ORNL/TM-2022/2681

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

Christian Engelmann and Suhas Somnath

September 30, 2022

ORNL/TM-2022/2681

Laboratory Directed Research and Development Program
Self-Driven Experiments for Science/Interconnected Science Ecosystem (INTERSECT) Initiative

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

Christian Engelmann and Suhas Somnath

September 30, 2022

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6283
managed by
UT-Battelle LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

AI	artificial intelligence. xiii, 1, 5, 31
DAG	directed acyclic graph. 14
DoDAF	DoD Architecture Framework. 4
DOE	U. S. Department of Energy. 1
GUI	graphical user interface. 32, 33
HPC	high-performance computing. 5
INTERSECT	Self-driven Experiments for Science / Interconnected Science Ecosystem. xiii, 1, 4
NAS	network attached storage. 33
OO	object-oriented. 5
OODA	observe, orient, decide, and act. 21–26, 28, 29
OPL	Our Pattern Language. 5
ORNL	Oak Ridge National Laboratory. xiii, 1, 4
SoS	system of systems. xiii, 1, 4
TBB	Thread Building Blocks. 5

ACKNOWLEDGEMENTS

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.

ABSTRACT

Oak Ridge National Laboratory (ORNL)'s Self-driven Experiments for Science / Interconnected Science Ecosystem (INTERSECT) architecture project, titled "An Open Federated Architecture for the Laboratory of the Future", creates an open federated hardware/software architecture for the laboratory of the future using a novel system of systems (SoS) and microservice architecture approach, connecting scientific instruments, robot-controlled laboratories and edge/center computing/data resources to enable autonomous experiments, "self-driving" laboratories, smart manufacturing, and artificial intelligence (AI)-driven design, discovery and evaluation. The project describes science use cases as design patterns that identify and abstract the involved hardware/software components and their interactions in terms of control, work and data flow. It creates a SoS architecture of the federated hardware/software ecosystem that clarifies terms, architectural elements, the interactions between them and compliance. It further designs a federated microservice architecture, mapping science use case design patterns to the SoS architecture with loosely coupled microservices, standardized interfaces and multi programming language support. The primary deliverable of this project is an INTERSECT Open Architecture Specification, containing the science use case design pattern catalog, the federated SoS architecture specification and the federated microservice architecture specification. This document represents the science use case design pattern catalog of the INTERSECT Open Architecture Specification.

REVISION RECORD

Version	Date	Description
0.5	30/09/2022	Public release with the following changes: • Further clarified terminology in pattern component diagrams (using "Test") • Added section about design pattern compositions • Removed sections for INTERSECT case studies from public release due to lack of maturity
0.4	30/06/2022	Internal draft release with the following changes: • Added experiment result to terminology and pattern descriptions/diagrams • Added optional post-processing of raw experiment result to all architectural patterns • Clarified language about remote components in architectural patterns • Added legend about data/control paths to diagrams in architectural patterns • Added a section for building solutions using science use case design patterns • Added sections for INTERSECT case studies
0.3	04/04/2022	Internal draft release with the following changes: • Clarified terminology and concepts • Improved science use case design pattern introduction • Extended science use case design pattern format by clarifying existing pattern properties and adding new pattern properties • Reorganized science use case design pattern classification • Added 2 architectural patterns (local and remote experiment control) • Significantly extended the descriptions of all patterns in the catalog, added new pattern property descriptions and improved pattern component diagrams
0.2	01/06/2022	Internal draft release
0.1	12/31/2021	Initial, unpublished draft

1. INTRODUCTION

The U. S. Department of Energy (DOE)'s Artificial intelligence (AI) for Science report [21] outlines the need for intelligent systems, instruments, and facilities to enable science breakthroughs with autonomous experiments, "self-driving" laboratories, smart manufacturing, and AI-driven design, discovery and evaluation. The DOE's Computational Facilities Research Workshop report [1] identifies intelligent systems/facilities as a challenge with enabling automation and Reducing human-in-the-loop needs as a cross-cutting theme.

Autonomous experiments, "self-driving" laboratories and smart manufacturing employ machine-in-the-loop intelligence for decision-making. Human-in-the-loop needs are reduced by an autonomous online control that collects experiment data, analyzes it, and takes appropriate operational actions to steer an ongoing or plan a next experiment. It may be assisted by an AI that is trained online and/or offline with archived data and/or with synthetic data created by a digital twin. Analysis and decision making may also rely on rule-based approaches, causal models, and advanced statistical methods. Human interaction for experiment planning, observation and steering is performed through a human-machine interface.

A federated hardware/software architecture for connecting instruments with edge and center computing resources is needed that autonomously collects, transfers, stores, processes, curates, and archives scientific data in common formats. It must be able to communicate with scientific instruments and computing and data resources for orchestration and control across administrative domains, and with humans for critical decisions and feedback. Standardized communication and programming interfaces are needed that leverage community and custom software for scientific instruments, automation, workflows and data transfer. Pluggability is required to permit quickly adaptable and deployable solutions, reuse of partial solutions for different use cases, and the use of digital twins, such as a virtual instrument, robot or experiment. This federated hardware/software architecture needs to be an open standard to enable adoption by DOE's science facilities.

Oak Ridge National Laboratory (ORNL)'s Self-driven Experiments for Science / Interconnected Science Ecosystem (INTERSECT) architecture project, titled "An Open Federated Architecture for the Laboratory of the Future", creates an open federated hardware/software architecture for the laboratory of the future using a novel system of systems (SoS) and microservice architecture approach, connecting scientific instruments, robot-controlled laboratories and edge/center computing/data resources to enable autonomous experiments, "self-driving" laboratories, smart manufacturing, and AI-driven design, discovery and evaluation.

The project describes science use cases as design patterns that identify and abstract the involved hardware/software components and their interactions in terms of control, work and data flow. It creates a SoS architecture of the federated hardware/software ecosystem that clarifies terms, architectural elements, the interactions between them and compliance. It further designs a federated microservice architecture, mapping science use case design patterns to the SoS architecture with loosely coupled microservices, standardized interfaces and multi programming language support. The primary deliverable of this project is an INTERSECT Open Architecture Specification, containing the science use case design pattern catalog, the federated SoS architecture specification and the federated microservice architecture specification. This document represents the science use case design pattern catalog of the INTERSECT Open Architecture Specification.

2. TERMINOLOGY AND CONCEPTS

This section briefly describes some of the used terminology and concepts in this document for clarification purposes. The descriptions are thematically grouped and ordered by dependence, i.e., their relationships. Although many of the used terminology and concepts have additional meanings in science and engineering, their descriptions are based on the context they are being used in this document. This is by no means an exhaustive list, but rather represents the core descriptions needed to understand this document.

General Terms:

- **Experiment:** A test under controlled conditions to demonstrate a known truth, examine the validity of a hypothesis, or determine the properties of something.
- **Multi-experiment workflow:** A set of experiments performed in serial (one after another) and/or in parallel (simultaneously).
- **Campaign:** A scientific endeavor that may consist of one or more experiments that may take place sequentially or in parallel to answer a broader overarching scientific question.

Experiment Devices:

- **Sensor:** A device for measuring something before, during and/or after running an experiment.
- **Actuator:** A device for moving or controlling something before, during and/or after running an experiment.
- **Instrument:** A device containing sensors and potentially actuators.
- **Robot:** An automated or autonomous device containing actuators and potentially sensors.

Experiment and Multi-experiment Workflow Data:

- **Experiment plan:** A list of actions that need to be executed while running an experiment.
- **Experiment design plan:** An initial experiment plan and a plan for creating new experiment plans based on experiment results.
- **Multi-experiment workflow plan:** A list of actions that need to be executed while running a multi-experiment workflow, i.e., a set of experiments in serial and/or parallel.
- **Experiment result:** The data collected from sensors before, during and/or after running an experiment.

Operational Experiment Properties:

- **Automated:** Executing an existing experiment or multi-experiment workflow plan, by performing its list of actions, without external or human intervention that can unnecessarily hold up execution.
- **Autonomous:** Creating a new or modifying an existing experiment or multi-experiment workflow plan and executing it, by performing its list of actions, without external or human intervention that can unnecessarily hold up execution.

Experiment and Workflow Control:

- **Loop control:** The devices and functions necessary to automatically or autonomously perform an experiment or a multi-experiment workflow.
- **Open loop control:** A loop control without feedback, except to monitor the experiment(s) for safety reasons.

- **Closed loop control:** A loop control with feedback, such as to monitor experiment(s) progress or result and to adapt experiment or multi-experiment workflow plans.
- **OODA loop control:** A closed loop control with 4 distinct components: (1) *Observe* the evolving situation, (2) *Orient* the observed information for decision making, (3) *Decide* on appropriate actions, and (4) *Act* on the made decisions.
- **Experiment controller:** A component that executes an experiment plan by performing its list of actions and collecting any feedback.
- **Experiment planner:** A component that creates an experiment plan based on an experiment design plan and experiment results.
- **Multi-experiment workflow controller:** A component that executes a multi-experiment workflow plan by performing its list of actions and collecting any feedback.

3. DESIGN PATTERNS FOR SCIENCE USE CASES

The primary deliverable of ORNL's INTERSECT architecture project, titled "An Open Federated Architecture for the Laboratory of the Future", is an INTERSECT Open Architecture Specification, containing a use case design pattern catalog, a SoS architecture specification and a microservice architecture specification. It essentially offers different viewpoints of the INTERSECT Open Architecture, similar to the DoD Architecture Framework (DoDAF) [23] approach with its (1) operational scenarios; (2) composition, interconnectivity and context; (3) services and their capabilities; (4) policies, standards and guidance; and (5) capability.

The SoS architecture specification primarily follows the DoDAF approach and provides operational, data, logical and physical views of the architecture, among others. The use case design pattern catalog contains elements of the use case view, but is a separate document, as it follows the design pattern approach. Similarly, the microservice architecture specification contains elements of the service view, but follows the microservice architecture approach. The microservice architecture maps use case design patterns to the SoS architecture with loosely coupled microservices and standardized interfaces.

3.1 INTRODUCTION TO DESIGN PATTERNS

A design pattern is a description of a generalized solution to a recurring problem within a well-defined context. Design patterns are often created from best practices and contain the essential elements of the problems they tackle and their corresponding solutions. They offer a template on how to solve a specific problem that may apply to different situations. They may also describe different solution alternatives to a specific problem.

The concept of design patterns originates in civil architecture and engineering. Design patterns captured the detailed designs of towns and neighborhoods, houses, gardens and rooms with the goal of designing functional and aesthetically beautiful living spaces and structures. They identify and catalog solutions to recurrent problems encountered during the process of building and planning. Each pattern describes a problem that occurs repeatedly in our environment and the core of the solution to that problem in such a way that it may be used a million times over, without ever doing it the same way twice [2].

