SCIENCE
PASSION
TECHNOLOGY

Design Patterns

448.058 (VO), 448.059 (UE)

Michael Krisper
Georg Macher

Plan for Today

- 13:00 – 16:00

- Introduction to „Design Patterns“
- Course Organisation
- Survey of Needs, Expectations, and Prerequisites

(Break, 15 minutes)

- What is a Design Pattern?
- Self-Assessment

Team

Michael Krisper
michael.krisper@tugraz.at
discord: Michael Krisper#5968
Uncertainty and Risk Propagation
Expert Judgment for Cyber-Security

Georg Macher
georg.macher@tugraz.at

Safety & Security
in Automotive & Autonomous Driving

ITI - Institute for Technical Informatics

Bachelor / Master Thesis
TOPICS PRESENTATION

Tuesday, October 13, 2020
2:30 – 4:00 PM
Institute of Technical Informatics
Discord: https://discord.gg/rFXPjW3

▸ Discord

▸ www.iti.tugraz.at

ITI
RESEARCH AREAS

Networked Embedded Systems

Prof. Kay Römer
roemer@tugraz.at

Embedded Automotive Systems

Prof. Marcel Baunach
baunach@tugraz.at

Industrial Informatics

Dr. Georg Macher
georg.macher@tugraz.at

Hardware/Software Codesign

Prof. Christian Steger
steger@tugraz.at

