Science Use Case Design Patterns for Autonomous Experiments

Christian Engelmann
Suhas Somnath
engelmannc@ornl.gov
somnaths@ornl.gov
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA

ABSTRACT

Connecting scientific instruments and robot-controlled laboratories with computing and data resources at the edge, the Cloud or the high-performance computing (HPC) center enables autonomous experiments, self-driving laboratories, smart manufacturing, and artificial intelligence (AI)-driven design, discovery and evaluation. The Self-driven Experiments for Science / Interconnected Science Ecosystem (INTERSECT) Open Architecture enables science breakthroughs using intelligent networked systems, instruments and facilities with a federated hardware/software architecture for the laboratory of the future. It relies on a novel approach, consisting of (1) science use case design patterns, (2) a system of systems architecture, and (3) a microservice architecture. This paper introduces the science use case design patterns of the INTERSECT Architecture. It describes the overall background, the involved terminology and concepts, and the pattern format and classification. It further offers an overview of the 12 defined patterns and 4 examples of patterns of 2 different pattern classes. It also provides insight into building solutions from these patterns. The target audience are computer, computational, instrument and domain science experts working in the field of autonomous experiments.

CCS CONCEPTS

• **Software and its engineering → Software architectures**.

KEYWORDS

smart laboratories, autonomous experiments, federated ecosystem, system architecture, design patterns

ACM Reference Format:
Christian Engelmann and Suhas Somnath. 2023. Science Use Case Design Patterns for Autonomous Experiments. In *EuroPLoP '23: 28th European Conference on Pattern Languages of Programs, July 05–09, 2023, Kloster Irsee, Germany*. ACM, New York, NY, USA, 14 pages. https://doi.org/XXXXXXX.XXXXXXX

EuroPLoP '23, July 05–09, 2023, Kloster Irsee, Germany

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The U.S. Department of Energy (DoE)'s AI for Science reports [8, 45] outline 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 [52]. The DoE's Computational Facilities Research Workshop report [11] 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 control that collects experiment data, analyzes it, and takes appropriate 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 or physics-based models, and advanced statistical methods. Human interaction for experiment planning, observation and steering is performed through appropriate interfaces.

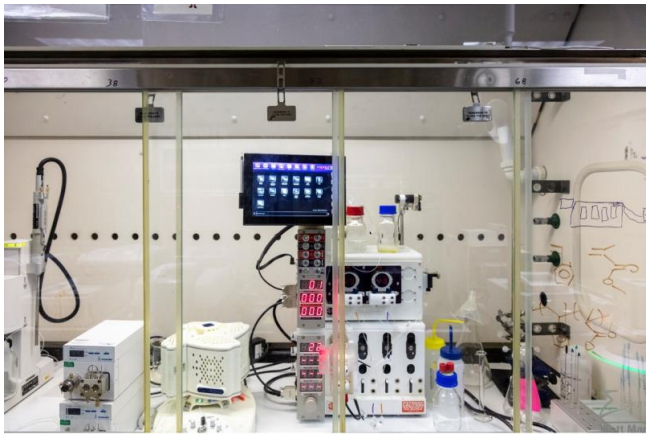


Figure 1: The INTERSECT autonomous robotic chemistry laboratory operates 24/7 using analysis of experimental data for the design of experiments.

For example, both the rate and output of traditional materials synthesis and discovery are currently too slow and too small to efficiently provide needed advances. An autonomous robotic chemistry laboratory (ACL) (Figure 1) can operate 24/7 with high precision to greatly accelerate materials discovery and innovation. It relies on the design of a laboratory utilizing robotic and autonomous tools for the manipulation of laboratory equipment and characterization

tools. A robotic platform with three major components is used: a mobile base, a robotic arm, and software/characterization tools including integration/feedback with AI.

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. 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 architecture needs to be an open standard to enable adoption.

This paper introduces the science use case design patterns of the INTERSECT Open Architecture Specification. The basic template for a science use case design pattern is defined in a loop control problem paradigm. At the moment, there are two classes of science use case design patterns, based (1) on high-level solution methods using experiment control architecture features at a very coarse granularity and (2) on more specific solution methods using hardware and software architecture features at a finer granularity. The classification scheme itself is open for extension, such as for adding new patterns for each class or new classes entirely. For example, a new class may map the existing patterns to other workflow properties, such as (a) data-intensive, (b) time-sensitive and (c) long-term experiment campaigns. The paper describes the overall background, the involved terminology and concepts, and the pattern format and classification. It further offers an overview of the defined patterns and 4 examples of patterns of 2 different pattern classes. It also provides insight into building solutions from these patterns. The target audience are computer, computational, instrument and domain science experts working in the field of autonomous experiments.

2 BACKGROUND AND RELATED WORK

The following discusses the overall background of the presented work, the overarching approach of the INTERSECT Open Architecture. It also describes relevant work related to the individual components of the architecture, including research and development in scientific workflows, design patterns, system of systems (SoS) architectures, and microservice architectures.

2.1 The INTERSECT Open Architecture

The INTERSECT Open Architecture approach [14] roughly follows the U.S. Department of Defense Architecture Framework (DoDAF) [49] with its different architectural viewpoints, such as (i) operational scenarios, (ii) composition, interconnectivity and context, (iii) services and their capabilities, (iv) policies, standards and guidance, and (v) capability. The major difference is that the INTERSECT Open Architecture splits these views over three different parts: (1) science use case design patterns [15], (2) a SoS architecture [32], and (3) a microservice architecture [5].

Science use cases for autonomous experiments, self-driving laboratories, smart manufacturing, and AI-driven design, discovery and evaluation are described as design patterns that identify and abstract the involved components and their control, work and data flow interactions. The SoS architecture clarifies used terms, architectural elements, the interactions between them, and compliance. The microservice architecture maps the patterns to the SoS architecture with loosely coupled microservices and standardized interfaces.

