中国科学技术大学
University of Science and Technology of China

Software Architecture

SSE USTC　Qing Ding
dingqing@ustc.edu.cn
http://staff.ustc.edu.cn/~dingqing

Architecture and Architect

Learning Objectives

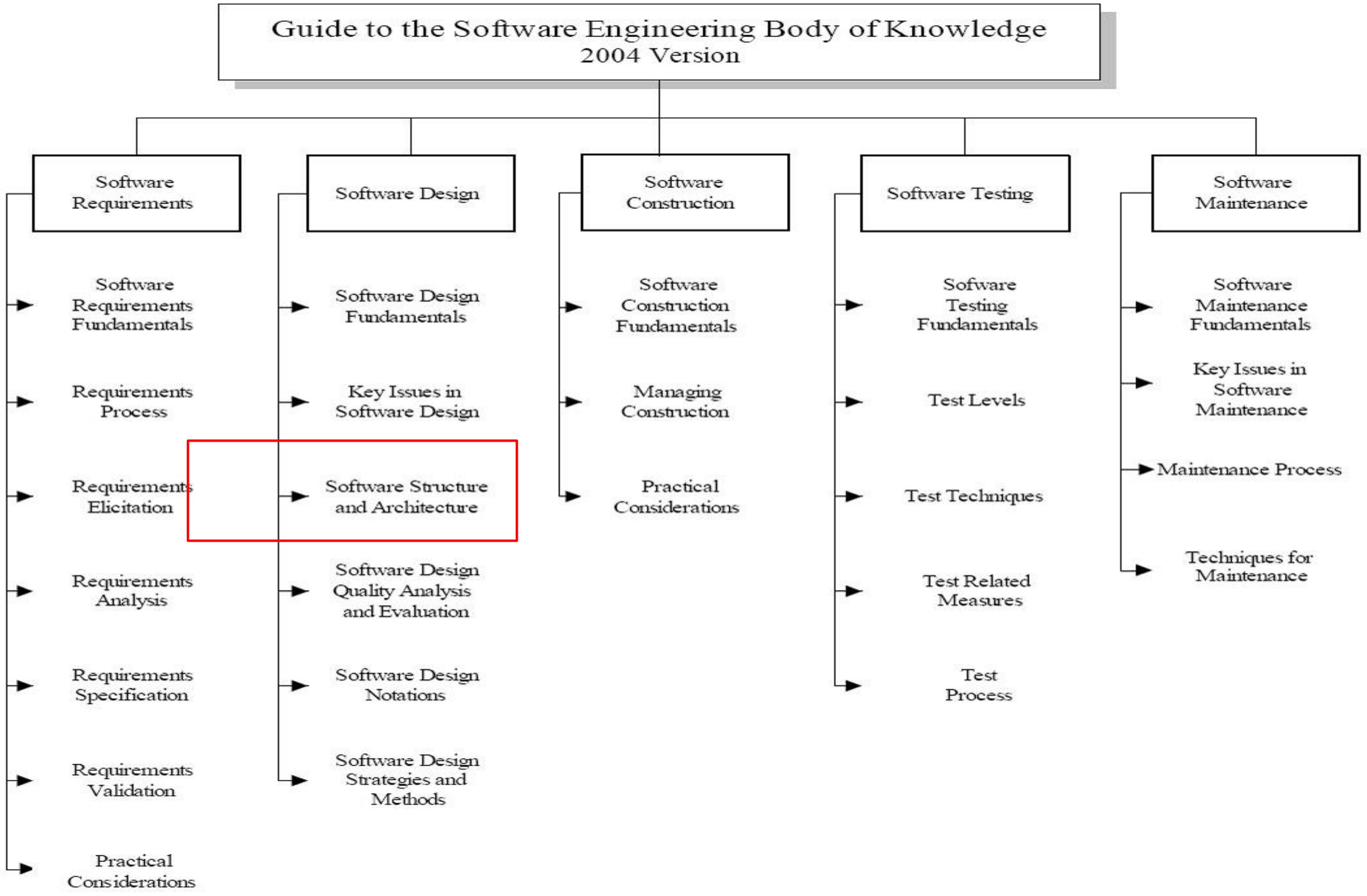
中国科学技术大学
University of Science and Technology of China

