Introduction to Software Design

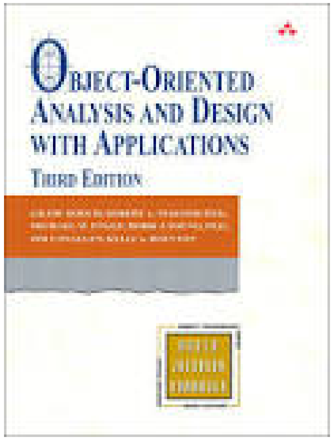
UA.DETI.PDS - 2024/25

José Luis Oliveira

UNIVERSIDADE DE AVEIRO

Resources & Credits

- **Object-Oriented Analysis and Design with Applications**
  Grady Booch, Robert A. Maksimchuk,
  Michael W. Engle, Bobbi J. Young,
  Jim Conallen, Kelli A. Houston
  Addison-Wesley Professional; 3rd edition

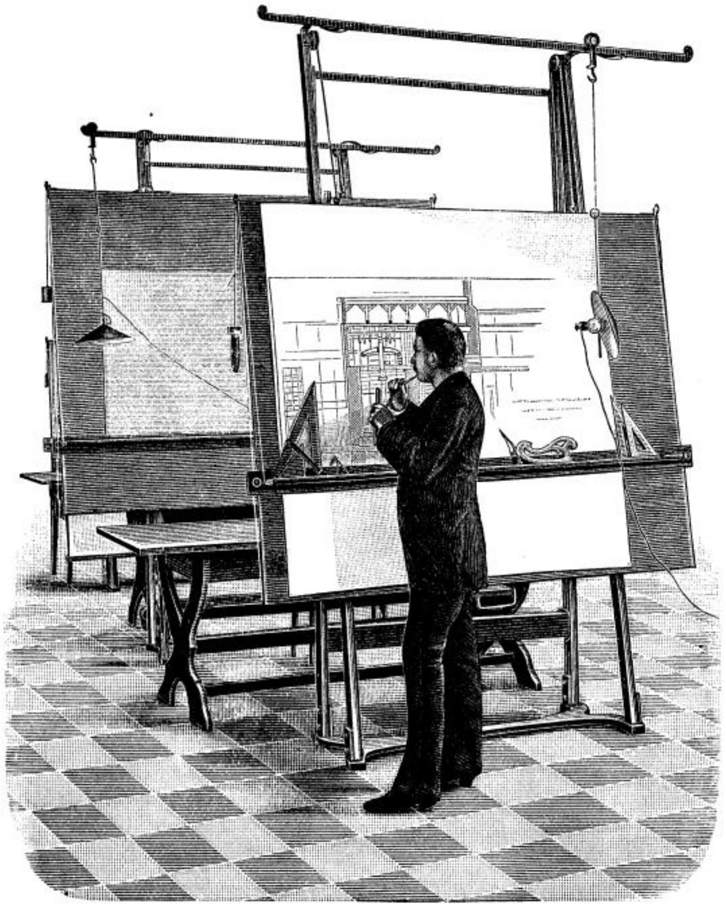


Object-Oriented Analysis and Design with Applications
Third Edition

- **Programming in the Large with Design Patterns**
  Eddie Burris
  Pretty Print Press

Programming in the Large with
Design Patterns
Eddie Burris

Design



"You can use an eraser on the drafting table or a sledgehammer on the construction site. "

--Frank Lloyd Wright

Design is a Universal Activity

- Any product that is an aggregate of more primitive elements, can benefit from the activity of design.