This approach permits separating coarse-, mid- and fine-grain architectural decisions. Coarse-grain architectural decisions define as what objective a particular self-driving laboratory has and how that objective is being achieved. Mid-grain architectural decisions clarify which instruments, robots, networks and computing systems are part of this self-driving laboratory and how do they communicate with each other. Fine-grain architectural decisions describe which particular experiment control, data transfer and compute microservices are being used and how. The science use case design patterns, SoS architecture and microservice architecture complement each other, just like the different viewpoints of the DoDAF. Additionally, the SoS architecture itself offers complementary viewpoints, such as user, data, operational, logical, physical and standards view.

2.2 Scientific Workflows

There are about 300 workflow solutions for instrument science and data analysis [2]. Only few holistic automated solutions exist.

The National Energy Research Scientific Computing Center (NERSC) Superfacility framework [37] integrates instruments with computational/data facilities for automation. For example, it connects the SLAC National Accelerator Laboratory's Linac Coherent Light Source via ESnet to the Cori supercomputer for photosynthesis research [48]. Its RESTful Superfacility API offers access to common supercomputer functions [36]. Oak Ridge National Laboratory (ORNL) offers federated environments for connecting instruments with computational/data resources. It leverages software containerization and softwarization of hardware for processing data from ORNL's Spallation Neutron Source and High Flux Isotope Reactor [1, 38]. Data transfer and workflow tools developed at Argonne National Laboratory (ANL) and the University of Chicago, such as Globus Automate [22], Gladier [21] and Balsam [3], permit automated analysis of instrument data. For example, they connect ANL's Advanced Photon Source with the Theta supercomputer for real-time analysis [24]. Other solutions exist, such as the ACL at the University of Liverpool [43], the FireCrest RESTful API at the Swiss National Supercomputing Centre [16], and the design of experiments as a Cloud service by Kebotix [29].

2.3 Design Patterns

Design patterns systematize software development using proven engineering paradigms and methodologies [7]. In object-oriented programming, design patterns provide methods for defining class interfaces, inheritance hierarchies and class relationships [19]. Pattern systems also exist for concurrent and networked object-oriented environments [44], resource management [30], and distributed systems [6]. Design patterns have been discovered in other domains, such as for natural language processing [46], user interface design [4], Web design [13], visualization [23], software security [12],

HPC resilience [25, 26], and data processing for automation of business processes [18].

Execution patterns, not design patterns, for workflows in general describe the functionality of a workflow [50], such as execution graphs, decision points and synchronization points. Common motifs in scientific workflows [20] start making the connection between the functionality of a workflow and certain common execution patterns, such as data movement and data analysis steps. Similar workflow execution patterns, not design patterns, have been recently proposed for instrument science [51].

2.4 System of Systems Architectures

The SoS approach designs a highly complex system by decomposing it into many smaller and easier to design systems [35, 41]. The set of systems interact to provide a unique capability that none of the individual systems can accomplish on its own [27]. A SoS has five key characteristics [33]: operational independence of systems, managerial independence of systems, geographical distribution, emergent behavior, and evolutionary development. Systems are individually developed and evolved, as the architecture of a SoS is the system interfaces [34, 42]. A recent example is Defense Advanced Research Projects Agency (DARPA)'s System of Systems Integration Technology and Experimentation (SoSITE) [10] System-of-systems Technology Integration Tool Chain for Heterogeneous Electronic Systems (STITCHES) [9, 17]. The DoDAF [49] is an overarching, comprehensive framework for the development of architectures from different viewpoints. It is used across the U.S. Department of Defense (DoD) for developing SoS architectures.

2.5 Microservice Architectures

Microservice architectures emerged from service-oriented architectures, initially realized with Web services [53]. They have since become the modern approach to decompose complex software systems. For example, Netflix created an open source microservice architecture for their internal applications [39, 40]. Kubernetes uses a microservice architecture for automating deployment, scaling, and management of containerized applications [31].

3 TERMINOLOGY AND CONCEPTS

This section describes the relevant terminology and concepts, thematically grouped and ordered by their relationships. This is by no means an exhaustive list, but rather represents the core descriptions needed to understand this paper.

General Terms:

- **Test:** A procedure or a method to evaluate the characteristics of a product, service, or system under specific conditions. For example, characterizing the chemical composition of a compound in a gas chromatograph.
- **Experiment:** A test under controlled conditions to demonstrate a known truth or examine the validity of a hypothesis. For example, creating a compound based on the hypothesis that it has a certain chemical composition, characterizing the chemical composition of the compound in a gas chromatograph, and analyzing the result to examine the validity of the hypothesis.
- **Multi-experiment workflow:** A set of experiments performed in serial (one after another) and/or in parallel (simultaneously). For example, a created compound is characterized with different tools, including a gas chromatograph, to examine the validity of multiple hypotheses. This may be performed by splitting the compound up and performing the experiments simultaneously (parallel), or by reusing the compound in subsequent (serial) experiments.
- **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. For example, performing multiple experiments involving a gas chromatograph in which different compounds are created and characterized to find an optimal compound for a specific practical application.

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 multiple experiments in a workflow, i.e., a set of experiments in serial and/or parallel. Each experiment in this workflow still has its own experiment plan.
- **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.
- **Self-driving:** Synonymous with autonomous operation.

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.
- **Laboratory:** A room or building equipped with experiment devices, such as sensors, actuators, instruments, and robots.

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.
- **Observe, orient, decide, and act (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 [47].
- **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.

Other Terms:

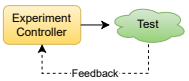
- **Smart manufacturing:** Computer-integrated manufacturing with high levels of adaptability and rapid design changes, treating the manufacturing process as series of experiments that improve the product through feedback.
- **AI-driven design, discovery and evaluation:** The use of AI technology in product design, scientific discovery, or product evaluation/testing.