- Define software architecture and its elements
- Distinguish between accidental 偶然的 and essential 必然的 difficulties in software and enumerate them
- Compare and contrast 对比 software architecture to building architecture and understand the role of the architect
- Identify 确定，辨别 essential elements of the architecture of the WWW and the UNIX pipes and filters paradigm 范例
- Understand software product lines and the role of reuse in software design

SWEBOK

中国科学技术大学 University of Science and Technology of China

- Guide to the Software Engineering Body of Knowledge
  - A project of the IEEE Computer Society Professional Practices Committee

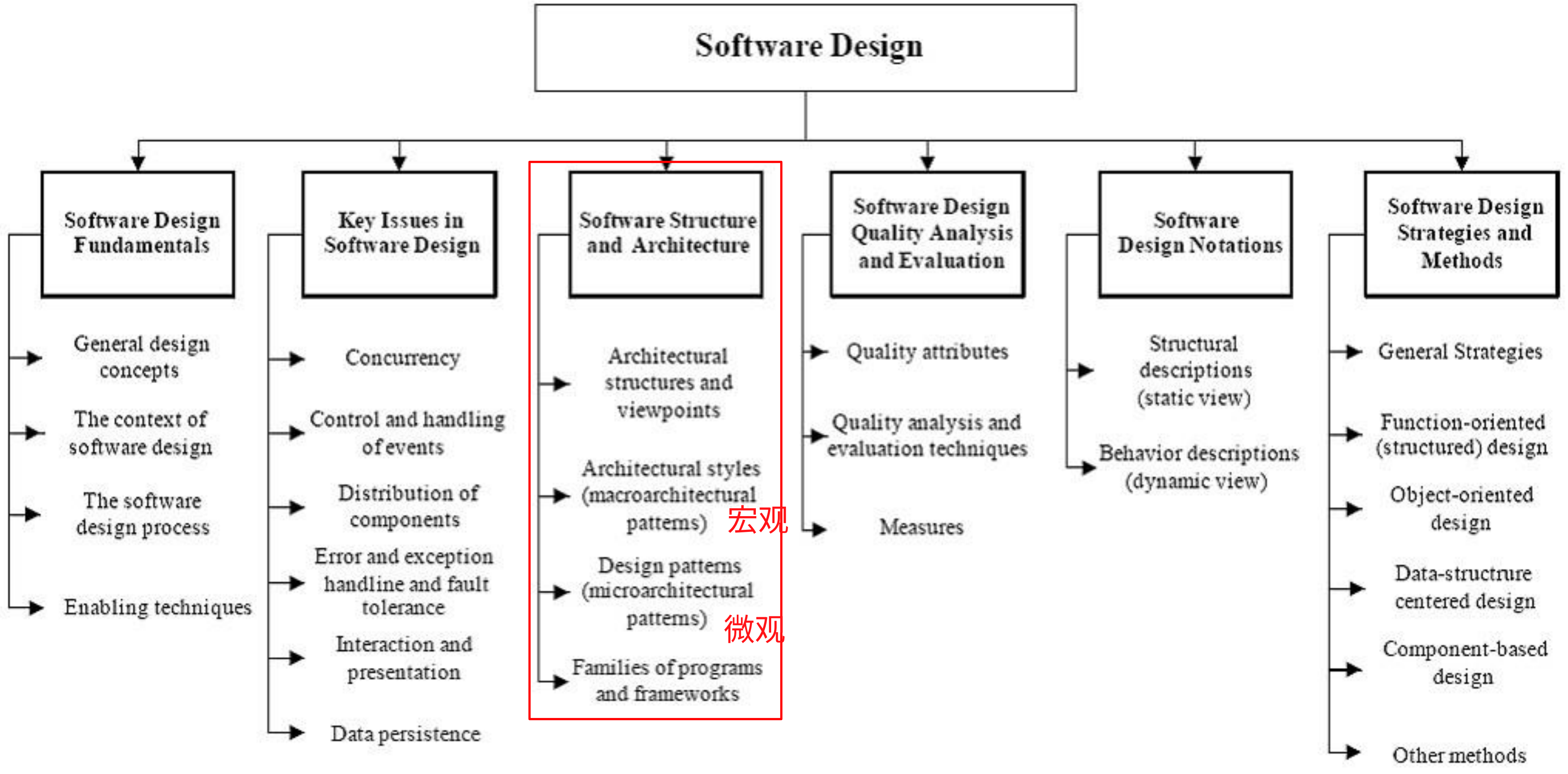


Guide to the Software Engineering Body of Knowledge 2004 Version
Software Requirements
Software Design
Software Construction
Software Testing
Software Maintenance
Software Requirements Fundamentals
Requirements Process
Requirements Elicitation
Requirements Analysis
Requirements Specification
Requirements Validation
Practical Considerations
Software Design Fundamentals
Key Issues in Software Design
Software Structure and Architecture
Software Design Quality Analysis and Evaluation
Software Design Notations
Software Design Strategies and Methods
Software Construction Fundamentals
Managing Construction
Practical Considerations
Sofware Testing Fundamentals
Test Levels
Test Techniques
Test Related Measures
Test Process
Software Maintenance Fundamentals
Key Issues in Software Maintenance
Maintenance Process
Techniques for Maintenance

SWEBOK

University of Science and Technology of China

- Guide to the Software Engineering Body of Knowledge
  - A project of the IEEE Computer Society Professional Practices Committee



Software Design
Software Design Fundamentals
General design concepts
The context of software design
The software design process
Enabling techniques
Key Issues in Software Design
Concurrency
Control and handling of events
Distribution of components