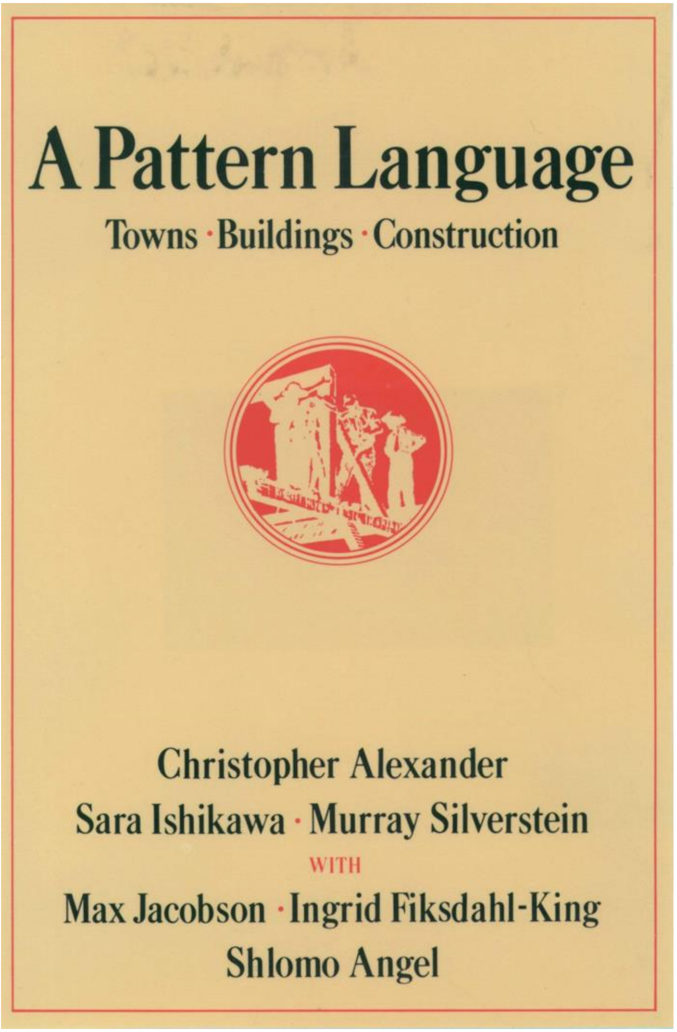
Smart Services

Prof. Eugen Brenner
brenner@tugraz.at

Founder of Pattern Movement: Christopher Alexander, Architect

The architect Christopher Alexander in 2012

(CC BY SA 4.0) Michaelmehaffy



A Pattern Language

Towns · Buildings · Construction

Christopher Alexander

Sara Ishikawa · Murray Silverstein

WITH

Max Jacobson · Ingrid Fiksdahl-King

Shlomo Angel

A Pattern Language, 1977

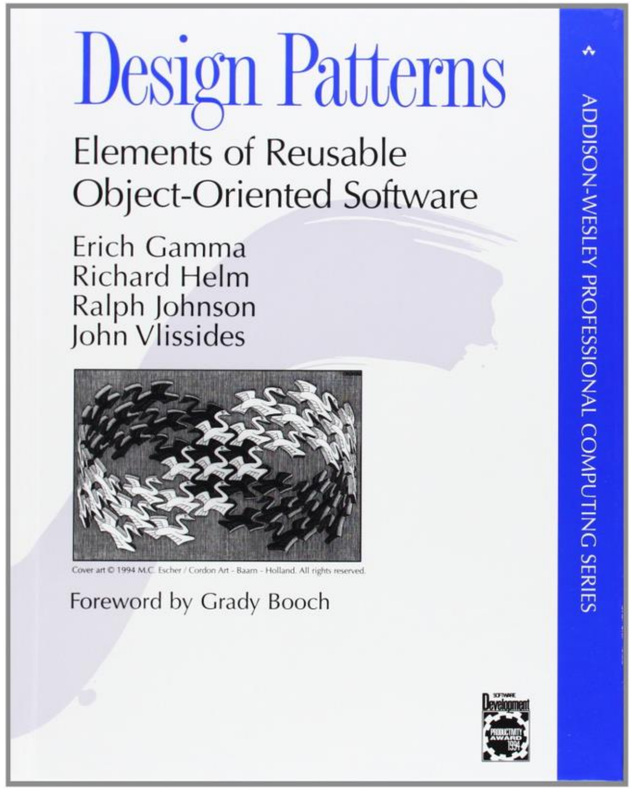
Kickstarters of Design Patterns for Software Development: Gang of Four: Johnson, Gamma, Helm, Vlissides

Ralph Johnson

Erich Gamma

Richard Helm

John Vlissides



Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.

Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Design Patterns, 1994

Grady Booch

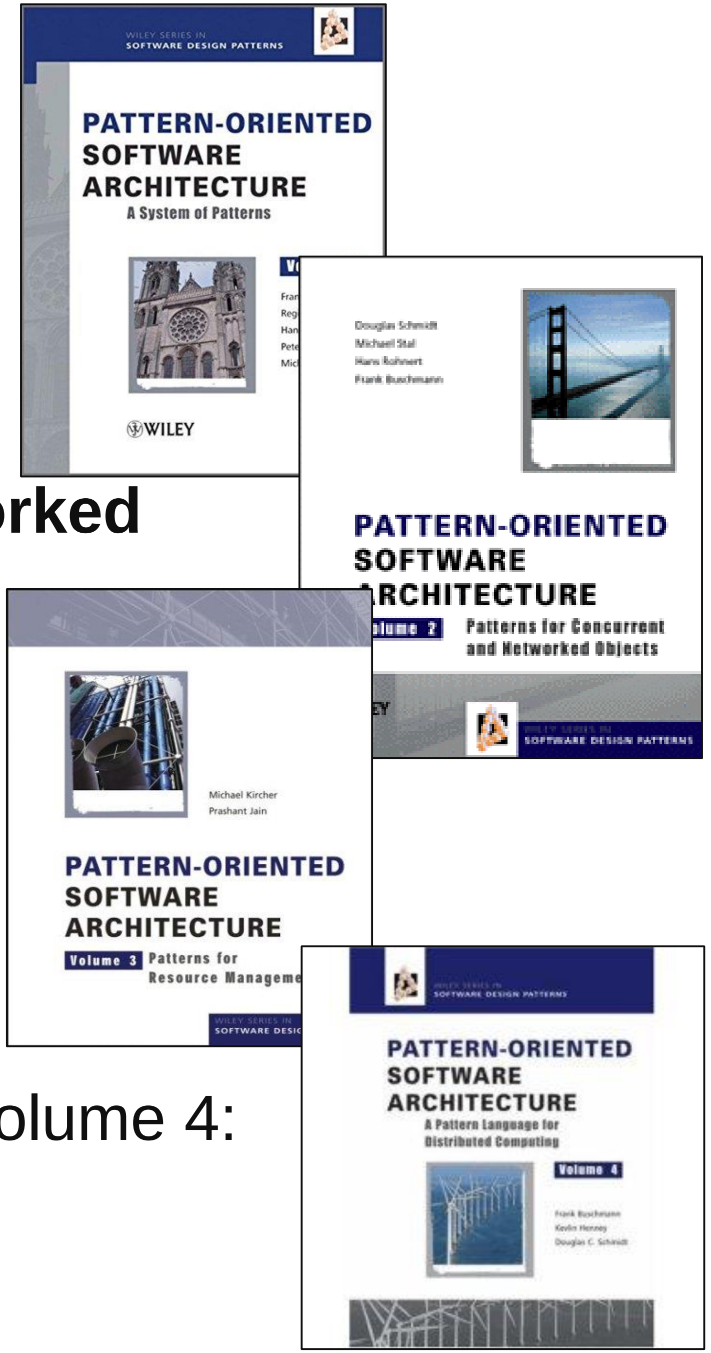
More Pattern Books…

POSA1: Pattern-Oriented Software Architecture Volume 1: **A system of patterns** (Buschmann, Meunier, et al., 1996)

POSA2: Pattern-Oriented Software Architecture Volume 2: **Patterns for Concurrent and Networked Objects** (Schmidt et al., 2000)

