Software Systems Architecture

FEUP-M.EIC-ASSO-2023-2024

Ademar Aguiar, Neil Harrison

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Mark Andreessen
Renowned VC

Software is eating the world,
in all sectors. "The Wall Street Journal" in 2011

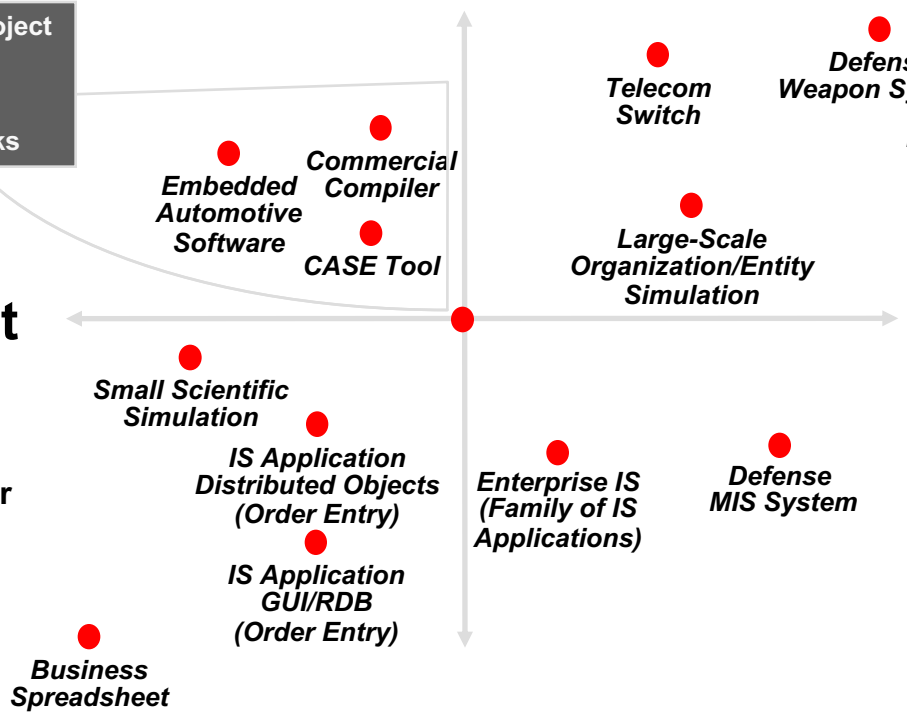
In the future
every company will become
a software company

Software Complexity

Higher technical complexity

- Custom, unprecedented, architecture reengineering

An average software project
- 5-10 people
- 3-9 month duration
- 3-5 external interfaces
- Some unknowns & risks



Lower management complexity

- Small scale
- Informal
- Single stakeholder
- "Products"

Higher management complexity

- Large scale
- Many stakeholders
- "Projects"