<u>Building Design</u>

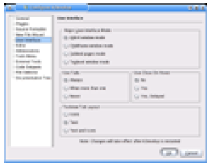
Doors, windows, plumbing fixtures, …

Wood, steel, concrete, glass, …

<u>Landscape Design</u>

Trees, flowers, grass, rocks, mulch, …

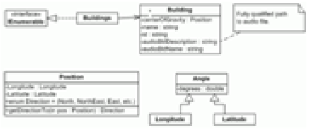
<u>User Interface Design</u>



Tree view, table view, File chooser, …

Buttons, labels, text boxes, …

<u>Software Design</u>

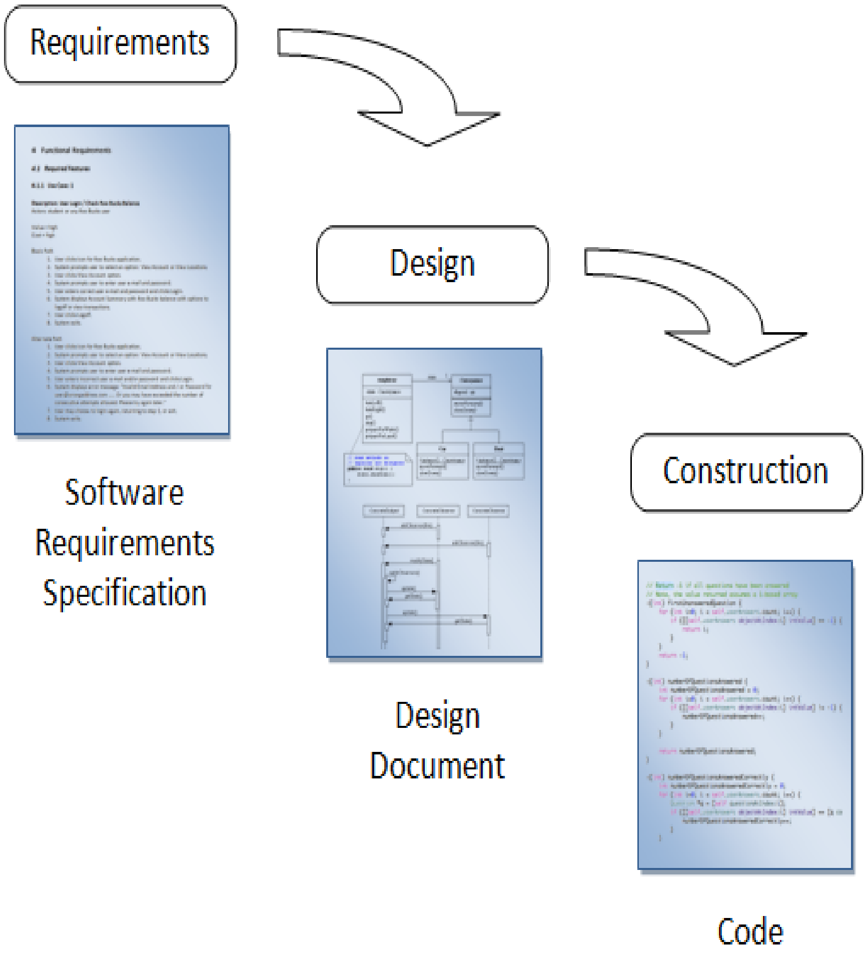


Classes, procedures, functions, …

Data declaration, expressions, control flow statements, …

What is Software Design?

- Design bridges the gap
  - between knowing what is needed (software requirements specification)
  - to entering the code that makes it work (the construction phase).



Requirements

Software Requirements Specification

Design

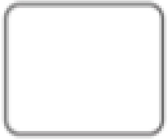
Design Document

Construction

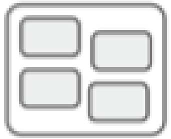
Code

What is Software Design?

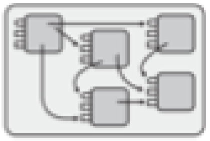
- Design is needed at several different levels of detail in a system:



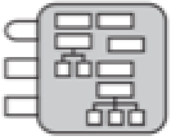
– system



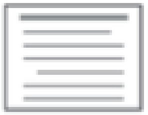
– subsystems or packages: user interface, data storage, application-level classes, graphics . . .



– classes within packages, class relationships, interface of each class, public methods