POSA3: Pattern-Oriented Software Architecture Volume 3: **Patterns for Resource Management** (Kircher and Jain, 2004)

POSA4: Pattern-Oriented Software Architecture Volume 4: **Pattern Language for Distributed Computing** (Buschmann, Henney, and Schmidt, 2007)



WILEY SERIES IN SOFTWARE DESIGN PATTERNS

PATTERN-ORIENTED SOFTWARE ARCHITECTURE

A System of Patterns

WILEY

Douglas Schmidt
Michael Stal
Hans Rohnert
Frank Buschmann

PATTERN-ORIENTED SOFTWARE ARCHITECTURE

Patterns for Concurrent and Networked Objects

WILEY SERIES IN SOFTWARE DESIGN PATTERNS

Michael Kircher
Prashant Jain

PATTERN-ORIENTED SOFTWARE ARCHITECTURE

Volume 3 Patterns for

WILEY SERIES IN SOFTWARE DESIGN PATTERNS

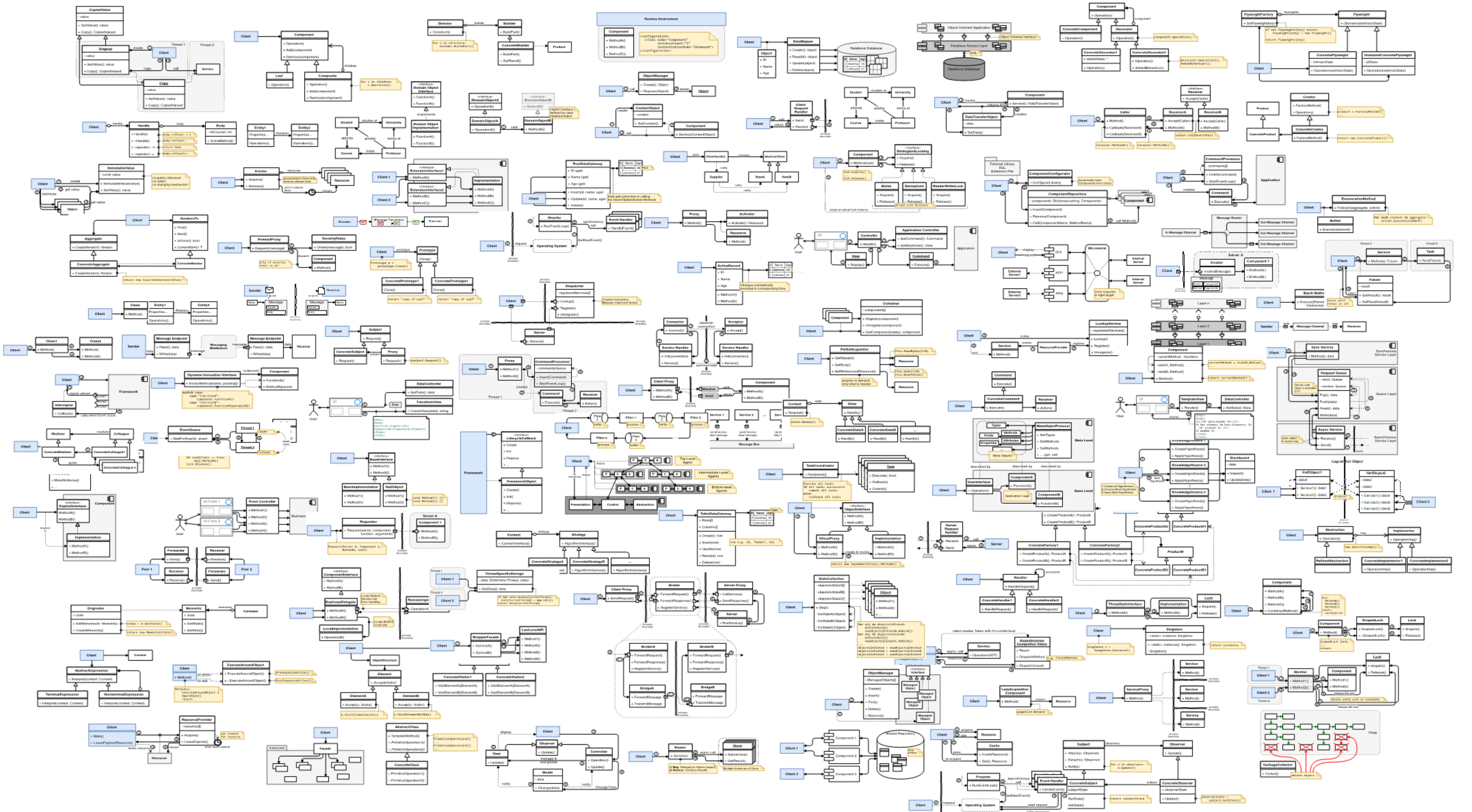
PATTERN-ORIENTED SOFTWARE ARCHITECTURE

A Pattern Language for Distributed Computing

Volume 4

Frank Buschmann
Kevlin Henney
Douglas C. Schmidt

Patterns in this course…



Patterns in this course…

- **Wrapping**: Adapter, Façade, Decorator, Proxy
- **Creation**: Factory Method, Abstract Factory, Builder, Prototype, Singleton, Flyweight
- **Behaviour**: Strategy, Command, State
- **Architecture**: Layers, Pipes & Filters, Broker, Master-Slave, Client-Server
- **Collections**: Iterator, Visitor, Composite, Null-Object
- **Communication**: Observer, Bridge, Broker, Mediator, Blackboard, Microkernel, Client-Dispatcher-Server/Lookup, Messaging & Service-Orientation: Message, Message-Endpoint, Message-Translator, Message-Router, MVC
- **Concurrency**: Locks, Monitor, Active Object, Future, Scoped Locking, Thread-Specific Storage, Double-Checked-Locking, Async/Await, Proactor, Reactor
- **Resources**: Lazy Acquisition, Eager Acquisition, Partial Acquisition, Caching & Pooling, Leasing, Garbage Collector
- **Others**: Memento, Counted Pointer, Chain of Responsibility, Interpreter/Abstract Syntax Tree

When will you need Design Patterns?

- Every time you develop and design software!

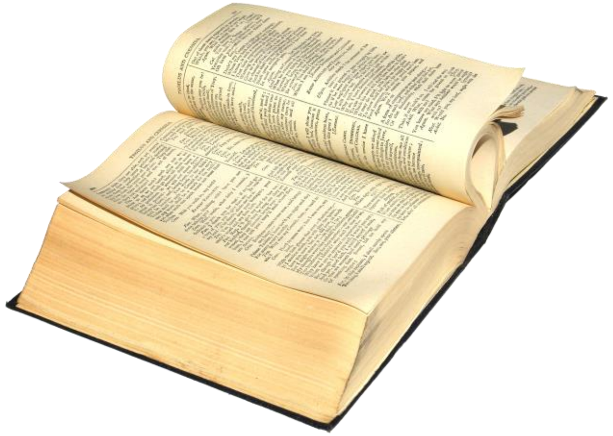