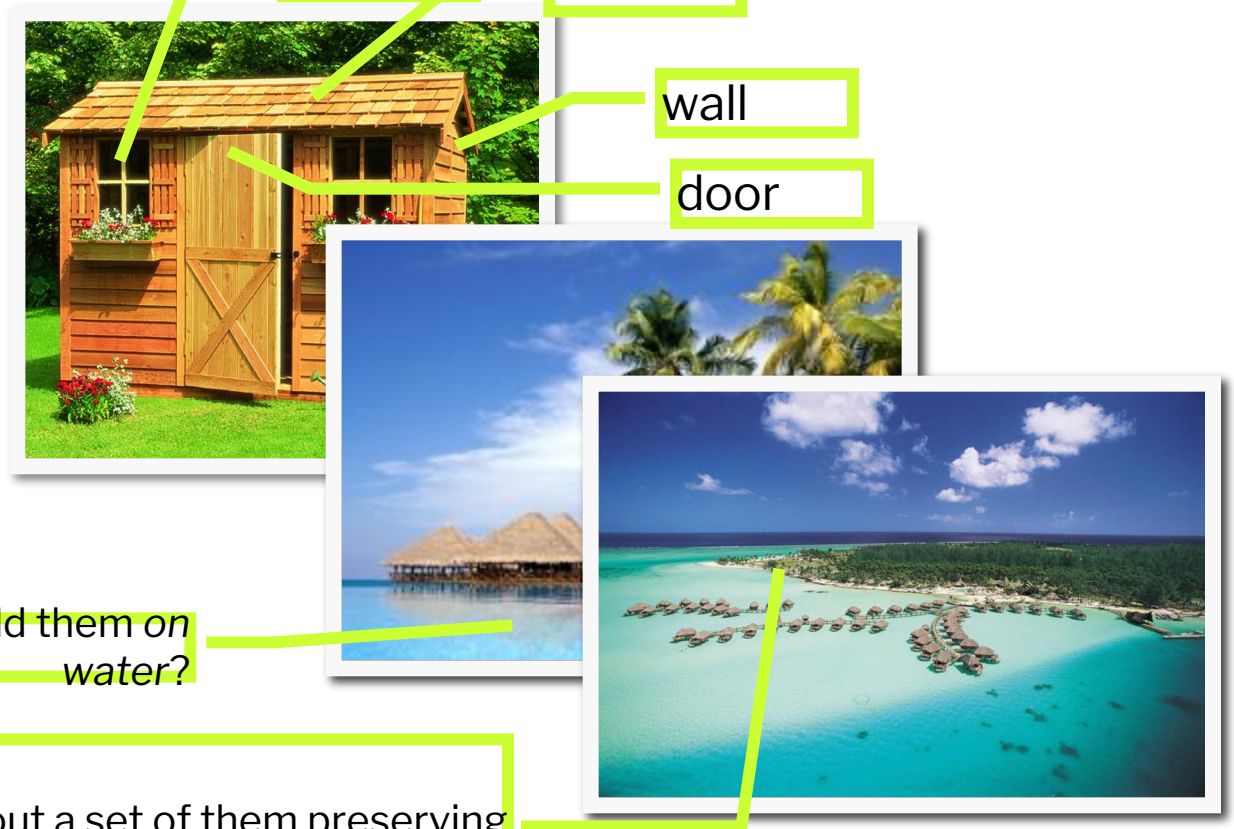
Lower technical complexity

- Mostly 4GL, or component-based
- Application reengineering
- Interactive performance

Walker Royce

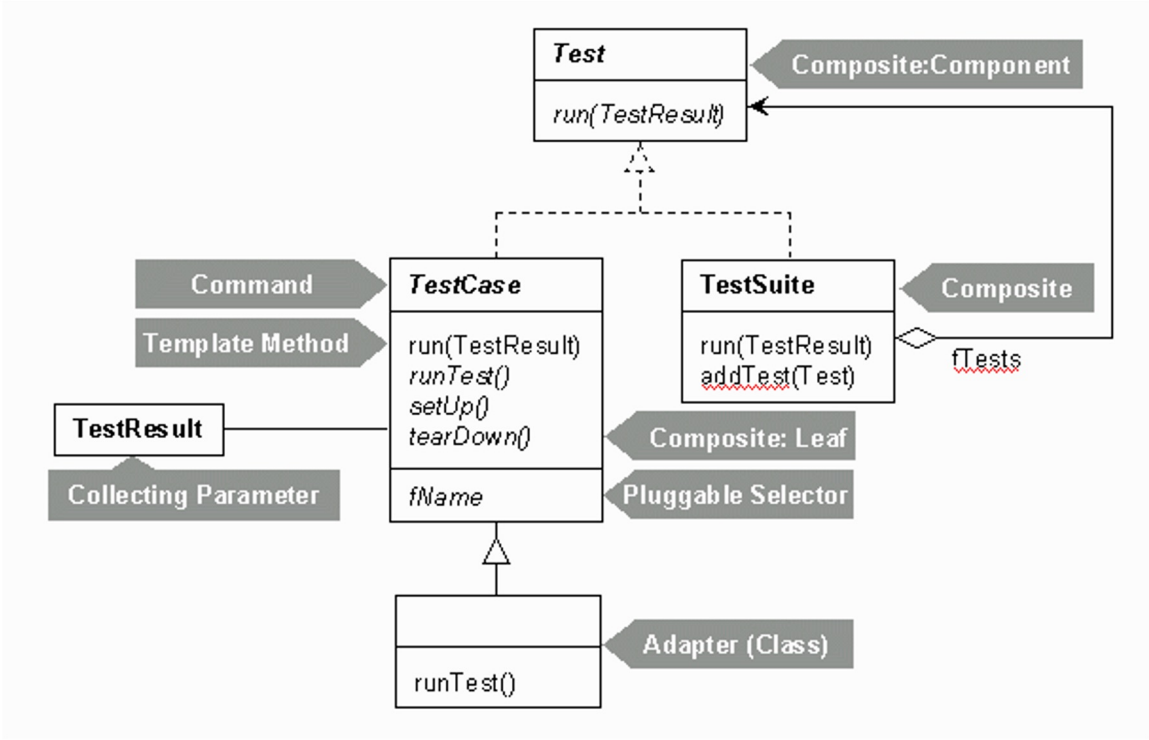
Architecting...

To architect a small house “seems” to be very easy...



Architecting JUnit...

