Design Patterns

A Pragmatic and Scalable Approach to Data and Software Management Protocols

CHORUS Forum June 2022

Jane Wyngaard (UCT), jane.wyngaard@uct.ac.za
Shelley Stall (AGU), sstall@agu.org
Chris Lenhardt (RENCI), clenhardt@renci.org
Chris Erdmann (AGU), CErdmann@agu.org

Overview

1. Our common challenge
2. Design Patterns in software engineering
3. Proposed Data Design Patterns concept
4. Proposed development approach
5. Seeking your input

Our common challenge

- “Common”: Across scientific disciplines
- Barriers to FAIR-er Scientific Objects
  - Technological / Cultural / Legal / Institutional
  - Practical:
    - With what time & money?
    - What data?
    - What metadata?
    - How: what tool/format/process?
    - How open?
    - Where: archive/journal/lab/other?
    - When?

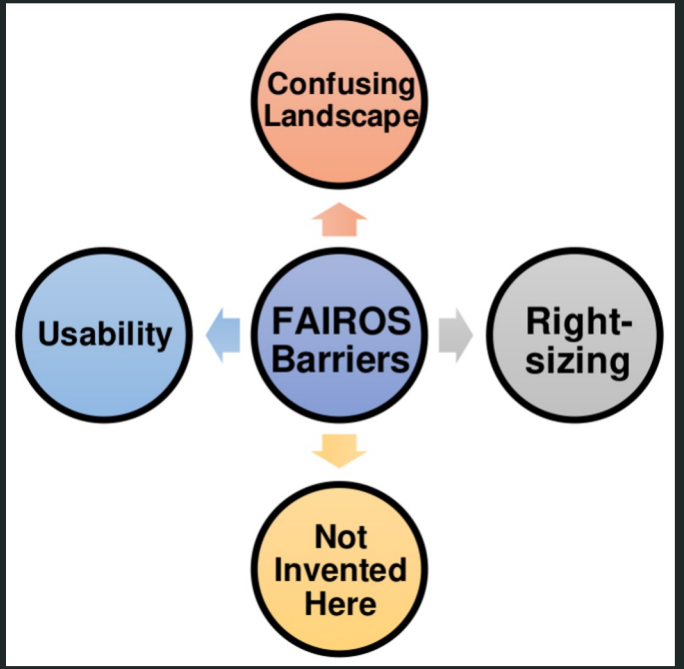


Image credit: Chris Lenhardt

Design Patterns in software engineering

Design Pattern: A reusable template created to address a common problem

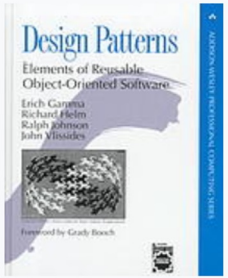
Advantages:

- Facilitate communicating complex problems
- Promoting re-use
- Capturing community practices
- Improving maintainability
- Provide a structure that is known to work for a given problem class when approaching a complex problem

Design Patterns in software engineering

Design Pattern: A reusable template created to address a common problem

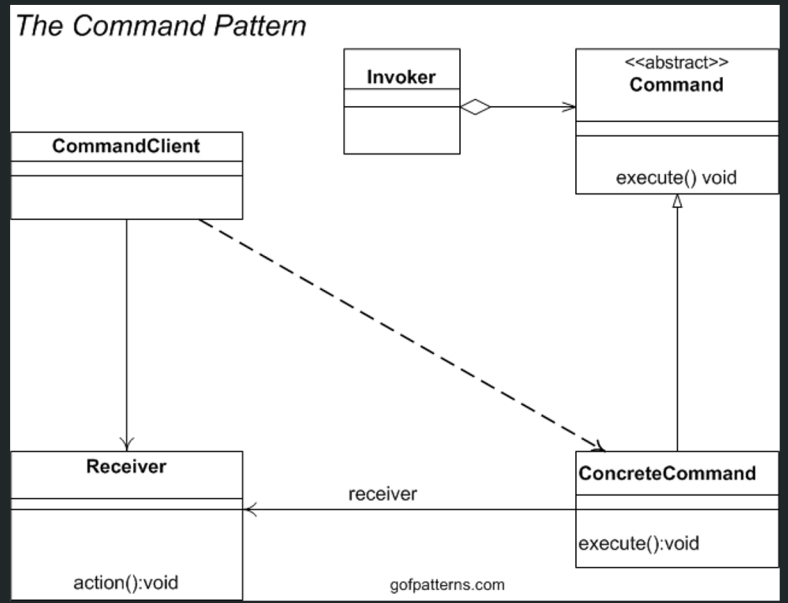
Design Patterns: Elements of Reusable Object-Oriented Software