In general, a design pattern identifies the key aspects of a solution and creates an abstract description that makes it useful in the creation of a reusable design element. Patterns don't describe a concrete design or an implementation - they are intended to be templates that may be applied by a designer in various contexts and modified to suit the problem at hand. Patterns are also free from constraints of detail associated with the level of system abstraction at which the solution is implemented. Patterns also describe the design decisions that must be made when applying a certain solution. This enables a designer to reason about the impact of the design decisions on a system's flexibility or scalability as well as consider implementation issues. Design patterns must address a specific problem at hand, and yet must be general enough to remain relevant to future requirements of systems.

In the domain of software design, patterns were introduced in an effort to create reusable solutions in the design of software and bring discipline to the art of programming. The intent of software design patterns isn't to provide a finished design that may be transformed directly into code; rather, design patterns are used to enhance the software development process by providing proven development paradigms. With the use of design patterns, there is sufficient flexibility for software developers to adapt their implementation to accommodate any constraints, or issues that may be unique to specific programming paradigms, or the target platform for the software. Related to design patterns, the concept of algorithmic skeletons was introduced [6] and further refined [7].

In the context of object-oriented (OO) programming, design patterns provide a catalog of methods for defining class interfaces and inheritance hierarchies, and establish key relationships among the classes [11]. In many object-oriented systems, reusable patterns of class relationships and communications between objects are used to create flexible, elegant, and ultimately reusable software design. There are three categories of OO patterns: (i) **creational** patterns for ways to do instantiation of objects, (ii) **structural** patterns concerned with class and object composition, and (iii) **behavioral** patterns for communication between objects. Patterns have also been defined in the design of software architectures [4] to capture repeatedly used methodologies in software engineering practice. Pattern systems have also been developed for cataloging concurrent and networked object-oriented environments [20], resource management [16], and distributed software systems [5].

In the pursuit of quality and scalable parallel software, patterns for programming paradigms were developed [17] as well as a pattern language, called Our Pattern Language (OPL) [15]. These parallel patterns are used as means to systematically describe parallel computation and communication when architecting parallel software. In an effort to enable a more structured approach to designing and implementing parallel applications, particularly for many-core processors, a catalog of parallel patterns enables programmers to compose parallel algorithms, which may be easily implemented using various programming interfaces such as OpenMP, OpenCL, Cilk Plus, ArBB, Thread Building Blocks (TBB) [18]. For the design of parallel algorithms, deterministic patterns support the development of systems that automatically avoid unsafe race conditions and deadlock [19].

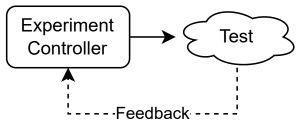
Design patterns have been identified in a variety of other domains for codifying the best-known solutions to common problems, including natural language processing [22], user interface design [3], web design [9], visualization [12], software security [8] and high-performance computing (HPC) resilience [14, 13]. Patterns have also been defined for enterprise applications that involve data processing in support or automation of business processes [10] in order to bring structure to the construction of enterprise application architectures. In each of these domains of design, patterns capture the essence of effective solutions in a succinct form that may be easily applied in similar form to other contexts and problems.

3.2 ANATOMY OF A SCIENCE USE CASE DESIGN PATTERN

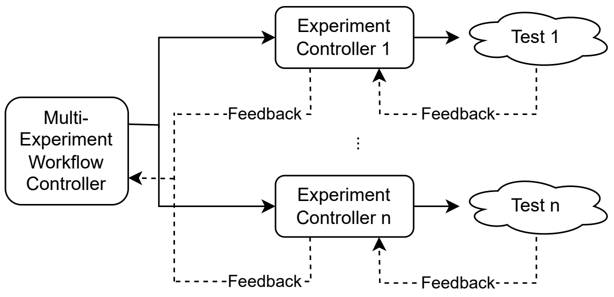
Reducing human-in-the-loop requirements with machine-in-the-loop capabilities by connecting scientific instruments, robot-controlled laboratories and edge/center computing/data resources to enable autonomous experiments, "self-driving" laboratories, smart manufacturing, and AI-driven design, discovery and evaluation is an inherent open or closed loop control problem. Therefore, the basic template for a science use case design pattern is defined in a loop control problem paradigm. The abstract science use case design pattern consists of a behavior and a set of interfaces in the context of performing a single or a set of experiments in an open or closed loop control. Such an abstract definition creates universal patterns that describe solutions free of implementation details.

3.3 FORMAT OF A SCIENCE USE CASE DESIGN PATTERN

Design patterns for science use cases are expressed in a written form and in a highly structured format, which permits quick identification of relevant patterns given a certain problem to be solved and easy



(a) Experiment loop control problem



(b) Multi-experiment workflow loop control problem

Figure 3-1. Science use case anatomy as experiment loop control problem and as multi-experiment workflow loop control problem

comparison of patterns regarding their applicability and capabilities. The format for describing science use case design patterns consists of individual descriptions of pattern properties, including text, diagrams, and mathematical models. It can be extended over time by adding more pattern properties and their descriptions. The current science use case design pattern format is as follows:

Name: A descriptive name that distinctly identifies the pattern and enables designers to think about designs in an abstract manner and communicate their design choices to others.

Problem: A description of the problem that provides insight on when it is appropriate to apply the pattern. Multiple patterns may address the same problem differently.

Context: The preconditions under which the pattern is relevant, including a description of the system before the pattern is applied.

Forces: A description of the relevant forces and constraints, and how they interact or conflict with each other and with the intended goals and objectives.

Solution: A description of the solution that defines the abstract elements that are necessary for the composition of the design solution as well as their relationships, responsibilities, and collaborations.

Capabilities: The specific capabilities provided by this pattern in terms of the loop control problem it solves.

Resulting Context: A brief description of the post-conditions arising from the application of the pattern. There may be trade-offs between competing optimization parameters that arise due to the implementation of a solution using this pattern.

Related Patterns: The relationships between this pattern and other relevant patterns. Other patterns may be predecessor or successor patterns. This pattern may complement or enhance other patterns. There may also be dependencies between patterns to provide a complete solution.

4. CLASSIFICATION OF SCIENCE USE CASE DESIGN PATTERNS

As explained in Section 3.1, there can be different categories, or classes, of design patterns, depending on context. A classification of patterns helps to identify groups of patterns that address similar problems in different ways or that describe solutions at different levels of granularity or from different points of view. A classification scheme codifies these relationships between patterns and enables designers to better understand individual pattern capabilities and relationships. It also further helps to understand how patterns rely on each other and can be composed to form a complete solution.

The classification of science use case design patterns is work in progress at this stage of this document. At this point, there are two classes of science use case design patterns (Figure 4-1): (1) strategy patterns that define high-level solution methods using experiment control architecture features at a very coarse granularity, and (2) architectural patterns that define more specific solution methods using hardware and software architecture features at a finer granularity. While the architectural patterns do inherit the features of certain parent strategy patterns, they also address additional problems that are not exposed at the high abstraction level of the strategy patterns.

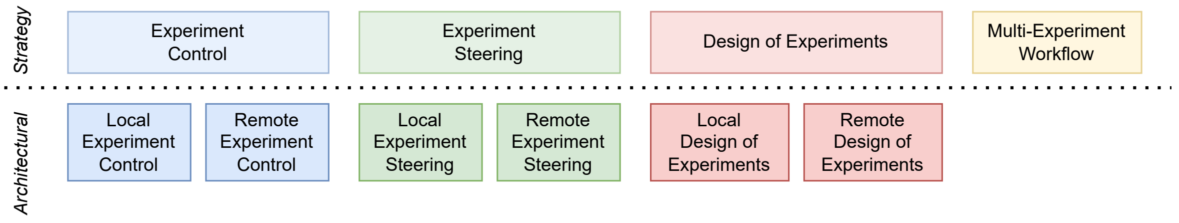

Strategy
Experiment Control
Experiment Steering
Design of Experiments
Multi-Experiment Workflow
Architectural
Local Experiment Control
Remote Experiment Control
Local Experiment Steering
Remote Experiment Steering
Local Design of Experiments
Remote Design of Experiments

Figure 4-1. Classification of the science use case design patterns

4.1 STRATEGY PATTERNS

The science use case strategy patterns define high-level solution methods using experiment control architecture features at a very coarse granularity. Their descriptions are deliberately abstract to enable architects to reason about the overall organization of the used techniques and their implications on the full system design. The catalog of science use case design patterns defines the following strategy patterns in Section 5.1:

- *Experiment Control*
- *Experiment Steering*
- *Design of Experiments*
- *Multi-Experiment Workflow*

The experiment control architecture features of these science use case strategy patterns and their relationships are compared in Table 4-1.

4.2 ARCHITECTURAL PATTERNS

Architectural patterns define more specific solution methods using hardware and software architecture features at a finer granularity. They offer more detailed descriptions, conveying different design choices for implementing strategy patterns and their abstract architectural features. Architectural patterns inherit the

Table 4-1. Feature comparison and relationships of the science use case strategy patterns

Feature	Experiment Control	Experiment Steering	Design of Experiments	Multi-Experiment Workflow
# of experiments	1	1	Multiple	Multiple
Control type	Open loop	Closed loop	Closed loop	Open loop
Operation type	Automated	Autonomous	Autonomous	Automated
Extends		Experiment Control		
Uses			Experiment Control	Experiment Control
May also use or use instead			Experiment Steering	Experiment Steering, Design of Experiments

features of their parent strategy patterns. However, they also address additional problems through specific design choices that are not exposed at the high abstraction level of the parent strategy patterns.

The architectural patterns provide abstractions for the different hardware/software architecture choices of implementing experiment control and workflow, such as using experiment-local, edge and/or center computing and data resources. The catalog of science use case design patterns defines the following architectural patterns in Section 5.2:

- *Local Experiment Control*
- *Remote Experiment Control*
- *Local Experiment Steering*
- *Remote Experiment Steering*
- *Local Design of Experiments*
- *Remote Design of Experiments*

Table 4-2 shows the architectural patterns and their relationships to the strategy patterns.

Table 4-2. Relationships of the science use case strategy and architectural patterns

Architectural Pattern	Implements Strategy Pattern
Local Experiment Control	Experiment Control
Remote Experiment Control	Experiment Control
Local Experiment Steering.	Experiment Steering
Remote Experiment Steering	Experiment Steering
Local Design of Experiments.	Design of Experiments
Remote Design of Experiments	Design of Experiments

5. CATALOG OF SCIENCE USE CASE DESIGN PATTERNS

5.1 STRATEGY PATTERNS

5.1.1 Experiment Control

Name: Experiment Control

Problem: Certain predetermined actions need to be performed while running an experiment.

