2 December 2024

KUEHNE+NAGEL

Patterns in Software

From Alexander’s Architecture Patterns to Pattern Languages and Beyond

João Pedro Dias
Architecture in da House – Kuehne+Nagel Porto IT Hub

$ whoami

KUEHNE+NAGEL

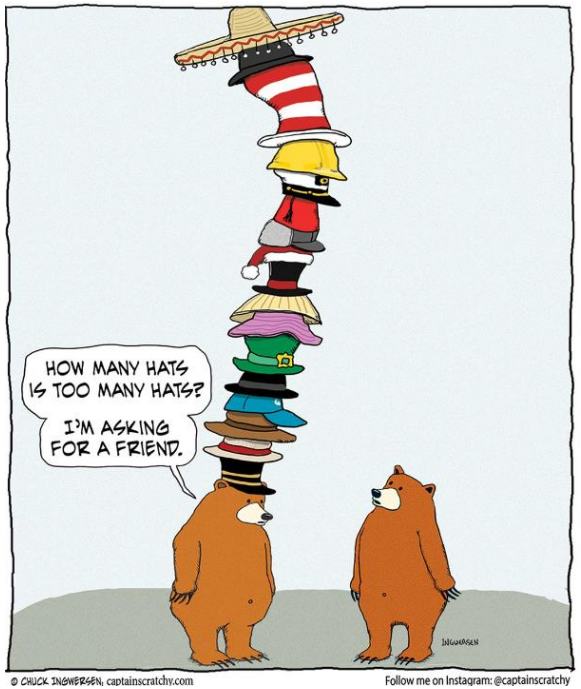
🏷️ João Pedro Dias

📍 Joined KN in April 2023, part of the MEP team @ KNITE+

- Team Lead (acting) and Lead Architect (acting)

🏕️ Life beyond KN:

- Academic background (professor and stuff)
- Ph.D. in Informatics Engineering focused in Software Engineering and IoT;
- Interested in all things Software Engineering, Security, and Internet-of-Things;
- Contributor to open-source & blog writter;
- Semi-pro photographer (landscapes & nature, https://500px.com/jpdias);
- Jack of all trades in my spare time: construction, woodworking, agriculture, ...



🕸️ Where to find me?

joao.dias@kuehne-nagel.com / jpdias@outlook.com / jpdias@pm.me
https://jpdias.me

Agenda

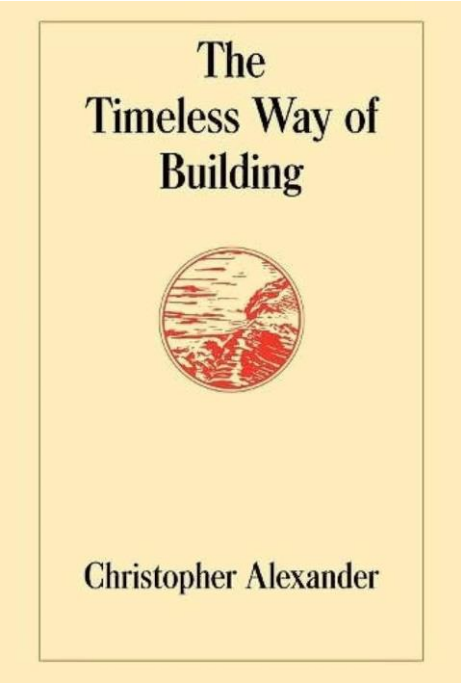
KUEHNE+NAGEL

- The origin of *patterns*
- What is a *pattern*?
- Writing a pattern
- Creating, or finding, patterns
- What is a pattern language?
- Patterns “workshop”
- Read more

The origin of *patterns*

KUEHNE+NAGEL

1977



1987

Using Pattern Languages for Object-Oriented Programs

Kent Beck, Apple Computer, Inc.
Ward Cunningham, Tektronix, Inc.

Technical Report No. CR-87-43
September 17, 1987

Submitted to the OOPSLA-87 workshop on the
Specification and Design for Object-Oriented Programming.