Error and exception handline and fault tolerance
Interaction and presentation
Data persistence
Software Structure and Architecture
Architectural structures and viewpoints
Architectural styles (macroarchitectural patterns)
宏观
Design patterns (microarchitectural patterns)
微观
Families of programs and frameworks
Software Design Quality Analysis and Evaluation
Quality attributes
Quality analysis and evaluation techniques
Measures
Software Design Notations
Structural descriptions (static view)
Behavior descriptions (dynamic view)
Software Design Strategies and Methods
General Strategies
Function-oriented (structured) design
Object-oriented design
Data-structrure centered design
Component-based design
Other methods

Architecture

What is architecture?

What is Software Architecture?

- It's all about software design
  - Architecture is software design, but not all design is software architecture
    - part of the design process

问题

- [Gorton] Architecture focuses on 'issues that will be difficult/impossible to change once the system is built'
  - E.g., quality attributes like security, performance
  - E.g., non-functional requirements like cost, deployment hardware

Definitions – Taylor et al.

- *[Software architecture is] the set of principal design decisions governing a system*

Definitions - ANSI/IEEE Std 1471-2000

- *"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**."*

Definitions - SEI

中国科学技术大学
University of Science and Technology of China

- *"The software architecture of a program or computing system is the **structure or structures** of the system, which comprise **software elements**, the **externally visible properties** of those elements, and the **relationships** among them."*

包含

外部

内容

What is Software Architecture?

中国科学技术大学
University of Science and Technology of China

- There are two common elements:
  - One is the highest-level breakdown of a system into its parts;
    分解
    自顶向下
  - the other, decisions that are hard to change.