Examples:

- You are a Software Developer and need to implement specific tasks in your product.
- You are a Senior Software Architect in a company and have to manage complex software requirements and design flexible software architectures.
- You are a startup founder and want to write software for a product which is extensible, and flexible.
- You are a student and have to solve a software problem for an exercise at the university.

Learning Goals

Design Patterns Theory

- What is a design pattern? Why do we need them?
- What are the core principles behind design patterns?
- How to describe design patterns?
- What is a pattern language?



Design Patterns in Detail

- Know core ideas and application of important design patterns! (~50)

Application of Design Patterns

- When to use what?

Learning Goals

- You know common design patterns and their core idea (approx. 50 patterns).
- You can apply them in software development regardless of the programming language or development environment.
- You can derive the consequences of design patterns and see the design decisions.
- You decide if the consequences of a pattern are acceptable or not.
- You avoid overengineering and misuse of patterns.
- You can make reasonable design decisions by balancing out the forces, consequences, and requirements for arbitrary problems and contexts.

Course Organisation

Organisation: „Digital First“

- TeachCenter: https://tc.tugraz.at/main/course/view.php?id=2199
- Lectures will be held in BigBlueButton
- Communication via eMail

Design Patterns, VO (approx. 110 students)

- Wednesdays, 13:00 - 16:00 (Attendance not required)
  - 13:00 – 14:00 Video Self-Lecture and Discussions
  - 14:00 – 14:15 Break
  - 14:15 – 16:00 Live Programming and Discussions

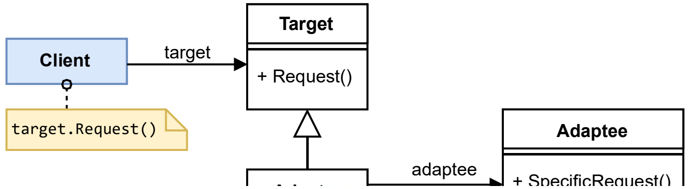
- Pattern videos and slides will be supplied
- Edited live recordings will be supplied

- **Exam: 27.01.2021** (in https://exam.tugraz.at/moodle/course/view.php?id=69)