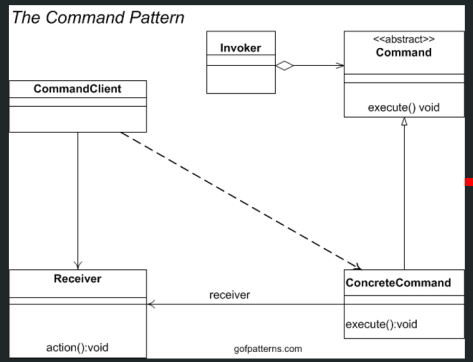
Author	The "Gang of Four": Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
Country	United States
Subject	Design patterns, software engineering, object-oriented programming
Publisher	Addison-Wesley
Publication date	1994

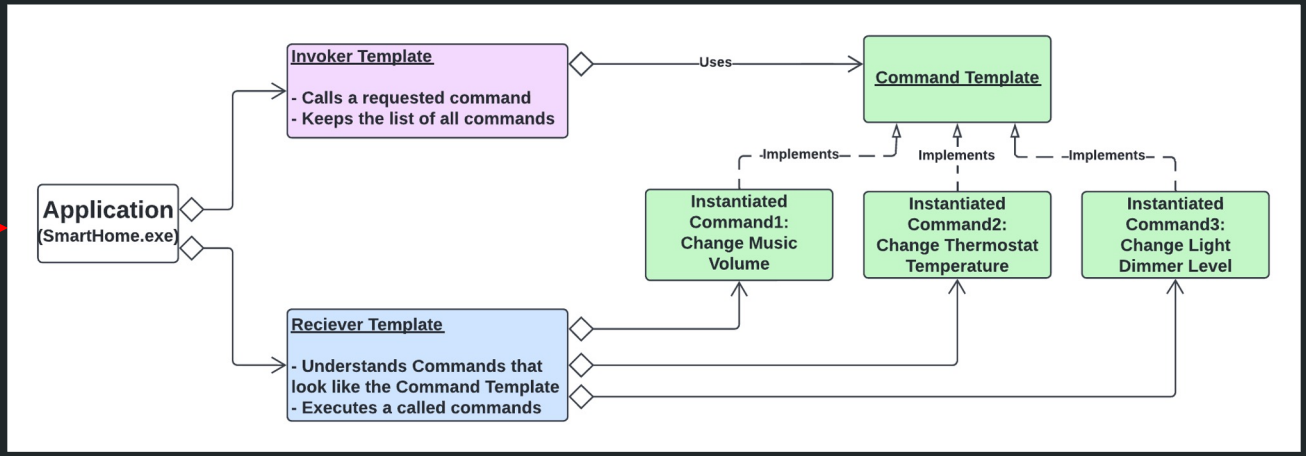
Design Patterns in software engineering

Eg: Command Pattern applied to a smart home controller example



Design Patterns in software engineering





Design Patterns in software engineering

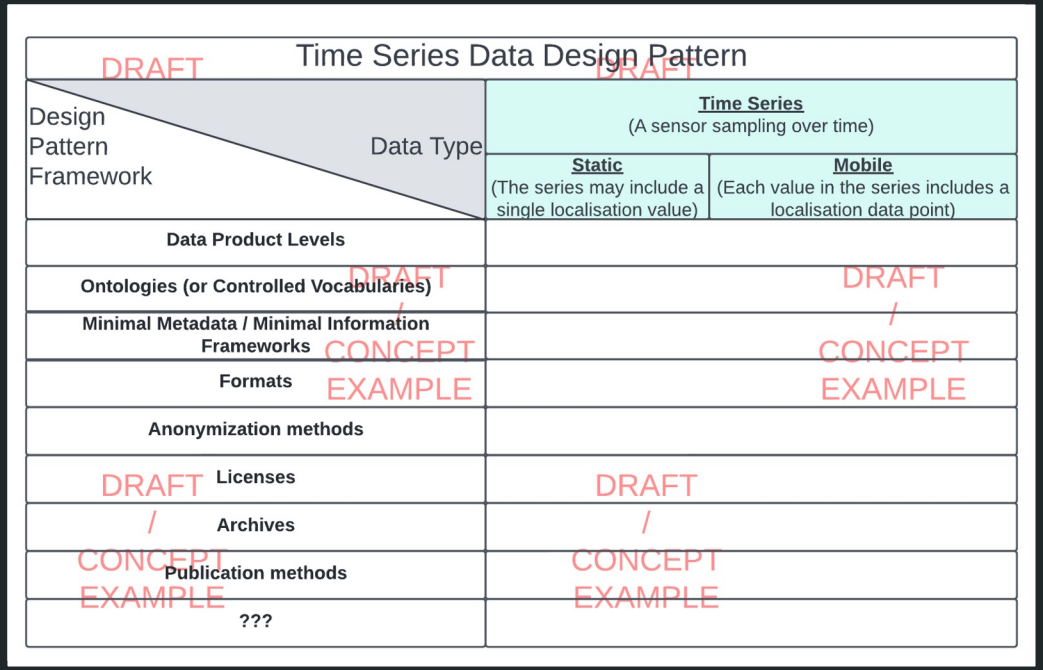
- Originally 25 in 3 functional classes, more have been developed since
- Each is applicable to a wide array of different real world software applications
- Actual code instantiation looks different in different software languages

By Purpose				
		Creational	Structural	Behavioral
By Scope	Class	• Factory Method	• Adapter (class)	• Interpreter • Template Method
	Object	• Abstract Factory • Builder • Prototype • Singleton	• Adapter (object) • Bridge • Composite • Decorator • Façade • Flyweight • Proxy	• Chain of Responsibility • Command • Iterator • Mediator • Memento • Observer • State • Strategy • Visitor

https://www.gofpatterns.com/sitemap.php

Proposed Data Design Patterns concept

Data Design Pattern: A reusable data management template created to address how to practice FAIR for a common data type:

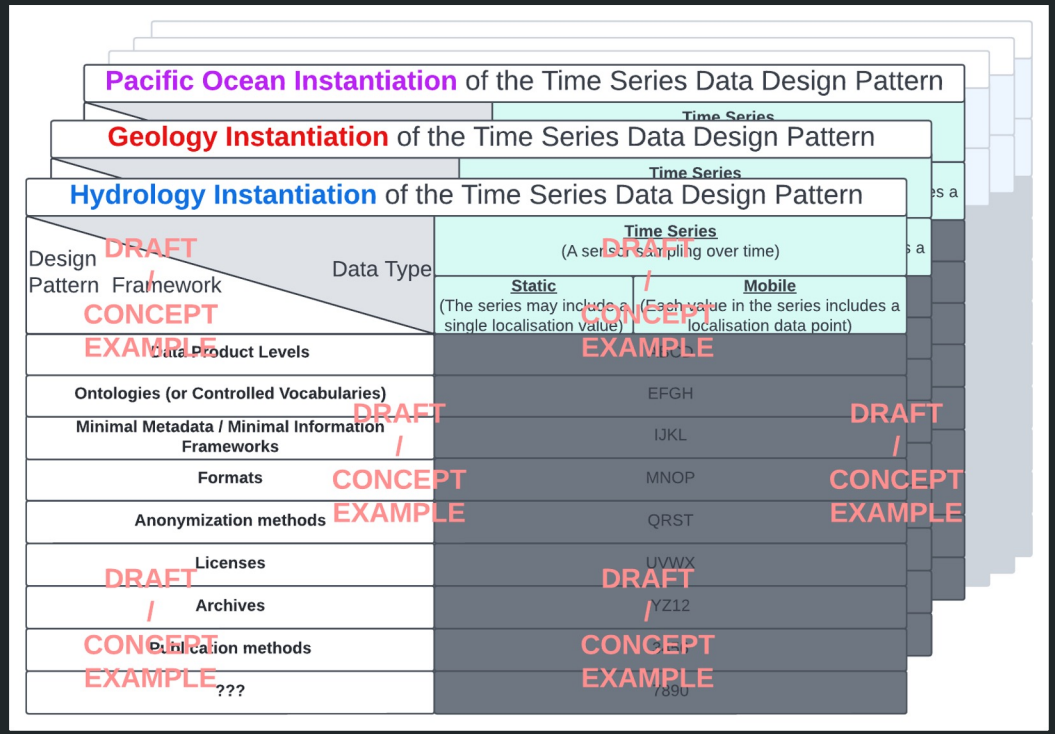


Time Series Data Design Pattern

Design Pattern Framework \ Data Type	Time Series (A sensor sampling over time)	
	Static (The series may include a single localisation value)	**Mobile** (Each value in the series includes a localisation data point)
Data Product Levels		
Ontologies (or Controlled Vocabularies)		
Minimal Metadata / Minimal Information Frameworks		
Formats		
Anonymization methods		
Licenses		
Archives		
Publication methods		
???		

Proposed Data Design Patterns concept

Data Design Pattern would be instantiated for different domains
(analogous to different coding languages' different instantiations of a software pattern)



Proposed Data Design Patterns concept

Different Data Design Pattern would exist for different data types/classes (analogous to different coding problems mapping to different software patterns)

Data Design Patterns

Design Pattern Framework \ Data Type	Time Series (A sensor sampling over time)		Raster data (2&3D) (Tesselation of a plane or 3D matrix of instantaneous data into a 2/3D matrix of cells/pixels)		Qualititative data (surveys, interviews)	Unstructured data (Information about textual data)	Physical samples	Complex Digital (Software, Model, Neural Network)
	Static (The series may include a single localisation value)	Mobile (Each value in the series includes a localisation data point)	Static (Multiple tesselations may be associated with a single localisation value)	Mobile (Each tesselation includes localisation data				
Data Product Levels	DRAFT / CONCEPT EXAMPLE				DRAFT / CONCEPT EXAMPLE			
Ontologies (or Controlled Vocabularies)								
Minimal Metadata / Minimal Information Frameworks								
Formats			DRAFT / CONCEPT EXAMPLE				DRAFT / CONCEPT EXAMPLE	
Anonymization methods								
Licenses								