Context: The pattern applies to a system with the following characteristics:

- An experiment plan exists that lists the predetermined actions to be performed while running the experiment.
- Actuators may exist to allow for moving or controlling something before, during and/or after running the experiment.
- Sensors may exist to allow for measuring something before, during and/or after running the experiment.
- Instruments may exist that contain sensors and potentially actuators.
- Robots may exist that contain actuators and potentially sensors and that execute predetermined actions from the experiment plan in an automated or autonomous fashion.

Forces: Only pre-experiment conditions are considered in performing the predetermined actions while running an experiment. Only safety-related conditions during the experiment may be considered. Other changing conditions during the experiment or post-experiment conditions are not considered.

Solution: An experiment controller executes an experiment using a predetermined experiment plan (Figure 5-1). The plan's execution is automated, performed in an open loop control and may involve human interaction. The controller may monitor the experiment for safety reasons. The plan contains a complete description of the predetermined actions to be performed for running the experiment, including any safety-related responses.

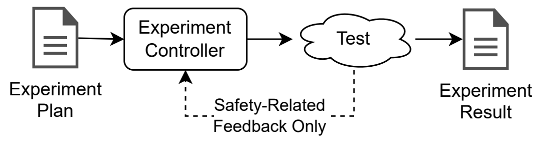


Figure 5-1. Experiment Control strategy pattern components and control/data flow

Capabilities: This pattern offers an open loop control with safety-related feedback on the experiment only. Experiment plan execution is automated, i.e., its list of actions is performed without external or human intervention that can unnecessarily hold up execution. Only 1 experiment is being controlled.

Resulting Context: An experiment is executed automatically using a predetermined plan.

Related Patterns: The Experiment Steering strategy pattern extends this strategy pattern with a closed loop control and feedback on experiment progress. The Multi-Experiment Workflow and Design of Experiments strategy patterns rely on this strategy pattern for automatically executing a predetermined experiment plan.

5.1.2 Experiment Steering

Name: Experiment Steering

Problem: Certain predetermined actions need to be performed while running an experiment, depending on experiment progress.

Context: The pattern applies to a system with the following characteristics:

- An experiment plan exists that lists the predetermined actions to be performed while running the experiment, including potential parameter changes based on experiment progress.
- Sensors exist to allow for measuring experiment progress.
- Actuators may exist to allow for moving or controlling something before, during and/or after running the experiment.
- Additional sensors may exist to allow for measuring something before, during and/or after running the experiment.
- Instruments may exist that contain sensors and potentially actuators.
- Robots may exist that contain actuators and potentially sensors and that execute predetermined actions from the experiment plan in an automated or autonomous fashion.

Forces: Only pre-experiment conditions and changing conditions during the experiment are considered in performing the predetermined actions while running an experiment. Post-experiment conditions are not considered.

Solution: An experiment controller executes an experiment using a predetermined experiment plan and changes the plan's parameters during execution based on experiment progress (Figure 5-2). The plan's execution is autonomous, performed in a closed loop control and may involve human interaction. The controller may monitor the experiment for safety reasons. The plan contains a complete description of the predetermined actions to be performed for running the experiment, including any safety-related responses and how to analyze and judge experiment progress and change the plan accordingly.

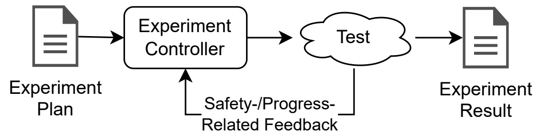


Figure 5-2. Experiment Steering strategy pattern components and control/data flow

Capabilities: This pattern offers a closed loop control with safety-related feedback on the experiment and feedback on experiment progress. Experiment plan execution is autonomous, i.e., its list of actions changes during execution based on feedback and is performed without external or human intervention that can unnecessarily hold up execution. Only 1 experiment is being controlled.

Resulting Context: An experiment is executed autonomously using a predetermined experiment plan, with the plan's parameters changing autonomously during the experiment based on experiment progress.

Related Patterns: This strategy pattern is an extension of the Experiment Control strategy pattern with an added closed loop control and feedback on experiment progress. The Multi-Experiment Workflow and Design of Experiments strategy patterns can be extended using this strategy pattern for autonomously executing a predetermined experiment plan, with the plan’s parameters changing autonomously during experiments based on experiment progress.

This strategy pattern is implemented by the Local Experiment Steering and Remote Experiment Steering architectural patterns.

5.1.3 Design of Experiments

Name: Design of Experiments

Problem: Certain predetermined actions need to be performed to run a set of similar experiments with different experiment plan parameters, depending on experiment results.

Context: The pattern applies to a system with the following characteristics:

- An experiment design plan exists that lists the predetermined actions to be performed for creating a new experiment plan based on prior experiment results.
- An initial experiment plan exists that lists the predetermined actions to be performed while running the experiment.
- Sensors exist to allow for measuring experiment results.
- Actuators may exist to allow for moving or controlling something before, during and/or after running the experiment.
- Additional sensors may exist to allow for measuring something before, during and/or after running the experiment.
- Instruments may exist that contain sensors and potentially actuators.
- Robots may exist that contain actuators and potentially sensors and that execute predetermined actions from the experiment plan in an automated or autonomous fashion.

Forces: Only pre- and post-experiment conditions are considered in performing the predetermined actions to run a set of similar experiments with different experiment plan parameters. Only safety-related conditions during the experiment may be considered. Other changing conditions during the experiments are not considered, unless the Experiment Steering strategy pattern is being used in conjunction with this strategy pattern.

Solution: An experiment controller executes each experiment using a predetermined experiment plan (Figure 5-3). The plan's execution is automated, performed in an open loop control and may involve human interaction. The controller may monitor the experiment for safety reasons. The experiment plan contains a complete description of the predetermined actions to be performed for running the experiment, including any safety-related responses. An experiment planner creates the experiment plan, based on an experiment design plan and prior experiment results (if any). The experiment plan change is autonomous, performed in a closed loop control and may involve human interaction. The experiment design plan contains an initial experiment plan and a plan for creating new experiment plans based on experiment results, including how to analyze and judge experiment results and change the plan accordingly.

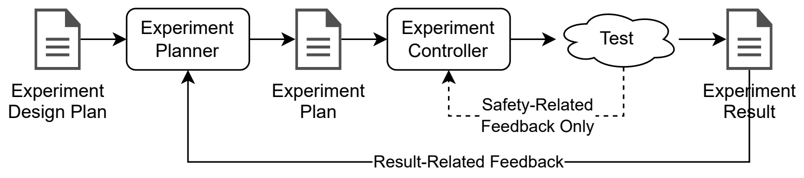


Figure 5-3. Design of Experiments strategy pattern components and control/data flow

Capabilities: This pattern offers an open loop control with safety-related feedback on the experiment and a separate closed loop control with feedback on experiment results. Experiment plan execution is automated within the open loop control, i.e., its list of actions is performed without external or human intervention that can unnecessarily hold up execution. Experiment design plan execution is autonomous, i.e., it creates a new experiment plan after each experiment based on experiment results and is performed without external or human intervention that can unnecessarily hold up execution. A set of similar experiments with different experiment plan parameters is controlled.

Resulting Context: An experiment is executed autonomously with different experiment plan parameters using a predetermined experiment plan, with the plan's parameters changing autonomously between experiments based on experiment results.

Related Patterns: This strategy pattern relies on the Experiment Control strategy pattern for automatically executing a predetermined experiment plan. This strategy pattern can be extended using the Experiment Steering strategy pattern (instead of the Experiment Control strategy pattern) for autonomously executing a predetermined experiment plan, with the plan's parameters changing autonomously during experiments based on experiment progress.

This strategy pattern is implemented by the Local Design of Experiments and Remote Design of Experiments architectural patterns.

5.1.4 Multi-Experiment Workflow

Name: Multi-Experiment Workflow

Problem: Certain predetermined actions need to be performed to run a set of experiments in serial (one after another) and/or in parallel (simultaneously).

Context: The pattern applies to a system with the following characteristics:

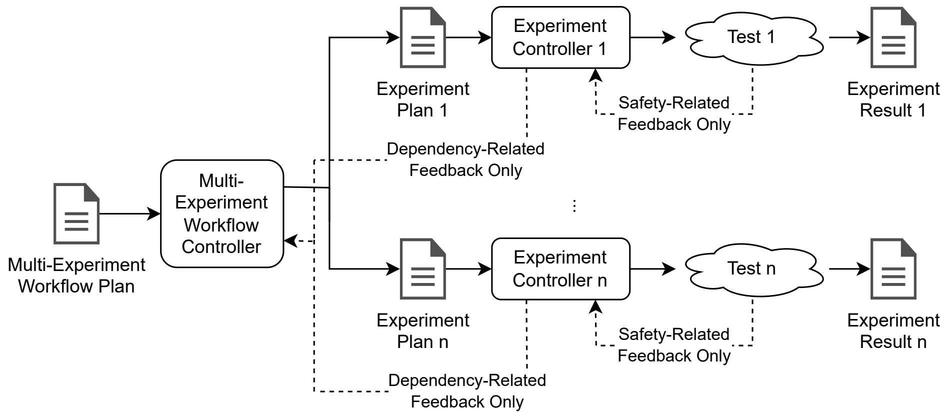
- A multi-experiment workflow plan exists that lists the predetermined actions to be performed for executing each experiment plan.
- An experiment plan exists for each experiment that lists the predetermined actions to be performed while running the experiment.
- Actuators may exist to allow for moving or controlling something before, during and/or after running the experiment.
- Sensors may exist to allow for measuring something before, during and/or after running the experiment.
- Instruments may exist that contain sensors and potentially actuators.
- Robots may exist that contain actuators and potentially sensors and that execute predetermined actions from the experiment plan in an automated or autonomous fashion.

Forces: Only pre- and post-experiment conditions are considered in performing the predetermined actions to run the set of experiments in serial and/or parallel. Safety-related conditions during the experiments may be considered. Only pre-experiment conditions are considered in performing the predetermined actions while running each experiment individually, unless the Experiment Steering strategy pattern or the Design of Experiments strategy pattern are being used for some or all experiments. If the Experiment Steering strategy pattern is being used for a particular experiment, then changing conditions during this experiment are considered in performing the predetermined actions while running it. If the Design of Experiments strategy pattern is being used for a particular experiment, then post-experiment conditions are considered in performing the predetermined actions to run it with different experiment plan parameters.