Schedule Design Patterns VO

Date	from	to	Content
07.10.2020	13:00	16:00	Introduction, Organisation
14.10.2020	13:00	16:00	Theory, Principles, and Guidelines
21.10.2020	13:00	16:00	Adapter, Facade, Decorator, Proxy
28.10.2020	13:00	16:00	Layers, Broker, Pipes & Filters, Master/Slave, Client/Server
04.11.2020	13:00	16:00	Factory Method, Abstract Factory, Builder, Singleton, Prototype, Memento, State, Flyweight
11.11.2020	13:00	16:00	Iterator, Visitor, Strategy, Command, Composite, Template Method, Map/Reduce, Fluent Interface
18.11.2020	13:00	16:00	Mediator, Bridge, Blackboard, Microkernel, Broker, Messages (Message, Endpoint, Translator, Router)
25.11.2020	13:00	16:00	Locks (Mutex,Semaphor, Condition Variable), Scoped Locking, Double Checked Locking, Monitor, Future/Asynchronous Completion Token, Active Object, Thread Specific Storage
02.12.2020	13:00	16:00	Lazy Acquisition, Eager Acquisition, Partial Acquisition, Caching, Pooling, Leasing, Garbage Collector, Scoped Resource, Active Record
09.12.2020	13:00	16:00	Chain of Responsibility, Counted Pointer/Smart Pointer/Unique Pointer, Interpreter/Abstract Syntax Tree
16.12.2020	13:00	16:00	Forwarder/Receiver, Proactor, Reactor, Async/Await, Coroutines
13.01.2021	13:00	16:00	Model-View-Controller, Model-View-Viewmodel, Model-View-Presenter, Presentation-Abstraction-Control
20.01.2021	13:00	16:00	Summary
27.01.2021	**13:00**	**16:00**	**Exam**

What is a Design Pattern?



What is a pattern?

“A proven solution for a (recurring) problem.”

A solution idea, scheme, or template.

Patterns are a universal principle:

- Economics (Etzioni, 1964)
- Social Interaction (Newell,Simon, 1972)
- Architecture (Alexander et. al., 1975)
- Software (General awareness from 1990’s on)

Purpose of Design Patterns

- Easier knowledge transfer
- Efficient problem solving by reusing existing ideas
  “Don’t reinvent the wheel”
- Establishes a common vocabulary, terminology, or language
- Increases usefulness of an idea by generalizing the solution

Types of Design Patterns

Architectural Patterns

- Fundamental structural patterns
- Stencils for whole architectures
- Examples: Layers, Pipes-And-Filters, Broker, Model-View-Controller, Microkernel, Async-Await

Design Patterns

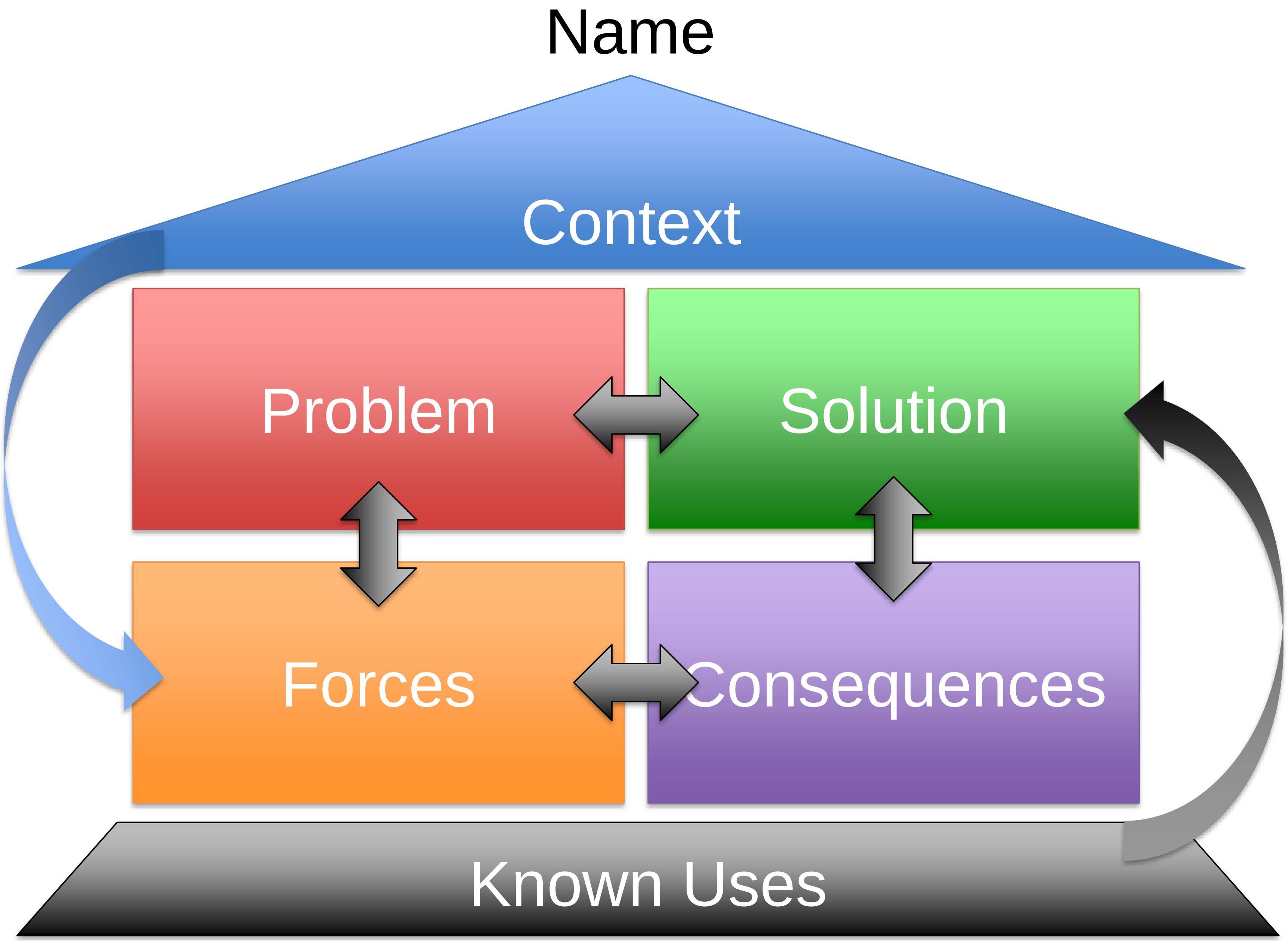
- Solution templates for more isolated problems
- Examples: Composite, Adapter, Proxy, Factory