Archives	DRAFT / CONCEPT EXAMPLE				DRAFT / CONCEPT EXAMPLE			
Publication methods								
Other???								

Proposed Advantages

- Reduce the cognitive load on a researcher related to operationalizing FAIROS:
  - Provide practical and specific guidance for practicing FAIR when dealing with a given type of data
- Address common problems encountered by groups pursuing similar goals scopes the solution to a particular problem and allows for a level of modularity
  - Enable common tooling & training stacks to be built that serve multiple disciplines
  - Enable sharing of resources (human/compute/storage/algorithmic)
- Reusable across domains, and inherently cross cutting as they are not encoding specific solutions to transcends disciplinary parochialisms
- Are NOT: Best practices, standards, documentation.
  - Nor necessarily static: As technology and science evolve, design patterns may be updated to maintain their utility

Proposed Development Approach

AGU is planning a *RCN-Like* development and engagement effort

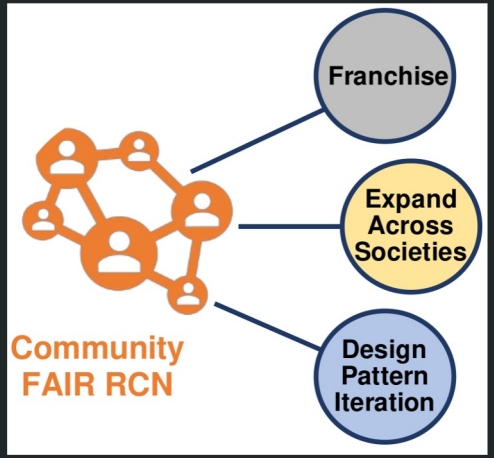
Beginning now with a demonstration collaboration with the Hydrology community represented by CUASHI

CUAHSI

https://www.cuahsi.org/

Future:

1. Full Design Pattern Creation
2. Communications, Adoption, Dissemination, Engagement, and Education
3. Diversity, Equity, Inclusion, and Accessibility
4. Maintain and Sustain
5. Assessment and Testing



Seeking your input

- Data Design Patterns will need to strike a balance between sufficiently detailed as to be useful, and not so detailed as to be cumbersome / constraining / monolithic.
- Initial engagement with Hydrology community - join us?
- Other? What else haven’t we thought of?
- Would you be interested in working on this? Either on pattern definition or trialing instantiating them for your domain?