Solution: A multi-experiment workflow controller orchestrates the execution of the experiments using a predetermined multi-experiment workflow plan (Figure 5-4). The multi-experiment workflow plan's execution is automated, performed in an open loop control and may involve human interaction. The multi-experiment workflow controller may monitor one or more experiment controllers for dependency reasons. The multi-experiment workflow plan contains a complete description of the predetermined actions to be performed for orchestrating the execution of the experiments including any dependency-related responses.

Multiple experiment controllers execute their experiments using their predetermined experiment plan. Each plan's execution is automated, performed in an open loop control and may involve human interaction. Each experiment controller may monitor the experiment for safety reasons. Each experiment plan contains a complete description of the predetermined actions to be performed for running its experiment, including any safety-related responses.

Some experiments may be executed in parallel, as they do not depend on each other, while other experiment may be executed in serial due to dependencies. The orchestration of the execution follows a directed acyclic graph (DAG) with the experiments as vertices and the edges as dependencies (Figure 5-5). A dependency between experiments may arise when one experiment needs the result of another.



Multi-Experiment Workflow Plan
Multi-Experiment Workflow Controller
Experiment Plan 1
Experiment Controller 1
Test 1
Experiment Result 1
Safety-Related Feedback Only
Dependency-Related Feedback Only
Experiment Plan n
Experiment Controller n
Test n
Experiment Result n
Safety-Related Feedback Only
Dependency-Related Feedback Only

Figure 5-4. Multi-Experiment Workflow strategy pattern components and control/data flow

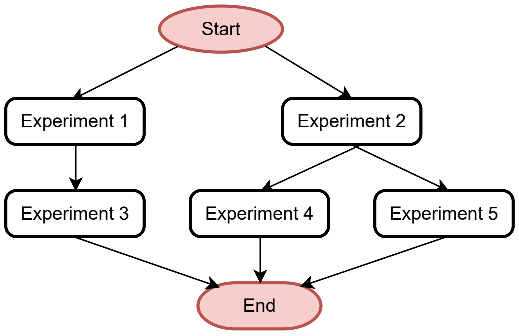


Figure 5-5. Example of a Multi-Experiment Workflow strategy pattern directed acyclic graph

Capabilities: This pattern offers an open loop control with safety-related feedback on each experiment and a separate loop control with safety-related feedback for each experiment. Experiment plan execution is automated within the open loop control for each experiment, i.e., its list of actions is performed without external or human intervention that can unnecessarily hold up execution. Multi-experiment workflow plan execution is automated within the open loop control for all experiments, i.e., its list of actions is performed without external or human intervention that can unnecessarily hold up execution. A set of serial and/or parallel experiments is controlled.

Resulting Context: Experiments are executed automatically in serial and/or parallel using a predetermined plan.

Related Patterns: This strategy pattern relies on the Experiment Control strategy pattern for automatically executing each predetermined experiment plan. This strategy pattern can be extended using the Experiment Steering strategy pattern (instead of the Experiment Control strategy pattern) for autonomously executing some or all predetermined experiment plans, with each plan's parameters changing autonomously during experiments based on progress. This strategy pattern can also be extended using the Design of Experiments strategy pattern for autonomously executing some or all predetermined experiment plans, with each plan's parameters changing autonomously between experiments based on results. The Experiment Control, Experiment Steering and Design of Experiments strategy patterns can be used together in conjunction with this strategy pattern, individually for each experiment of the multi-experiment workflow. However, the

Experiment Control and Experiment Steering strategy patterns are mutually exclusive for the same experiment, as the Experiment Steering strategy pattern extends the Experiment Control strategy pattern.

5.2 ARCHITECTURAL PATTERNS

5.2.1 Local Experiment Control

Name: Local Experiment Control

Problem: Certain predetermined actions need to be performed while running an experiment. A local experiment controller executes an experiment. There are no remote components that could incur a significant communication delay.

Context: The pattern applies to a system with the following characteristics:

- An experiment plan exists that lists the predetermined actions to be performed while running the experiment.
- A local experiment controller exists that executes the predetermined actions to be performed while running the experiment.
- Actuators may exist to allow for moving or controlling something before, during and/or after running the experiment.
- Sensors may exist to allow for measuring something before, during and/or after running the experiment.
- Instruments may exist that contain sensors and potentially actuators.
- Robots may exist that contain actuators and potentially sensors and that execute predetermined actions from the experiment plan in an automated or autonomous fashion.
- A component may exist that post-processes raw experiment data, such as to identify features.

Forces: Only pre-experiment conditions are considered in performing the predetermined actions while running an experiment. Only safety-related conditions during the experiment may be considered. Other changing conditions during the experiment or post-experiment conditions are not considered.

Experiment actions are controlled without significant communication delay to remote components. Proper control capability must be present locally to be able to act in time.

Solution: The is pattern implements the Experiment Control strategy pattern. A local experiment controller executes an experiment using a predetermined experiment plan (Figure 5-7). The plan's execution is automated, performed in an open loop control and may involve human interaction. The controller may monitor the experiment for safety reasons. The plan contains a complete description of the predetermined actions to be performed for running the experiment, including any safety-related responses. Raw experiment data may be post-processed by an optional component, such as to identify features.

As all components are local, a shared storage device may be used between them for sensor and controller data. Control messages between these components orchestrate the control flow.

Capabilities: This pattern offers an open loop control with safety-related feedback on the experiment only. Experiment plan execution is automated, i.e., its list of actions is performed without external or human intervention that can unnecessarily hold up execution. Only 1 experiment is being controlled. There is no significant communication delay to remote components in the open loop control, as the experiment controller is local.

Resulting Context: An experiment is executed automatically using a predetermined plan executed by a local experiment controller, i.e., without significant communication delay to remote components.

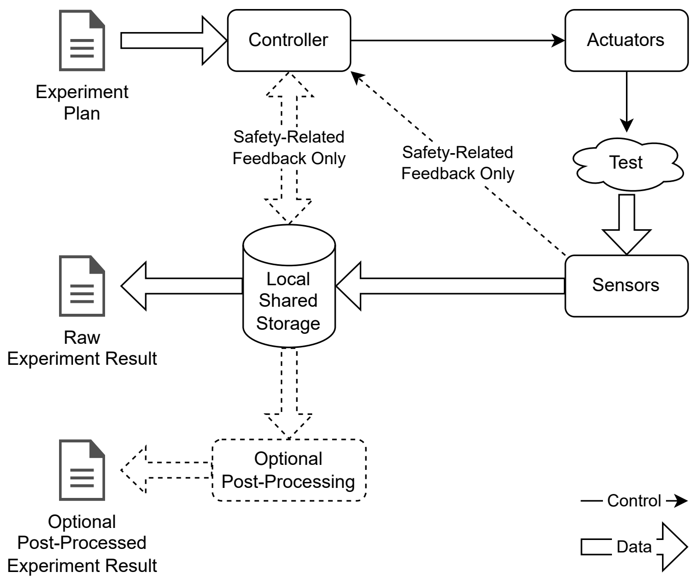


Figure 5-6. Local Experiment Control architectural pattern components and control/data flow

Related Patterns: This architectural pattern implements the Experiment Control strategy pattern.

In contrast to this architectural pattern, the Remote Experiment Control architectural pattern executes a predetermined plan using a remote experiment controller, i.e., with significant communication delay to remote components.

5.2.2 Remote Experiment Control

Name: Remote Experiment Control

Problem: Certain predetermined actions need to be performed while running an experiment. A remote experiment controller executes an experiment, incurring a potentially significant communication delay.

Context: The pattern applies to a system with the following characteristics:

- An experiment plan exists that lists the predetermined actions to be performed while running the experiment.
- A remote experiment controller exists that executes the predetermined actions to be performed while running the experiment.
- Actuators may exist to allow for moving or controlling something before, during and/or after running the experiment.
- Sensors may exist to allow for measuring something before, during and/or after running the experiment.
- Instruments may exist that contain sensors and potentially actuators.
- Robots may exist that contain actuators and potentially sensors and that execute predetermined actions from the experiment plan in an automated or autonomous fashion.
- A component may exist that post-processes raw experiment data, such as to identify features.

Forces: Only pre-experiment conditions are considered in performing the predetermined actions while running an experiment. Only safety-related conditions during the experiment may be considered. Other changing conditions during the experiment or post-experiment conditions are not considered.

Experiment actions are controlled with significant communication delay to remote components. Proper control capability does not need to be local, but must be able to act in time.

Solution: The is pattern implements the Experiment Control strategy pattern. A remote experiment controller executes an experiment using a predetermined experiment plan (Figure 5-7). The plan's execution is automated, performed in an open loop control and may involve human interaction. The controller may monitor the experiment for safety reasons. The plan contains a complete description of the predetermined actions to be performed for running the experiment, including any safety-related responses. Raw experiment data may be post-processed by an optional component, such as to identify features.

As the experiment controller is remote, component-local storage and explicit data transfer between components may be used for sensor and controller data. Control messages between these components orchestrate the control flow.

Capabilities: This pattern offers an open loop control with safety-related feedback on the experiment only. Experiment plan execution is automated, i.e., its list of actions is performed without external or human intervention that can unnecessarily hold up execution. Only 1 experiment is being controlled. There is a significant communication delay to remote components in the open loop control, as the experiment controller is remote.

Resulting Context: An experiment is executed automatically using a predetermined plan executed by a remote experiment controller, i.e., with significant communication delay to remote components.

Related Patterns: This architectural pattern implements the Experiment Control strategy pattern.

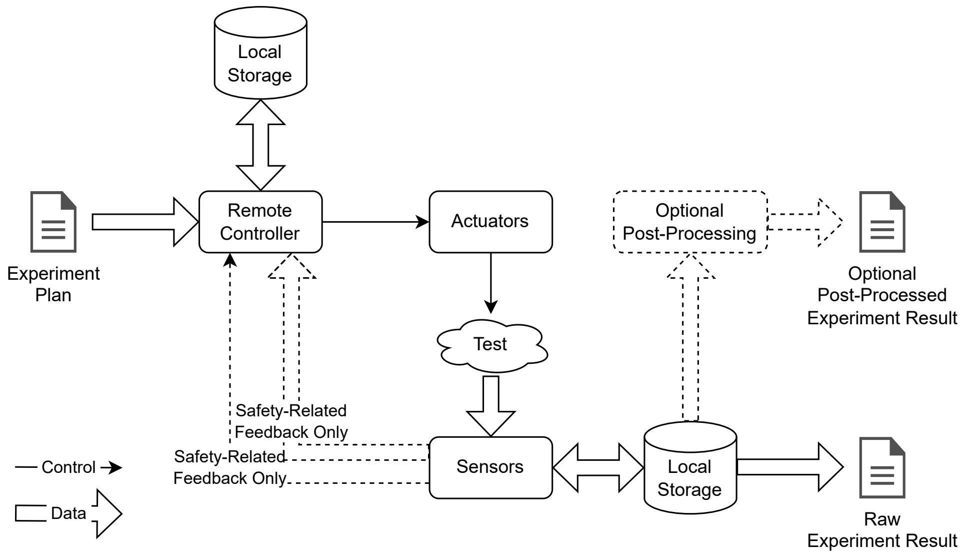


Figure 5-7. Remote Experiment Control architectural pattern components and control/data flow

In contrast to this architectural pattern, the Local Experiment Control architectural pattern executes a predetermined plan using a local experiment controller, i.e., without significant communication delay to remote components.