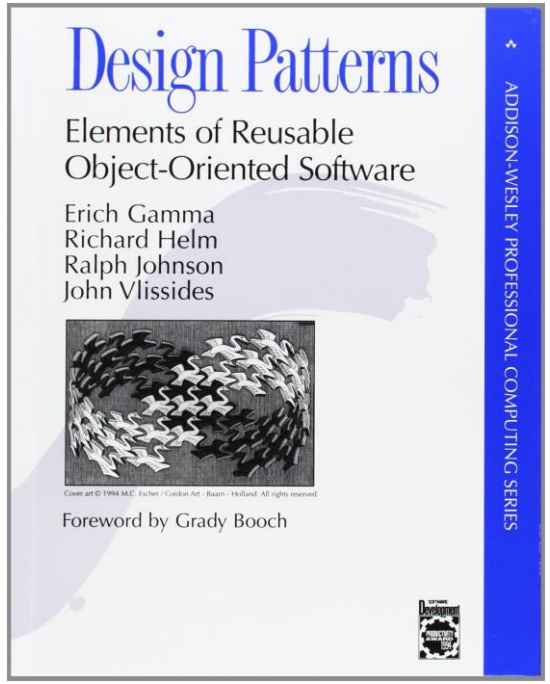
Abstract

We outline our adaptation of Pattern Language to object-oriented programming. We sumarize a system of five patterns we have successfuly used for designing window-based user interfaces and present in slightly more detail a single pattern drawn from our current effort to record a complete pattern language for object-oriented programs.

The search for an appropriate methodology for object-oriented programming has seen the usual rehash of tired old ideas, but the fact is that OOP is so different that no mere force-fit o structured analysis or entity-relationship methods will provide access to the potential inheren in OOP. In particular, neither of these methods address the user interface design issues that have obviously become of paramount importance. In addition, while E-R seems to be "object oriented" it is not suited to the dynamic nature of objects as in Smalltalk and encourages the use of a global perspective while designing, a sure lose in object-oriented programming.

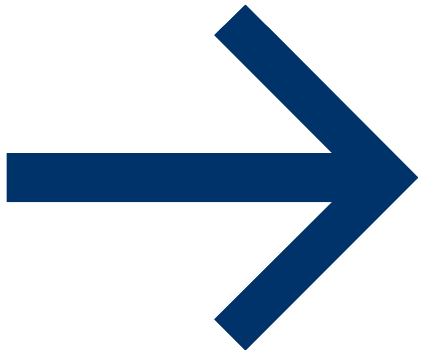
We propose a radical shift in the burden of design and implementation, using concepts adapted from the work of Christopher Alexander, an architect and founder of the Center for Environmental Structures. Alexander proposes homes and offices be designed and built by their eventual occupants. These people, he reasons, know best their requirements for a particular structure. We agree, and make the same argument for computer programs. Computer users should write their own programs. The idea sounds foolish when one considers the size and complexity of both buildings and programs, and the years of training for the design professions. Yet Alexander offers a convincing scenario. It revolves around a concept called a "pattern language."

1994



...

KUEHNE+NAGEL



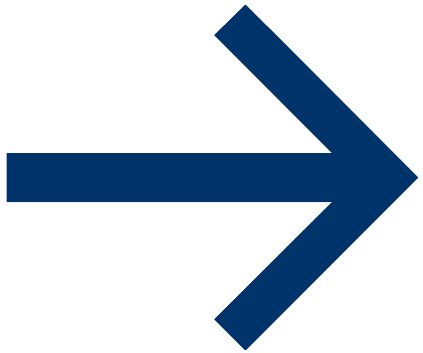
“Alexander proposes homes and offices be designed and built by their eventual occupants. These people, he reasons, know best their requirements for a particular structure. We agree, and make the same argument for computer programs. **Computer users should write their own programs.** The idea sounds foolish when one considers the size and complexity of both buildings and programs, and the years of training for the design professions. Yet Alexander offers a convincing scenario. It revolves around **a concept called a “pattern language”.**

Using Pattern Languages for Object-Oriented Programs, *Kent Beck and Ward Cunningham,*
https://c2.com/doc/oopsla87.html

KUEHNE+NAGEL

What is a *pattern*?

KUEHNE+NAGEL



“A pattern is the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts.

Each pattern is a **three-part rule**, which expresses a relation between a **certain context**, a certain **system of forces** which occurs repeatedly in that context, and a **certain software configuration which allows these forces to resolve themselves**.

Patterns and Software: Essential Concepts and Terminology by Brad Appleton, https://www.bradapp.com/docs/patterns-intro.html

KUEHNE+NAGEL

Pattern Checklist

Doug Lea,
https://hillside.net/index.php/pattern-writing-checklist

- ✓ Describes a **single kind of problem**.
- ✓ Describes the **context** in which the problem occurs.
- ✓ Describes the solution as a **constructable**.
- ✓ Describes **design steps or rules** for constructing the solution.
- ✓ Describes the **forces leading to the solution**.
- ✓ Describes **evidence that the solution optimally resolves forces**.
- ✓ Describes **details that are allowed to vary, and those that are not**.
- ✓ Describes **at least one actual instance of use**.
- ✓ Describes **evidence of generality across different instances**.
- ✓ Describes or refers to variants and subpatterns.
- ✓ Describes or refers to other patterns that it relies upon.
- ✓ Describes or refers to other patterns that rely upon this pattern.
- ✓ **Relates to other patterns with similar contexts, problems, or solutions**.