4 PATTERN FORMAT

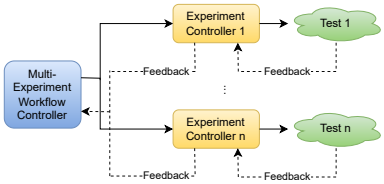
Reducing human-in-the-loop requirements with machine-in-the-loop capabilities by connecting scientific instruments, robot-controlled laboratories and edge/center computing/data resources to enable autonomous experiments, self-driving laboratories, smart manufacturing, and AI-driven design, discovery 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.

Figure 2 shows two different loop control problems. Figure 2a describes a closed loop control of an experiment that performs a test with some feedback to an experiment controller running the test. Figure 2b describes a multi-experiment workflow with a closed loop control of multiple experiments, each with their own a closed loop control. There are a number of different loop control problems that the science use case design patterns systematize and categorize.

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 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



(a) Experiment loop control problem



(b) Multi-experiment workflow loop control problem

Figure 2: Science use case as a loop control problem

descriptions. Patterns are described in the traditional design pattern paradigm: from context to problem to solution to resulting context. 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.

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

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.

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. The specific capabilities provided by this pattern.

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.

Examples: A description of one or more examples, including their specific pattern properties, that illustrate the use of the pattern for solving concrete problems.

Known Uses: A list of known applications of the pattern in existing systems, including any practical considerations and limitations.

5 PATTERN CLASSIFICATION

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

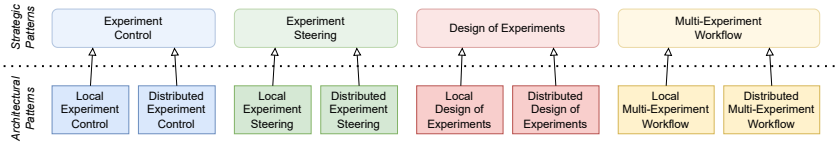


Figure 3: Classification of the science use case design patterns

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.

There are currently two classes of science use case design patterns (Figure 3): (1) strategic 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 strategic patterns, they also address additional problems that are not exposed at the high abstraction level of the strategic patterns.

Strategic patterns currently focus on the differences in experiment control features, such as steering of an ongoing experiment using live experimental data vs. design of the next experiment(s) using past experimental data. The key differences in features between the 4 strategic patterns are (1) no feedback, (2) feedback for the same experiment, (3) feedback for the next experiment, and (4) workflow of multiple experiments.

The primary feature currently explored by the architectural patterns is the distinction between local and remote components used by a corresponding strategic pattern, where local means that there is not a potentially significant communication delay to a component and remote means that there is a potentially significant communication delay to a component. Other architectural features may be explored in the future with different patterns.

For example, the Experiment Steering strategic pattern is used in every experiment, where live feedback of experiment data is being used to autonomously change parameters during the experiment. Known uses range from a simple proportional–integral–derivative (PID) controller to complex probabilistic approaches or domain science informed AI in the feedback loop. The Distributed Experiment Steering architectural pattern inherits all the properties of the Experiment Steering strategic pattern, but has the architectural property of potentially significant communication delay between the experiment and a remote analysis. This severely restricts real-time feedback. In contrast, the Local Experiment Steering architectural pattern also inherits all the properties, but experiment progress is analyzed and judged locally, i.e., without significant communication delay to remote components.

This classification scheme is open for extension. New patterns may be added for each class if new strategic or architectural patterns emerge that do not fit in the existing patterns. New classes may be added if new pattern features emerge that express commonalities across workflows that are not covered by patterns. For example, a new class may map the existing patterns to data-intensive, time-sensitive and long-term experiment campaigns, which are workflow features that are orthogonal to the current pattern classes. Another new class may focus on the algorithms used in the feedback loop, such as probabilistic (e.g., Bayesian) vs. domain science based (e.g., physics informed) algorithms.

6 STRATEGIC PATTERNS

The science use case strategic 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 strategic patterns:

- *Experiment Control*: Certain predetermined actions need to be performed while running an experiment. This pattern would be used in all automated experiments that do not have feedback for steering the ongoing or designing the next experiment. Since autonomous operation requires to first figure out automation, this pattern provides a basic solution that covers most experiments performed at this point.
- *Experiment Steering*: Certain predetermined actions need to be performed while running an experiment to positively influence experiment progress. This pattern involves feedback for the ongoing experiment as an extension to Experiment Control. It offers autonomous operation and is used in experiments that require live feedback to adjust parameters.
- *Design of Experiments*: Certain predetermined actions need to be performed to run a set of similar experiments with different experiment plan parameters, depending on (prior) experiment results. This pattern makes use of either Experiment Control or Experiment Steering and additionally offers feedback between experiments, typically to define the parameters of the next experiment or next series of experiments. It is typically used in conjunction with probabilistic (e.g., Bayesian) or domain science based (e.g., physics informed) analysis of experiment results. This pattern is predominantly used in large-scale parameter studies, such as to find the optimal conditions of a chemical catalysis.
- *Multi-Experiment Workflow*: Certain predetermined actions need to be performed to run a set of experiments in serial (one after another) and/or in parallel (simultaneously). This pattern utilizes the other 3 patterns to orchestrate multiple experiments that may depend on each other. An example use case is the creation of a certain material using physical and/or chemical processes (e.g., catalysis) and the analysis of the properties of the created material in multiple experiments (e.g., spectroscopy and stress testing).

The features of these science use case strategic patterns and their relationships are compared in Table 1.

Table 1: Feature comparison and relationships of the science use case strategic 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

In the following, the Experiment Steering and Design of Experiments strategic patterns are described as examples. The full design pattern catalog is beyond the scope of this paper. A preliminary version of the catalog has already been published [15] and an improved version based on this paper is forthcoming.

6.1 Example: Experiment Steering

Name: Experiment Steering

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.

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

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 4). 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.

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.