5.2.3 Local Experiment Steering

Name: Local Experiment Steering

Problem: Certain predetermined actions need to be performed while running an experiment, depending on experiment progress. Experiment progress is analyzed and judged locally. There are no remote components that could incur a significant communication delay.

Context: The pattern applies to a system with the following characteristics:

- An experiment plan exists that lists the predetermined actions to be performed while running the experiment, including potential parameter changes based on experiment progress.
- A local experiment controller exists that executes the predetermined actions to be performed while running the experiment.
- A local experiment analyzer exists that orients the observed information for the experiment controller.
- Sensors exist to allow for measuring experiment progress.
- Actuators may exist to allow for moving or controlling something before, during and/or after running the experiment.
- Additional sensors may exist to allow for measuring something before, during and/or after running the experiment.
- Instruments may exist that contain sensors and potentially actuators.
- Robots may exist that contain actuators and potentially sensors and that execute predetermined actions from the experiment plan in an automated or autonomous fashion.
- A component may exist that post-processes raw experiment data, such as to identify features.

Forces: Only pre-experiment conditions and changing conditions during the experiment are considered in performing the predetermined actions while running an experiment. Post-experiment conditions are not considered.

Experiment progress is analyzed and judged without significant communication delay to remote components. Proper computational analysis and decision making capability must be present locally to be able to respond within a certain amount of time.

Solution: The is pattern implements the Experiment Steering strategy pattern using an observe, orient, decide, and act (OODA) loop control. All components of the OODA loop control are local, i.e., physically located and connected in a way that does not incur a significant communication delay between the components.

As in the Experiment Steering strategy pattern, an experiment controller executes an experiment using a predetermined experiment plan and changes the plan's parameters during execution based on experiment progress (Figure 5-8). The plan's execution is autonomous, performed in a closed loop control and may involve human interaction. The controller may monitor the experiment for safety reasons. The plan contains a complete description of the predetermined actions to be performed for running the experiment, including any safety-related responses and how to analyze and judge experiment progress and change the plan accordingly. Raw experiment data may be post-processed by an optional component, such as to identify features.

The OODA loop control is formed by sensors that observe the experiment, an analyzer that orients the observed information, an experiment controller that decides on appropriate actions and actuators that perform the appropriate actions. As all components of the OODA loop control are local, a shared storage device may be used between them for sensor, analyzer and controller data. Control messages between these components orchestrate the control flow.

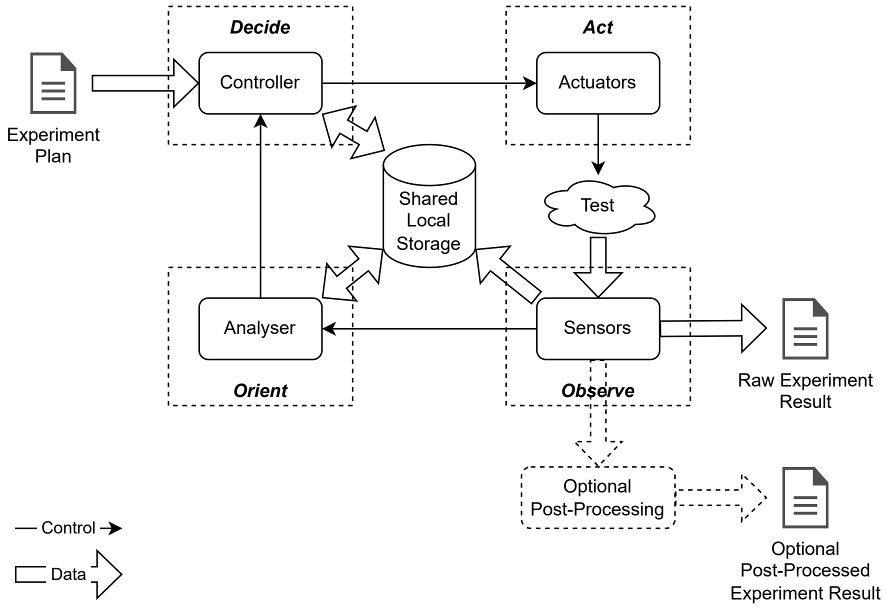


Figure 5-8. Local Experiment Steering architectural pattern components and control/data flow

Capabilities: This pattern offers a closed OODA loop control with safety-related feedback on the experiment and feedback on experiment progress. Experiment plan execution is autonomous, i.e., its list of actions changes during execution based on feedback and is performed without external or human intervention that can unnecessarily hold up execution. Only 1 experiment is being controlled. There is no significant communication delay to remote components in the closed OODA loop control, as the experiment progress analysis is local and the experiment controller is local as well.

Resulting Context: An experiment is executed autonomously using a predetermined experiment plan, with the plan's parameters changing autonomously during the experiment based on experiment progress. Experiment progress is analyzed and judged locally, i.e., without significant communication delay to remote components.

Related Patterns: This architectural pattern implements the Experiment Steering strategy pattern.

In contrast to this architectural pattern, the Remote Experiment Steering architectural pattern analyzes and potentially also judges experiment progress remotely, i.e., with significant communication delay to remote components.

5.2.4 Remote Experiment Steering

Name: Remote Experiment Steering

Problem: Certain predetermined actions need to be performed while running an experiment, depending on experiment progress. Experiment progress is analyzed and optionally also judged/controlled remotely, incurring a potentially significant communication delay.

Context: The pattern applies to a system with the following characteristics:

- An experiment plan exists that lists the predetermined actions to be performed while running the experiment, including potential parameter changes based on experiment progress.
- A local or remote experiment controller exists that executes the predetermined actions to be performed while running the experiment.
- A remote experiment analyzer exists that orients the observed information for the experiment controller.
- Sensors exist to allow for measuring experiment progress.
- Actuators may exist to allow for moving or controlling something before, during and/or after running the experiment.
- Additional sensors may exist to allow for measuring something before, during and/or after running the experiment.
- Instruments may exist that contain sensors and potentially actuators.
- Robots may exist that contain actuators and potentially sensors and that execute predetermined actions from the experiment plan in an automated or autonomous fashion.
- A component may exist that post-processes raw experiment data, such as to identify features.

Forces: Only pre-experiment conditions and changing conditions during the experiment are considered in performing the predetermined actions while running an experiment. Post-experiment conditions are not considered.

Experiment progress is analyzed and optionally also judged with significant communication delay to remote components. Proper computational analysis and decision making capability does not need to be local, but must be able to respond within a certain amount of time.

Solution: The is pattern implements the Experiment Steering strategy pattern using an OODA loop control. The *Orient* component and optionally the *Decide* component of the of the OODA loop control are remote, i.e., physically located and connected in a way that does incur a significant communication delay between the components.

As in the Experiment Steering strategy pattern, an experiment controller executes an experiment using a predetermined experiment plan and changes the plan's parameters during execution based on experiment progress (Figure 5-9). The plan's execution is autonomous, performed in a closed loop control and may involve human interaction. The controller may monitor the experiment for safety reasons. The plan contains a complete description of the predetermined action to be performed for running the experiment, including any safety-related responses and how to analyze and judge experiment progress and change the plan accordingly. Raw experiment data may be post-processed by an optional component, such as to identify features.

The OODA loop control is formed by sensors that observe the experiment, an analyzer that orients the observed information, an experiment controller that decides on appropriate actions and actuators that perform the appropriate actions. As some components of the OODA loop control are remote, component-local storage and explicit data transfer between components may be used for sensor, analyzer and controller data. Control messages between these components orchestrate the control flow.

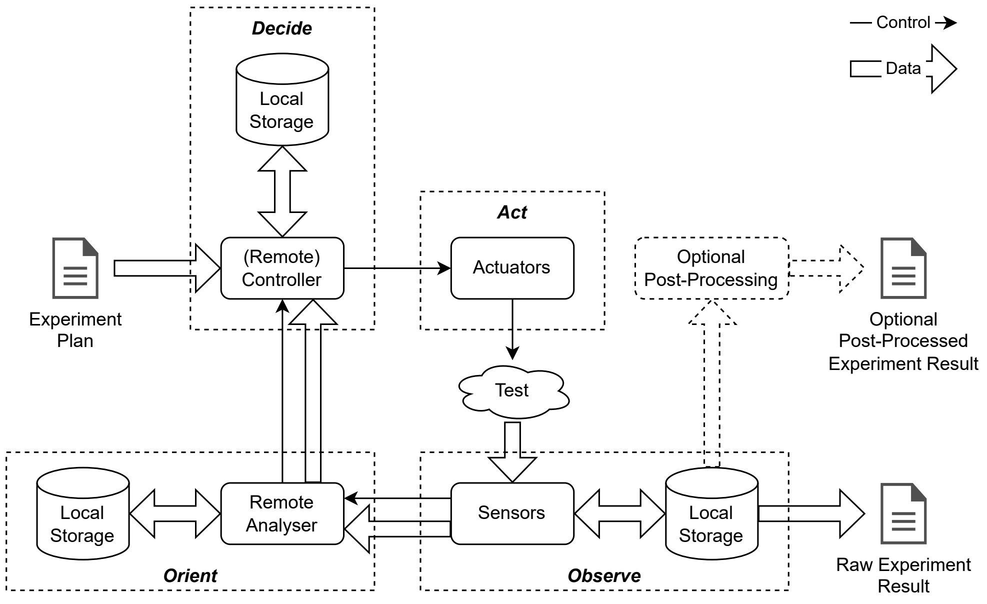


Figure 5-9. Remote Experiment Steering architectural pattern components and control/data flow

Capabilities: This pattern offers a closed OODA loop control with safety-related feedback on the experiment and feedback on experiment progress. Experiment plan execution is autonomous, i.e., its list of actions changes during execution based on feedback and is performed without external or human intervention that can unnecessarily hold up execution. Only 1 experiment is being controlled. There is a significant communication delay to remote components in the closed OODA loop control, as the experiment progress analysis is remotely and the experiment controller may be remote as well.