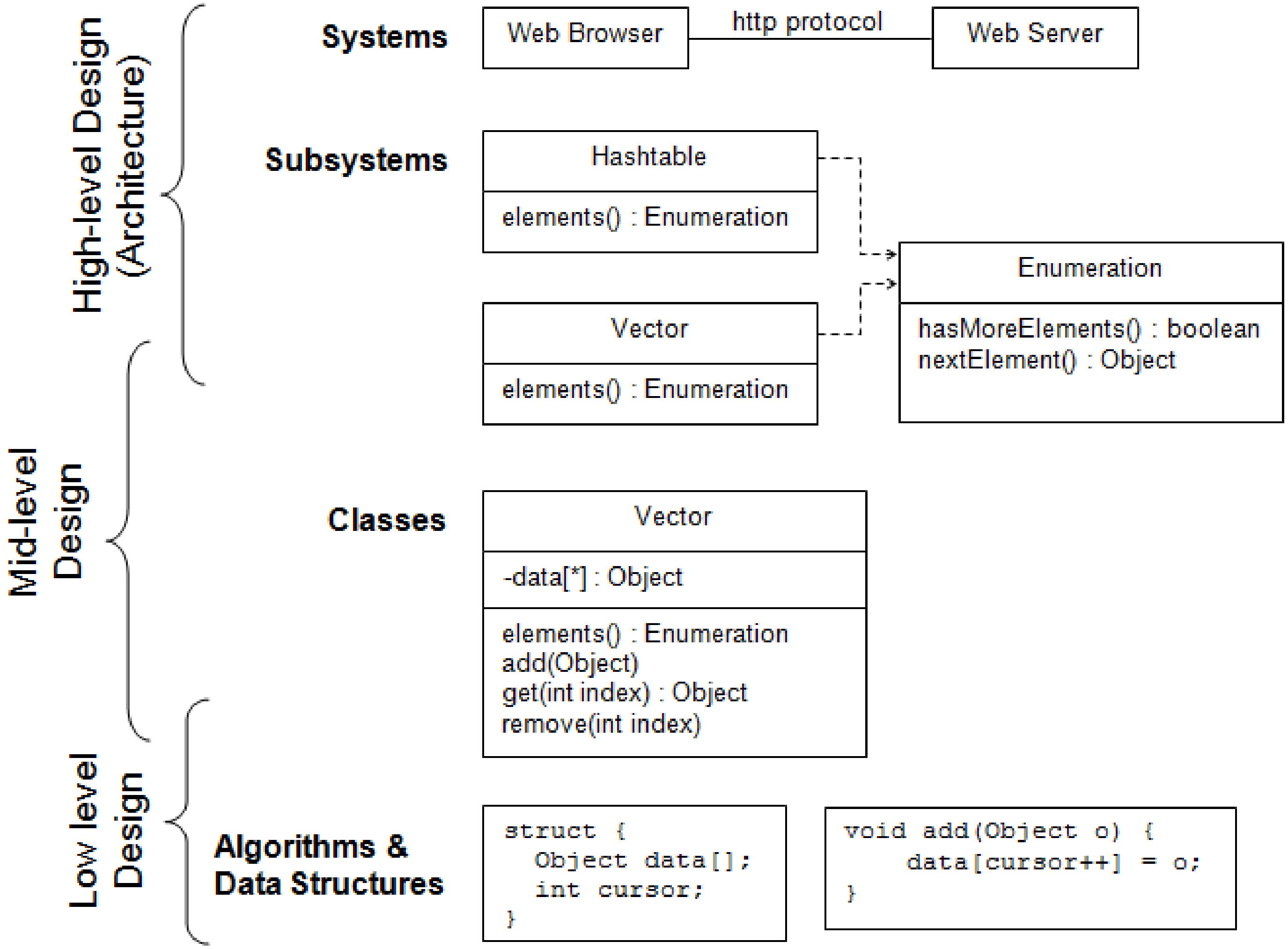


– attributes, private methods, inner classes . . .



