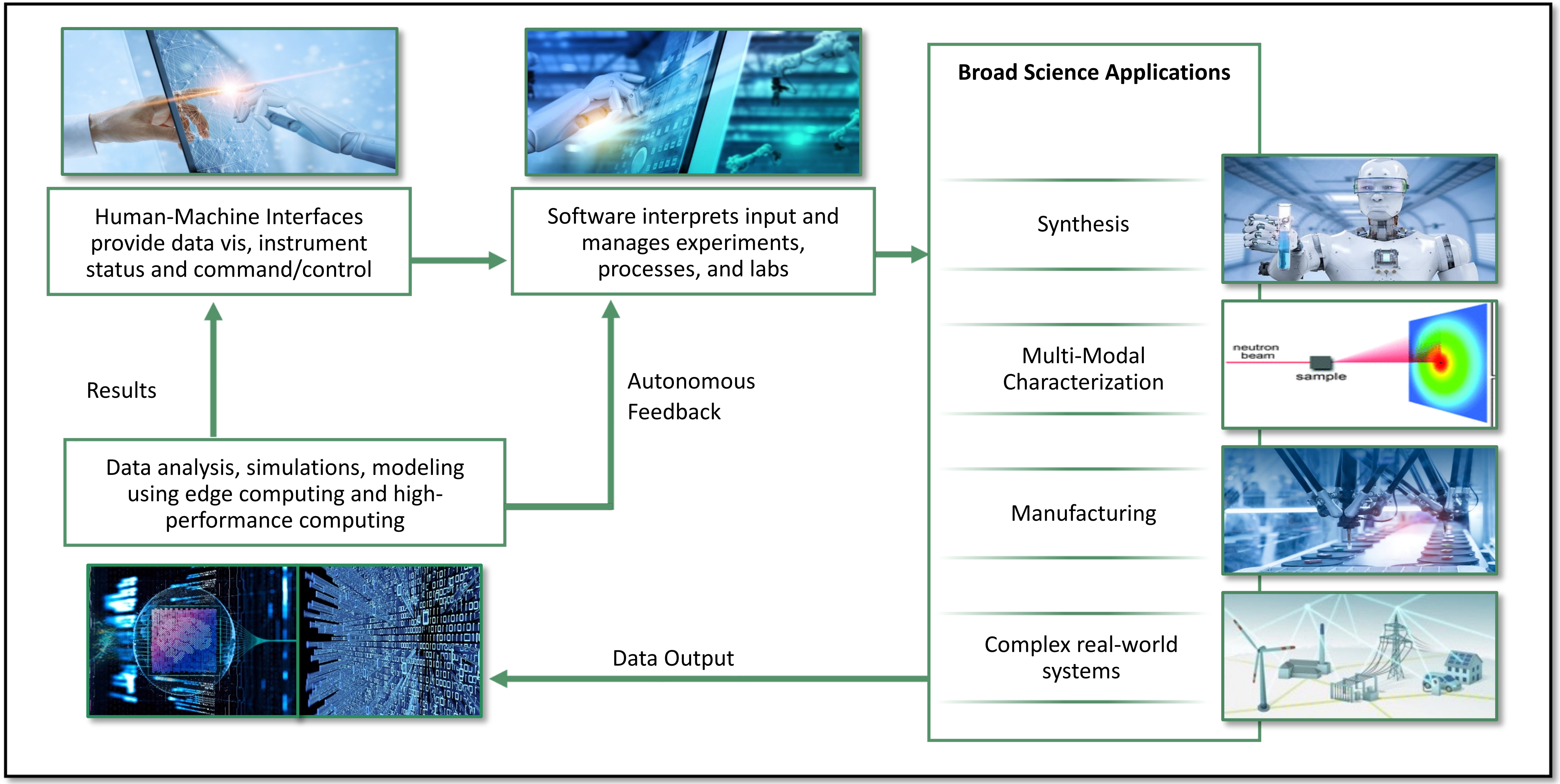
alexa

Siri

“Hey Oakley,
Help me solve a science problem!”

Bixby

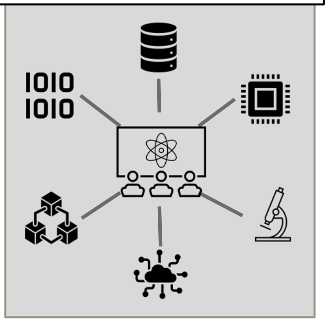
Reality of Autonomous Laboratories



Ecosystem solutions must be <u>Scalable</u>, <u>Flexible</u>, and <u>Interoperable</u>

INTERSECT Software Ecosystem

Interoperable Autonomous Labs



Smart Lab Marketplace

Development and Operations (DevOps) Env.
- Replicates operational environment for sandbox software development

Adopter's Web Portal
- Easy access to software capabilities

DevOps Tools
- Containers
- CI/CD

Web Portal
- Market Webpage
- ...

Ecosystem Software Services

Abstract Service Bus and Common Messages
- System and Software Interoperability
- Software Reuse

Microservice Architecture
- Breaks Monolithic Software
- Incremental Software Development and/or Updates
- Reuse Individual Services

Exposing Resources
- System Adapter
- Subsystem Adapter

Orchestration
- Workflow Manager
- Approval Manager

Autonomy / Automation
- Contingency Manager
- Autonomy

Cybersecurity
- Authent. / Author.
- Cyber Monitor

Human Machine Interface
- Web Portal

Data Management
- Storage
- Movement

Data Analysis/Computation
- AI Analysis
- Simulation

Other
- ...
- ...

Abstract Service Bus

Abstracts Protocols and Networks
- Rapid Integration of New Technology with Limited Software Rewrite

Standard Requirements
- Interoperability Across Implementations

Control Plane
- Message Abstraction Layer
- MQTT
- RestAPIs
- ...

Data Plane
- Scientific Data Layer
- Globus
- Stream Data
- ...

INTERSECT Programmatic Structure

Domain Science Projects

Autonomous Chemistry Lab	AutoflowS	Autonomous Microscopy	Additive Manufacturing	Quantum Accelerator	Electric Grid
Establish an **autonomous robotic chemistry lab** for catalytic synthesis that operates 24/7	**Develop an autonomous flow chemistry system** by combining in-situ analysis capabilities with AI enabled feedback	Establish **data streaming, on-the-fly data analysis and simulation for AI enabled feedback** for microscopes at CNMS	**Autonomous additive manufacturing (AM)** by combining AM build system, in-situ analysis, and on-the-fly simulations	Integrate a **trapped ion quantum resource** into the INTERSECT ecosystem for use as a quantum accelerator	Establish a scalable platform for hardware in the loop **emulation of large-scale power grids** to test new power controllers

Crosscutting Projects

INTERSECT Architecture	Develop an **open architecture** that is scalable across scientific domains
INTERSECT Software Development Framework	Develop the **software framework and tools** required to interconnect systems
INTERSECT Integration	**Integrate** networks, systems, and software across all projects

INTERSECT Architecture Overview

INTERSECT Architecture

Science Use Case Design Patterns

- Strategy patterns
  - Experiment control
  - Experiment steering
  - Design of experiments
  - Multi-experiment workflow
- Architectural Patterns
  - Local vs. distributed

System of Systems Architecture

- User view
- Data view
- Logical view
- Physical view
- Operational view
- Standards view

Microservice Architecture

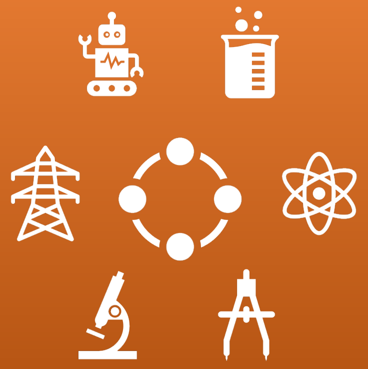
- Interaction patterns
- Capabilities catalog
- Orchestration and deployment patterns

INTERSECT Software Development Kit

INTERSECT Integration

Interconnected Smart Laboratories

- Automated/ Autonomous
- Robot-controlled
- Chemistry
- Microscopy
- Additive Manufacturing
- Electric Grid Emulation



The INTERSECT Open Architecture Specification

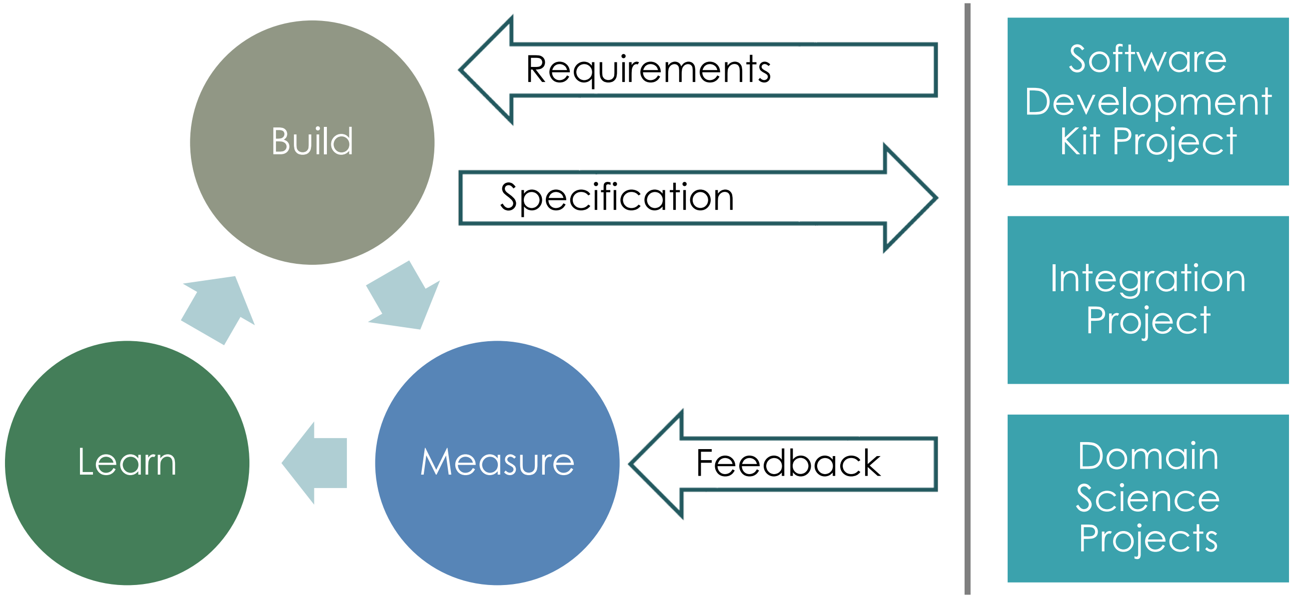
A written documentation of the INTERSECT Architecture, like a blueprint

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

- Current approach: 3 reports (PDF) released in intervals

Agile Development of the INTERSECT Architecture

- Iteratively develop and refine the INTERSECT Architecture
- Interact with the Software Development Kit, Integration and Domain Science Projects for
  - Requirements analysis
  - Feedback on drafts and releases
  - Assuring architecture compliance
  - Understanding implementation nuances



Fine-Grain Cycle for Specification Document Draft
Coarse-Grain Cycle for Specification Document Release

Science Use Case Design Pattern Specification

- Abstract descriptions of the involved hardware and software components and their work, data and control flows.

ORNL/TM-XXXX/XXX

INTERSECT Architecture Specification: Use Case Design Patterns (Version 0.5)

Christian Engelmann and Suhas Somnath

September 30, 2022

Approved for public release. Distribution is unlimited.

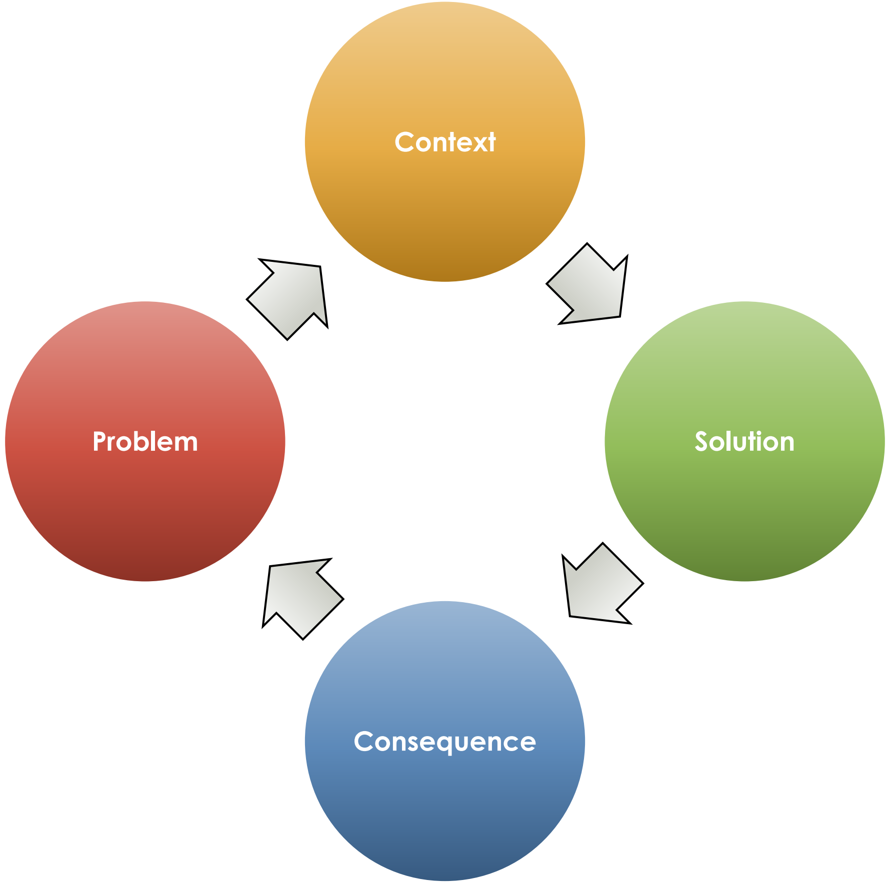
OAK RIDGE National Laboratory

CONTENTS

OAK RIDGE National Laboratory

Why Design Patterns?