Resulting Context: An experiment is executed autonomously using a predetermined experiment plan, with the plan's parameters changing autonomously during the experiment based on experiment progress. Experiment progress is analyzed and potentially also judged remotely, i.e., with significant communication delay to remote components.

Related Patterns: This architectural pattern implements the Experiment Steering strategy pattern.

In contrast to this architectural pattern, the Local Experiment Steering architectural pattern analyzes and judges experiment progress locally, i.e., without significant communication delay to remote components.

5.2.5 Local Design of Experiments

Name: Local Design of Experiments

Problem: Certain predetermined actions need to be performed to run a set of similar experiments with different experiment plan parameters, depending on experiment results. Experiment results are analyzed and judged locally. There are no remote components that could incur a significant communication delay.

Context: The pattern applies to a system with the following characteristics:

- An experiment design plan exists that lists the predetermined actions to be performed for creating a new experiment plan based on prior experiment results.
- An initial experiment plan exists that lists the predetermined actions to be performed while running the experiment.
- A local experiment planner exists that creates the new experiment plan based on prior experiment results.
- A local experiment controller exists that executes the predetermined actions to be performed while running the experiment.
- A local experiment analyzer exists that orients the observed information for the experiment planner.
- Sensors exist to allow for measuring experiment results.
- Actuators may exist to allow for moving or controlling something before, during and/or after running the experiment.
- Additional sensors may exist to allow for measuring something before, during and/or after running the experiment.
- Instruments may exist that contain sensors and potentially actuators.
- Robots may exist that contain actuators and potentially sensors and that execute predetermined actions from the experiment plan in an automated or autonomous fashion.
- A component may exist that post-processes raw experiment data, such as to identify features.

Forces: Only pre- and post-experiment conditions are considered in performing the predetermined actions to run a set of similar experiments with different experiment plan parameters. Only safety-related conditions during the experiment may be considered. Other changing conditions during the experiments are not considered, unless the Experiment Steering strategy pattern is being used in conjunction with this architectural pattern, such as by using the Local Experiment Steering or Remote Experiment Steering architectural patterns.

Experiment results are analyzed and judged without significant communication delay to remote components. Proper computational analysis and decision making capability must be present locally to be able to respond within a certain amount of time.

Solution: The is pattern implements the Design of Experiments strategy pattern using an OODA loop control. All components of the OODA loop control are local, i.e., physically located and connected in a way that does not incur a significant communication delay between the components.

As in the Design of Experiments strategy pattern, an experiment controller executes each experiment using a predetermined experiment plan (Figure 5-10). The plan's execution is automated, performed in an open loop control and may involve human interaction. The controller may monitor the experiment for safety reasons. The experiment plan contains a complete description of the predetermined actions to be performed for running the experiment, including any safety-related responses. An experiment planner creates the experiment plan, based on an experiment design plan and prior experiment results (if any). The experiment plan change is autonomous, performed in a closed loop control and may involve human interaction. The experiment design plan contains an initial experiment plan and a plan for creating new experiment plans based on experiment results, including how to analyze and judge experiment results and change the plan accordingly. Raw experiment data may be post-processed by an optional component, such as to identify features.

The OODA loop control is formed by sensors that observe the experiment, an analyzer that orients the observed information, an experiment planner that decides on appropriate actions, and an experiment controller and actuators that perform the appropriate actions. As all components of the OODA loop control are local, a shared storage device may be used between them for sensor, analyzer, planner and controller data. Control messages between these components orchestrate the control flow.

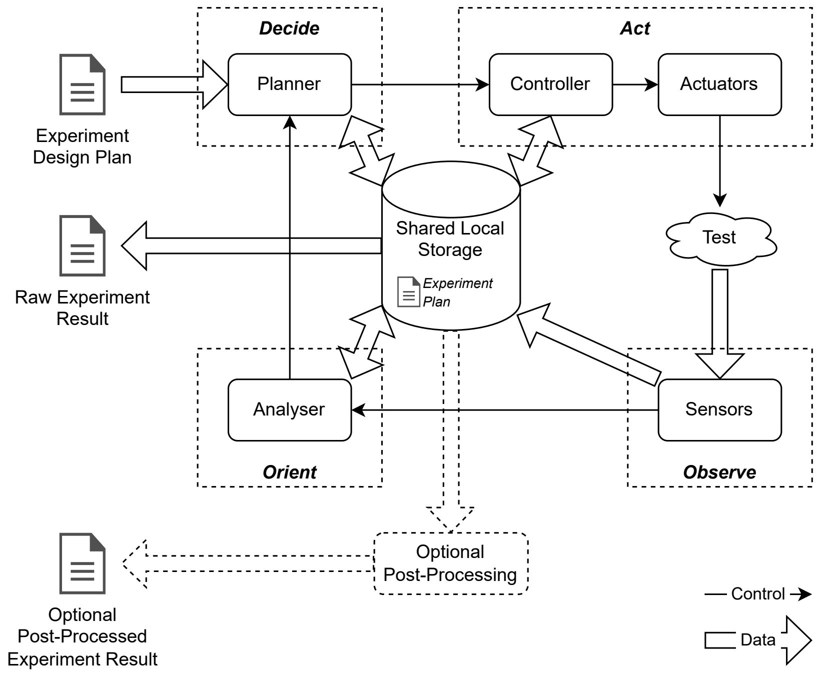


Figure 5-10. Local Design of Experiments architectural pattern components and control/data flow

Capabilities: This pattern offers an open loop control with safety-related feedback on the experiment and a separate closed OODA loop control with feedback on experiment results. Experiment plan execution is automated within the open loop control, i.e., its list of actions is performed without external or human intervention that can unnecessarily hold up execution. Experiment design plan execution is autonomous, i.e., it creates a new experiment plan after each experiment based on experiment results and is performed without external or human intervention that can unnecessarily hold up execution. A set of similar experiments with different experiment plan parameters is controlled. There is no significant communication delay to remote components in the open loop control, as the experiment controller is local. There is also no significant communication delay to remote components in the closed OODA loop control, as the experiment result analysis and experiment planner are local as well.

Resulting Context: An experiment is executed autonomously with different experiment plan parameters using a predetermined experiment plan, with the plan’s parameters changing autonomously between experiments based on experiment results. Experiment results are analyzed and judged locally, i.e., without significant communication delay to remote components.

Related Patterns: This architectural pattern implements the Design of Experiments strategy pattern. It relies on the Experiment Control strategy pattern for automatically executing a predetermined experiment plan. This architectural pattern can be extended using the Experiment Steering strategy pattern (instead of the Experiment Control strategy pattern) for autonomously executing a predetermined experiment plan, with the plan’s parameters changing autonomously during experiments based on experiment progress. Such extension may involve the Local Experiment Steering or Remote Experiment Steering architectural patterns.

In contrast to this architectural pattern, the Remote Design of Experiments architectural pattern analyzes and potentially also judges experiment results remotely, i.e., with significant communication delay to remote components.

5.2.6 Remote Design of Experiments

Name: Remote Design of Experiments

Problem: Certain predetermined actions need to be performed to run a set of similar experiments with different experiment plan parameters, depending on experiment results. Experiment results are analyzed and optionally also judged/controlled remotely, incurring a potentially significant communication delay.

Context: The pattern applies to a system with the following characteristics:

- An experiment design plan exists that lists the predetermined actions to be performed for creating a new experiment plan based on prior experiment results.
- An initial experiment plan exists that lists the predetermined actions to be performed while running the experiment.
- A local or remote experiment planner exists that creates the new experiment plan based on prior experiment results.
- A local experiment controller exists that executes the predetermined actions to be performed while running the experiment.
- A remote experiment analyzer exists that orients the observed information for the experiment planner.
- Sensors exist to allow for measuring experiment results.
- Actuators may exist to allow for moving or controlling something before, during and/or after running the experiment.
- Additional sensors may exist to allow for measuring something before, during and/or after running the experiment.
- Instruments may exist that contain sensors and potentially actuators.
- Robots may exist that contain actuators and potentially sensors and that execute predetermined actions from the experiment plan in an automated or autonomous fashion.
- A component may exist that post-processes raw experiment data, such as to identify features.

Forces: Only pre- and post-experiment conditions are considered in performing the predetermined actions to run a set of similar experiments with different experiment plan parameters. Only safety-related conditions during the experiment may be considered. Other changing conditions during the experiments are not considered, unless the Experiment Steering strategy pattern is being used in conjunction with this architectural pattern, such as by using the Local Experiment Steering or Remote Experiment Steering architectural patterns.

Experiment results are analyzed and optionally also judged with significant communication delay to remote components. Proper computational analysis and decision making capability does not need to be local, but must be able to respond within a certain amount of time.

Solution: The is pattern implements the Design of Experiments strategy pattern using an OODA loop control. The *Orient* component and optionally the *Decide* component of the of the OODA loop control are remote, i.e., physically located and connected in a way that does incur a significant communication delay between the components.

As in the Design of Experiments strategy pattern, an experiment controller executes each experiment using a predetermined experiment plan (Figure 5-11). The plan's execution is automated, performed in an open loop control and may involve human interaction. The controller may monitor the experiment for safety reasons. The experiment plan contains a complete description of the predetermined actions to be performed for running the experiment, including any safety-related responses. An experiment planner creates the experiment plan, based on an experiment design plan and prior experiment results (if any). The experiment plan change is autonomous, performed in a closed loop control and may involve human interaction. The experiment design plan contains an initial experiment plan and a plan for creating new experiment plans based on experiment results, including how to analyze and judge experiment results and change the plan accordingly. Raw experiment data may be post-processed by an optional component, such as to identify features.

The OODA loop control is formed by sensors that observe the experiment, an analyzer that orients the observed information, an experiment planner that decides on appropriate actions, and an experiment controller and actuators that perform the appropriate actions. As some components of the OODA loop control are remote, component-local storage and explicit data transfer between components may be used for sensor, analyzer, planner and controller data. Control messages between these components orchestrate the control flow.