KUEHNE+NAGEL

Writing a *pattern*

A Pattern Language for Pattern Writing, Gerard Meszaros,
https://hillside.net/index.php/a-pattern-language-for-pattern-writing

How to write a pattern?, A rough guide for first-time pattern authors, Tim Wellhausen and Andreas Fießer
https://europlop.net/wp-content/uploads/2022/10/How-to-write-a-pattern.pdf

Quick Overview

KUEHNE+NAGEL

Context: sets the stage where the pattern takes place.

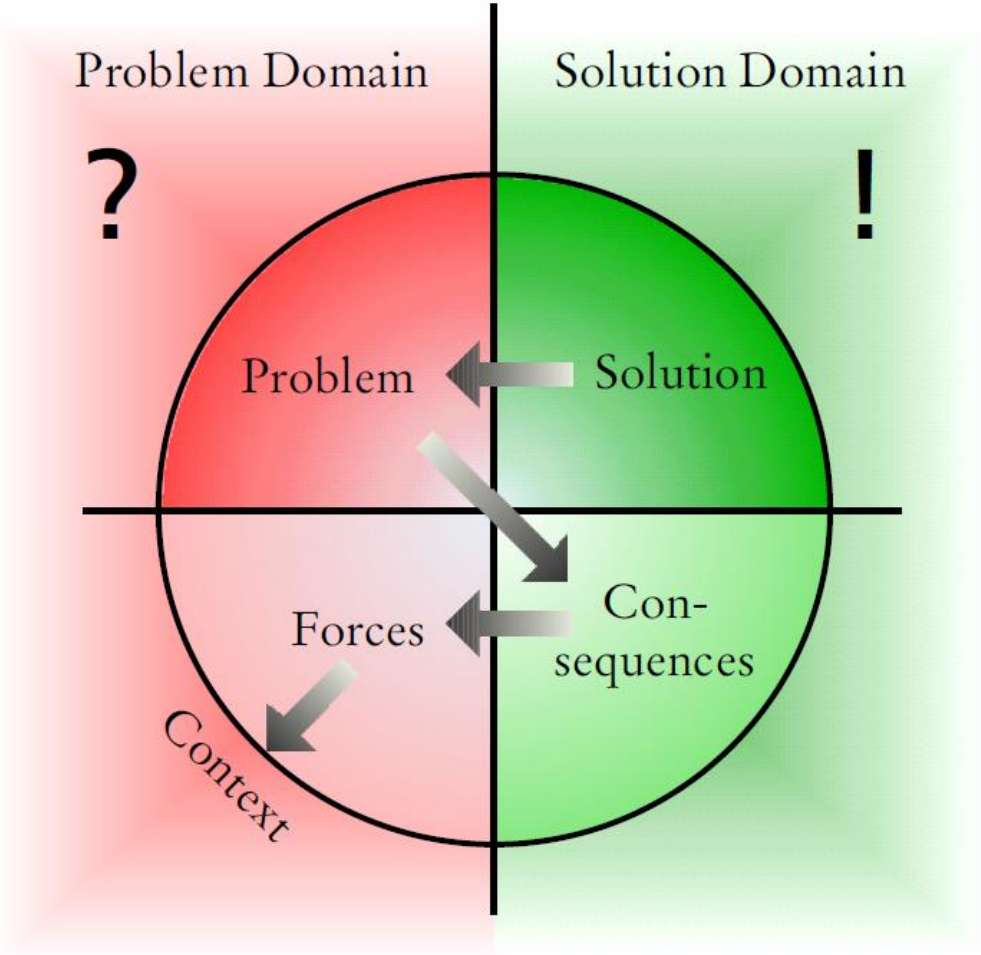
Problem: explains what the actual problem is.

Forces: describes why the problem is difficult to solve.

START HERE

Solution: explains the solution in detail.

Consequences: demonstrates what happens when you apply the solution (positive/negative tradeoffs).



The key and lock “thingy”

What is your pattern all about?

- You are an expert in the field, thus with **extensive knowledge about a specific *thing* you care about**.

- ❑ What makes the “thing” special?
- ❑ What does it include and what does it not include?
- ❑ How can it be distinguished from similar ideas, things or processes?
- ❑ What examples do you know?

Solution

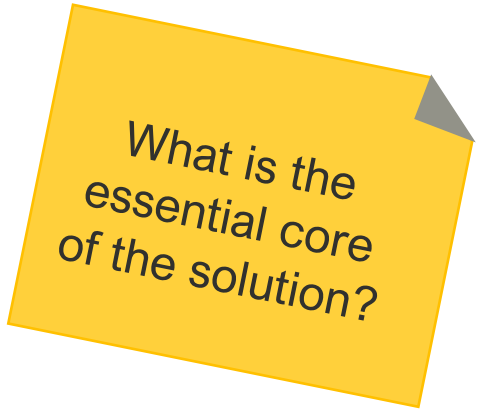
KUEHNE+NAGEL

What is the essential core of the solution?