- Systematize recurring problems by describing generalized solutions based on best practices
- Offer solution templates to solve specific problems that may apply to different situations
- Provide different solution alternatives to specific problems
- Identify the key aspects of solutions and create abstract descriptions to develop reusable design elements
- Communicate problems and solutions with clear terms and abstract concepts



Science Use Case Design Patterns: Anatomy

- ***Approach: Focus on the control problem***
  - Open vs. closed loop control
  - Single vs. multiple experiment control
  - Steering vs. designing experiments
  - Local vs. remote compute in the loop
- 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*

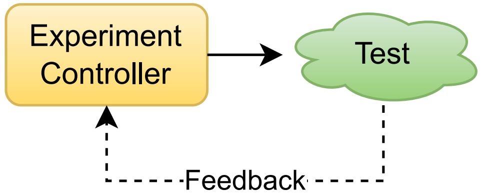


Figure: Single experiment control

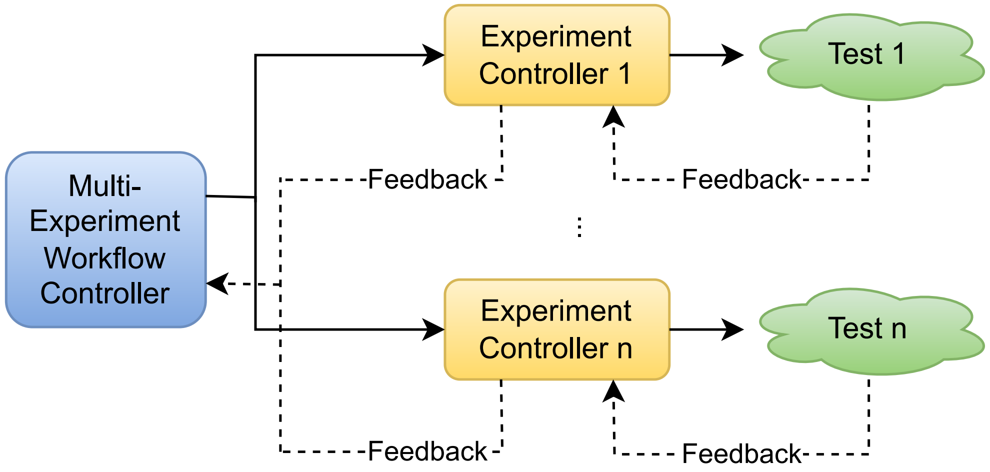


Figure: Multi-experiment control

Science Use Case Design Patterns: Classification

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

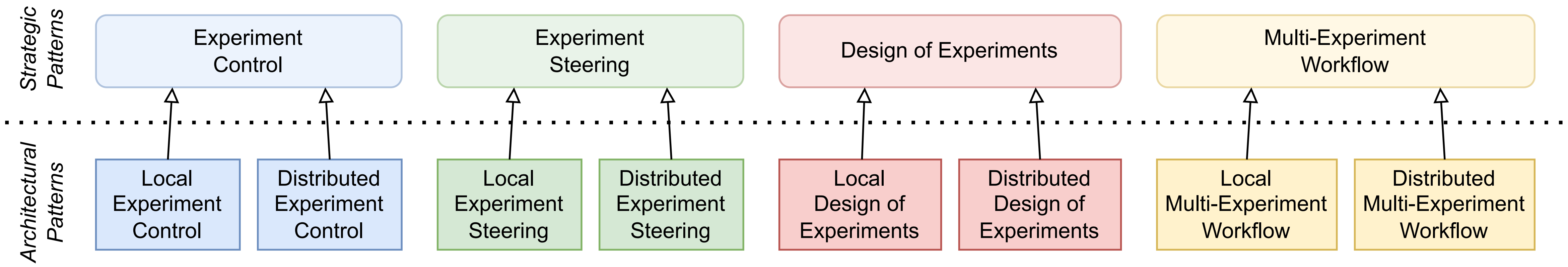
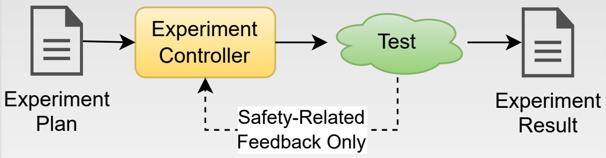


Figure: Pattern classification scheme

Science Use Case Design Patterns: Strategic Patterns

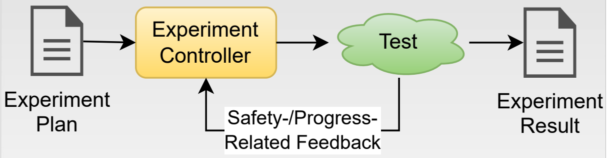
Experiment Control



Executes an existing plan

- Open loop control
- Automated operation

Experiment Steering

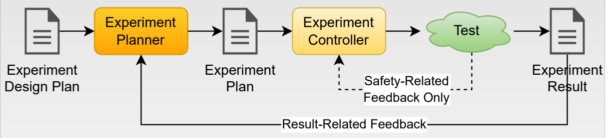


Executes an existing plan, depending on progress

- Closed loop control
- Autonomous operation

- Extends patterns:
    - *Experiment Control*

Design of Experiments



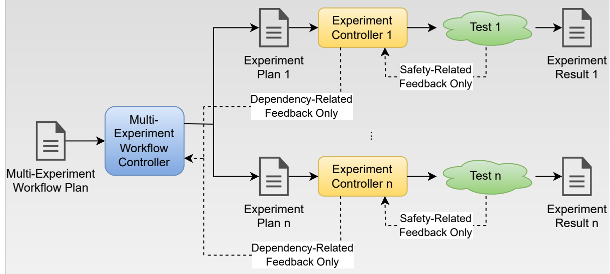
Creates/executes a plan, based on prior result

- Closed loop control
- Autonomous operation

- Uses patterns:
    - *Experiment Control*

- May use patterns:
    - *Experiment Steering*

Multi-Experiment Workflow



Executes existing plans (workflow of 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
Local vs. Distributed Experiment Control

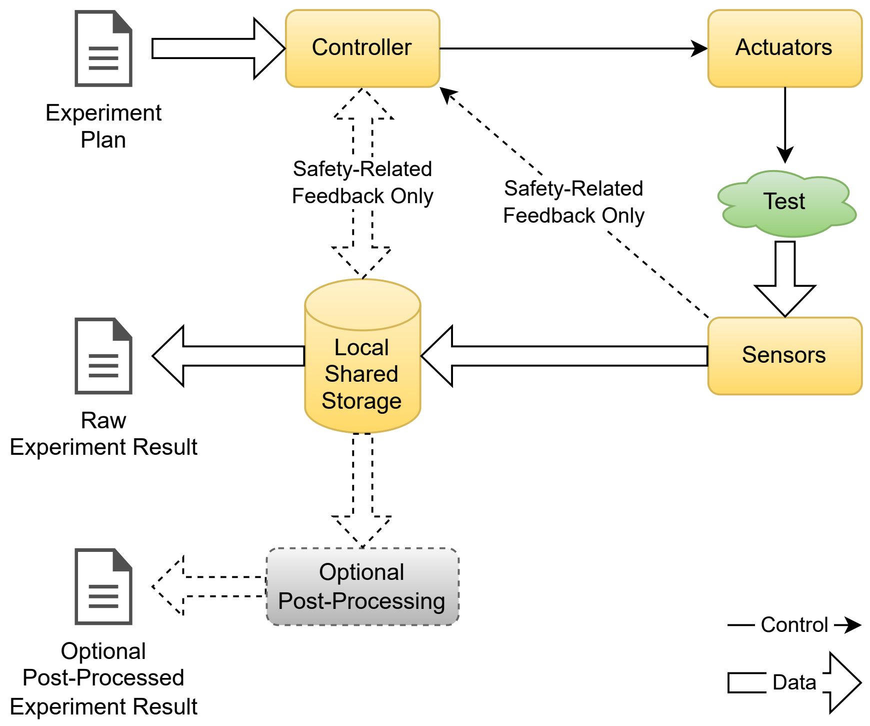


Figure: Local Experiment Control

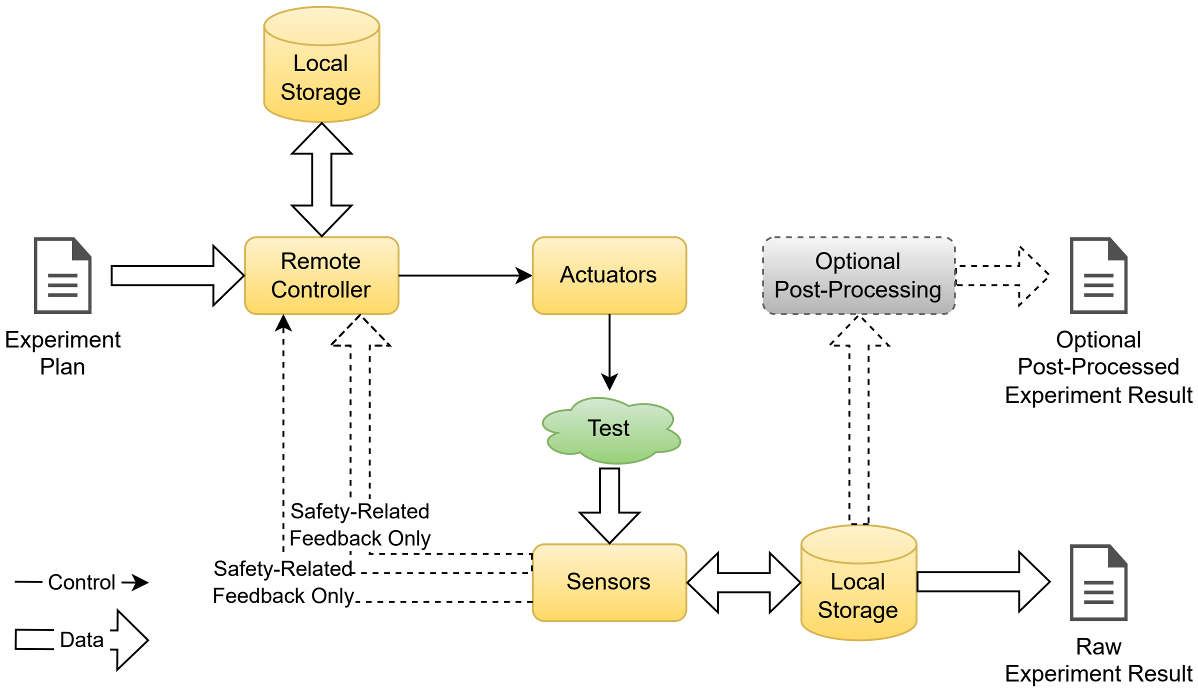


Figure: Distributed Experiment Control

Science Use Case Design Patterns: Architectural Patterns
Local vs. Distributed Experiment Steering

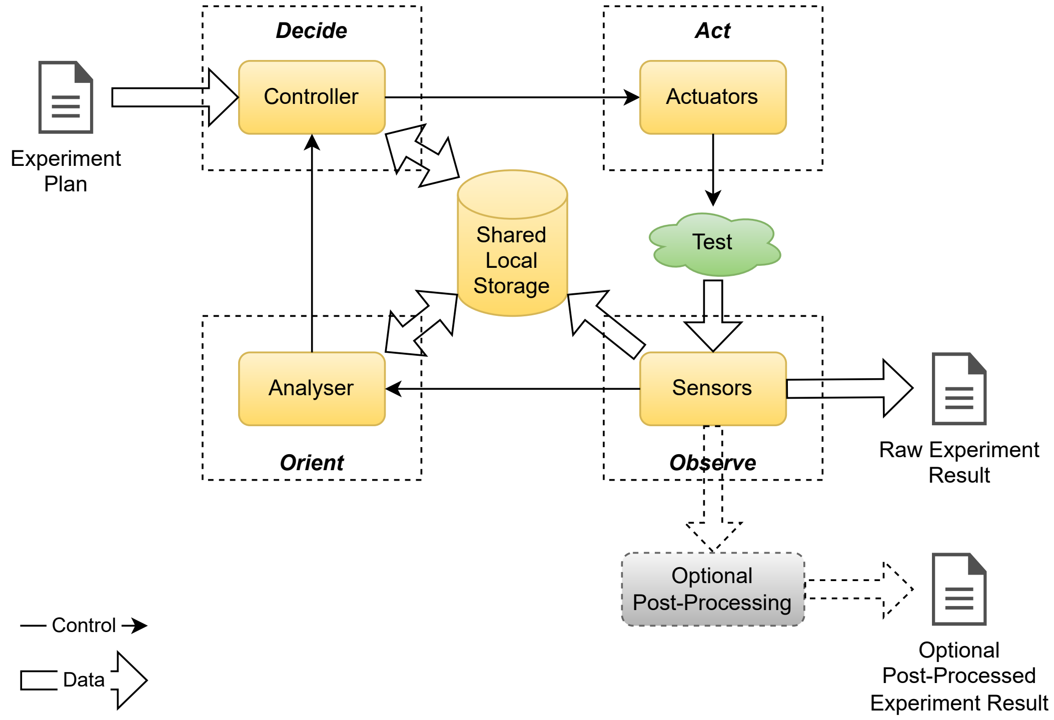


Figure: Local Experiment Steering

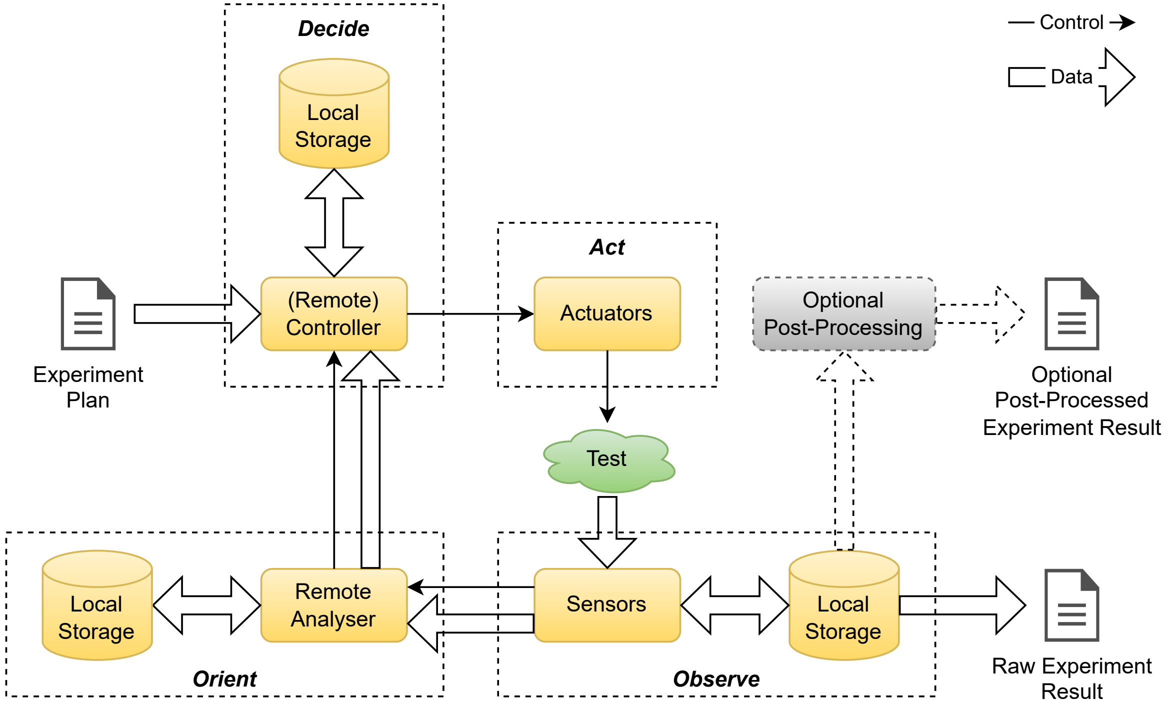


Figure: Distributed Experiment Steering

Science Use Case Design Patterns: Architectural Patterns
Local vs. Distributed Design of Experiments