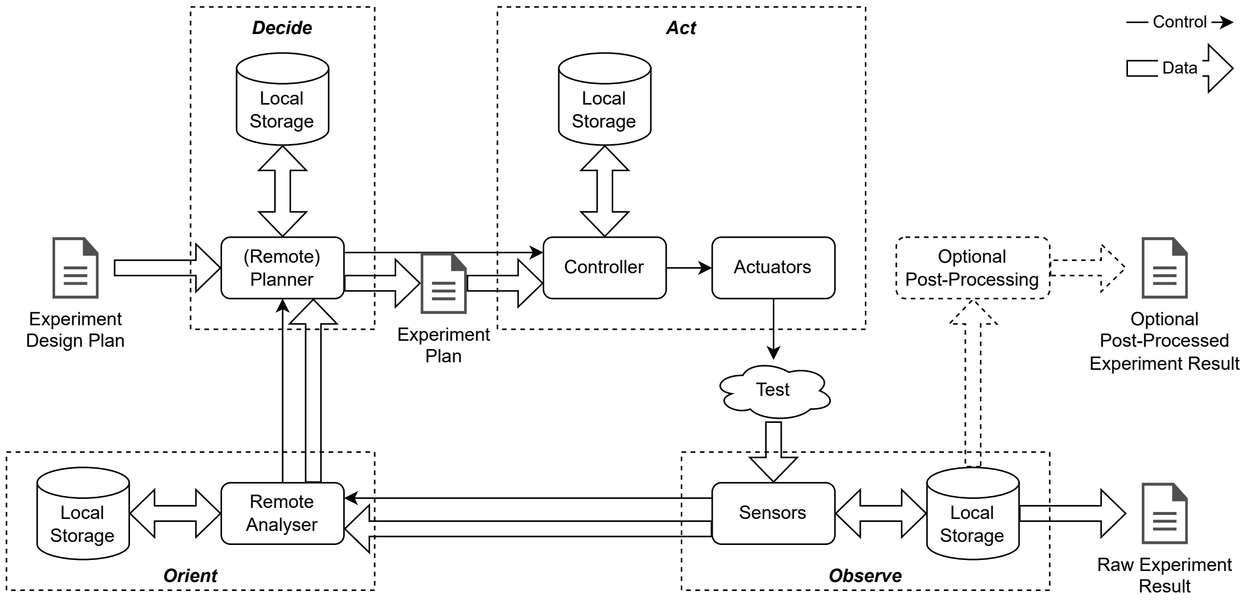


Figure 5-11. Remote Design of Experiments architectural pattern components and control/data flow

Capabilities:

This pattern offers an open loop control with safety-related feedback on the experiment and a separate closed OODA loop control with feedback on experiment results. Experiment plan execution is automated within the open loop control, i.e., its list of actions is performed without external or human intervention that can unnecessarily hold up execution. Experiment design plan execution is autonomous, i.e., it creates a new experiment plan after each experiment based on experiment results and is performed without external or human intervention that can unnecessarily hold up execution. A set of similar experiments with different experiment plan parameters is controlled. There is no significant communication delay to remote components in the open loop control, as the experiment controller is local. There is a significant communication delay to remote components in the closed OODA loop control, as the experiment results are analyzed remotely and the experiment planner may be remote as well.

Resulting Context: An experiment is executed autonomously with different experiment plan parameters using a predetermined experiment plan, with the plan's parameters changing autonomously between experiments based on experiment results. Experiment results are analyzed and potentially also judged remotely, i.e., with significant communication delay to remote components.

Related Patterns: This architectural pattern implements the Design of Experiments strategy pattern. It relies on the Experiment Control strategy pattern for automatically executing a predetermined experiment plan. This architectural pattern can be extended using the Experiment Steering strategy pattern (instead of the Experiment Control strategy pattern) for autonomously executing a predetermined experiment plan, with the plan's parameters changing autonomously during experiments based on experiment progress. Such extension may involve the Local Experiment Steering or Remote Experiment Steering architectural patterns.

In contrast to this architectural pattern, the Local Design of Experiments architectural pattern analyzes and judges experiment results locally, i.e., without significant communication delay to remote components.

6. BUILDING SOLUTIONS USING SCIENCE USE CASE DESIGN PATTERNS

The science use case design patterns detailed in the catalog focus on the inherent open or closed loop control problem as a common problem to be solved in reducing human-in-the-loop requirements with machine-in-the-loop capabilities. Scientific instruments, robot-controlled laboratories and edge/center computing/data resources are connected in a loop control to enable autonomous experiments, "self-driving" laboratories, smart manufacturing, and AI-driven design, discovery and evaluation. Each science use case design pattern consists of a behavior and a set of interfaces in the context of performing a single or a set of experiments in an open or closed loop control. The abstract design pattern definitions describe solutions free of implementation details.

The science use case design patterns are divided into two different classes: (1) strategy patterns that define high-level solution methods using experiment control architecture features at a very coarse granularity, and (2) architectural patterns that define more specific solution methods using hardware and software architecture features at a finer granularity. The architectural patterns inherit the features of their parent strategy patterns, but also address additional problems that are not exposed at the high abstraction level of the strategy patterns.

6.1 A STEP-BY-STEP GUIDE

Building a complete solution from an existing science use case requires dissecting the science use case by the open or closed loop control problem or problems it contains. Each loop control problem needs to be identified, including its properties and hardware/software architectural features. A step-by-step decomposition process would work as follows:

1. Clearly define the experiment or experiments that are being performed
2. Identify the loop control problem or problems that exist for each experiment
3. Classify each loop control problem by a strategy pattern
4. Identify the individual components of each loop control problem and associated strategy pattern
5. Classify each loop control problem by an architectural pattern that matches its strategy pattern
6. Match the identified components with the components of the architectural patterns
7. Design the hardware/software architecture of the solution based on the architectural patterns and the corresponding matched components, using the pattern properties as design guidelines

What is the experiment?: It is important to clearly define the experiment or experiments, as the wrong definition ultimately leads the designer down the wrong path. It is often easier to think of an experiment as a concrete test process that demonstrates a specific known truth, examines the validity of a specific hypothesis, or determines specific properties of something. Clearly identifying the experiment devices, such as sensors, actuators, instruments and robots, is part of that definition as well. It is quite possible that one experiment in a laboratory tries to accomplish multiple objectives, in which case a single multi-objective experiment could be split up into multiple experiments, especially if it involves a workflow or completely separated loop control problems. There is no hard rule on this and any such split would be on a case-by-case basis.

Which loop control problems exist?: Separating out what is being controlled and how is the key to identifying the loop control problem or problems that exist for each experiment. In pretty much all cases, there is some type of simple open loop control, as described in the Experiment Control strategy pattern. Additional loop control problems may exist that may extend the simple open loop control, such as to the

Experiment Steering strategy pattern, or uses/relies on the simple open loop control, such as with the Design of Experiments strategy pattern. There also may be multiple loop control problems for the same experiment, such as a combination of the the Experiment Steering and Design of Experiments strategy patterns. Similarly, a multi-objective experiment may have multiple loop control problems for different parts of the experiment, potentially requiring it to be split up into multiple experiment. Obviously, a multi-experiment workflow may have loop control problems for each experiment in the workflow. Pattern combinations that solve such issues are discussed in Section 6.2.

Who is in control?: The science use case design patterns have one controller component and some have an additional planner component. These are not necessarily physical standalone components. Instead, an analyzer may already contain the decision-making logic and also act as a controller or planner. Similarly, the controller or planner may require human input or may be a human itself. While the goal is to reduce human-in-the-loop requirements with machine-in-the-loop capabilities, this may be a process that requires a transition and some human-in-the-loop requirements may not necessarily completely eliminated.

Which strategy pattern?: The key differences in features between the 4 strategy patterns are (1) no feedback, (2) feedback for the same experiment, (3) feedback for the next experiment, and (4) workflow of multiple experiments. If there is no feedback, then Experiment Control is the right strategy pattern. If there is feedback for the same experiment, such as changing a parameter based on a measurement to observe how that or another measurement changes, then Experiment Steering is the right strategy pattern. If there is feedback for the next experiment, such as to change the parameters and re-run the experiment, then Design of Experiments is the right strategy pattern. There are experiments, where the experiment plan constantly evolves as the experiment is performed, based on measurements. In this case, either Experiment Steering or Design of Experiments may be used, whichever is closer. In this case, using Design of Experiments splits the experiment into multiple separate experiments with different experiment plans. Multi-Experiment Workflow is used whenever there are multiple experiments without feedback. There could be a greater feedback loop over multiple experiments in a workflow. In this case, a separate strategy pattern is employed (see Section 6.2).

What is local? What is remote?: The architectural science use case design patterns distinguish between local and remote components based on communication delay. Any potentially significant communication delay to a component makes it a remote component. The term "significant communication delay" is purposely not clearly defined to give designers room for interpretation. There may be other reasons for defining a component as remote, such as when a component is physically located at an entirely different location that does not necessarily incur a significant communication delay but requires a special way of communication. A human that acts as a planner and communicates with the rest of the system via e-mail or a graphical user interface (GUI) would likely also be considered a remote component.

6.2 PATTERN COMPOSITIONS

A solution may require the composition of science use case design patterns. A simple example from the pattern catalog is the Design of Experiments strategy pattern that already uses the Experiment Control strategy pattern, but could use the Experiment Steering strategy pattern instead. Similarly, the Multi-Experiment Workflow strategy pattern already uses the Experiment Control strategy pattern, but could use the Experiment Steering strategy pattern, the Design of Experiments strategy patterns, or a combination of Experiment Control, Experiment Steering and Design of Experiments strategy patterns instead. This composition of strategy patterns is then also reflected in composition of architectural patterns.

The decision to compose a solution from multiple science use case design patterns depends on the actual properties of the solution. The most significant indicator is the need for multiple, different control loops. Another indicator is the existence of a Multi-Experiment Workflow with different experiments that have different control loops. The number and properties of the control loops typically define the composition of science use case design patterns, from strategy to architecture. Note that there may be more than one control loop implementing the same strategy and even architectural pattern, but with different properties. For example, there may be multiple Local Experiment Steering control loops that are independent from each other. They may operate with different timing requirements, perform analysis on different computational resources and modify different parameters independent from each other.

The following example illustrates the composition of science use case design patterns. In this solution, there is a control loop for Experiment Steering to change parameters based on observation as the experiment is progressing. There is also a second control loop for Design of Experiments to change the Experiment Plan based on the prior experiment result after each experiment. Figure 6-1 illustrates the involved components and control/data flow of the Experiment Steering and the Design of Experiments strategy pattern composition. The Experiment Design Plan and the Experiment Planner are exclusive parts of the Design of Experiments strategy pattern, while the other components are part of the Experiment Steering strategy pattern that the Design of Experiments strategy pattern is using as its experiment to control from an Experiment Plan perspective.