– source code implementing methods

Design Occurs at Different Levels

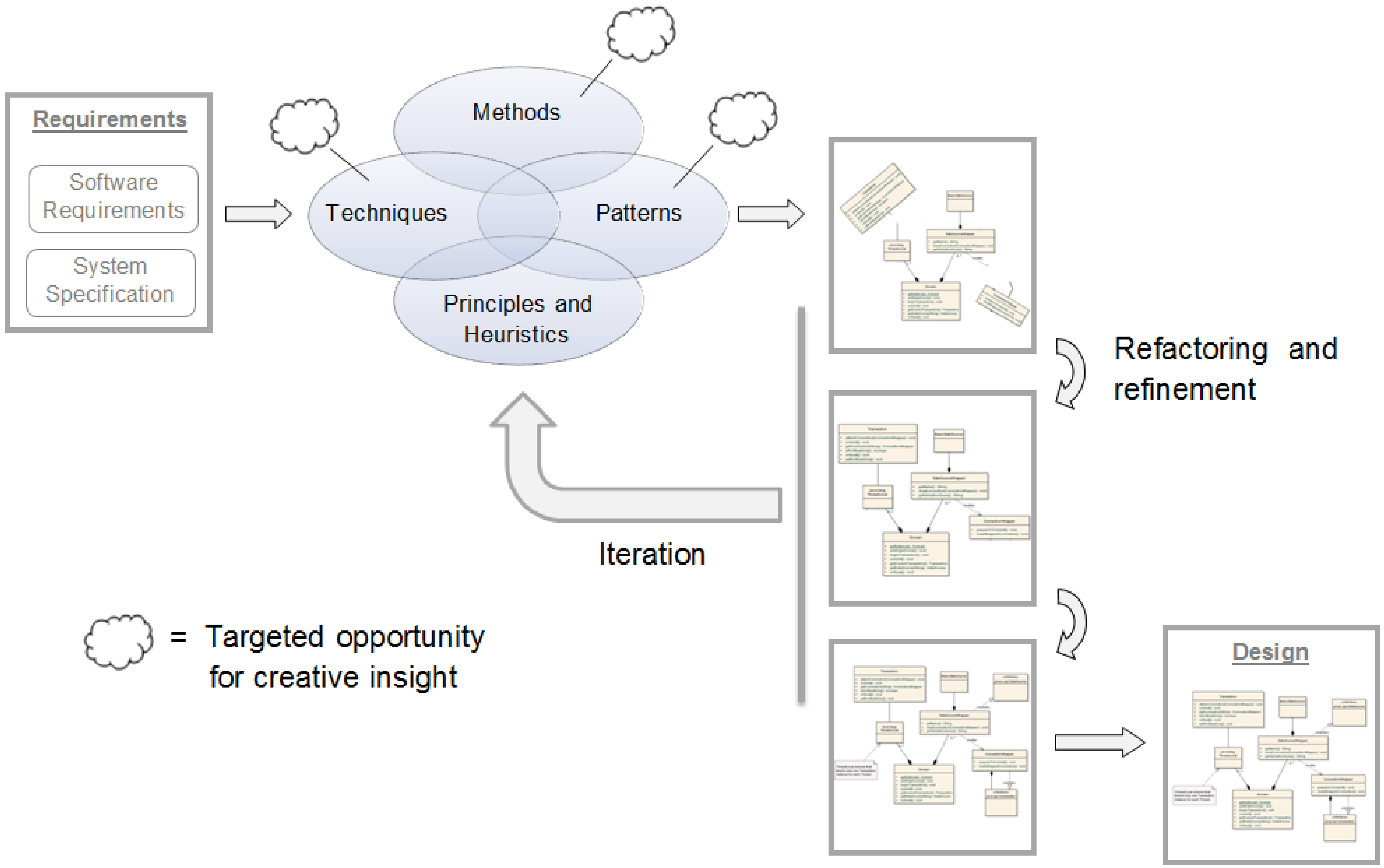


```
struct {
  Object data[];
  int cursor;
}
```

```
void add(Object o) {
    data[cursor++] = o;
}
```

Importance of Software Design

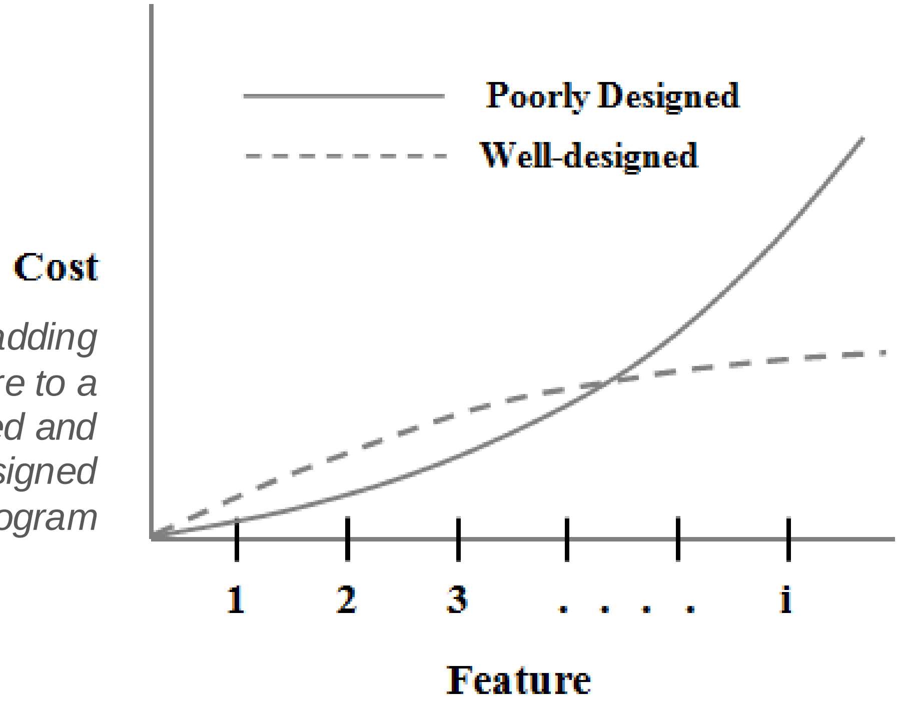
- The design process can be made more systematic and predictable through the application of methods, techniques and patterns, all applied according to principles and heuristics.