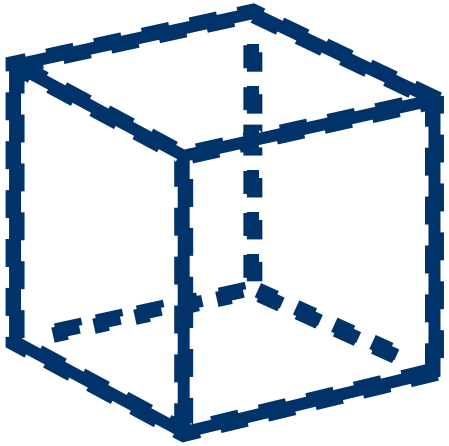
Solution

KUEHNE+NAGEL

Integrate a mechanism into a door that can be repeatedly operated from both sides of the door by a matching key to allow or prevent the door from being opened.



Problem



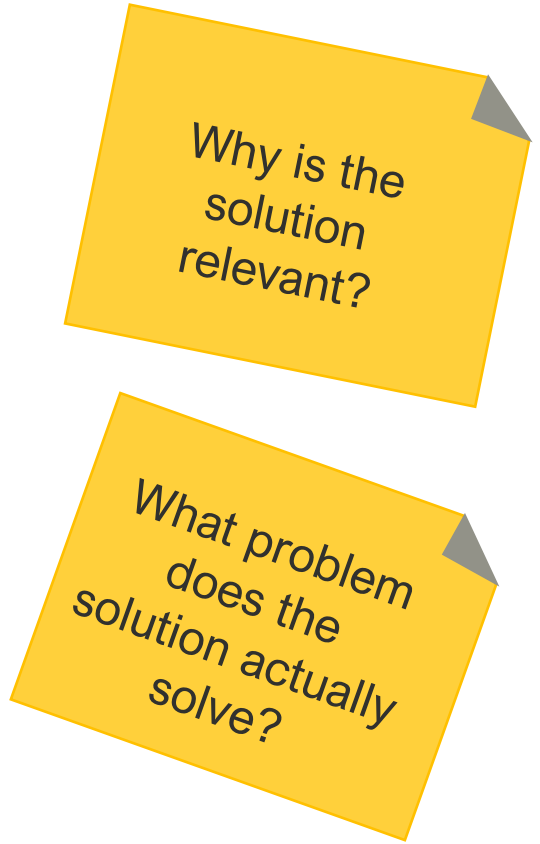
Why is the solution relevant?

What problem does the solution actually solve?

Problem

KUEHNE+NAGEL

You want to control access to a building.



Consequences

KUEHNE+NAGEL

What happens if the solution is applied?

Consequences

KUEHNE+NAGEL

Benefit:

You can easily lock the door when you are inside your property and when you are leaving. It is hard for someone without the key to enter.

Liability:

Because the keys must be difficult to duplicate, you typically only have a limited number of them. So you cannot give access to your property to as many people in parallel as you may like.

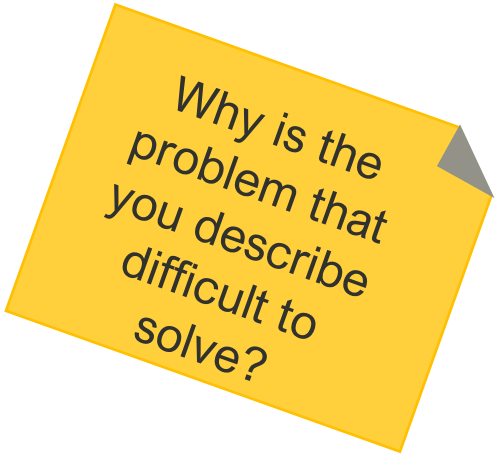
Forces

Why is the problem that you describe difficult to solve?

Forces

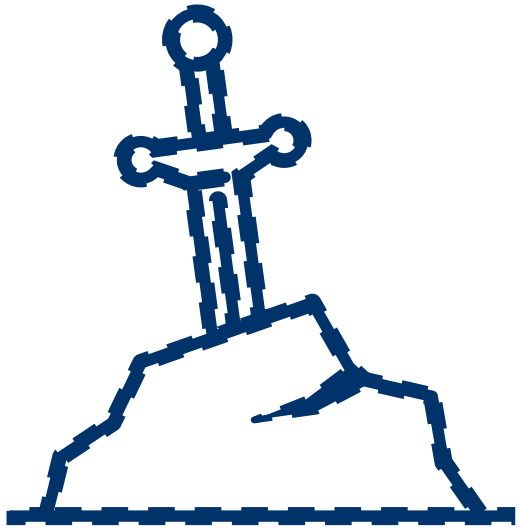
KUEHNE+NAGEL

You want to protect your property, but you cannot be at home all the time or ask somebody else to watch your belongings. Even when you are at home you might want to protect yourself together with your belongings to have a peaceful sleep.