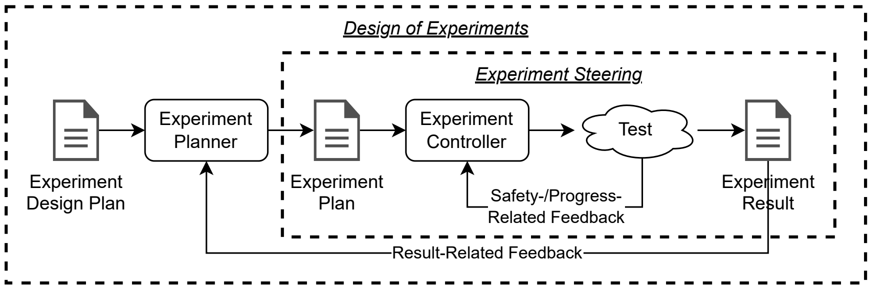


Figure 6-1. Example: Components and control/data flow of Experiment Steering and Design of Experiments strategy pattern composition

In the given science use case example, the Experiment Steering utilizes a local shared storage device, such as a small network attached storage (NAS), for all sensor data and its analysis results. It also relies on a local computational resource, such as an NVIDIA Jetson Nano, for analysis and decision making. The Design of Experiments transfers the sensor data of the entire experiment from the shared storage device to a remote analyzer, such as an NVIDIA DGX system. Its analysis results are evaluated and a new experiment plan is created by the Controller on a desktop system running a GUI. The corresponding involved components and control/data flow of the Local Experiment Steering and the Remote Design of Experiments architectural pattern composition is shown in Figure 6-2.

This is just an example of how a solution may require the composition of science use case design patterns. Different logical components may utilize the same physical components, such as when different control loops use the same storage device or the same computational resource for analysis and/or control. For example, separate controllers for different Experiment Steering control loops may use exactly the same physical component, such as a Raspberry Pi, for storing and analyzing sensor data and for issuing different, non-conflicting control commands to a robot.

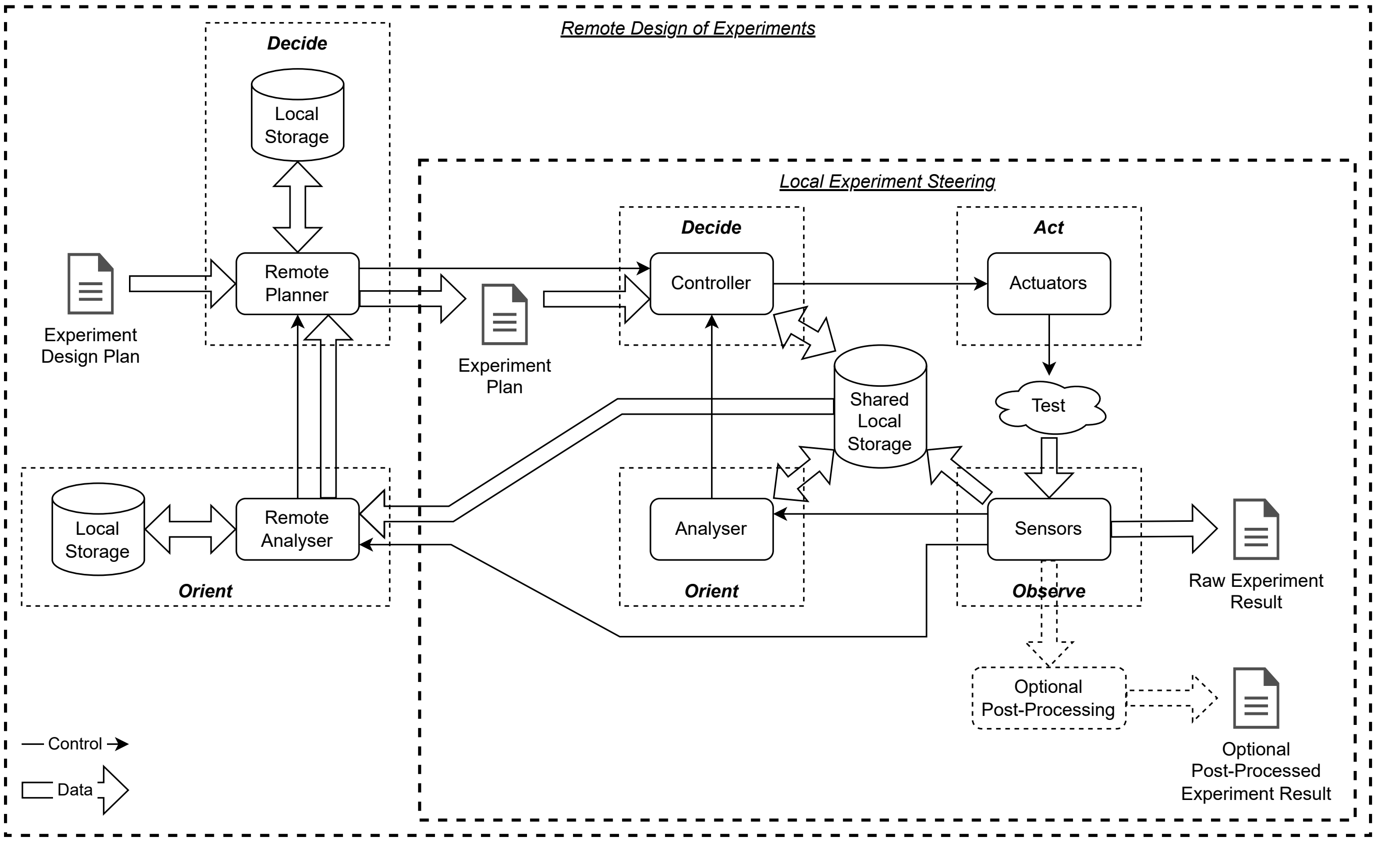


Figure 6-2. Example: Components and control/data flow of Local Experiment Steering and Remote Design of Experiments architectural pattern composition

REFERENCES

[1] DOE national laboratories' computational facilities – Research workshop report. Technical Report ANL/MCS-TM-388, Argonne National Laboratory, Lemont, IL, USA, February 2020. URL https://publications.anl.gov/anlpubs/2020/02/158604.pdf.

[2] Christopher Alexander, Sara Ishikawa, and Murray Silverstein. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, August 1977. ISBN 978-0195019193.

[3] Jan Borchers. *A Pattern Approach to Interaction Design*. John Wiley & Sons, Inc., New York, NY, USA, 2001. ISBN 978-0471498285.

[4] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture - Volume 1: A System of Patterns*. Wiley Publishing, August 1996. ISBN 978-0-471-95869-7.

[5] Frank Buschmann, Kevlin Henney, and Douglas C. Schmidt. *Pattern-Oriented Software Architecture - Volume 4: A Pattern Language for Distributed Computing*. Wiley Publishing, 2007. ISBN 978-0470059029.

[6] Murray Cole. *Algorithmic Skeletons: Structured Management of Parallel Computation*. MIT Press, Cambridge, MA, USA, 1991. ISBN 978-0273088073.

[7] Murray Cole. Bringing skeletons out of the closet: A pragmatic manifesto for skeletal parallel programming. *Parallel Computing*, 30(3):389–406, March 2004. ISSN 0167-8191. doi: 10.1016/j.parco.2003.12.002. URL https://dx.doi.org/10.1016/j.parco.2003.12.002.

[8] Chad Dougherty, Kirk Sayre, Robert Seacord, David Svoboda, and Kazuya Togashi. Secure design patterns. Technical Report CMU/SEI-2009-TR-010, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2009. URL https://dx.doi.org/10.1184/R1/6583640.v1.

[9] Douglas K. Van Duyne, James Landay, and Jason I. Hong. *The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002. ISBN 978-0201721492.

[10] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002. ISBN 978-0321127426.

[11] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Professional, 1994. ISBN 978-0201633610.

[12] Jeffrey Heer and Maneesh Agrawala. Software design patterns for information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):853–860, September 2006. ISSN 1077-2626. doi: 10.1109/TVCG.2006.178. URL https://dx.doi.org/10.1109/TVCG.2006.178.

[13] Saurabh Hukerikar and Christian Engelmann. Resilience design patterns: A structured approach to resilience at extreme scale (version 1.2). Technical Report ORNL/TM-2017/745, Oak Ridge National Laboratory, Oak Ridge, TN, USA, August 2017. URL https://dx.doi.org/10.2172/1436045.

[14] Saurabh Hukerikar and Christian Engelmann. Resilience design patterns: A structured approach to resilience at extreme scale. *Journal of Supercomputing Frontiers and Innovations (JSFI)*, 4(3):4–42,

October 2017. ISSN 2409-6008. doi: 10.14529/jsfi170301. URL https://dx.doi.org/10.14529/jsfi170301.

[15] Kurt Keutzer and Timothy Mattson. Our pattern language (OPL): A design pattern language for engineering (parallel) software. 2009. URL https://patterns.eecs.berkeley.edu/?page_id=98.

[16] Michael Kircher and Prashant Jain. *Pattern-Oriented Software Architecture, Volume 3: Patterns for Resource Management*. Wiley Publishing, 2004. ISBN 978-0470845257.

[17] Timothy Mattson, Beverly Sanders, and Berna Massingill. *Patterns for Parallel Programming*. Addison-Wesley Professional, first edition, 2004. ISBN 978-0321228116.

[18] Michael McCool, James Reinders, and Arch Robison. *Structured Parallel Programming: Patterns for Efficient Computation*. Elsevier, 2012. ISBN 978-0-12-415993-8.

[19] Michael D. McCool. Structured parallel programming with deterministic patterns. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Parallelism (HotPar) 2010*. USENIX Association, June 2010. URL https://www.usenix.org/legacy/event/hotpar10/tech/full_papers/McCool.pdf.

[20] Douglas C. Schmidt, Michael Stal, Hans Rohnert, and Frank Buschmann. *Pattern-Oriented Software Architecture Volume 2: Patterns for Concurrent and Networked Objects*. Wiley Publishing, 2000. ISBN 978-0471606956.

[21] Rick Stevens, Valerie Taylor, Jeff Nichols, Arthur Barney Maccabe, Katherine Yelick, and David Brown. AI for science report, March 2020. URL https://www.anl.gov/ai-for-science-report.

[22] Jerry Talton, Lingfeng Yang, Ranjitha Kumar, Maxine Lim, Noah Goodman, and Radomír Měch. Learning design patterns with bayesian grammar induction. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST) 2012*, pages 63–74, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1580-7. doi: 10.1145/2380116.2380127. URL https://dx.doi.org/10.1145/2380116.2380127.

[23] U.S. Department of Defense. The DoDAF architecture framework version 2.02, August 2010. URL https://dodcio.defense.gov/Library/DoD-Architecture-Framework.