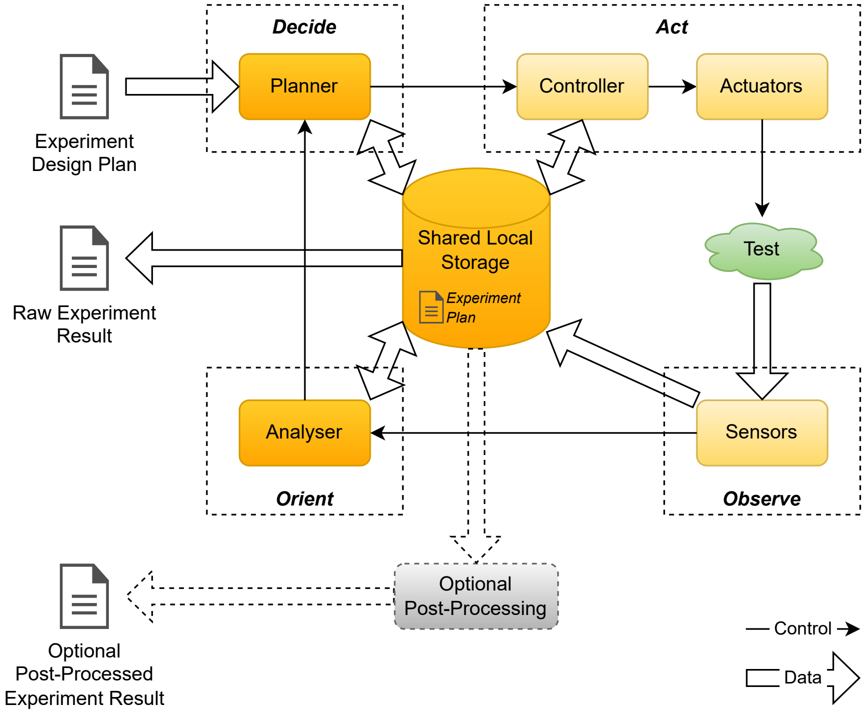


Figure: Local Design of Experiments

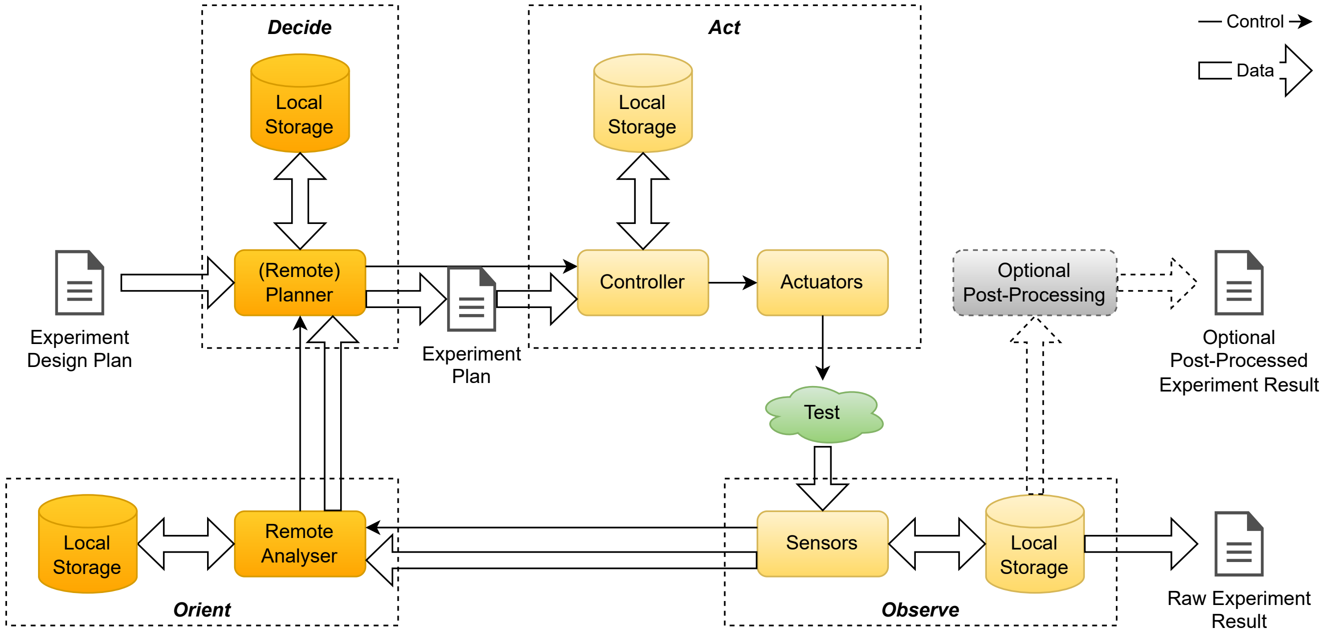


Figure: Distributed Design of Experiments

Science Use Case Design Patterns: Architectural Patterns
Local vs. Distributed Multi-Experiment Workflow

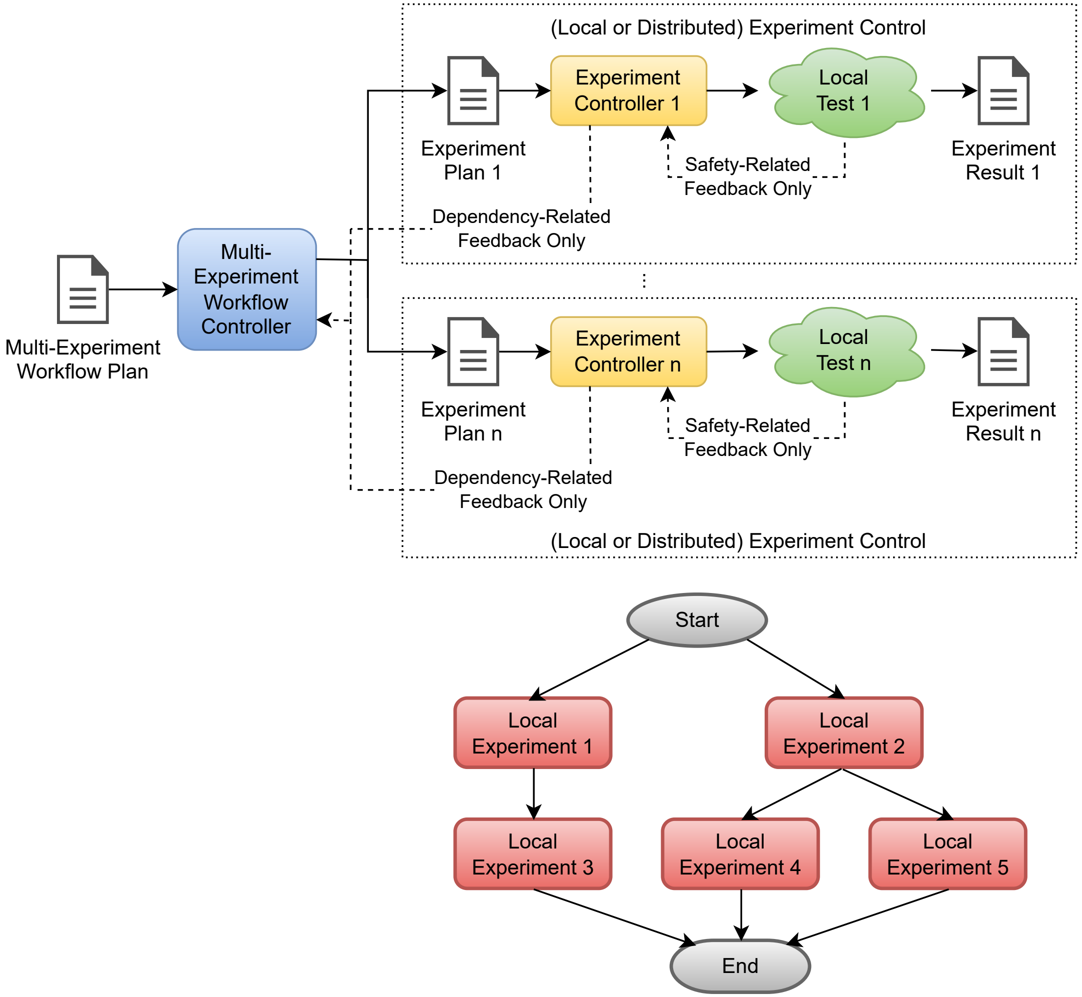


Figure: Local Multi-Experiment Workflow

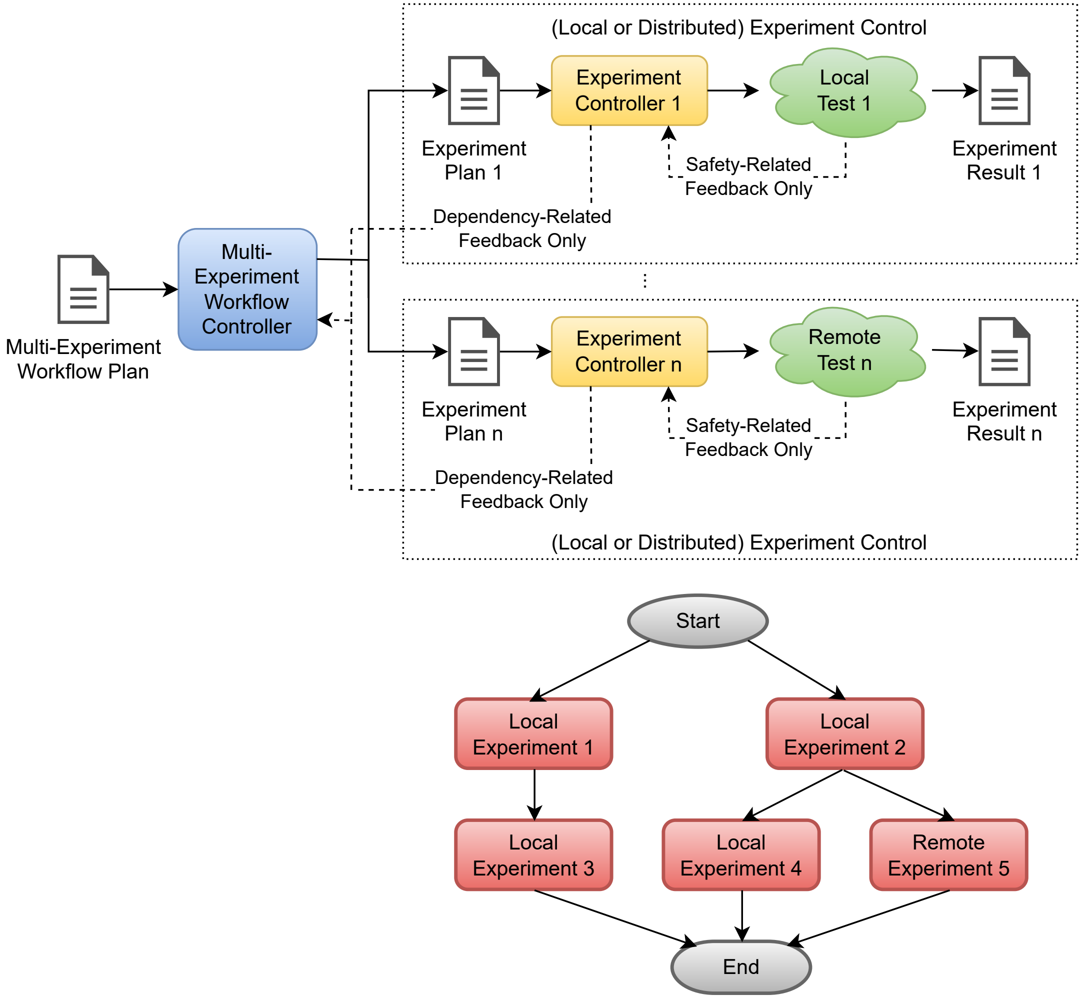


Figure: Distributed Multi-Experiment Workflow

Science Use Case Design Patterns: Compositions

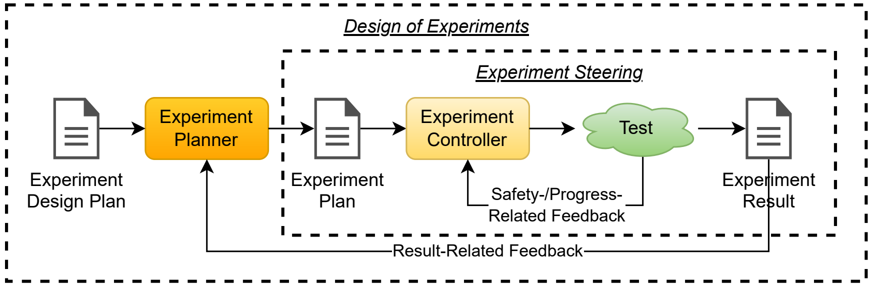


Figure: Strategic pattern composition

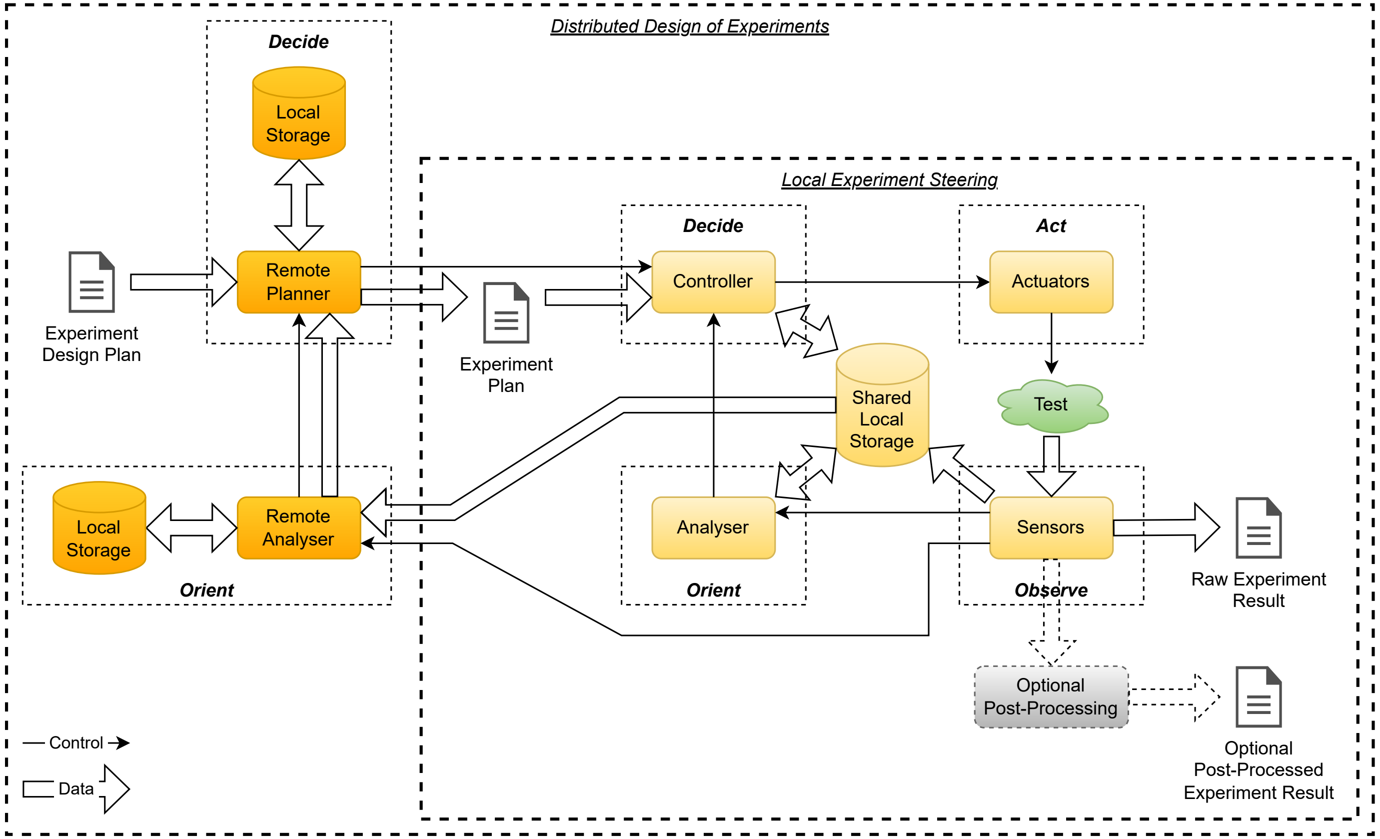


Figure: Architectural pattern composition

System of Systems Architecture Specification

- Detailed design decisions about the involved hardware and software components from different points of view.