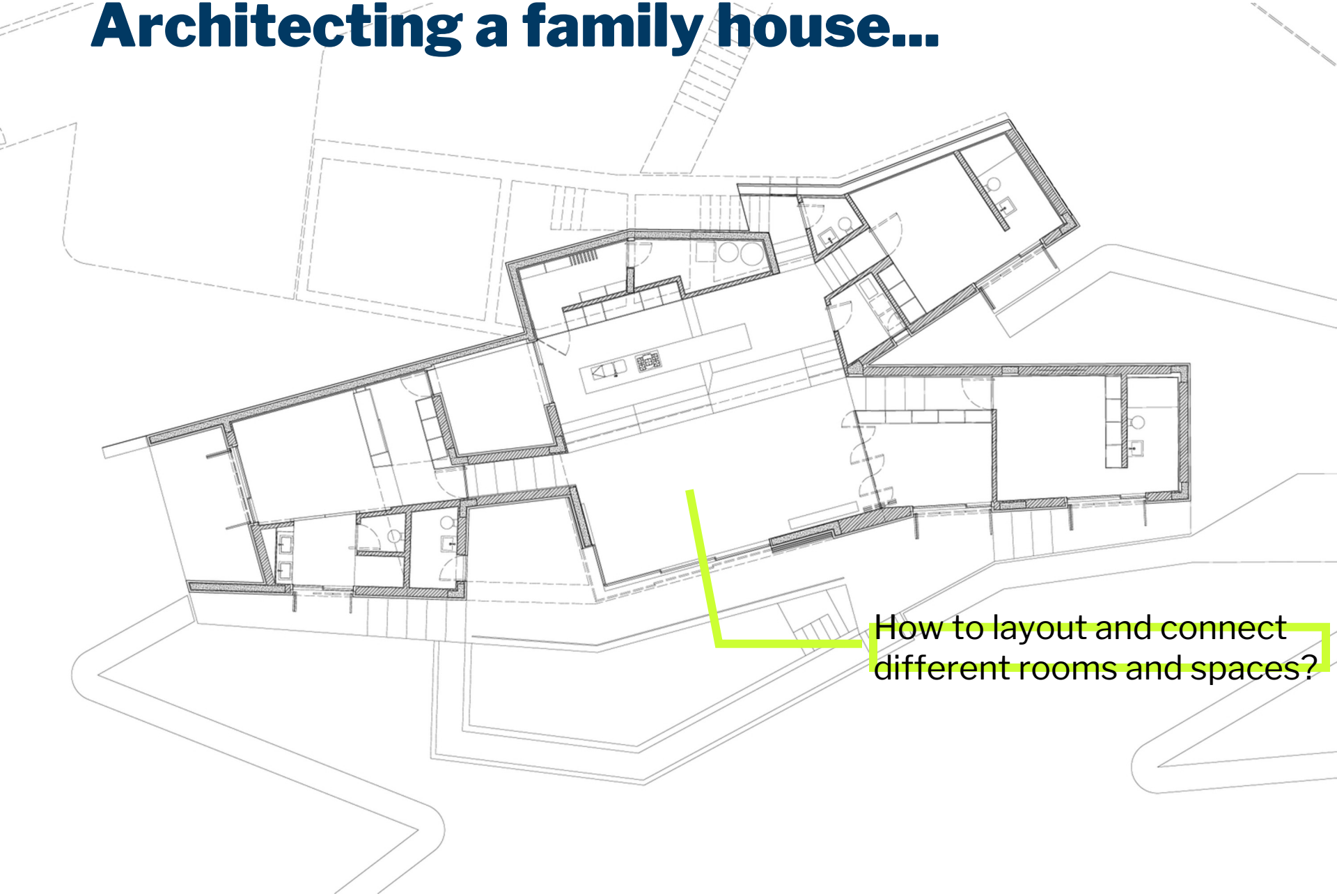
"Voilá" a well know"little software house": xUnit



Which building elements can we identify here? Hmm...

Architecting a family house...

Architecting a family house...



How to layout and connect different rooms and spaces?

Architecting a multi-story building...

How to layout and connect several houses?

Architecting a multi-story building…

How to build and maintain such building, and make it live?

Software cannot fail!!!


Software cannot fail!!!

Architecting...

Main diferences between all these buildings:

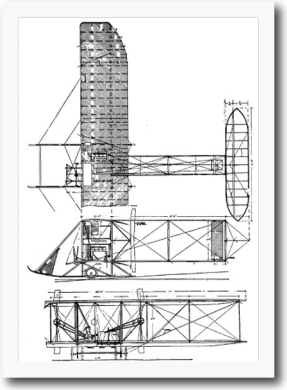
- Scale
- Cost, Time, Teams
- Process, skills
- Materials and technologies
- Risks
- Robustness
- Longevity...

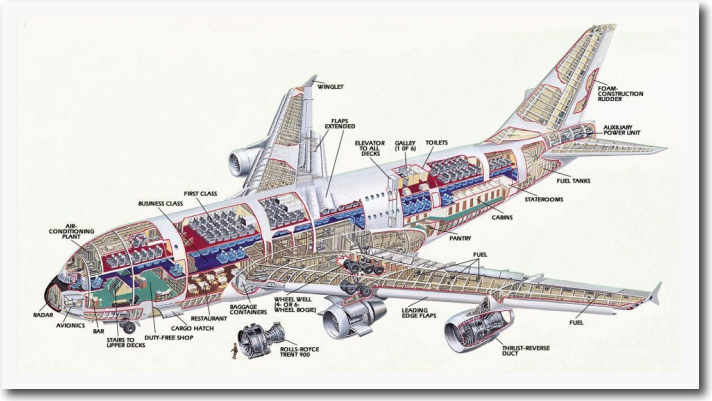
Traditional Engineering

Traditional engineering have evolved based on “stable systems”, mature, good references

Stable systems have evolved over time

- By trial and error
- By constant adaptation and specific needs
- By reuse and refinement of well-proven solutions
- By several advances in materials, processes and standards





Software Construction: is it harder? Yes!

Invisible constructions (logical product)

Temporal nature of software is complex and hard to understand (discrete systems)

Big needs to change and evolve along lifespan

Software must be very easy to adapt; each case is a case

Underlying technologies evolve very fast

Short history (since 1960?)