Match the Forces with the Consequences

KUEHNE+NAGEL



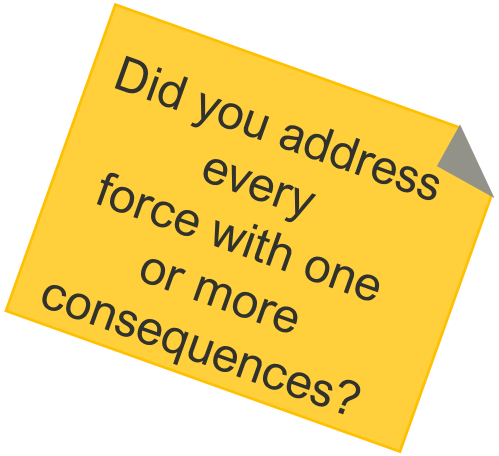
Did you address every force with one or more consequences?

Match the Forces with the Consequences

KUEHNE+NAGEL

Force:

Storage of permission: Permission should be easy to use and reliable to store.



Consequences:

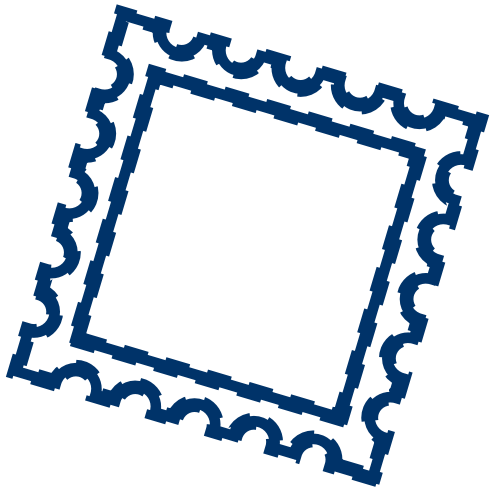
Benefit:

Storage of permission: As the key is a physical object, you can easily take it with you any time so that you can always lock and unlock the door as desired.

Liability:

Storage of permission: Because the keys must be difficult to duplicate, you typically only have a limited number of them. So you cannot give access to your property to as many people in parallel as you may like.

Context

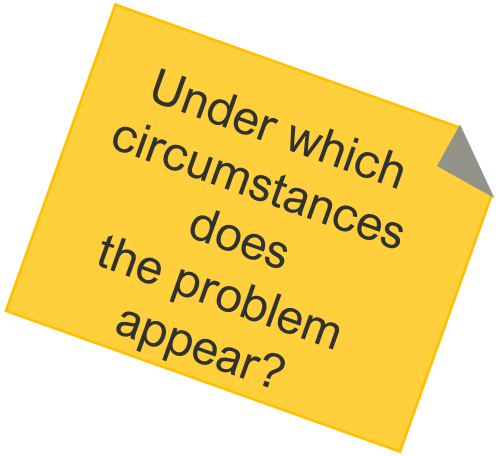


Under which circumstances does the problem appear?

Context

KUEHNE+NAGEL

You are the owner of a house in an area where you cannot leave your property freely accessible to all people because this might attract thieves. The building can be accessed through Doors.



Name, Examples, and Related Patterns

KUEHNE+NAGEL

What name helps us remember the solution?

What examples do you know?

What other patterns exist that share similar characteristics?