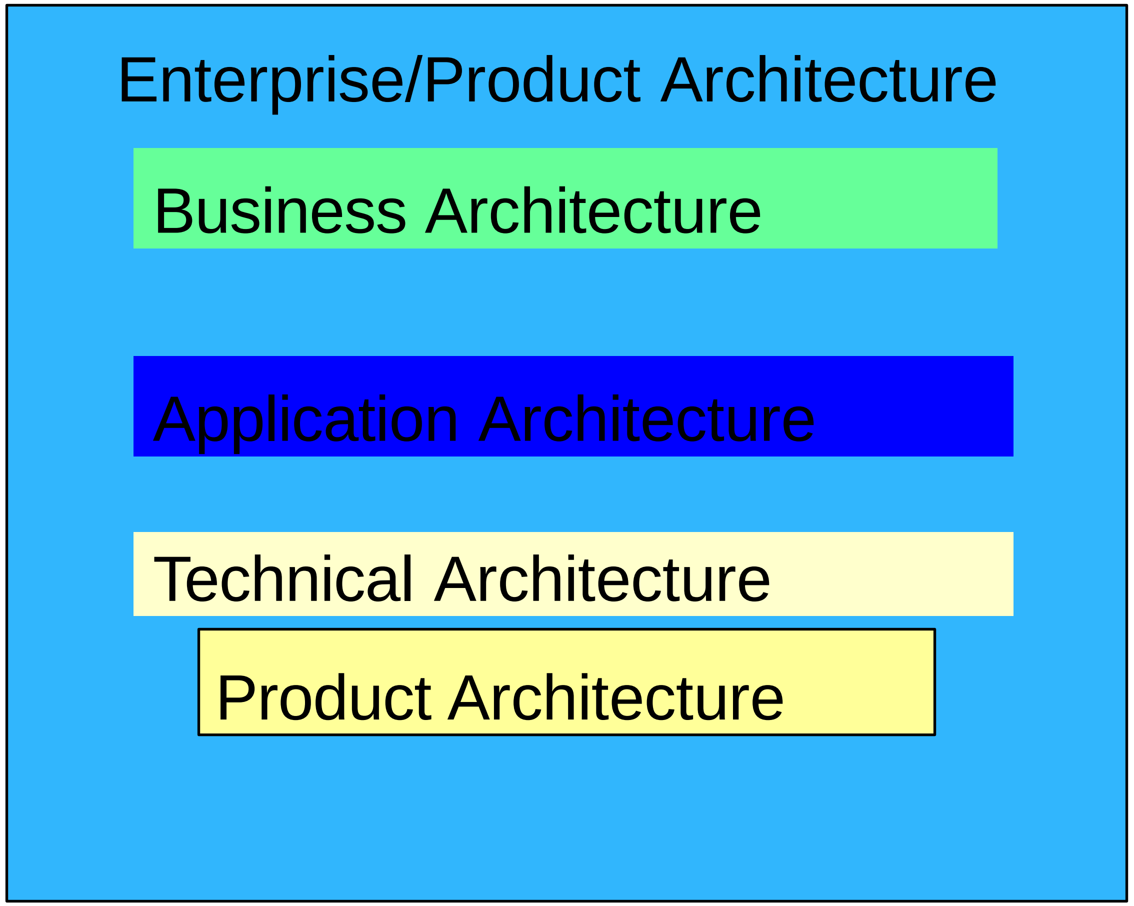
- It's also increasingly realized that there isn't just one way to state a system's architecture;
  - rather, there are multiple architectures in a system, and the view of what is architecturally significant is one that can change over a system's lifetime.

Main point: Architecture Defines Structure

- 分解 组件
  Decomposition of system into components/modules/subsystems
- Architecture defines:
  - Component interfaces
    - What a component can do
  - Component communications and dependencies
    - How components communicate
  - Component responsibilities
    - Precisely what a component will do

Typical System Architecture

中国科学技术大学
University of Science and Technology of China



What is Software Architecture

中国科学技术大学
University of Science and Technology of China

- Software Architecture
  - transforms Business Architecture into an set of designs and guidelines to realize business process in an information systems

- Software Architecture
  - relies on Technical Architecture to provide an efficient/scalable/secure environment

可扩展的

Software Engineering Difficulties

中国科学技术大学
University of Science and Technology of China

- Software engineers deal with unique set of problems
  - Young field with tremendous expectations
  - Building of vastly complex, but intangible systems
  - Often software is not useful on its own
    - (e.g., unlike a car)
    - thus it must conform to changes in other engineering areas
- Some problems can be eliminated
  - “accidental difficulties”
- Other problems can be lessened 减弱, but not eliminated
  - “essential difficulties”