Idioms

- Fine-Grained Patterns for problems in specific programming languages or environments
- Examples: Counted Pointer, Scoped Locking, Variadic Macros

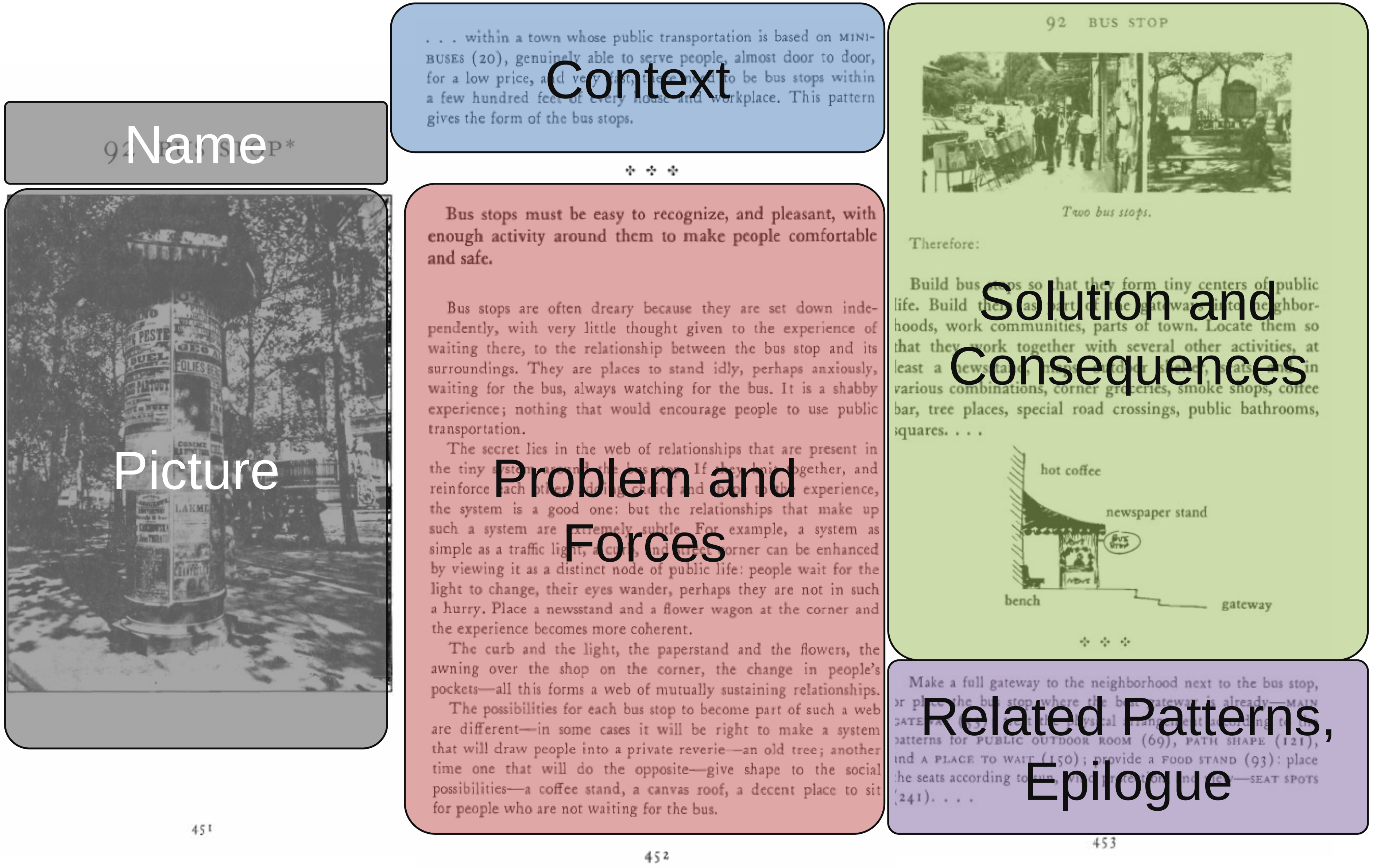
Pattern format

- **Name**: A catchy name for the pattern
- **Context**: The situation where the problem occurs
- **Problem**: General Problem Description
- **Forces**: Requirements and Constraints - Why does the problem hurt in this context?
- **Solution**: Generic Description of a proven solution.
  Static Structures, Dynamic Behaviour, Actionable Steps
- **Consequences (Rationale, Resulting Context):**
  - What are the benefits and drawbacks? Pro and Contra?
  - What are the liabilities, limitations and tradeoffs?
  - How are the forces resolved?
- **Known-Uses**: Real Life Examples



Name
Context
Problem
Solution
Forces
Consequences
Known Uses

Alexandrian Pattern Format



Name

92 BUS STOP*

Picture

Context

. . . within a town whose public transportation is based on MINIBUSES (20), genuinely able to serve people, almost door to door, for a low price, and very fast, there need to be bus stops within a few hundred feet of every house and workplace. This pattern gives the form of the bus stops.

✣ ✣ ✣

Problem and Forces

Bus stops must be easy to recognize, and pleasant, with enough activity around them to make people comfortable and safe.

Bus stops are often dreary because they are set down independently, with very little thought given to the experience of waiting there, to the relationship between the bus stop and its surroundings. They are places to stand idly, perhaps anxiously, waiting for the bus, always watching for the bus. It is a shabby experience; nothing that would encourage people to use public transportation.