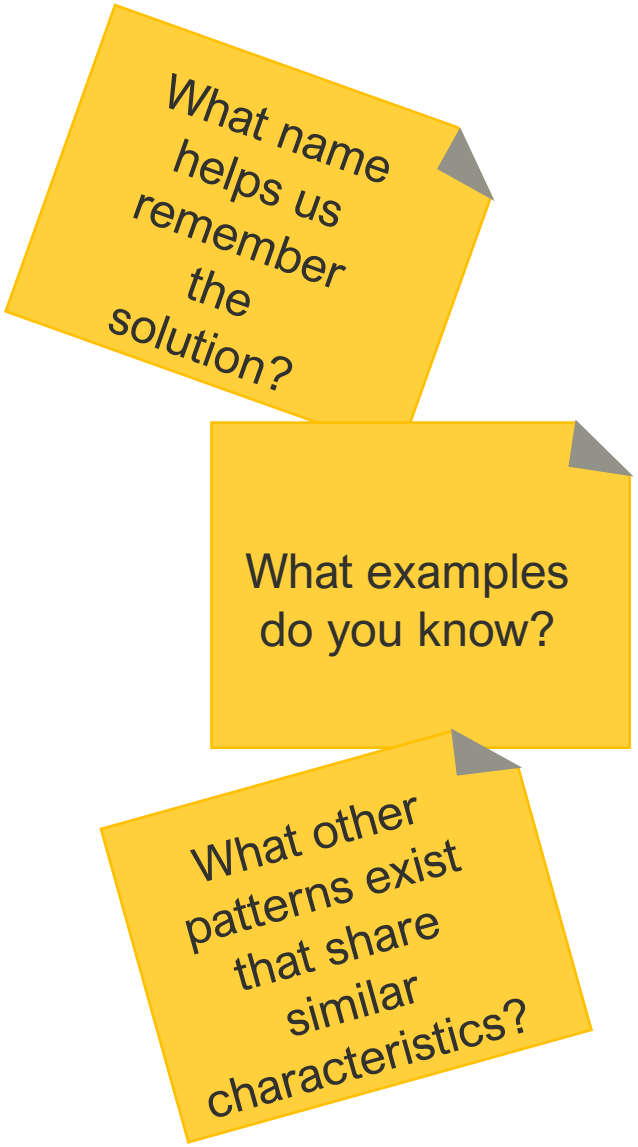
Name, Examples, and Related Patterns

KUEHNE+NAGEL

Name:
Door Lock

Example:
Garage door to protect your car from being damaged or stolen.

Related Patterns:
Padlock, Combination Lock



KUEHNE+NAGEL

Creating, or *finding*, patterns

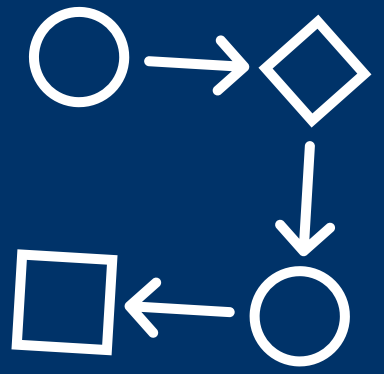
Pattern mining

KUEHNE+NAGEL

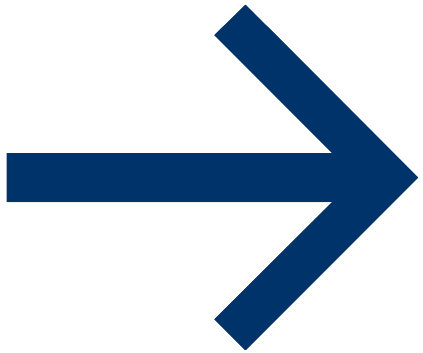
- **Patterns are derived from practical experience and not deduced from theories.** Discovering a pattern is called pattern mining.
- It encompasses the **analysis of existing design structures and the implicit knowledge of experts**.
- The process of pattern mining reveals “**nuggets of wisdoms**” from the structure and form of artefacts and the decision making of their creators.
- To **expose the invariant structure and discriminate it from the surface structure (non-essential features)** is the main task of pattern mining.