Figure 4: Experiment Steering strategic pattern components and control/data flow

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

Examples: An autonomous microscopy science use case [28] (Figure 5) implements the Experiment Steering strategic pattern, as an ongoing scanning transmission electron microscopy (STEM) experiment is controlled by analyses of periodic experiment data. At the strategic pattern level of abstraction, the individual pattern components are as follows:

- The experiment plan contains a complete description of the predetermined actions to be performed for running the experiment, including any parameters for operating the STEM, safety-related responses and how to analyze and judge experiment progress and change the plan accordingly. The experiment plan also contains the goal of the STEM experiment to steer it in the right direction and to stop its closed loop control upon completion.
- The experiment controller executes a STEM experiment using a predetermined experiment plan and changes the plan's parameters during execution based on experiment progress. The plan's execution is autonomous, performed in a closed loop control and may involve human interaction.
- The test performed in a STEM experiment determines the properties of microscopic structures.
- The experiment result is a combination of raw and analyzed STEM data and insights derived from this data.

Known Uses: This strategic pattern is used in every experiment, where live feedback of experiment data is being used to autonomously change parameters during the experiment. Known uses range from

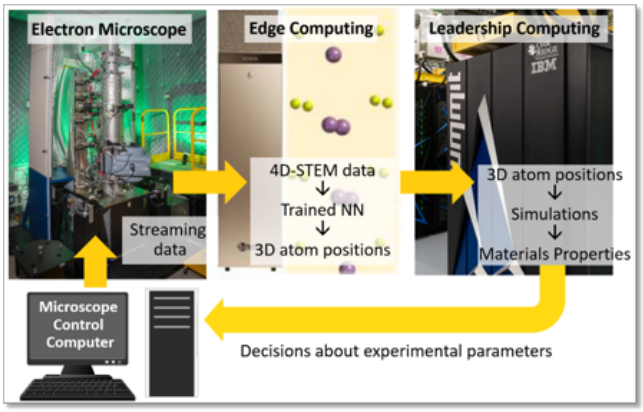


Figure 5: The autonomous INTERSECT scanning transmission electron microscopy experiment investigates and modifies samples at atom-scale using analysis of periodic experimental data.

having simple PID controller to complex probabilistic or domain science informed AI in the feedback loop.

6.2 Example: Design of Experiments

Name: Design of Experiments

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.

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

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 strategic pattern is being used in conjunction with this strategic pattern.

Solution: An experiment controller executes each experiment using a predetermined experiment plan (Figure 6). 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.

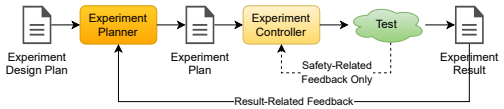


Figure 6: Design of Experiments strategic pattern components and control/data flow

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

Examples: An ACL science use case (Figure 1) implements the Design of Experiments strategic pattern, as a robot automates experiment execution and the software/characterization tools in the feedback loop to plan the experiments to be performed. At the strategic pattern level of abstraction, the individual pattern components are as follows:

- The experiment design plan describes the goal, which is the desired chemical compound, and the logic necessary to craft subsequent experiments towards achieving the goal.
- The experiment planner is the subject matter expert (SME) that may be substituted by a machine learning or deep learning model for autonomous operation to decide on the next experiment plan, given the results from past experiments.

- The experiment plan is the sequence of predetermined steps and associated parameters necessary to run the experiment. The predetermined steps include the parameters for synthesizing the chemical compound, route navigation instructions for the robots to move the sample between the different synthesis and characterization stations, parameters for characterizing the synthesized chemical compound, and safety related feedback instructions.
- The test performed in an experiment characterizes the synthesized chemical compound.
- The experiment controller is a central workstation that is able to command and control the robots, synthesis equipment, analytical instruments, and any data and computing resources for analyzing the measurement data.
- The experiment result is a combination of the sample characterization results.

The experiment is a complex sequence of steps involving multiple instruments, actuators, sensors, etc. Thus, the experiment itself could be considered a Multi-Experiment Workflow strategic pattern using a sequence of Experiment Control strategic patterns. Examples of steps that constitute the Multi-Experiment Workflow strategic include the synthesis step and each of the individual characterization steps, such as the gas chromatography, high performance liquid chromatography, and X-ray microscopy. Some of these steps could potentially be performed in parallel if the sample were broken down into pieces such that the pieces could be analyzed be the characterization instruments in parallel.

Known Uses: This strategic pattern is used in every experiment, where feedback of experiment results is being used to autonomously change the parameters of the next experiment(s). Known uses range from having simple linear or random parameter scan to complex probabilistic approaches (e.g., Bayesian design of experiments) or domain science informed AI (e.g., physics-informed design of experiments) in the feedback loop.

7 ARCHITECTURAL PATTERNS

Architectural patterns define more specific solution methods using hardware and software architecture features at a finer granularity. They inherit the features of their parent strategic patterns. However, they address additional problems through design choices that are not exposed at the high abstraction level of the parent strategic patterns. They address different choices of implementing experiment control and workflow, such as using experiment-local, edge and/or center computing/data resources. The catalog of science use case design patterns defines the following architectural patterns:

- *Local Experiment Control*: A local experiment controller executes an experiment. There are no remote components that could incur a significant communication delay.
- *Distributed Experiment Control*: A remote experiment controller executes an experiment, incurring a potentially significant communication delay.
- *Local Experiment Steering*: Experiment progress is analyzed and judged locally. There are no remote components that could incur a significant communication delay.
- *Distributed Experiment Steering*: Experiment progress is analyzed and optionally also judged/controlled remotely, incurring a potentially significant communication delay.
- *Local Design of Experiments*: Experiment results are analyzed and judged locally. There are no remote components that could incur a significant communication delay.
- *Distributed Design of Experiments*: Experiment results are analyzed and optionally also judged/controlled remotely, incurring a potentially significant communication delay.
- *Local Multi-Experiment Workflow*: All experiments are local. There are no remote experiments that could incur a significant communication delay.
- *Distributed Multi-Experiment Workflow*: One or more experiments are remote, incurring a potentially significant communication delay.

Table 2 shows the architectural patterns and their relationships to the strategic patterns.