The secret lies in the web of relationships that are present in the tiny system around the bus stop. If they knit together, and reinforce each other, adding choice and hope to the experience, the system is a good one: but the relationships that make up such a system are extremely subtle. For example, a system as simple as a traffic light, a curb, and street corner can be enhanced by viewing it as a distinct node of public life: people wait for the light to change, their eyes wander, perhaps they are not in such a hurry. Place a newsstand and a flower wagon at the corner and the experience becomes more coherent.

The curb and the light, the paperstand and the flowers, the awning over the shop on the corner, the change in people's pockets—all this forms a web of mutually sustaining relationships.

The possibilities for each bus stop to become part of such a web are different—in some cases it will be right to make a system that will draw people into a private reverie—an old tree; another time one that will do the opposite—give shape to the social possibilities—a coffee stand, a canvas roof, a decent place to sit for people who are not waiting for the bus.

Solution and Consequences

Two bus stops.

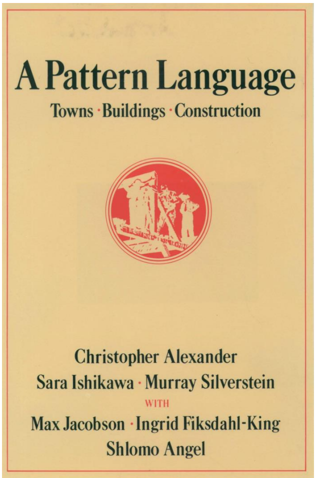
Therefore:

Build bus stops so that they form tiny centers of public life. Build them as part of the gateways into neighborhoods, work communities, parts of town. Locate them so that they work together with several other activities, at least a newsstand, maps, outdoor shelter, seats, and in various combinations, corner groceries, smoke shops, coffee bar, tree places, special road crossings, public bathrooms, squares. . . .

✣ ✣ ✣

Related Patterns, Epilogue

Make a full gateway to the neighborhood next to the bus stop, or place the bus stop where the best gateway is already—MAIN GATEWAY (53); treat the physical arrangement according to the patterns for PUBLIC OUTDOOR ROOM (69), PATH SHAPE (121), and A PLACE TO WAIT (150); provide a FOOD STAND (93): place the seats according to sun, wind protection, and view—SEAT SPOTS (241). . . .



A Pattern Language

Towns · Buildings · Construction

Christopher Alexander
Sara Ishikawa · Murray Silverstein
WITH
Max Jacobson · Ingrid Fiksdahl-King
Shlomo Angel

How Design Patterns emerge?

Design Patterns are found - not invented!