Requirements
Software Requirements
System Specification
Methods
Techniques
Patterns
Principles and Heuristics
Iteration
Refactoring and refinement
Design
= Targeted opportunity for creative insight

Importance of Managing Complexity

- Poorly designed programs are difficult to understand and modify.
- The larger the program, the more pronounced are the consequences of poor design.



Cost of adding the i^{th} feature to a well-designed and poorly designed program

Two Types of Complexity in Software

- **Essential complexities**
  - complexities that are inherent in the problem.
- **Accidental/incidental complexities**
  - complexities that are artifacts of the solution.

- The total amount of complexity in a software solution is:
  - Essential Complexities + Accidental complexities

- The primary purpose of design is to control complexity
  - Goal: manage essential complexity while avoiding the introduction of additional accidental complexities

Dealing with Software Complexity

- Modularity – subdivide the solution into smaller easier to manage components. (divide and conquer)
- Abstraction – use abstractions to suppress details in places where they are unnecessary.
- Information Hiding – hide details and complexity behind simple interfaces
- Inheritance – general components may be reused to define more specific elements.
- Composition – reuse of other components to build a new solution

Design is a wicked problem

- A wicked problem is one that can only be clearly defined by solving it.

> *"TEX would have been a complete failure if I had merely specified it and not participated fully in its initial implementation. The process of implementation constantly led me to unanticipated questions and to new insights about how the original specifications could be improved."*
>
> *Donald Knuth*

Characteristics of Software Design

- **Non-deterministic**
  - No two designers or design processes are likely to produce the same output.
- **Heuristic**
  - Design techniques tend to rely on heuristics and rules-of-thumb rather than repeatable processes.
- **Emergent**
  - The final design evolves from experience and feedback. Design is an iterative and incremental process where a complex system arises out of relatively simple interactions.

A Generic Design Process

- Understand the problem (software requirements).
- Construct a "black-box" model of solution (system specification).
  - System specifications are typically represented with use cases (especially when doing OOD).
- Look for existing solutions (e.g., architecture and design patterns) that cover some or all of the software design problems identified.
- Consider building prototypes
- Document and review design
- Iterate over solution (Refactor)
  - Evolve the design until it meets functional requirements and maximizes non-functional requirements

Inputs to the design process

- User requirements and system specification
  - including any constraints on design and implementation options