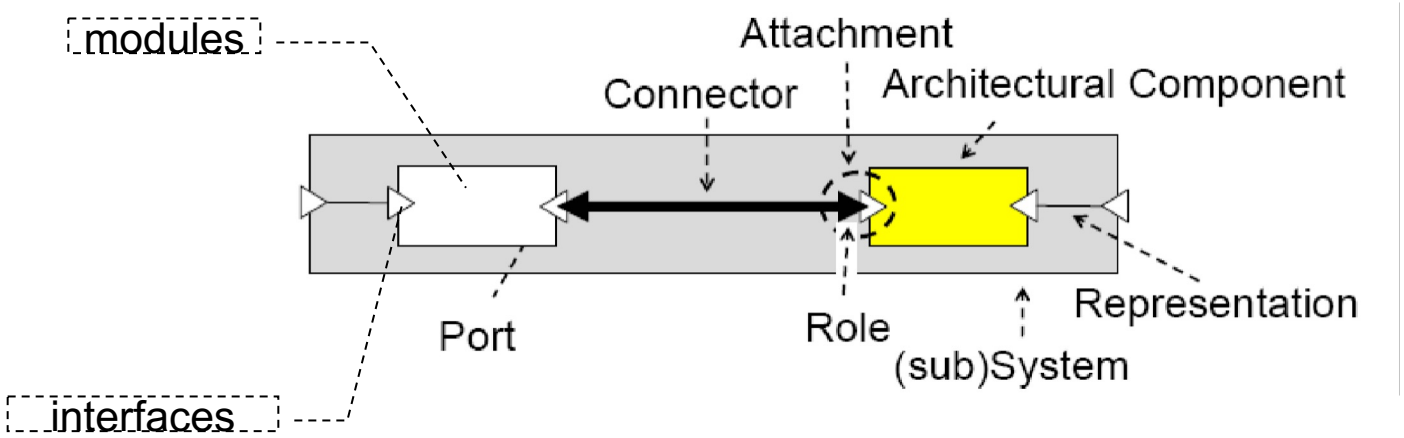
No “nature laws” or “phyical laws” to conform to...

Software Architecture

Software Architecture is the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.

IEEE Recommended Practice for Architectural Description of Software Intensive Systems (IEEE 1471-2000)

- Which **structural elements** and **interfaces** is the system made of?
- Which **composition mechanisms** are used to compound elements?
- Which **collaboration behaviors** between elements?
- Which main **architectural style** enforms the system?



The Software Architecture Role

Architectural Drivers
Understanding the goals; capturing, refining, and challenging the requirements and constraints

Designing Software
Creating the technical strategy, vision and roadmap

Technical Risks
Identifying, mitigating and owning the technical risks to ensure that the architecture "works"

Architecture Evolution
Continuous technical leadership and ownership of the architecture throughout the software delivery

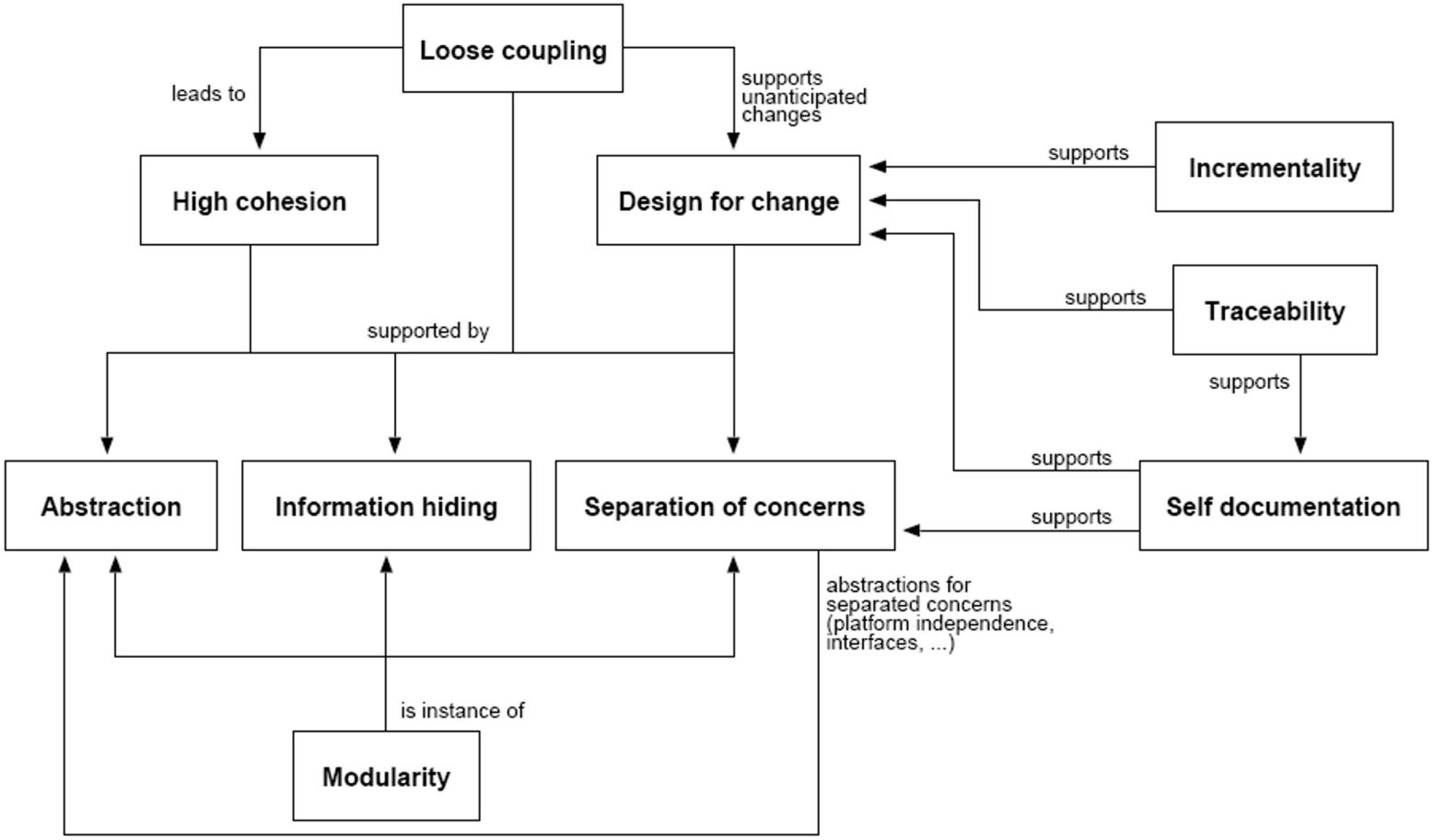
Coding
Involvement in the hands-on elements of the software delivery