ORNL/TM-XXXX/XXX

INTERSECT Architecture Specification: System of System Architecture (Version 0.5)

Olga A. Kuchar, Swen Boehm, Thomas Naughton, Suhas Somnath, Ben Mintz, Jack Lange, Scott Atchley

September 16, 2022

Approved for public release. Distribution is unlimited.

OAK RIDGE National Laboratory

ORNL IS MANAGED BY UT-BATTELLE LLC FOR THE US DEPARTMENT OF ENERGY

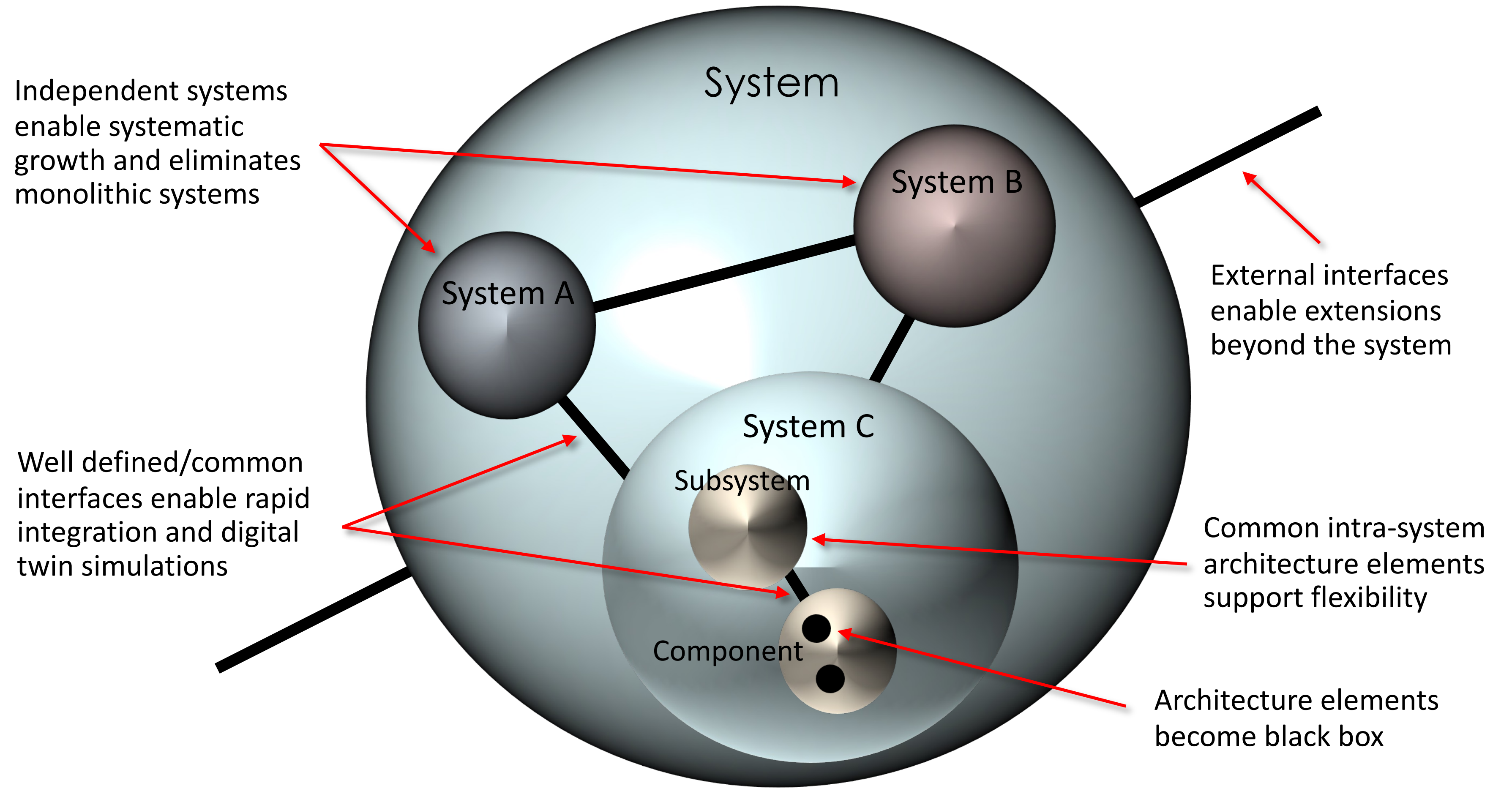
CONTENTS

iii

OAK RIDGE National Laboratory

Why System of Systems?

Common Architecture Elements



Common Messages

System

SystemStatus
SystemControlStatus
SystemControlRequest
SystemControlRequestStatus
SystemTask
SystemTaskStatus

Subsystem

SubsystemStatus
SubsystemControlRequest
SubsystemControlRequestStatus
X_Capability
X_CapabilityStatus
X_CapabilityCommand
X_CapabilityCommandStatus
X_CapabilityActivity

Component

ComponentControlStatus
ComponentCommand
ComponentCommandStatus

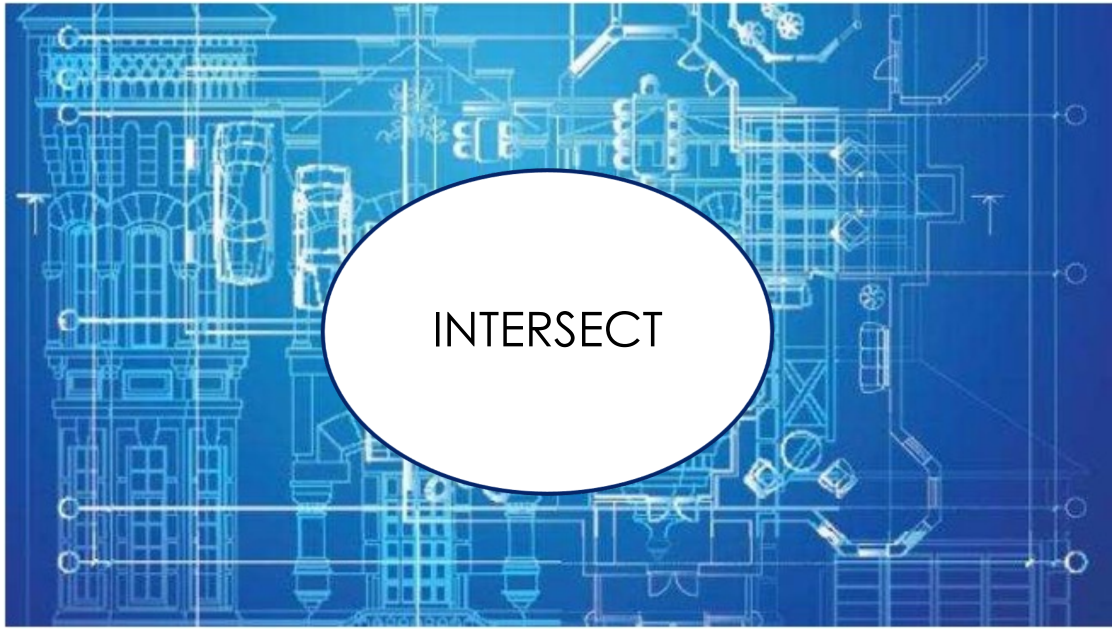
Enable Scalable, Flexible, and Interoperable Development, Deployment and Operation

System of Systems Architecture Views

User View

Data View

Operational View



Logical View

Physical View

Standards View

System of Systems Architecture: Stakeholder Roles

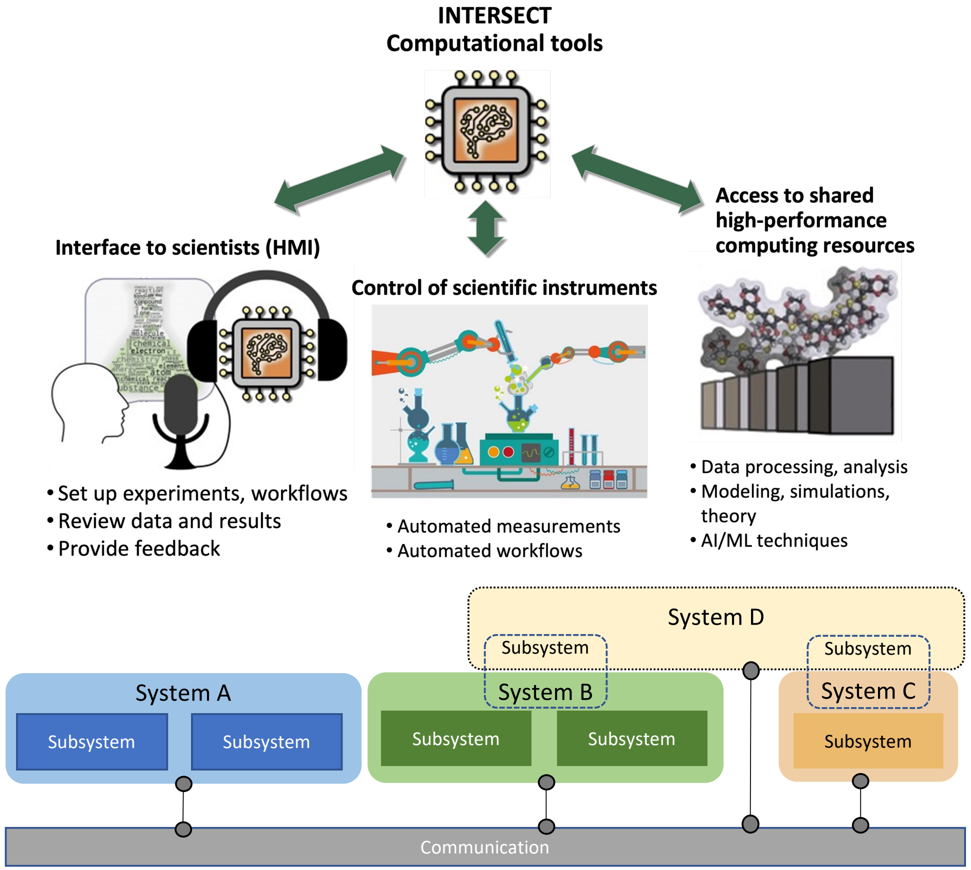


Figure: A federated ecosystem for autonomous experiments and self-driving labs with a system of systems architecture

Table 1-1. Stakeholder roles and the views within this document.

	View Chapters					
Role	**User**	**Data**	**Operational**	**Logical**	**Physical**	**Standards**
Application software developers	X	X		X		X
Infrastructure software developers		X	X		X	X
End users	X			X		
Application and platform hardware engineers						
Security Engineers		X	X		X	
Communications engineers		X				X
System-of-system engineers		X	X	X	X	X
Chief engineer/scientists	X	X		X		X
Lead System Integrator		X	X		X	
System Integration and test engineers	X	X	X	X		
External test agencies	X	X	X	X	X	
Operational system managers	X	X	X			

System of Systems Architecture: Logical View

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

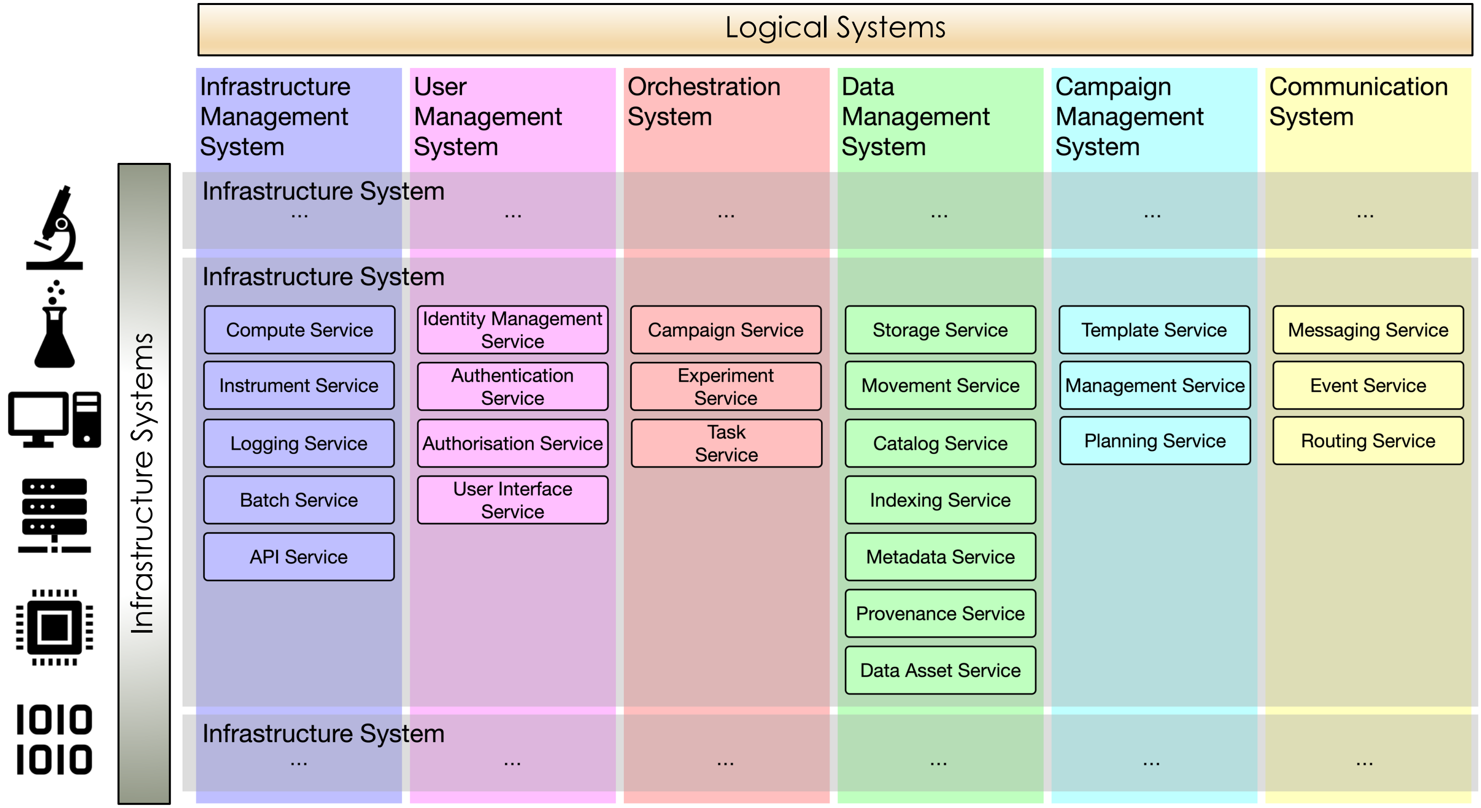


Figure: Relationships between infrastructure and logical systems and their services