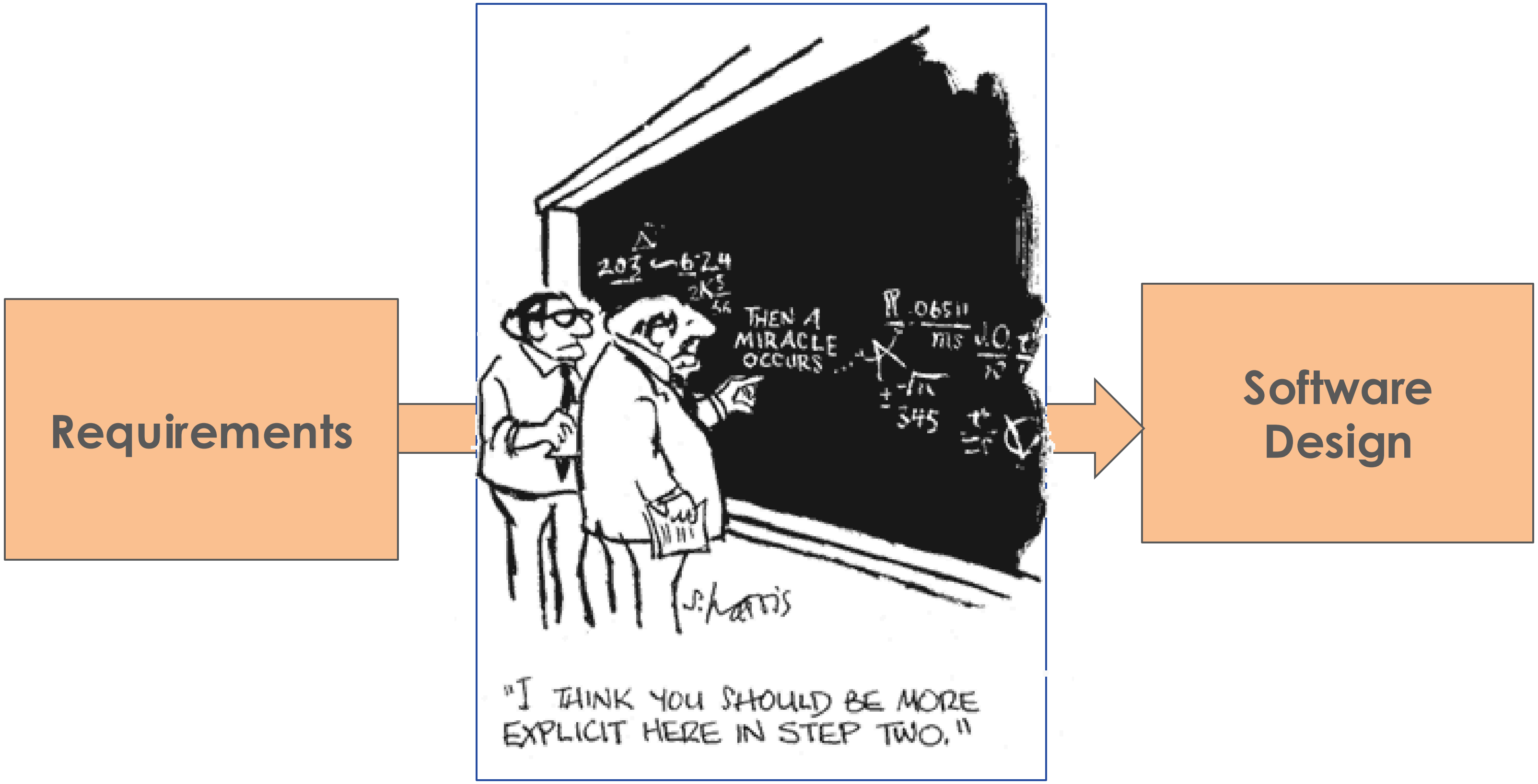
- Domain knowledge
  - For example, if it’s a healthcare application the designer will need some knowledge of healthcare terms and concepts.

- Implementation knowledge
  - capabilities and limitations of eventual execution environment

Desirable Internal Design Characteristics

- **Minimal complexity** – Keep it simple. Maybe you don't need high levels of generality.
- **Loose coupling** – minimize dependencies between modules
- **Ease of maintenance** – Your code will be read more often then it is written.
- **Extensibility** – Design for today but with an eye toward the future. Note, this characteristic can be in conflict with "minimize complexity". Engineering is about balancing conflicting objectives.
- **Reusability** – reuse is a hallmark of a mature engineering discipline
- **Portability** – works or can easily be made to work in other environments
- **High fan-in** on a few utility-type modules and low-to-medium fan-out on all modules. High fan-out is typically associated with high complexity.
- **Leanness** – when in doubt, leave it out. The cost of adding another line of code is much more than the few minutes it takes to type.
- **Stratification** – Layered. Even if the whole system doesn't follow the layered architecture style, individual components can.
- **Standard techniques** – sometimes it's good to be a conformist! Boring is good. Production code is not the place to try out experimental techniques.

Software Design methods



Requirements

Software Design

THEN A MIRACLE OCCURS . . .
S. Harris
"I THINK YOU SHOULD BE MORE EXPLICIT HERE IN STEP TWO."

Design Methods

- Design methods provide a procedural description for obtaining a design solution
- Most methods include:
  - A representation part or notation for representing problem and intermediate forms of the design solution (usually from different view points). Examples: UML, pseudocode.
  - Process part or procedures to following in developing the solution
  - Heuristics – guidelines and best practices for making decisions and assessing intermediate and final results. Remember, design isn't deterministic.