Quality Assurance
Introduction and adherence to standards, guidelines, principles, etc.

Ultimate challenge
"to architect and design timeless and (ultra) large-scale software…"
(not only buildings, but cities of buildings…)

Brown, Simon. Software architecture for developers. Leanpub, 2015.

Software Architecture: key principles



Software Quality Attributes

Architectural **evolvability** is about preserving a system's software qualities over time

- Accessibility
- Accountability
- Accuracy
- Adaptability
- Administrability
- Affordability
- Agility
- Auditability
- Autonomy
- Availability
- Compatibility
- Composability
- Configurability
- Correctness
- Credibility
- Customizability
- Debuggability
- Determinability
- Demonstrability
- Dependability
- Deployability
- Discoverability
- Distributability
- Durability
- Effectiveness
- Efficiency
- Extensibility
- Failure-transparency
- Fault-tolerance
- Fidelity
- Flexibility
- Inspectability
- Installability
- Integrity
- Interchangeability
- ...

Reusing architectural knowledge

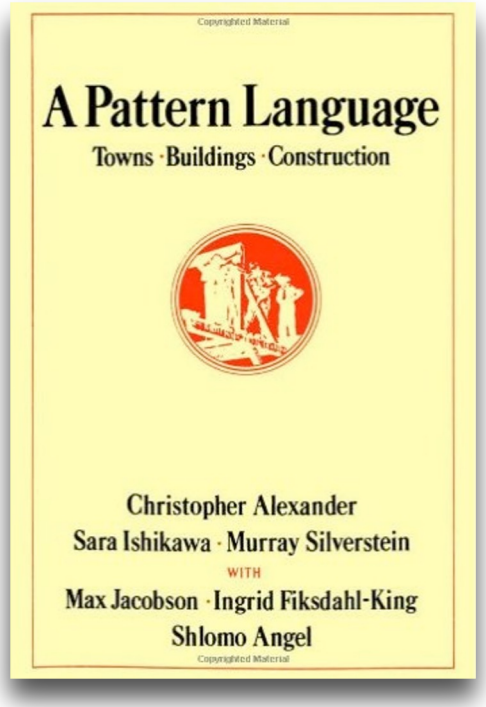
To architect solutions from scratch is hard and not effective in most cases

Good architects **reuse** solutions that worked well before and adapt them smoothly and incrementally to new situations.

Therefore it is very important:

- to know well proven solutions
- to reuse generic solutions
- to adapt existing solutions
- and then to add innovation over them to fit the needs.

Architectural Patterns, 1977



Christopher Alexander in this book presents a pattern language, an ordered collection of 253 patterns.

The goal was to enable non-experts to architect and design their own houses and communities.

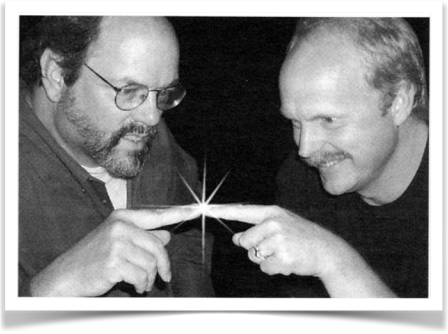
Patterns describe current practice and provide vision for the future.

Patterns integrate knowledge from diverse sources and link theory and practice.

Patterns have a strong “bottom up” orientation.

http://c2.com/cgi/wiki?ChristopherAlexander

Software Patterns, 1987..1994..