System of Systems Architecture: Data View

- ***Highlights the system's data needs and framework***
- Includes data flow between systems and data definitions, schemas and exchange sequence diagrams
- Does not include specifications for scientific, instrument, or experiment data

Entity Name	Description
User	A user of an INTERSECT-compliant system or application. May participate in authentication or authorization processes.
User Profile	Profile information (contact/address/miscellaneous) for an INTERSECT user.
Project	Accounting abstraction for resource allocation in an INTERSECT system.
Campaign	A collection of related experimental activity which uses INTERSECT resources. A Campaign is associated with a Project and may have multiple Users associated with it. Campaigns have explicit durations and discrete sets of resources assigned to them.
Campaign Result	Outcomes of INTERSECT Campaigns. There may be several different result states represented.
Campaign Error	"Error" outcomes for INTERSECT Campaigns. As with Campaign Result, there may be several different "flavors" of error/failure results.
Campaign Template	It may prove useful to memoize a Campaign structure as a template, so that it may be quickly replicated by users. Such repllicated new Campaigns are assigned the tamplated INTERSECT resources.
Recipe	Users may also wish to reuse resource structures at a finer granularity than Campaign. Recipies allow this usage to be memoized.
Approved User Resources Approved Administrator Resources Approved Operator Resources	Resource allocations are tracked with approval durations for each of Users, Administrators, and Operators.
INTERSECT Resource Type	Additional information about an INTERSECT resource.
INTERSECT Resource Action	Detail on the operations/functions available from a given INTERSECT resource.
INTERSECT Resources	Experimental/physical, computational, or virtual facilities available within the INTERSECT system or application.
Computational Resource	Additional information about computational resources available to the INTERSECT system or application.
Resource Support	An INTERSECT resource may be large and complex, requiring specialized support procedures and/or personnel for operation. Computational resources, for example, may have multiple such support staff, organized into tiers or functional areas.
Resource Capability	Resources provide INTERSECT capabilities, which allow them to be composed into systems and applications within the INTERSECT Architecture.

Table 6-1. Names and descriptions of INTERSECT architecture data entities

System of Systems Architecture: Operational View

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

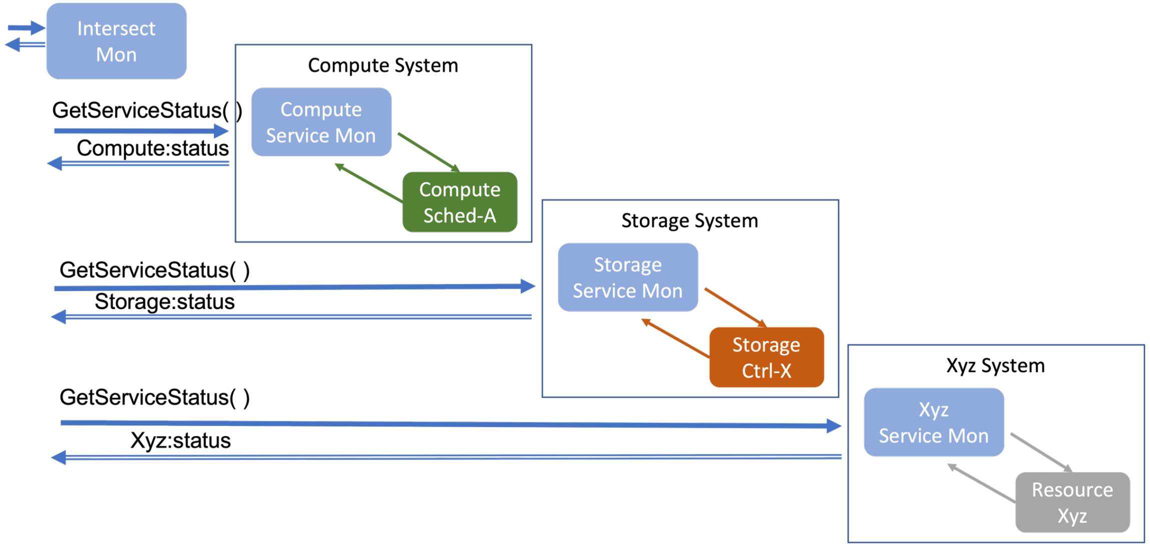


Figure: Components, interfaces, and message sequences involved in system status monitoring

System of Systems Architecture: Physical View

- ***Captures 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
- <u>Proprietary information is not part of the open architecture documentation!</u>

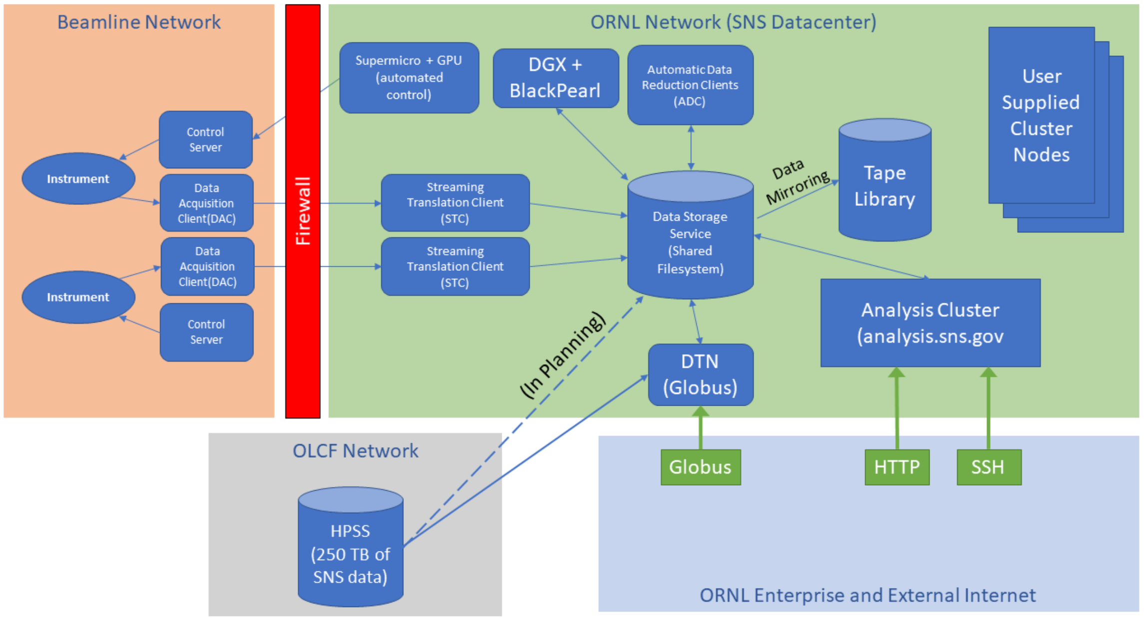
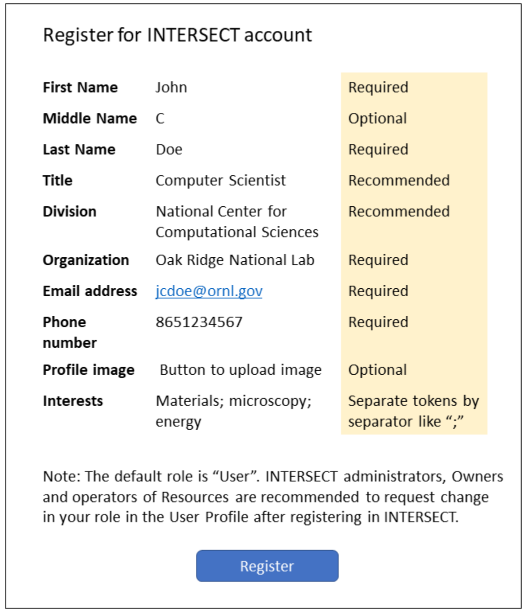


Figure: Schematic representation of resources at Oak Ridge National Laboratory's Spallation Neutron Source

System of Systems Architecture: User View

- ***Captures user-facing <u>functionality</u>***
- Does not include system-internal interactions
- Described activities:
  - Logging into dashboard
  - Experiment creation
  - Start experiment
  - Steer experiment
  - Experiment end
- Includes <u>examples</u> for graphical user interfaces



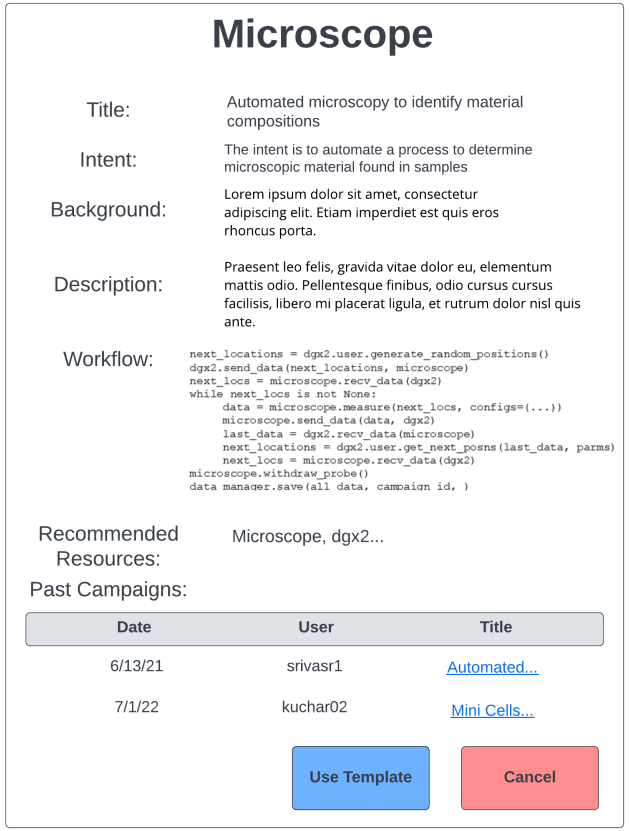


Figure: Examples of graphical user interfaces for different user interactions

System of Systems Architecture: Standards View

- ***Captures the 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 Specification

- Detailed design decisions about software microservices, including their functionalities, capabilities, compositions, with control, work, and data flows.

ORNL/TM-XXXX/XXX

INTERSECT Architecture Specification: Microservice Architecture (Version 0.5)

Michael J. Brim
Christian Engelmann

June 2022

Approved for public release.
Distribution is unlimited.

OAK RIDGE National Laboratory

ORNL IS MANAGED BY UT-BATTELLE LLC FOR THE US DEPARTMENT OF ENERGY

CONTENTS

iii

OAK RIDGE National Laboratory

Microservice Architecture: Microservice Capabilities

- System consists of
  - Subsystems, resources, and services
- Subsystem consists of
  - Services and resources
- Service consists of
  - Microservice capabilities

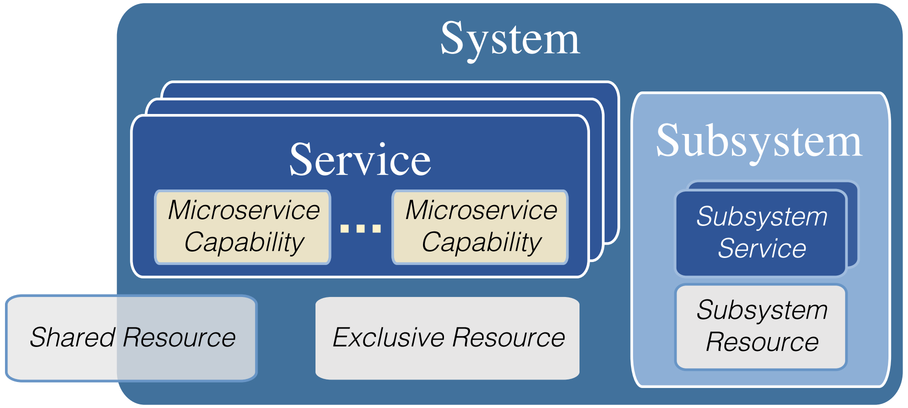


Figure: Systems, subsystems, services, and microservices

Capability: Unique Capability Name

Description: A short summary description of the domain of interest for this capability and the provided functionality.

Related Capabilities: Where applicable, provides references to related capabilities.

- *Extends*: A list of base capabilities that the functionality of this capability extends. A service implementing this capability must also implement the base capabilities.
- *Requires*: A list of required capabilities that are necessary to implement the functionality of this capability. The required capabilities are most often provided by other services, but may be implemented in the same service.

Custom Data Type: Where applicable, provides definitions of new data types or structures.

Interactions: Command

- `MethodName()`
  Purpose: A short description of the purpose of the current command method.
  Command Data: A list of input data for the current method formatted as:
  - `dataName` (`DataType`) : A description of the data, including any format or value constraints.

Interactions: Request-Reply

- `MethodName()`
  Purpose: A short description of the purpose of the current request method.
  Request Data: A list of input data for the current method formatted as:
  - `dataName` (`DataType`) : A description of the data, including any format or value constraints.

  Reply Data: A list of output data for the current method formatted as:
  - `dataName` (`DataType`) : A description of the data, including any format or value constraints.

Interactions: Asynchronous Event

- `EventName`
  Purpose: A description of the activity or state change that generates this event.
  Event Data: A list of data for the current event formatted as:
  - `dataName` (`DataType`) : A description of the data, including any format or value constraints.