Table 2: Relationships of the science use case strategic and architectural patterns

Architectural Pattern	Implements Strategic Pattern
Local Experiment Control	Experiment Control
Distributed Experiment Control	Experiment Control
Local Experiment Steering	Experiment Steering
Distributed Experiment Steering	Experiment Steering
Local Design of Experiments	Design of Experiments
Distributed Design of Experiments	Design of Experiments
Local Multi-Experiment Workflow	Multi-Experiment Workflow
Distributed Multi-Experiment Workflow	Multi-Experiment Workflow

In the following, the Distributed Experiment Steering and Local Design of Experiments architectural pattern are described.

7.1 Example: Distributed Experiment Steering

Name: Distributed Experiment Steering

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.

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

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 strategic 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 strategic 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 7). 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.

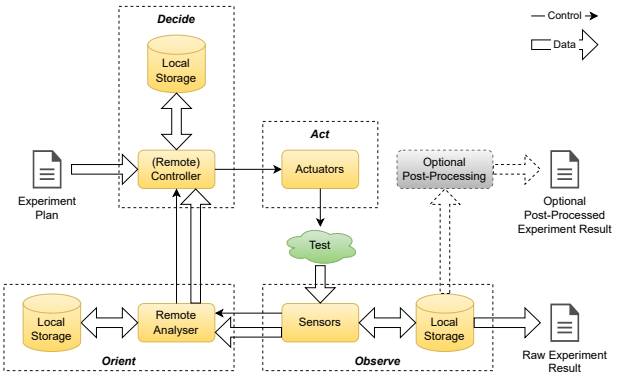


Figure 7: Distributed Experiment Steering architectural pattern components and control/data flow

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.

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 strategic 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.

Examples: The autonomous microscopy use case [28] (Figure 5) implements the Distributed Experiment Steering architectural pattern, as an ongoing STEM experiment is controlled by remote analyses of periodic experiment data. At the architectural pattern level of abstraction, the individual pattern components are as follows:

- In addition to the properties identified by the Experiment Steering strategic pattern, the experiment controller is either local or remote and may feature a graphical user interface (GUI) or some other human-machine interface (HMI).
- The actuator is part of the STEM and moves the scanning electron beam.
- The test is performed in the STEM experiment determines the properties of microscopic structures.
- The sensor is part of the STEM and provides the raw microscope data.

Known Uses: This architectural pattern is used in every experiment, where live feedback of remotely analyzed experiment data is being used to autonomously change experiment parameters. Given the potentially significant communication delay between the experiment and the remote analysis, real-time feedback is limited.

7.2 Example: Local Design of Experiments

Name: Local Design of Experiments

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.

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.

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 strategic pattern is being used in conjunction with this architectural pattern, such as by using the Local Experiment Steering or Distributed 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 strategic 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 strategic pattern, an experiment controller executes each experiment using a predetermined experiment plan (Figure 8). 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.

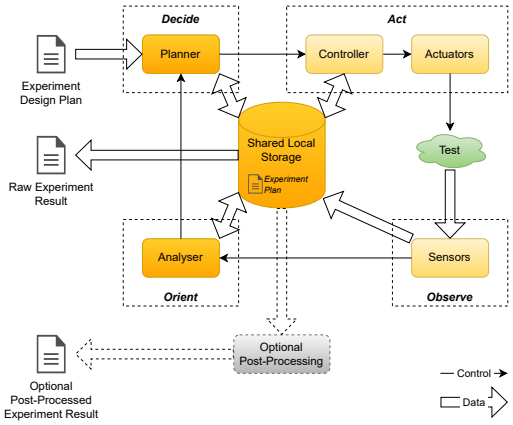


Figure 8: Local Design of Experiments architectural pattern components and control/data flow

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.

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 strategic pattern. It relies on the Experiment Control strategic pattern for automatically executing a predetermined experiment plan. This architectural pattern can be extended

using the Experiment Steering strategic pattern (instead of the Experiment Control strategic 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 Distributed Experiment Steering architectural patterns.

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

Examples: The ACL science use case (Figure 1) implements the Local Design of Experiments architectural pattern, as all components (planner, controller(s), robot, synthesis station(s), and characterization station(s)) are local, i.e., in close physical and logical proximity with no significant latency (for communication or sample movement) to remote components.

The experiment is a complex sequence of steps involving multiple instruments, actuators, sensors, etc. Thus, the experiment itself could be considered a Local Multi-Experiment Workflow architectural pattern using a sequence of Local Experiment Control architectural patterns. Examples of steps that constitute the Multi-Experiment Workflow architecture include the synthesis step and each of the individual characterization steps, such as the gas chromatography, high performance liquid chromatography, and X-ray microscopy. Some of these steps could potentially be performed in parallel if the sample were broken down into pieces such that the pieces could be analyzed be the characterization instruments in parallel. There is a significant overlap of the different components of the patterns, as the same shared storage is being used, for example.

Known Uses: This architectural pattern is used in every experiment, where feedback of experiment results is being used to autonomously change the parameters of the next experiment(s) using components that are all local, i.e., in close physical and logical proximity with no significant latency (for communication or sample movement) to remote components. Known uses range from having simple linear or random parameter scan to complex probabilistic approaches (e.g., Bayesian design of experiments) or domain science informed AI (e.g., physics-informed design of experiments) in the feedback loop.

8 BUILDING SOLUTIONS

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. Section 8.1 describes the involved steps and discusses the individual decision parameters in more detail. Section 8.2 discusses additional considerations when composing different design patterns, such as due to multiple loop control problems.

8.1 A STEP-BY-STEP GUIDE

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 strategic pattern.
(4) Identify the individual components of each loop control problem and associated strategic pattern.
(5) Classify each loop control problem by an architectural pattern that matches its strategic 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 strategic pattern. Additional loop control problems may exist that may extend the simple open loop control, such as to the Experiment Steering strategic pattern, or uses/relies on the simple open loop control, such as with the Design of Experiments strategic 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 strategic 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 8.2.