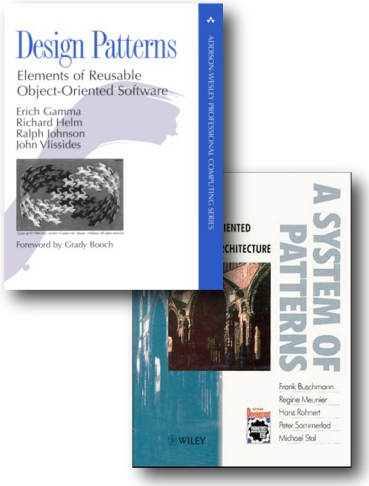


Kent Beck & Ward Cunningham, “Using Pattern Languages for Object-
Oriented Program”, *OOPSLA '87*, 1987.

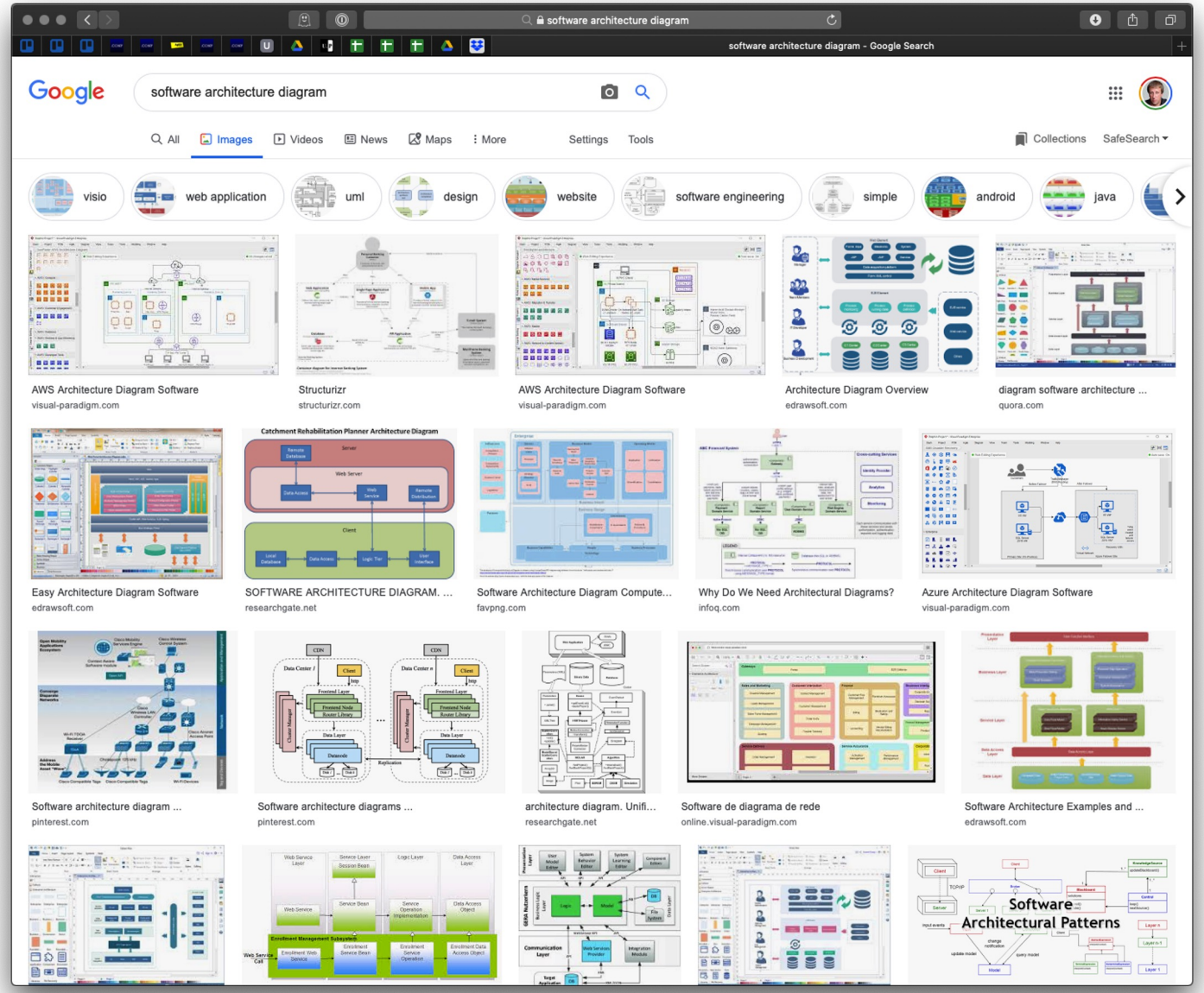
THE HILLSIDE GROUP

A new software patterns discipline started by a group later called “The Hillside Group”, 1993.