Figure 3-1. Microservice Capability Definition Format

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
  - Microservice architecture does not force a specific implementation

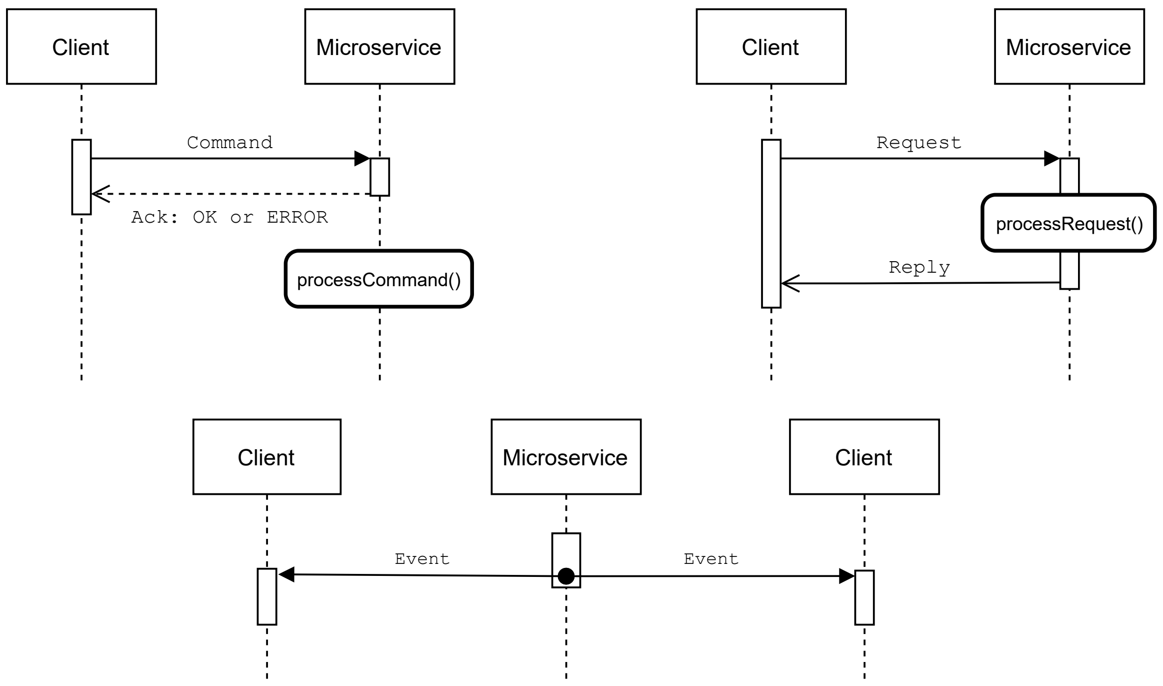


Figure: Command/acknowledgement, request/reply and asynchronous event interaction patterns for microservices

Microservice Architecture: Capabilities Catalog

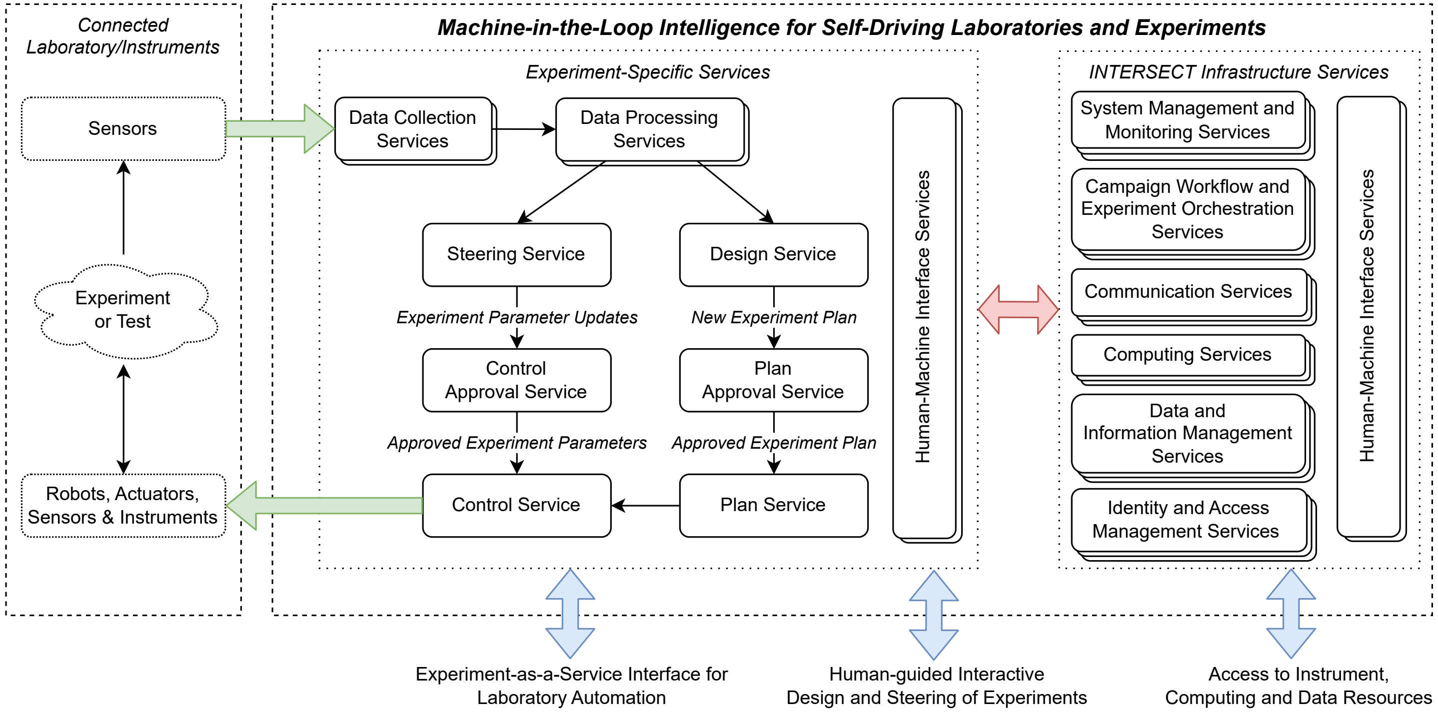


Figure: Experiment-specific and infrastructure services in the context of autonomous experiments and self-driving laboratories

- Example: Data Management
  - Data Transfer
    - File Transfer
    - Block Data Transfer
    - Streaming Data Transfer
    - Multi-party Data Transfer
  - Data Storage
    - File System Storage
    - Key-value Storage
    - Object Storage
    - Relational Database
    - Non-relational Database
  - …

Microservice Architecture: Orchestration and Deployment

- Microservice orchestration
  - Asynchronous messaging or/and RESTful services
  - Conductor vs. choreography
- Microservice deployment
  - Sidecar pattern, Ambassador Proxy, and Service Mesh deployment patterns

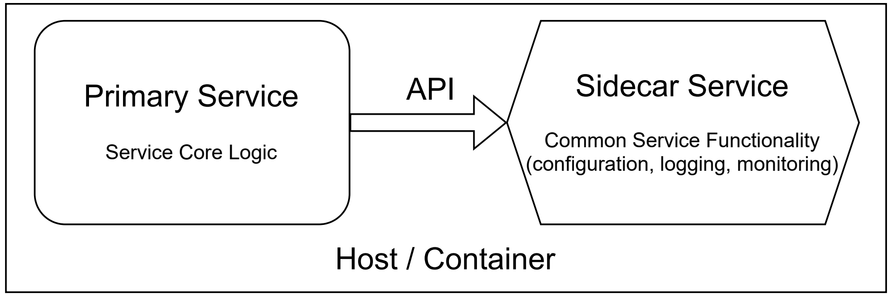


Figure: Sidecar deployment pattern

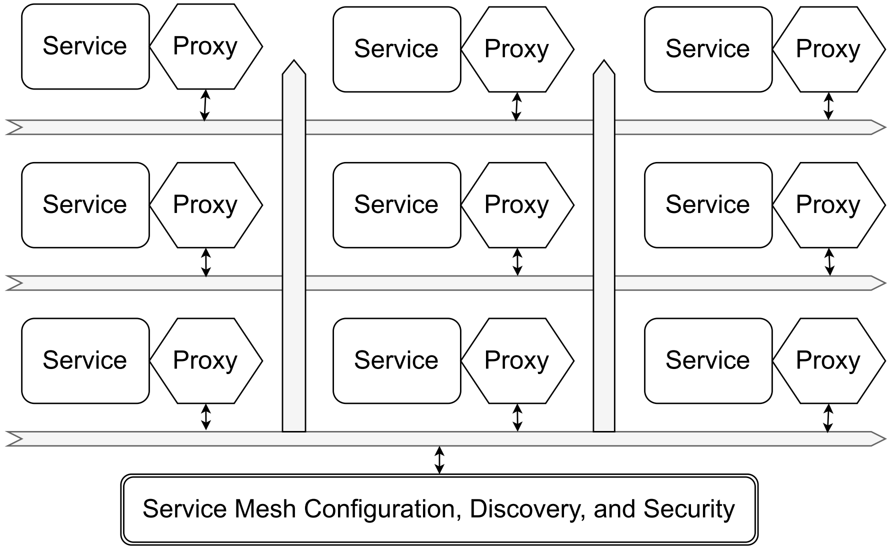


Figure: Service Mesh deployment pattern

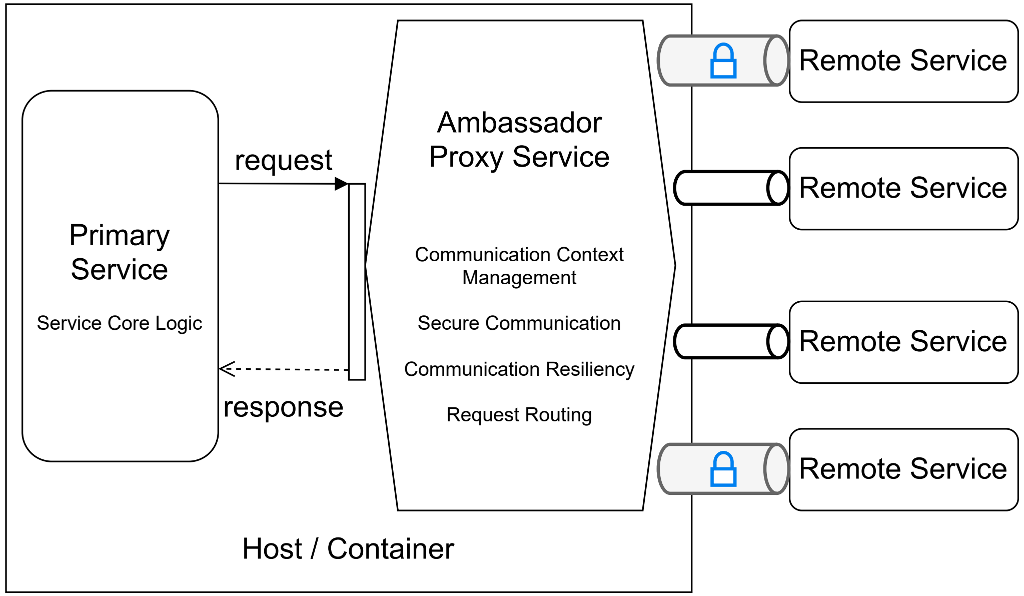
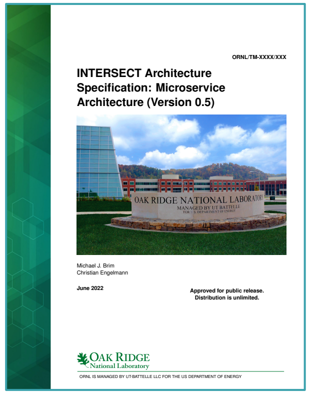


Figure: Ambassador Proxy deployment pattern

Current Status

- ***INTERSECT Open Architecture Specification***
  - Design pattern catalog that covers the science use cases in the INTERSECT Initiative
  - System-of-systems architecture specification with elements, communication and interfaces and some command and control and resource triad specifications
  - Initial microservice architecture that covers some INTERSECT science use cases
- ***v0.5 released as ORNL reports in 9/2022 (v0.9 to be released soon)***
  - INTERSECT Architecture: Use Case Design Patterns
  - INTERSECT Architecture: System of Systems Architecture
  - INTERSECT Architecture: Microservices Architecture



Future Roadmap: Capabilities to be Targeted

- **Campaign orchestration** (distributed and federated) and management templates (workflow repository)
- **Data plane architecture** (storage, movement, catalog, indexing, metadata, provenance, and asset management)
- **Standards view**: Requirements for INTERSECT and domain-specific standards (APIs, messages, and data formats)
- Architecture support for multi-tenancy (**multi-user**) and federation (**multi-site**)
- **Distributed and federated monitoring** architecture (for reliability, availability, serviceability and cybersecurity)
- **Error handling** concepts and interfaces (detection, notification, and isolation)
- **Resilience** concepts and interfaces (error/failure detection, notification, and mitigation)
- **Cybersecurity** architecture, including identity management adapters and access controls
- INTERSECT **documentation portal** targeting different audiences (e.g., developers and users)
- **Architecture for graphical user interfaces** that are independent from the business logic
- **INTERSECT as part of ORNL's Integrated Research Infrastructure**

INTERSECT Architecture Demonstration

ORNL/TM-XXXX/XXX

INTERSECT Architecture Specification: Use Case Design Patterns (Version 0.5)

CONTENTS

ORNL/TM-XXXX/XXX

INTERSECT Architecture Specification: System of System Architecture (Version 0.5)

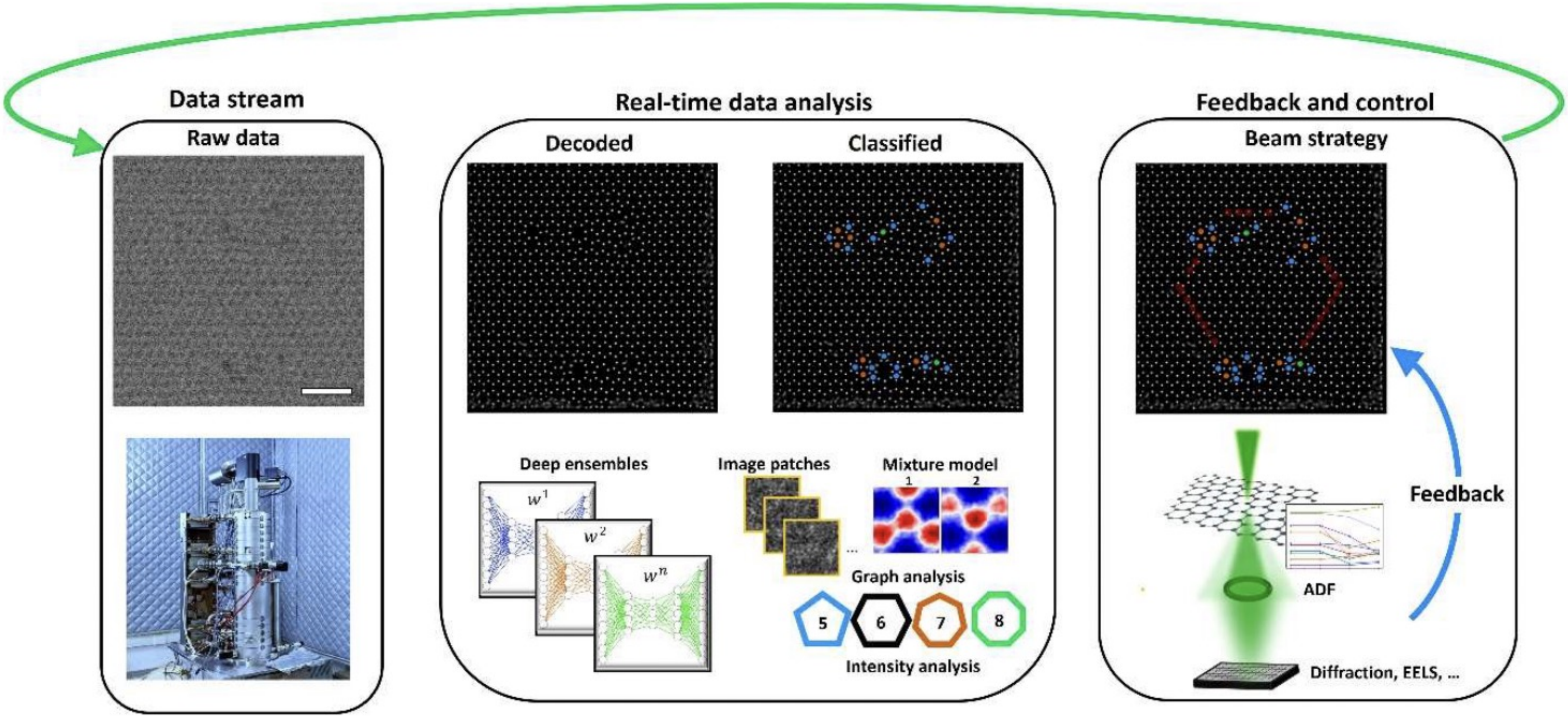
CONTENTS

ORNL/TM-XXXX/XXX

INTERSECT Architecture Specification: Microservice Architecture (Version 0.5)