They emerge out of real use-cases/known-uses

1. Find patterns in real solutions
   ➔ At least three Known-Uses, Real Projects!
2. Write down the core idea and experiences
   ➔ Name, Context, Problem, Forces, Solution, Consequences, Known Uses
3. Discuss with others (often & repeatedly)
4. Improve Pattern (and repeat discussions)
5. Publish! (Conferences, Books, Blogs)
6. Continue to improve, apply and discuss pattern

Pattern Languages

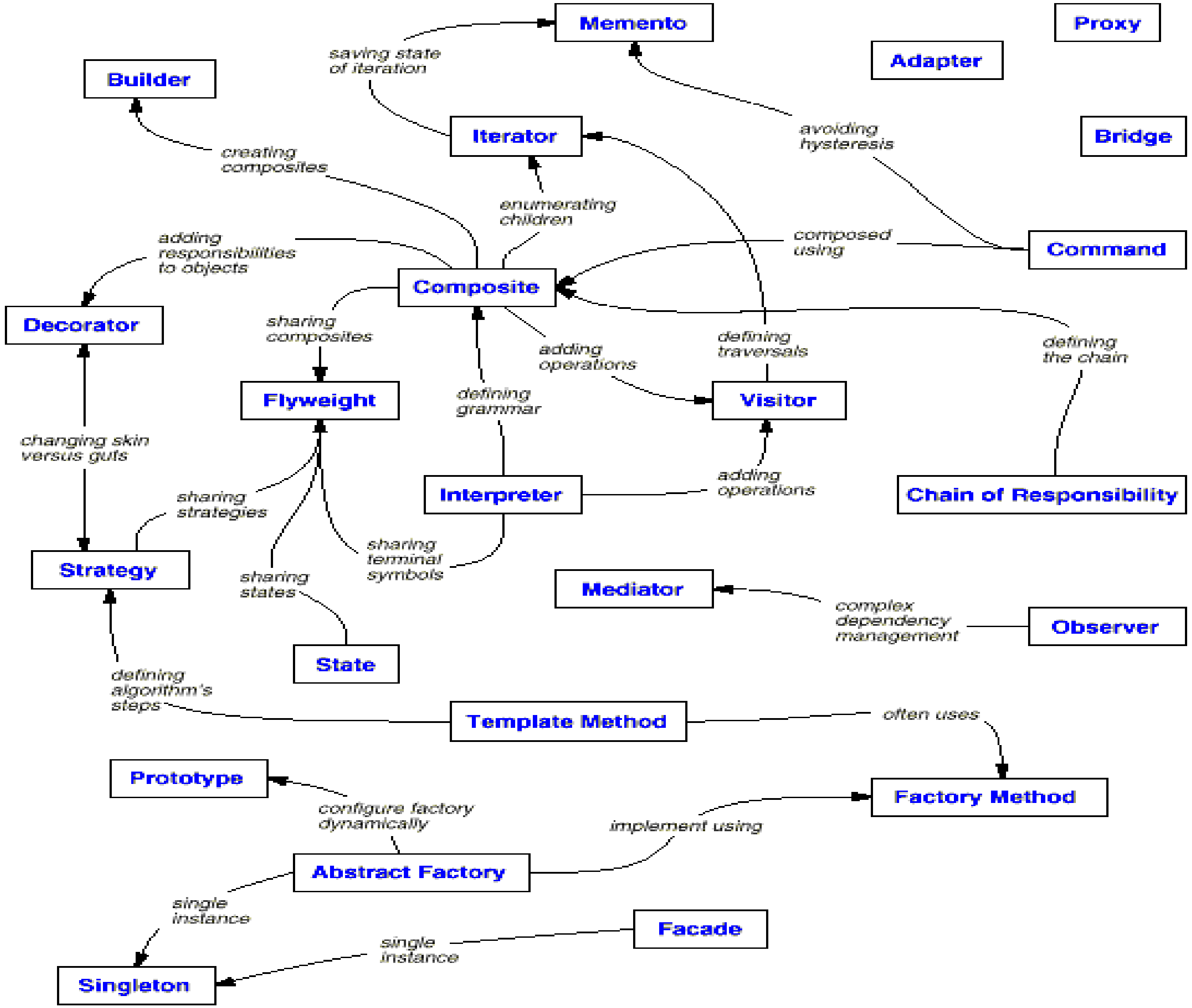
… are coherent systems of patterns.

Consisting of:

- Patterns
- Relations
- Principles (Guidelines for design and evolution):
  - How to create / implement
  - Beneficial combination of patterns
  - How to change/evolve

Daily Life Examples: Cooking, Sports, Crafts, Sailing, Architecture, Programming

GOF Pattern Language



Self-Assessment (9 Questions – 10 minutes)

1. When is the exam?
2. What is a design pattern?
3. Why are design patterns useful?
4. How can a design pattern be described? (Pattern format)
5. What are the essential parts of a design pattern?
6. Design patterns are invented. □ YES or □ NO?
7. What is a idiom and why is it different to an architectural design pattern?
8. What is a pattern language?
9. Can you name some real-life design pattern?