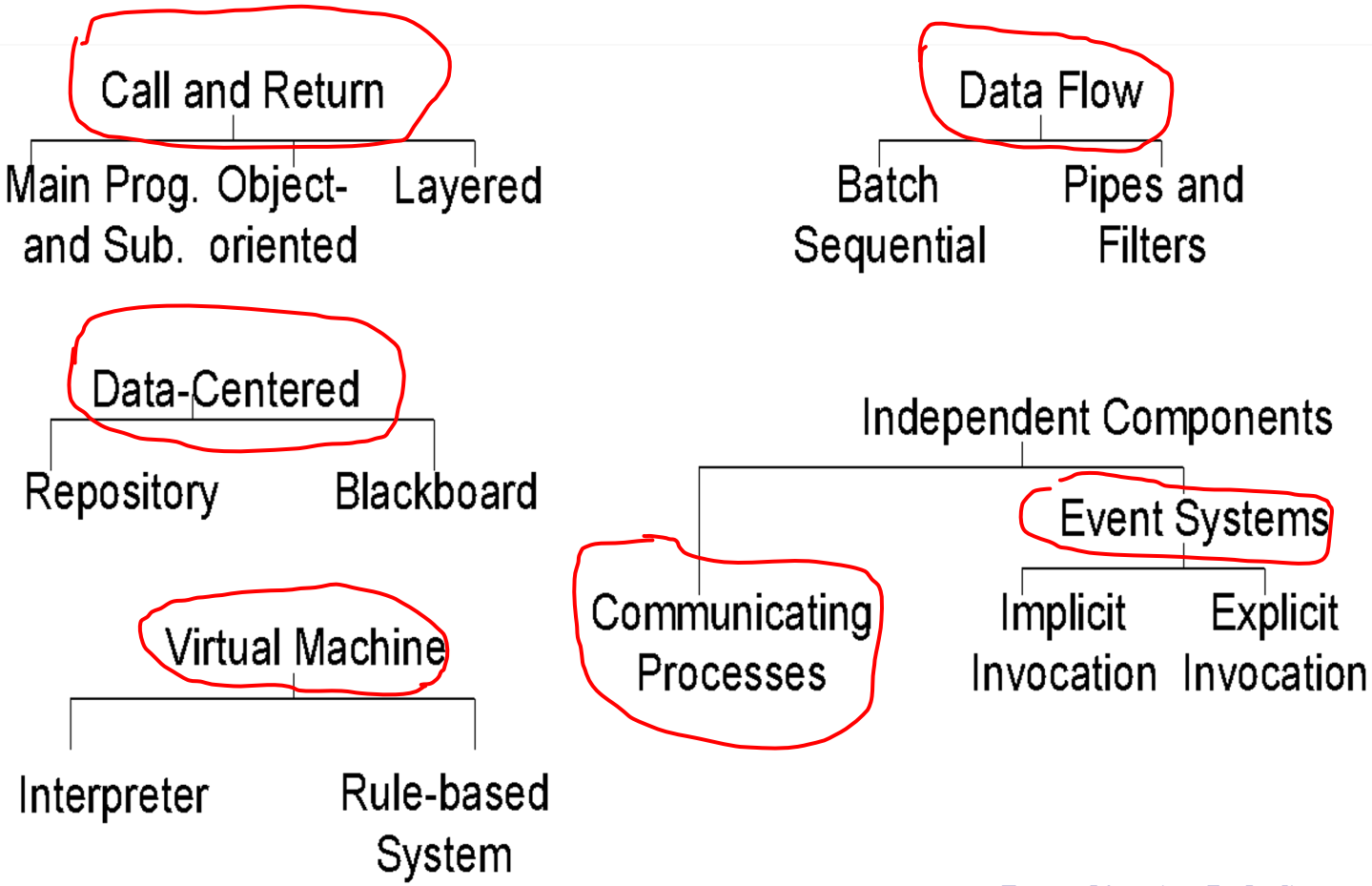
Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (aka Gang of Four), published the first book on “Design Pattens”, the “bible”, 1994.



Software Architecture: communicating



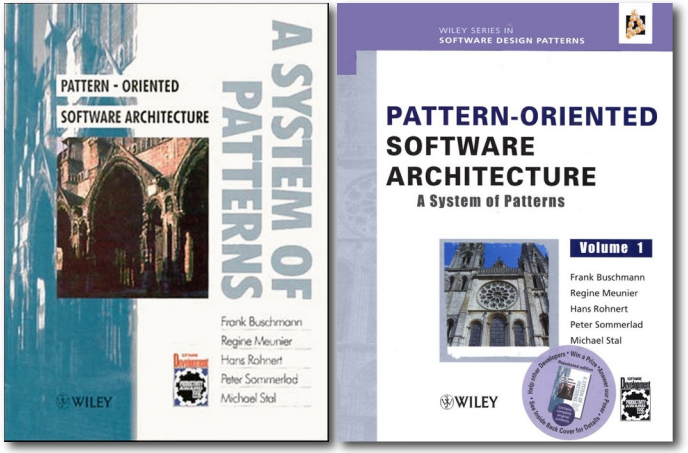
Architectural styles



From Chapter 5, Software Architecture in Practice, p. 95

POSA Patterns

"Pattern Oriented Software Architecture"