Design – Representational Forms

- Class diagrams for static structure
- Sequence diagrams for dynamic behavior
- Textual and visual form of use cases are used to create and validate analysis and design representational forms
- Other UML models are also useful for understanding the problem and conceptualizing a solution (state machine diagram, activity diagram, etc.)

Methods and Patterns

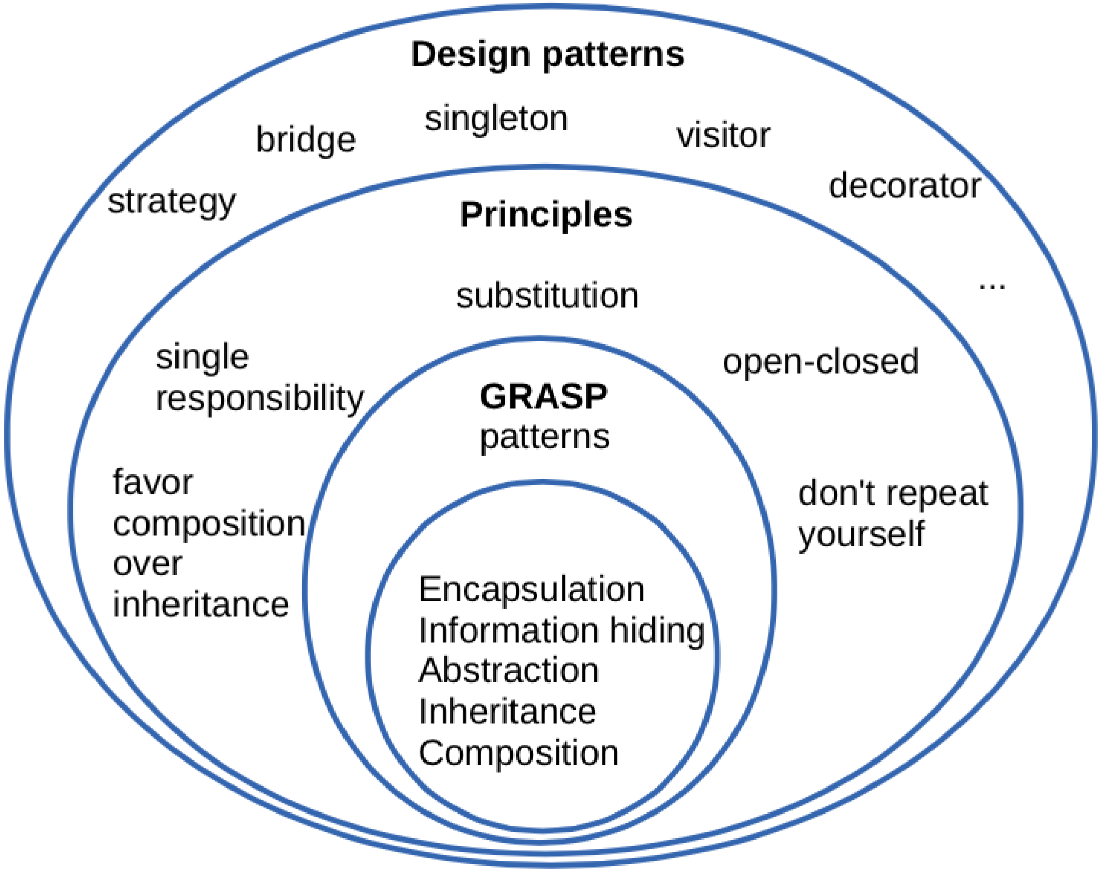
- Methods and patterns are the principle techniques for dealing with the challenges of design
- They are useful for:
  - Creating a design
  - Documenting and communicating a design
  - Transferring design knowledge and experience between practitioners

Patterns

- A design pattern is a reusable solution to a commonly occurring design problem
- Design patterns are adapted for the unique characteristics of the problem
- Just as there are levels of design, there are levels of design patterns:
  - Architecture Styles/Patterns
  - Design Patterns
  - Programming Idioms

What next? O-O Software Design

- There’s no a methodology to get the best object-oriented design, but there are principles, patterns, heuristics.


Design patterns
bridge
singleton
visitor
strategy
decorator
...
Principles
substitution
single responsibility
open-closed
favor composition over inheritance
don't repeat yourself
GRASP patterns
Encapsulation
Information hiding
Abstraction
Inheritance
Composition