KUEHNE+NAGEL

Pattern Languages



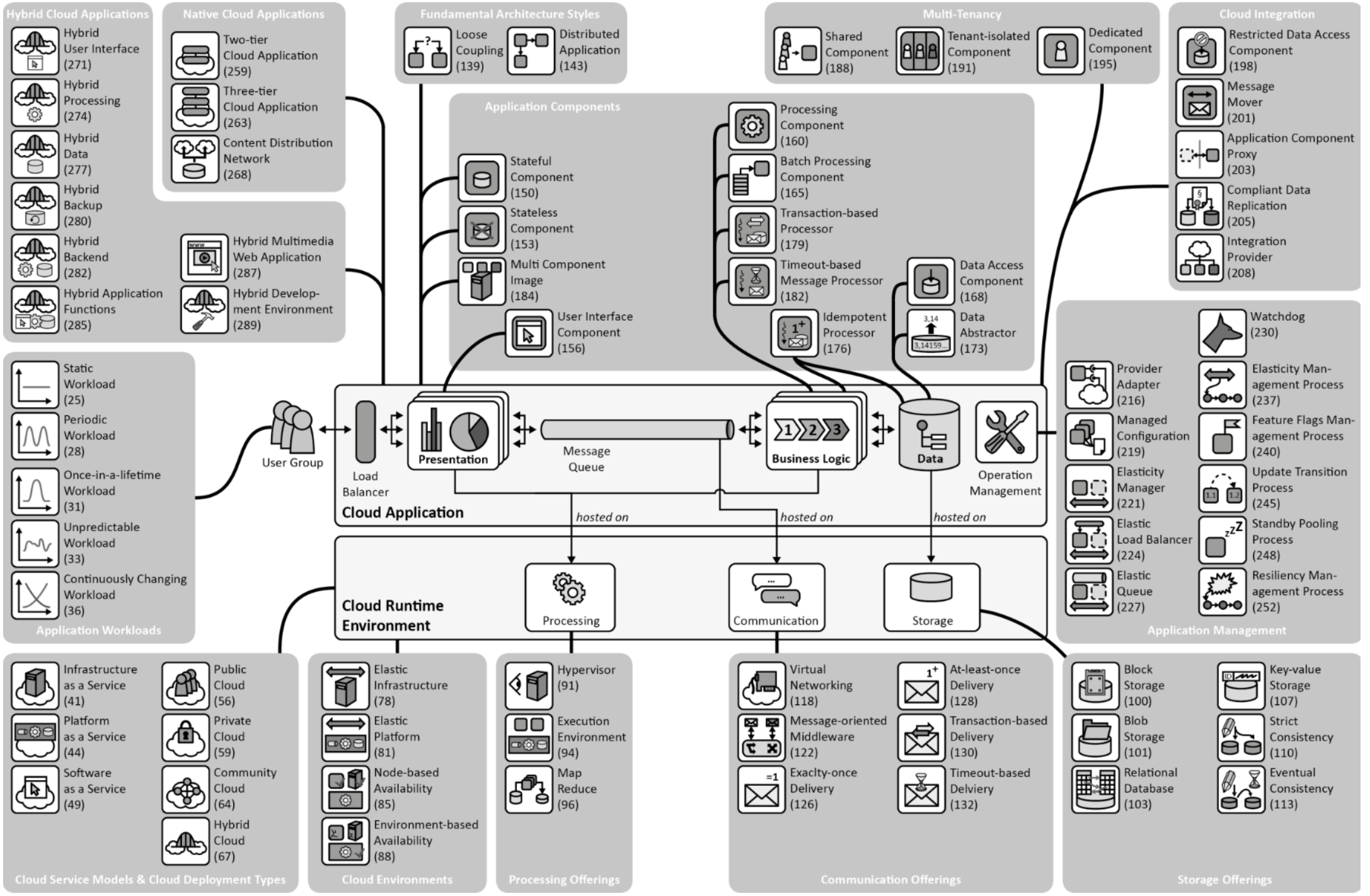
KUEHNE+NAGEL



“ A pattern language defines a collection of patterns and the rules to combine them into an architectural style. Pattern languages describe software frameworks or families of related systems.

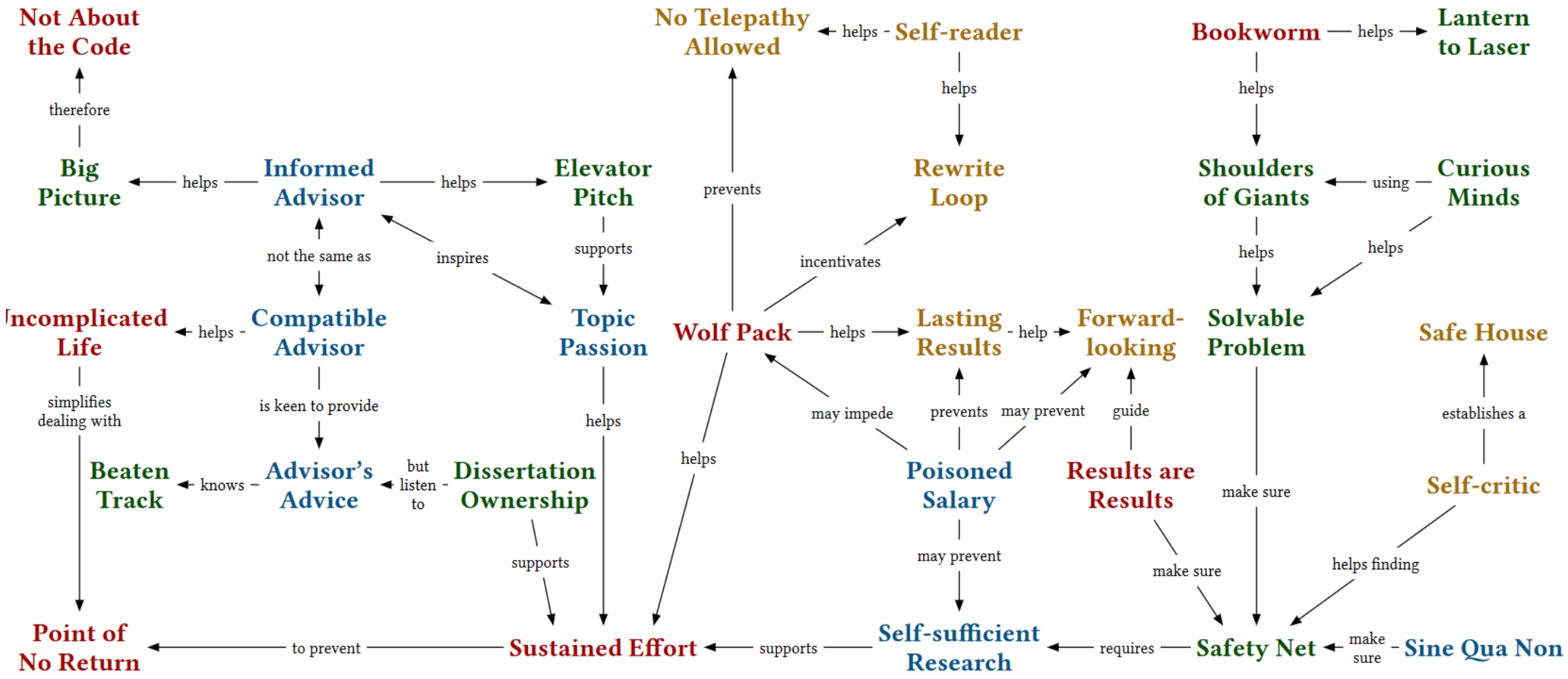
A pattern language may be regarded as a **lexicon of patterns plus a grammar** that defines how to weave them together into valid sentences. Ideally, **good pattern languages are generative, capable of generating all the possible sentences from a rich and expressive pattern vocabulary.**