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 strategic pattern?: The key differences in features between the 4 strategic 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 strategic 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 strategic 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 strategic 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 strategic pattern is employed (see Section 8.2).

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 GUI would likely also be considered a remote component.

8.2 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 strategic pattern that already uses the Experiment Control strategic pattern, but could use the Experiment Steering strategic pattern instead. Similarly, the Multi-Experiment Workflow strategic pattern already uses the Experiment Control strategic pattern, but could use the Experiment Steering strategic pattern, the Design of Experiments strategic patterns, or a combination of Experiment Control, Experiment Steering and Design of Experiments strategic patterns instead. This composition of strategic 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 strategic to architectural. Note that there may be more than one control loop implementing the same strategic 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 9 illustrates the involved components and control/data flow of the Experiment Steering and the Design of Experiments strategic pattern composition. The Experiment Design Plan and the Experiment Planner are exclusive parts of the Design of Experiments strategic pattern, while the other components are part of the Experiment Steering strategic pattern that the Design of Experiments strategic pattern is using as its experiment to control from an Experiment Plan perspective.

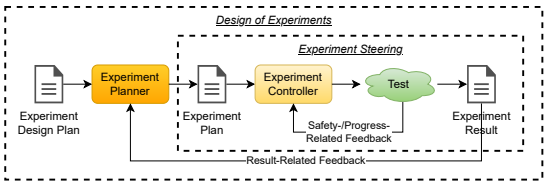


Figure 9: Example: Components and control/data flow of Experiment Steering and Design of Experiments strategic pattern composition

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 Distributed Design of Experiments architectural pattern composition is shown in Figure 10.

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.

9 CONCLUSION

This paper introduced the science use case design patterns of the INTERSECT Open Architecture Specification. Given the open nature of the architecture, the specification itself has been published

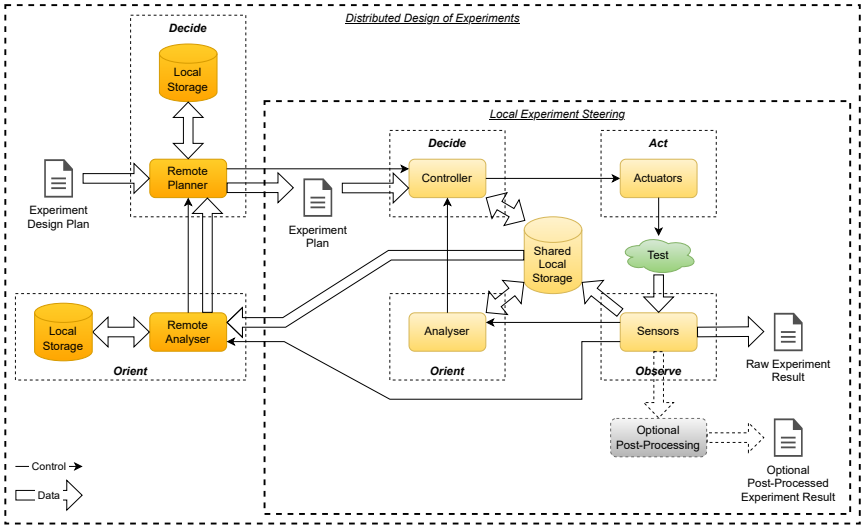


Figure 10: Example: Components and control/data flow of Local Experiment Steering and Distributed Design of Experiments architectural pattern composition

in 3 reports: (1) a science use case design pattern catalog [15], (2) the SoS architecture [32], and (3) the microservice architecture [5].

The basic template for a science use case design pattern is defined in a loop control problem paradigm, as the problems to be solved by the patterns have machine-in-the-loop requirements. Two classes of science use case design patterns have been identified: strategic patterns and architectural patterns. Strategic patterns define high-level solution methods using experiment control architecture features at a very coarse granularity. Architectural patterns define more specific solution methods using hardware and software architecture features at a finer granularity.

The current science use case design pattern catalog [15] defines 12 patterns (4 strategic and 8 architectural). The paper provides a step-by-step guide for building solutions using these patterns and details how solutions can be composed from multiple patterns. An improved version of the science use case design pattern catalog based on this paper is forthcoming.

Ongoing work focuses on refining the INTERSECT Open Architecture and applying it to the following six initial science use cases at ORNL: (1) automation for grid interconnected-laboratory emulation, (2) autonomous additive manufacturing, (3) autonomous continuous flow reactor synthesis, (4) autonomous microscopy, (5) an autonomous robotic chemistry laboratory, and (6) an ion trap quantum computing resource.

Future work seeks to clarify the relationships between the design patterns presented in this paper and execution patterns, scientific workflow motifs and workflow execution patterns [20, 50, 51], which are not design patterns but categorize behavioral commonalities of workflows. These patterns and motifs may be included as behavioral patterns in the science use case design pattern catalog.

ACKNOWLEDGMENTS

Research sponsored by the Laboratory Directed Research and Development Program's INTERSECT Initiative of Oak Ridge National Laboratory. This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

The authors also thank the EuroPLoP'23 shepherd, Aliaksandr Birukou, the participants of the EuroPLoP'23 conference for helping to improve this paper and providing valuable feedback.

REFERENCES

[1] Anees Al-Najjar, Nageswara Rao, Neena Imam, Thomas Naughton, Seth Hitefield, Lawrence Sorrillo, James Kohl, Wael Elwasif, Jean-Christophe Bilheux, Hassina Bilheux, Swen Boehm, and Jason Kincl. 2021. Virtual Framework for Development and Testing of Federation Software Stack. In *2021 IEEE 46th Conference on Local Computer Networks (LCN)*. 323–326. https://doi.org/10.1109/LCN52139.2021.9524993

[2] Peter Amstutz, Maxim Mikheev, Michael R. Crusoe, Nebojša Tijanić, Samuel Lampa, et al. 2022. Existing Workflow systems. https://s.apache.org/existing-workflow-systems

[3] Balsam 2022. Balsam Workflows. https://www.alcf.anl.gov/support-center/theta/balsam

[4] Jan Borchers. 2001. *A Pattern Approach to Interaction Design.* John Wiley & Sons, Inc., New York, NY, USA.