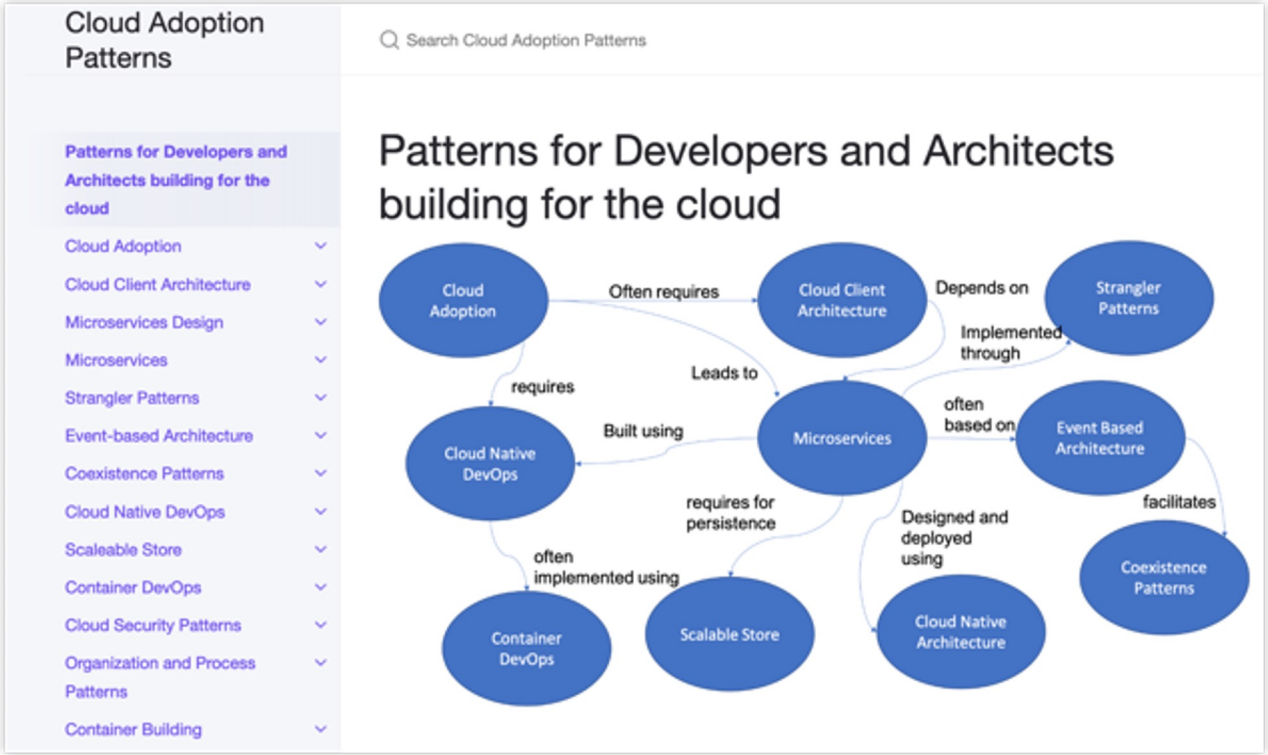


Pattern Oriented Software Architecture, A System of Patterns

Frank Buschmann
Regine Meunier
Hans Rohnert
Peter Sornmerlad
Michael Stal

Wiley, 1996

Cloud, Microservices & DevOps Patterns

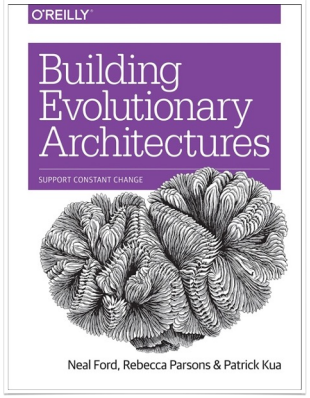


https://kgb1001001.github.io/cloudadoptionpatterns/

Evolutionary architecture

How is long-term planning possible when things are constantly changing in unexpected ways?

How can we build architectures that can gracefully change over time?



Building Evolutionary Architectures

Neal Ford
Rebecca Parsons
Patrick Kua

O'Reilly, 2017

UC Components

TP classes

- 14 weeks x 3 hours (1,5 T + 1,5 P) = 42h
- Lectures
- Case studies
- Development of projects
- Q&A sessions

Case studies & Projects

Case studies

- from web, publicly available
- from companies, by invited speakers

Architecture and Design Projects

- several small projects in groups (almost no coding)
- practice software design and architecture
- presentations to each other about their designs
- application of patterns

Grades

Distributed evaluation, no final exam, 1 mini-test.

Components

- 1 mini-test with access to materials, 90min, before May.
- 8-10 Assignments. Most are group projects that involve designing the architecture of a system.
- Individual assessment based on student's performance.

Formula

- (Mini-test x 30%) + (Asignments x 60%) + (Individual x 10%)

Self-formation of teams

Pick a team and add your GitHub username in the spreadsheet using your official Google account provided by the University.

Bibliography

[Alexander77] C. Alexander and S. Ishikawa and M. Silverstein, A Pattern Language, Oxford University Press, 1977.

[Alexander79] C. Alexander, A Timeless Way of Building, Oxford University Press, 1979.

[Gamma95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, Design Patterns – Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.

[Buschmann96] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, Pattern Oriented Software Architecture - a System of Patterns, John Wiley and Sons, 1996.

[Buschmann99] F. Buschmann, Building Software with Patterns, EuroPLoP'99 Proceedings.

...