CONTENTS

Autonomous Microscopy: Science Goal



Autonomous Microscopy: Science Use Case Design Patterns

- Strategic Pattern
  - Experiment Steering
  - Control of an ***ongoing*** STEM experiment via analysis of periodic experimental data
- Architectural Pattern
  - Distributed Experiment Steering
  - Local control of an ***ongoing*** STEM experiment via ***remote*** analysis of periodic experimental data

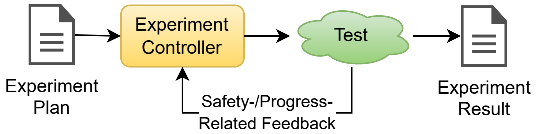


Figure: Strategy pattern: Experiment Steering

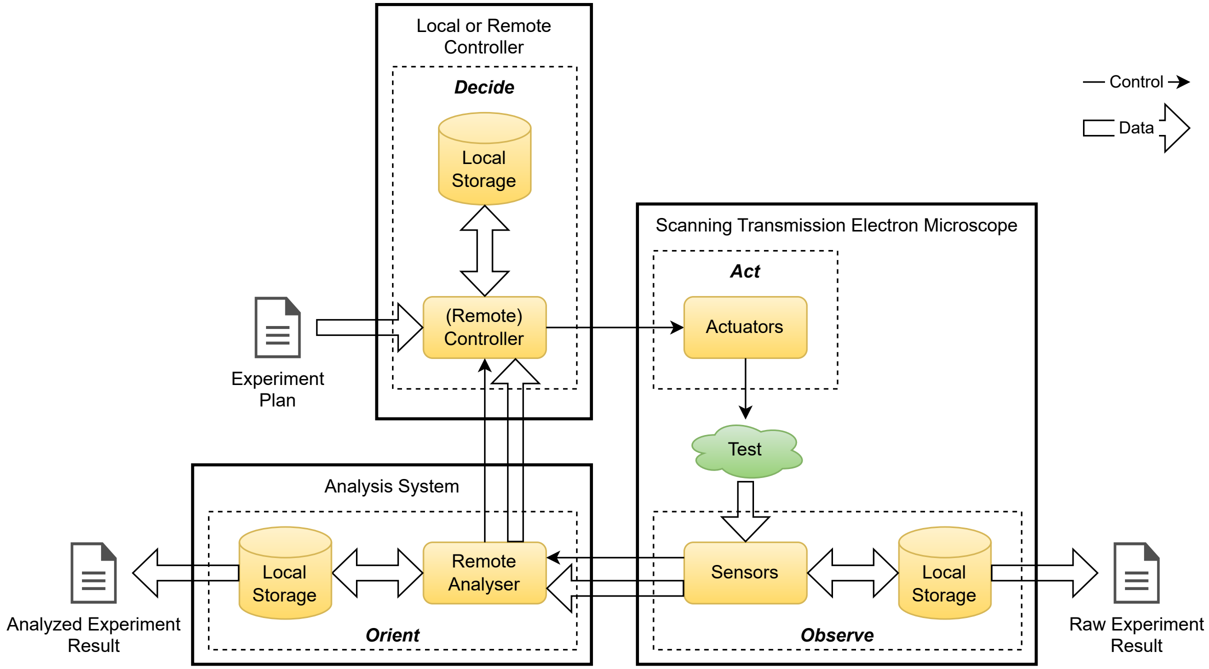
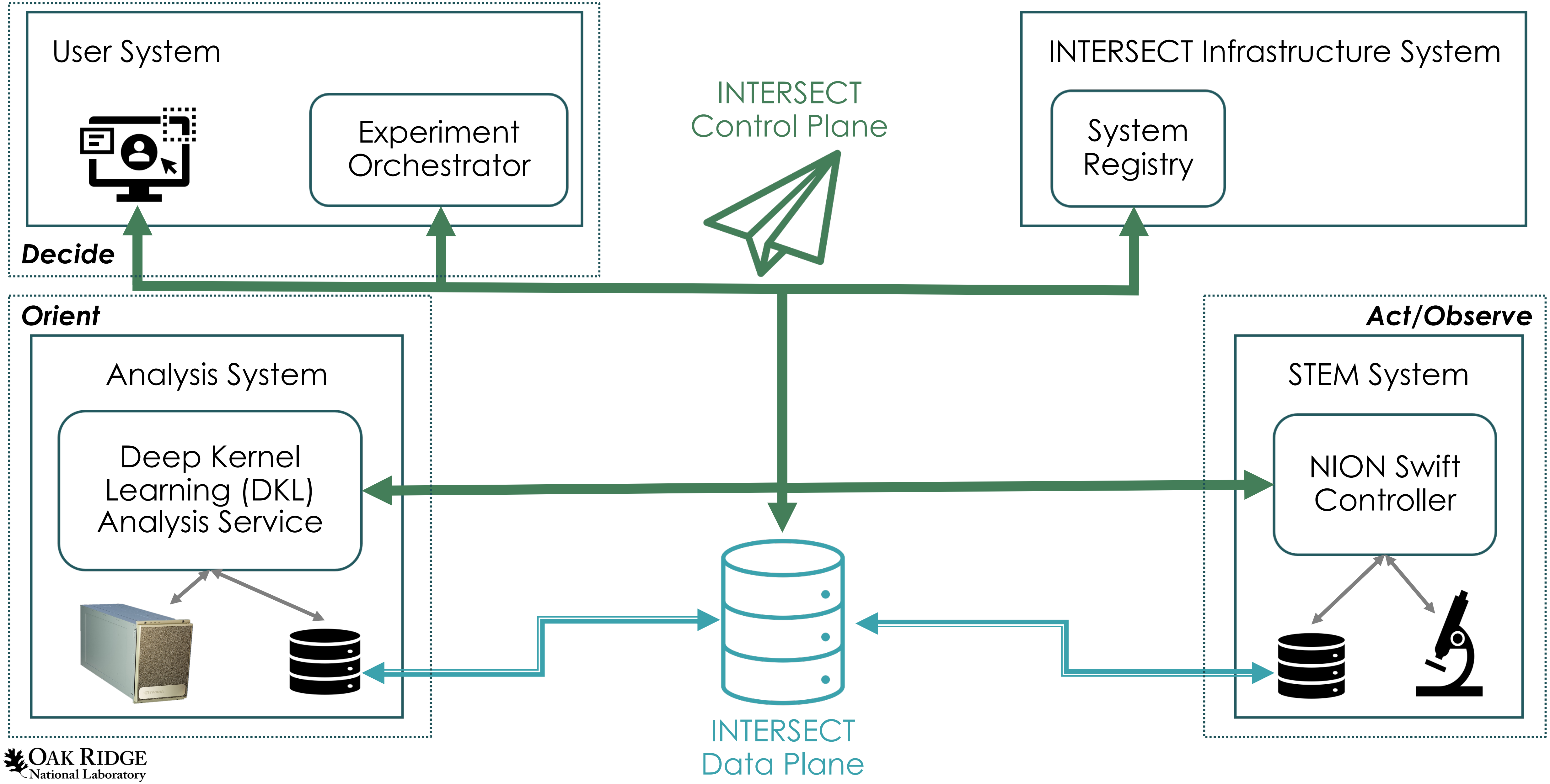


Figure: Architectural pattern: Remote Experiment Steering

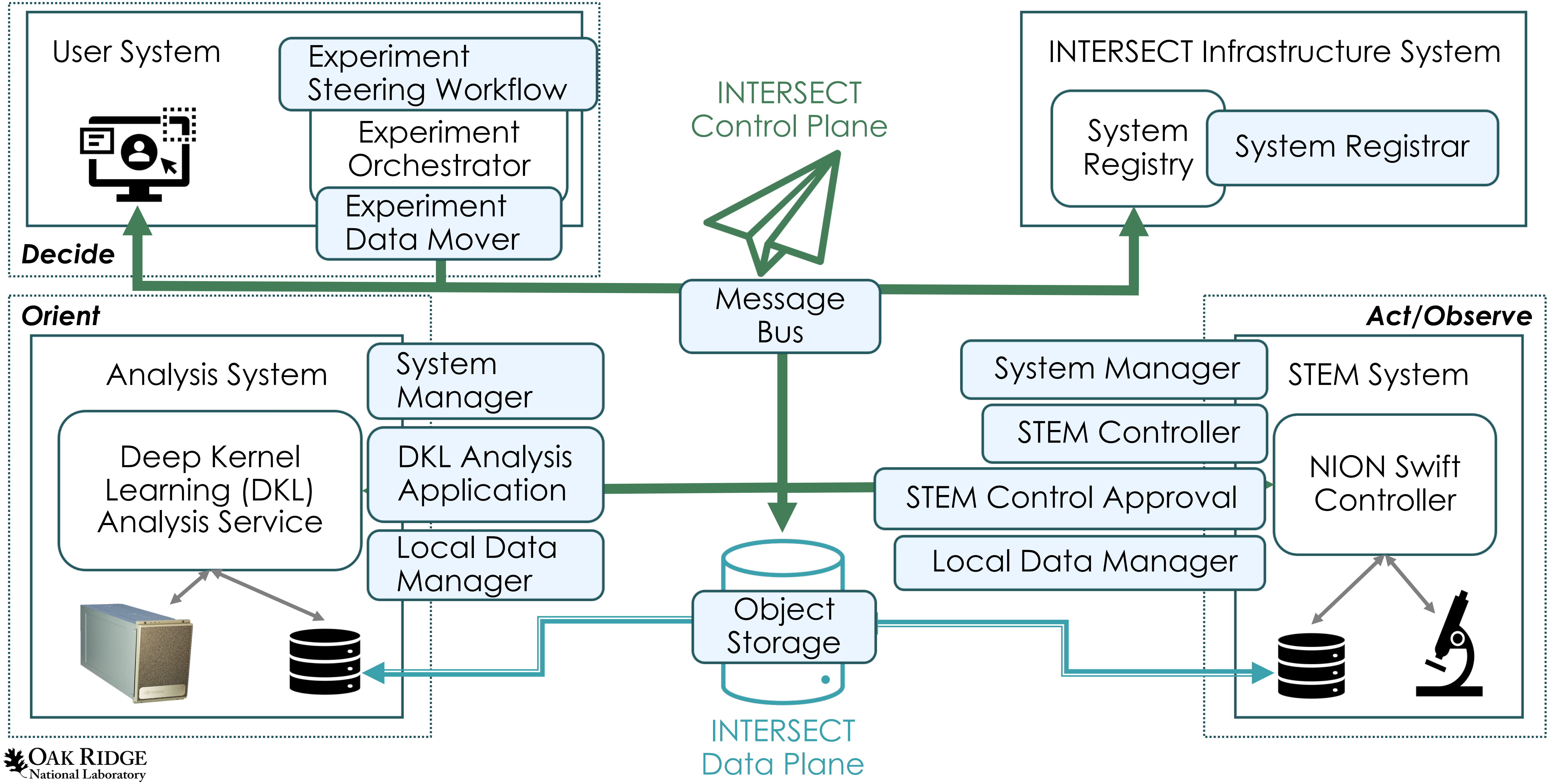
System of Systems Architecture

KEY: System Service



Microservice Architecture

KEY: Microservice Capability



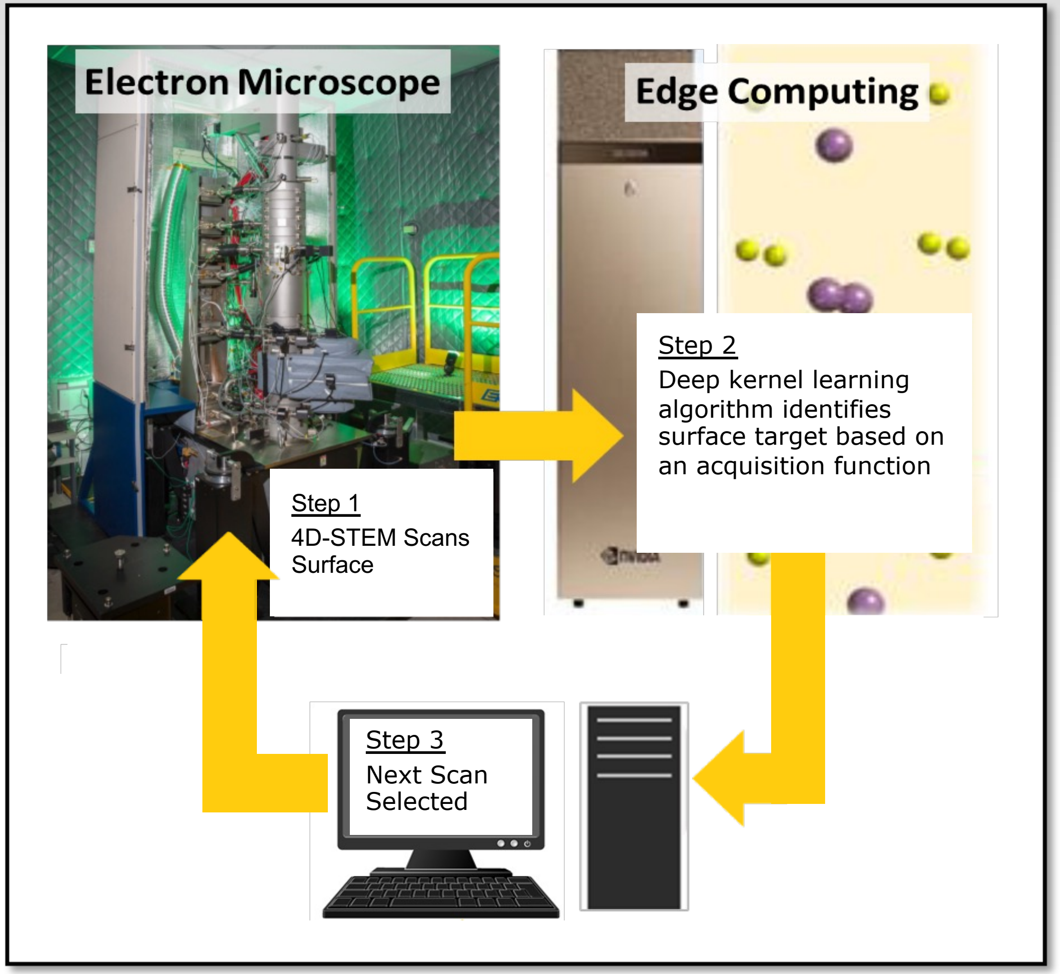
Automated Experiments in Electron Microscopy

PI: Ziatdinov, Maxim

Scientific Achievement	Demonstrated a fully autonomous electron microscopy loop that enabled discovery of local structures, symmetry-breaking distortions, and internal electric and magnetic fields in complex materials.