[5] Michael Brim and Christian Engelmann. 2022. *INTERSECT Architecture Specification: Microservice Architecture (Version 0.5).* Technical Report ORNL/TM-2022/2715. Oak Ridge National Laboratory, Oak Ridge, TN, USA. https://doi.org/10.2172/1902805

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

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

[8] Jonathan Carter, John Feddema, Doug Kothe, Rob Neely, Jason Pruet, and Rick Stevens. 2023. Advanced Research Directions on AI for Science, Energy, and Security. Available at https://www.anl.gov/ai-for-science-report.

[9] Defense Advanced Research Projects Agency, U.S. Department of Defense. 2022. Creating Cross-Domain Kill Webs in Real Time. https://www.darpa.mil/news-

events/2020-09-18a
[10] Defense Advanced Research Projects Agency, U.S. Department of Defense. 2022. System of Systems Integration Technology and Experimentation (SoSITE). https://www.darpa.mil/program/system-of-systems-integration-technology-and-experimentation
[11] DOE 2020. *DOE National Laboratories' Computational Facilities – Research Workshop Report.* Technical Report ANL/MCS-TM-388. Argonne National Laboratory, Lemont, IL, USA. https://publications.anl.gov/anlpubs/2020/02/158604.pdf
[12] Chad Dougherty, Kirk Sayre, Robert Seacord, David Svoboda, and Kazuya Togashi. 2009. *Secure Design Patterns.* Technical Report CMU/SEI-2009-TR-010. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA. https://doi.org/10.1184/R1/6583640.v1
[13] Douglas K. Van Duyne, James Landay, and Jason I. Hong. 2002. *The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
[14] Christian Engelmann, Olga Kuchar, Swen Boehm, Michael J. Brim, Thomas Naughton, Suhas Somnath, Scott Atchley, Jack Lange, Ben Mintz, and Elke Arenholz. 2022. The INTERSECT Open Federated Architecture for the Laboratory of the Future. In *Communications in Computer and Information Science (CCIS): Accelerating Science and Engineering Discoveries Through Integrated Research Infrastructure for Experiment, Big Data, Modeling and Simulation. 18th Smoky Mountains Computational Sciences & Engineering Conference (SMC) 2022*, Vol. 1690. Springer, Cham, 173–190. https://doi.org/10.1007/978-3-031-23606-8_11 Acceptance rate 32.4% (24/74).
[15] Christian Engelmann and Suhas Somnath. 2022. *INTERSECT Architecture Specification: Use Case Design Patterns (Version 0.5).* Technical Report ORNL/TM-2022/2681. Oak Ridge National Laboratory, Oak Ridge, TN, USA. https://doi.org/10.2172/1896984
[16] FireCrest 2022. FireCrest RESTful API. https://firecrest.readthedocs.io/en/latest/index.html
[17] Evan Fortunato. 2016. STITCHES - SoS Technology Integration Tool Chain for Heterogeneous Electronic Systems. https://ndiastorage.blob.core.usgovcloudapi.net/ndia/2016/systems/18869_Fortunato_SoSITE_STITCHES_Overview_Long_9Sep2016_.pdf
[18] Martin Fowler. 2002. *Patterns of Enterprise Application Architecture.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
[19] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1994. *Design Patterns: Elements of Reusable Object-oriented Software.* Addison-Wesley Professional.
[20] Daniel Garijo, Pinar Alper, Khalid Belhajjame, Oscar Corcho, Yolanda Gil, and Carole Goble. 2014. Common motifs in scientific workflows: An empirical analysis. *Future Generation Computer Systems* 36 (2014), 338–351. https://doi.org/10.1016/j.future.2013.09.018 Special Section: Intelligent Big Data Processing Special Section: Behavior Data Security Issues in Network Information Propagation Special Section: Energy-efficiency in Large Distributed Computing Architectures Special Section: eScience Infrastructure and Applications.
[21] Gladier 2022. Gladier experiment steering. https://labs.globus.org/projects/gladier.html
[22] Globus Automate 2022. Globus Automation Services. https://docs.globus.org/globus-automation-services
[23] Jeffrey Heer and Maneesh Agrawala. 2006. Software Design Patterns for Information Visualization. *IEEE Transactions on Visualization and Computer Graphics* 12, 5 (Sept. 2006), 853–860. https://doi.org/10.1109/TVCG.2006.178
[24] Nils Heinonen. 2020. Argonne researchers use Theta for real-time analysis of COVID-19 proteins. https://www.alcf.anl.gov/news/argonne-researchers-use-theta-real-time-analysis-covid-19-proteins
[25] Saurabh Hukerikar and Christian Engelmann. 2017. Resilience Design Patterns: A Structured Approach to Resilience at Extreme Scale. *Journal of Supercomputing Frontiers and Innovations (JSFI)* 4, 3 (Oct. 2017), 4–42. https://doi.org/10.14529/jsfi170301
[26] Saurabh Hukerikar and Christian Engelmann. 2017. *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. https://doi.org/10.2172/1436045
[27] ISO/IEC JTC 1/SC 7 Software and systems engineering. 2019. ISO/IEC/IEEE 21839:2019. https://www.iso.org/standard/71955.html
[28] Sergei V. Kalinin, Maxim Ziatdinov, Jacob Hinkle, Stephen Jesse, Ayana Ghosh, Kyle P. Kelley, Andrew R. Lupini, Bobby G. Sumpter, and Rama K. Vasudevan. 2021. Automated and Autonomous Experiments in Electron and Scanning Probe Microscopy. *ACS Nano* 15, 8 (2021), 12604–12627. https://doi.org/10.1021/acsnano.1c02104 arXiv:https://doi.org/10.1021/acsnano.1c02104