Cloud Computing Pattern Language



Christoph Fehling, Frank Leymann, Ralph Retter, Walter Schupeck, and Peter Arbitter.
Cloud Computing Patterns.
https://www.cloudcomputingpatterns.org/

Pattern language for the masters student



Towards a pattern language for the masters student, Hugo Sereno Ferreira, André Restivo, Tiago Boldt Sousa

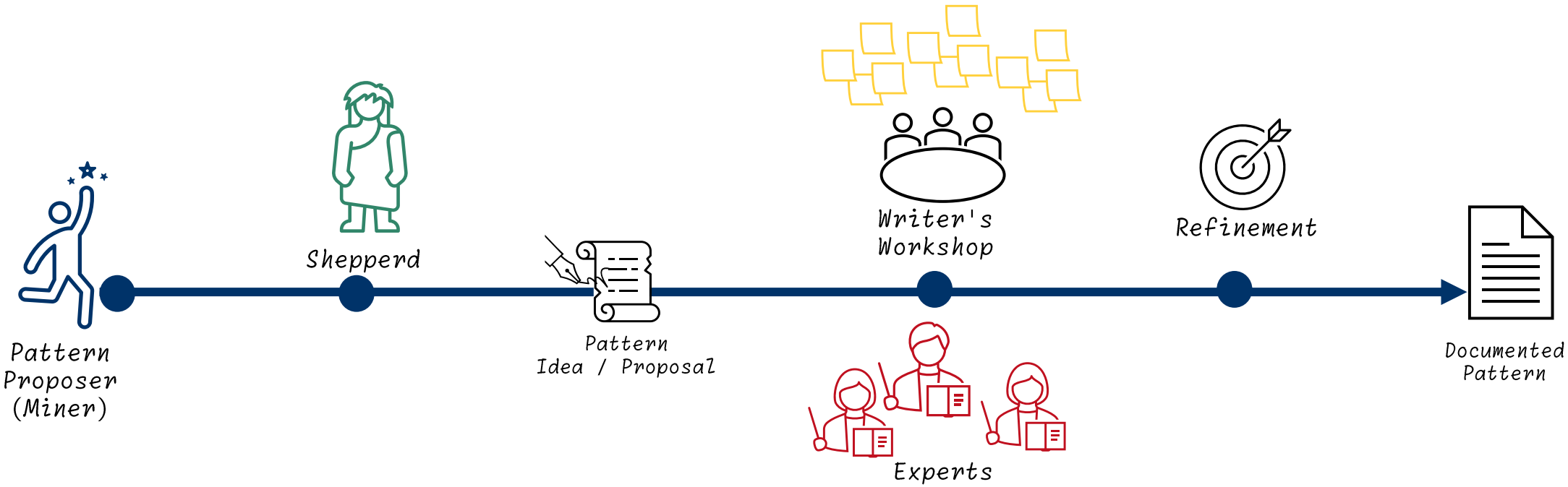
https://dl.acm.org/doi/10.1145/3361149.3361184

KUEHNE+NAGEL

Pattern Writing

Writing Workshop

KUEHNE+NAGEL



Read More

Read More

KUEHNE+NAGEL

- Pattern Languages of Programs, People, and Practices (PLoP) conferences, https://hillside.net/conferences/plop
- Patterns and Software: Essential Concepts and Terminology, https://www.bradapp.com/docs/patterns-intro.html
- A Pattern Language for Pattern Writing, https://hillside.net/index.php/a-pattern-language-for-pattern-writing
- Patterns Catalog, https://hillside.net/patterns/patterns-catalog
- A Learning Guide To Design Patterns, https://www.industriallogic.com/papers/learning.html

Inspire. Empower. Deliver.

KUEHNE+NAGEL