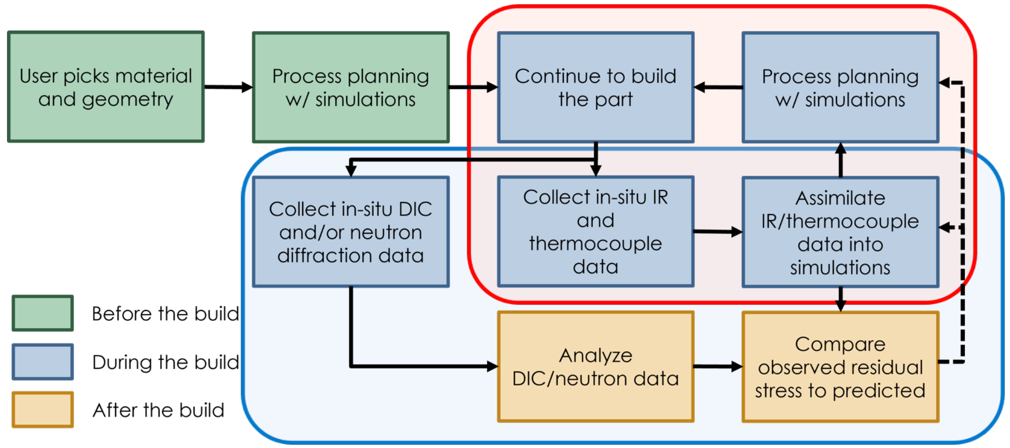
Accomplished	• Initial Implementation
	  ○ Connected STEM to a NVIDIA DGX edge server with custom software
	• Ecosystem Migration
	  ○ Open Architecture and Software teams created the Message Abstraction Layer
	  ○ Software team used the abstraction layer to demonstrate microscopy workflow
	  ○ Microservices are being leveraged across other projects

Microscopy Autonomous Workflow

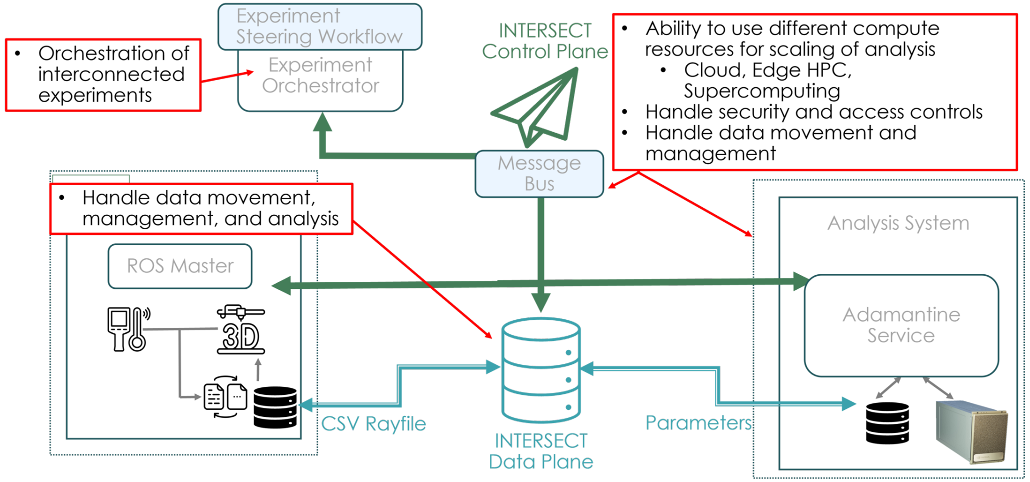


Autonomous Additive Manufacturing

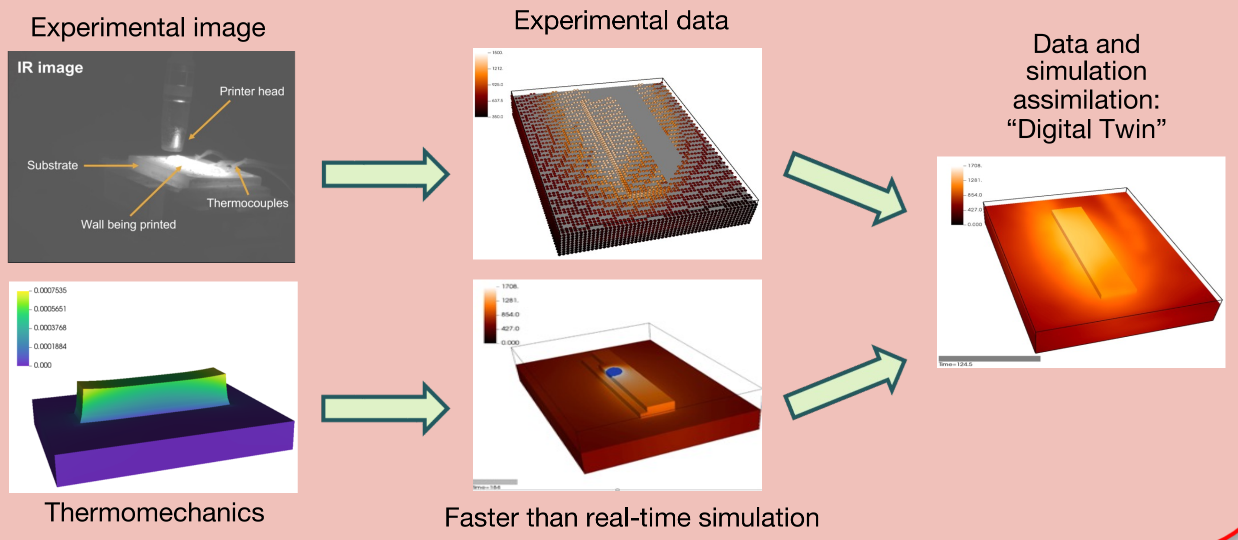
Additive Manufacturing Autonomous Workflow



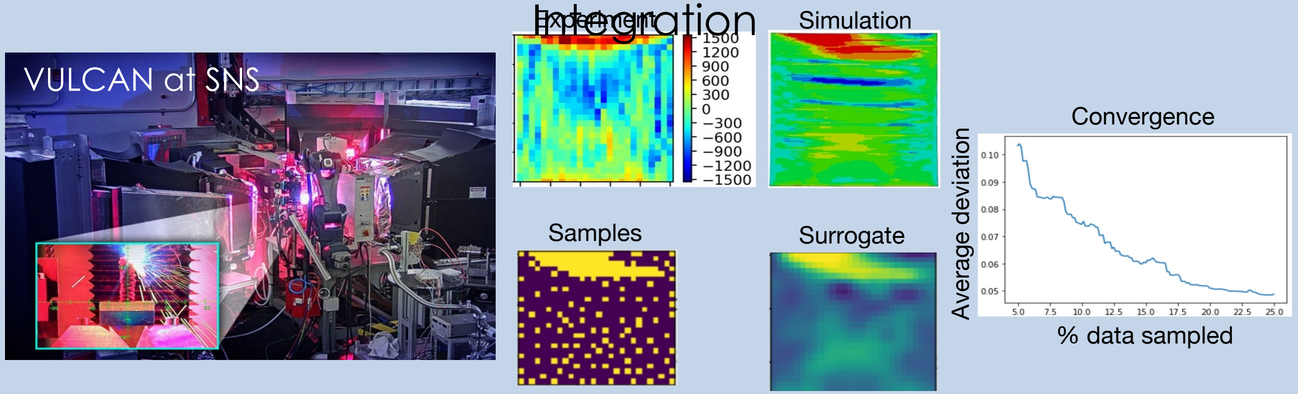
INTERSECT Software Services



(FY 23) Additive Manufacturing Digital Twin

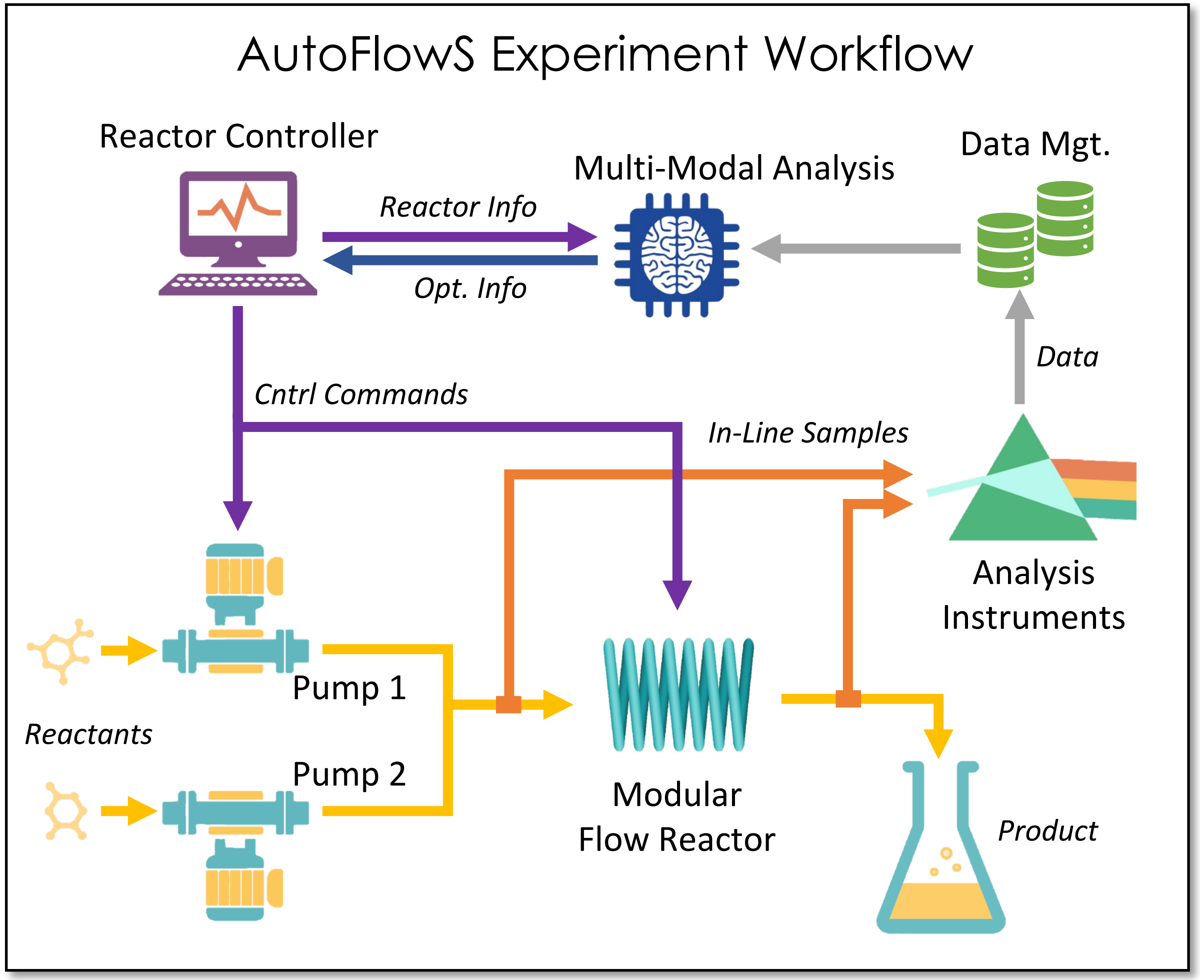


(FY 23/24) Additive Manufacturing / SNS Integration

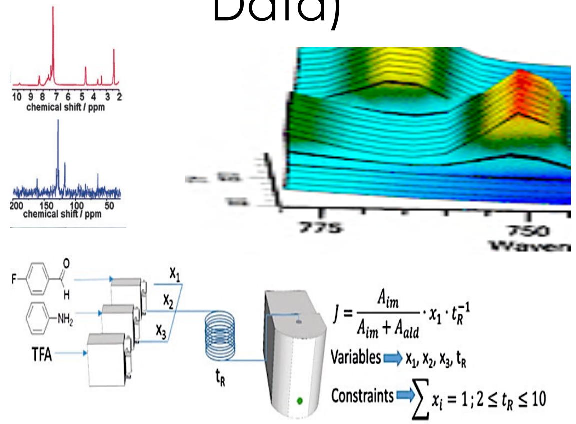


Oak Ridge National Laboratory

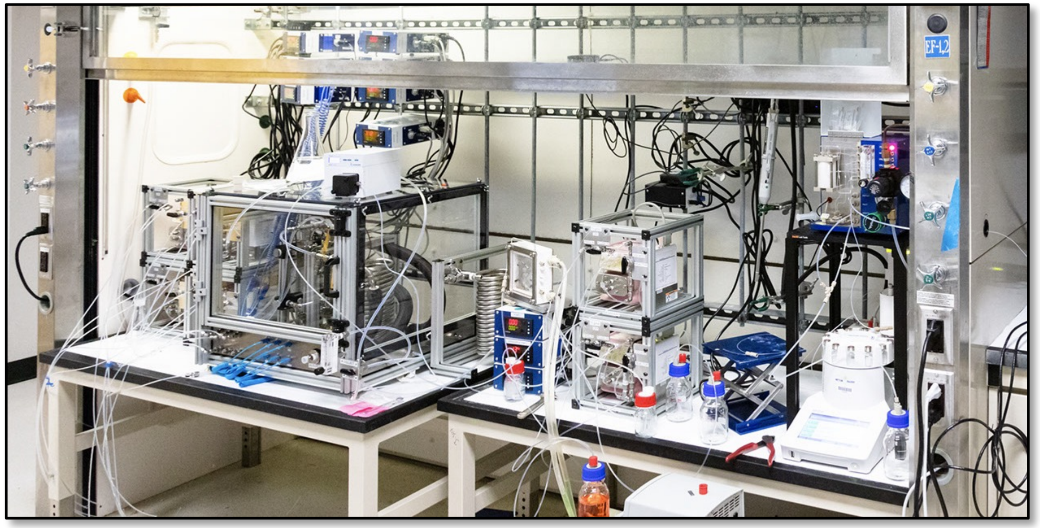
Autonomous Continuous Flow Reactor Synthesis (AutoFlowS)



Active Learning (Multi-Modal Data)



GC-MS, FTIR, Raman, (up to 10 modes)



Questions?