[29] Kebotix 2022. Kebotix. https://www.kebotix.com
[30] Michael Kircher and Prashant Jain. 2004. *Pattern-Oriented Software Architecture, Volume 3: Patterns for Resource Management.* Wiley Publishing.
[31] Kubernetes 2022. Kubernetes. https://kubernetes.io
[32] Olga A. Kuchar, Swen Boehm, Thomas Naughton, Suhas Somnath, Ben Mintz, Jack Lange, Scott Atchley, Rohit Srivastava, and Patrick Widener. 2022. *INTERSECT Architecture Specification: System-of-systems Architecture (Version 0.5).* Technical Report ORNL/TM-2022/2717. Oak Ridge National Laboratory, Oak Ridge, TN, USA. https://doi.org/10.2172/1968700
[33] Mark. W. Maier. 1998. Architecting Principles for System-of-Systems. *Systems Engineering* 1, 4 (Nov. 1998), 267–284.
[34] Mark. W. Maier and Eberhardt Rechtin. 2009. *The Art of Systems Architecting (Systems Engineering).* CRC Press, Boca Raton, FL, USA.
[35] William H. J. Manthorpe Jr. 1996. The Emerging Joint System of Systems: A Systems Engineering Challenge and Opportunity for APL. *John Hopkins APL Technical Digest* 17, 3 (July 1996), 305–313.
[36] National Energy Research Scientific Computing Center (NERSC). 2022. Superfacility API. https://api.nersc.gov
[37] National Energy Research Scientific Computing Center (NERSC). 2022. Superfacility project. https://www.nersc.gov/research-and-development/superfacility
[38] Thomas Naughton, Seth Hitefield, Lawrence Sorrillo, Nageswara Rao, James Kohl, Wael Elwasif, Jean-Christophe Bilheux, Hassina Bilheux, Swen Boehm, Jason Kincl, Satyabrata Sen, and Neena Imam. 2020. Software Framework for Federated Science Instruments. In *Driving Scientific and Engineering Discoveries Through the Convergence of HPC, Big Data and AI*, Jeffrey Nichols, Becky Verastegui, Arthur Barney Maccabe, Oscar Hernandez, Suzanne Parete-Koon, and Theresa Ahearn (Eds.). Springer International Publishing, 189–203.
[39] Netflix. 2022. Netflix OSS. https://netflix.github.io
[40] Netflix. 2022. Spring Cloud Netflix. https://spring.io/projects/spring-cloud-netflix
[41] Richard S. Pei. 2000. System of Systems Integration (SoSI) - A Smart Way of Acquiring Army C4I2WS Systems. In *Proceedings of the Summer Computer Simulation Conference 2000.* 574–579.
[42] Eberhardt Rechtin. 1990. *Systems Architecting: Creating & Building Complex Systems.* Prentice Hall.
[43] Katharine Sanderson. 2019. Automation: Chemistry shoots for the Moon. *Nature* 568 (April 2019), 577–579. https://doi.org/10.1038/d41586-019-01246-y
[44] Douglas C. Schmidt, Michael Stal, Hans Rohnert, and Frank Buschmann. 2000. *Pattern-Oriented Software Architecture Volume 2: Patterns for Concurrent and Networked Objects.* Wiley Publishing.
[45] Rick Stevens, Valerie Taylor, Jeff Nichols, Arthur Barney Maccabe, Katherine Yelick, and David Brown. 2020. AI for Science Report. Available at https://www.anl.gov/cels/reference/ai-for-science-report-2020.
[46] Jerry Talton, Lingfeng Yang, Ranjitha Kumar, Maxine Lim, Noah Goodman, and Radomír Měch. 2012. Learning Design Patterns with Bayesian Grammar Induction. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST) 2012* (Cambridge, Massachusetts, USA). ACM, New York, NY, USA, 63–74. https://doi.org/10.1145/2380116.2380127
[47] Mohammadreza Torkjazi and Ali K. Raz. 2022. A Taxonomy for System of Autonomous Systems. In *2022 17th Annual System of Systems Engineering Conference (SOSE)*. 198–203. https://doi.org/10.1109/SOSE55472.2022.9812673
[48] Keri Troutman. 2019. Superfacility Framework Advances Photosynthesis Research. https://www.nersc.gov/news-publications/nersc-news/science-news/2019/superfacility-framework-advances-photosynthesis-research
[49] U.S. Department of Defense. 2010. The DoDAF Architecture Framework Version 2.02. https://dodcio.defense.gov/Library/DoD-Architecture-Framework
[50] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. 2003. Workflow Patterns. *Distributed and Parallel Databases* 14, 1 (2003), 5–51. https://doi.org/10.1023/A:1022883727209
[51] Rafael Vescovi, Ryan Chard, Nickolaus D. Saint, Ben Blaiszik, Jim Pruyne, Tekin Bicer, Alex Lavens, Zhengchun Liu, Michael E. Papka, Suresh Narayanan, Nicholas Schwarz, Kyle Chard, and Ian T. Foster. 2022. Linking scientific instruments and computation: Patterns, technologies, and experiences. *Patterns* 3, 10 (2022), 100606. https://doi.org/10.1016/j.patter.2022.100606
[52] Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, Anima Anandkumar, Karianne Bergen, Carla P. Gomes, Shirley Ho, Pushmeet Kohli, Joan Lasenby, Jure Leskovec, Tie-Yan Liu, Arjun Manrai, Debora Marks, Bharath Ramsundar, Le Song, Jimeng Sun, Jian Tang, Petar Veličković, Max Welling, Linfeng Zhang, Connor W. Coley, Yoshua Bengio, and Marinka Zitnik. 2023. Scientific Discovery in the Age of Artificial Intelligence. *Nature* 620 (Aug. 2023), 47–60. https://doi.org/10.1038/s41586-023-06221-2
[53] Eberhard Wolff. 2016. *Microservices: Flexible Software Architectures.* Addison-Wesley Professional.