Accidental Difficulties

- Solutions exist
  - Possibly waiting to be discovered

- Past productivity increases were the result of overcoming
  - 不充分的 Inadequate programming constructs & abstractions
    - 弥补 Remedied by high-level programming languages
    - Increased productivity by factor of five
    - Complexity was never inherent in program at all

Accidental Difficulties (cont'd)

- Past productivity increases were the result of overcoming （cont’d)
  - Inadequate tools:
    - E.g., Viewing results of programming decisions took long time
      - Remedied by time–sharing
      - 转变
        Turnaround time approaching limit of human perception
  - 多样化的
    Difficulty of using heterogeneous programs
    - Addressed by integrated software development environments
    - Support task that was conceptually always possible

Essential Difficulties

- Only partial solutions exist for them, if any
- Cannot be abstracted away
  - Complexity
  - Conformity
  - Changeability
  - Intangibility

Complexity

- No two software parts are alike
  - If they are, they are abstracted away into one
- 超线性 Complexity grows super-linearly with size
  - E.g., it is impossible to enumerate all states of program
    - Except perhaps “toy” programs

Conformity

- Software is required to conform to its
  - Operating environment
  - Hardware
- Often “last kid on block”
- Perceived as most conformable

Changeability

有延展性的

- Software is viewed as infinitely malleable
- Change originates with
  - New applications, users, machines, standards, laws
  - Hardware problems

Intangibility

- Software is not embedded in space
  - Often no constraining physical laws
- No obvious representation
  - E.g., familiar geometric shapes

Promising Attacks On Complexity

- Buy vs. Build
- Requirements refinement & rapid prototyping
  - Hardest part is deciding what to build (or buy?)
  - Must show product to customer to get complete spec.
  - Need for iterative feedback

Promising Attacks On Complexity (cont'd)

- Incremental/Evolutionary/Spiral Development
  - Grow systems, don't build them
  - Good for morale 士气
  - Easy backtracking
  - Early prototypes
- Great designers
  - Good design can be taught; great design cannot
  - Nurture great designers

Primacy of Design

中国科学技术大学
University of Science and Technology of China

- Software engineers collect requirements, code, test, integrate, configure, etc.
- An architecture-centric approach to software engineering places an emphasis on design
  - Design pervades the engineering activity from the very beginning
- 着手处理
  But how do we go about the task of architectural design?

Analogy: Architecture of Buildings

- We all live in them
- (We think) We know how they are built
  - Requirements
  - Design (blueprints)
  - Construction
  - Use
- This is similar (though not identical) to how we build software

完全相同的

Some Obvious Parallels

- Satisfaction of customers’ needs
- Specialization of labor
- Multiple perspectives of the final product
- Intermediate points where plans and progress are reviewed

Deeper Parallels

- Architecture is different from, but linked with the product/structure
- 内容 Properties of structures are induced by the design of the architecture
- The architect has a distinctive role

Deeper Parallels (cont’d)

中国科学技术大学
University of Science and Technology of China

- Process is not as important as architecture
  - Design and resulting qualities are at the forefront
  - Process is a means, not an end
- Architecture has matured over time into a discipline 纪律
  - Architectural styles as sets of constraints
  - Styles also as wide range of solutions, techniques and palettes of compatible materials, colors, and sizes 调色板

More about the Architect

- A distinctive role in a project
- Very broad training
- Amasses (累积) and leverages extensive experience
- A keen (敏锐的) sense of aesthetics (美学)
- Deep understanding of the domain
  - Properties of structures, materials, and environments
  - Needs of customers

More about the Architect (cont'd)

中国科学技术大学
University of Science and Technology of China

一流的

- Even first-rate programming skills are insufficient for the creation of complex software applications
    - But are they even necessary?

Limitations of the Analogy…

- We know a lot about buildings, much less about software
- The nature of software is different from that of building architecture
- Software is much more malleable than physical materials
- The two “construction industries” are very different
- Software deployment has no counterpart in building architecture
- Software is a machine; a building is not

…But Still Very Real Power of Architecture

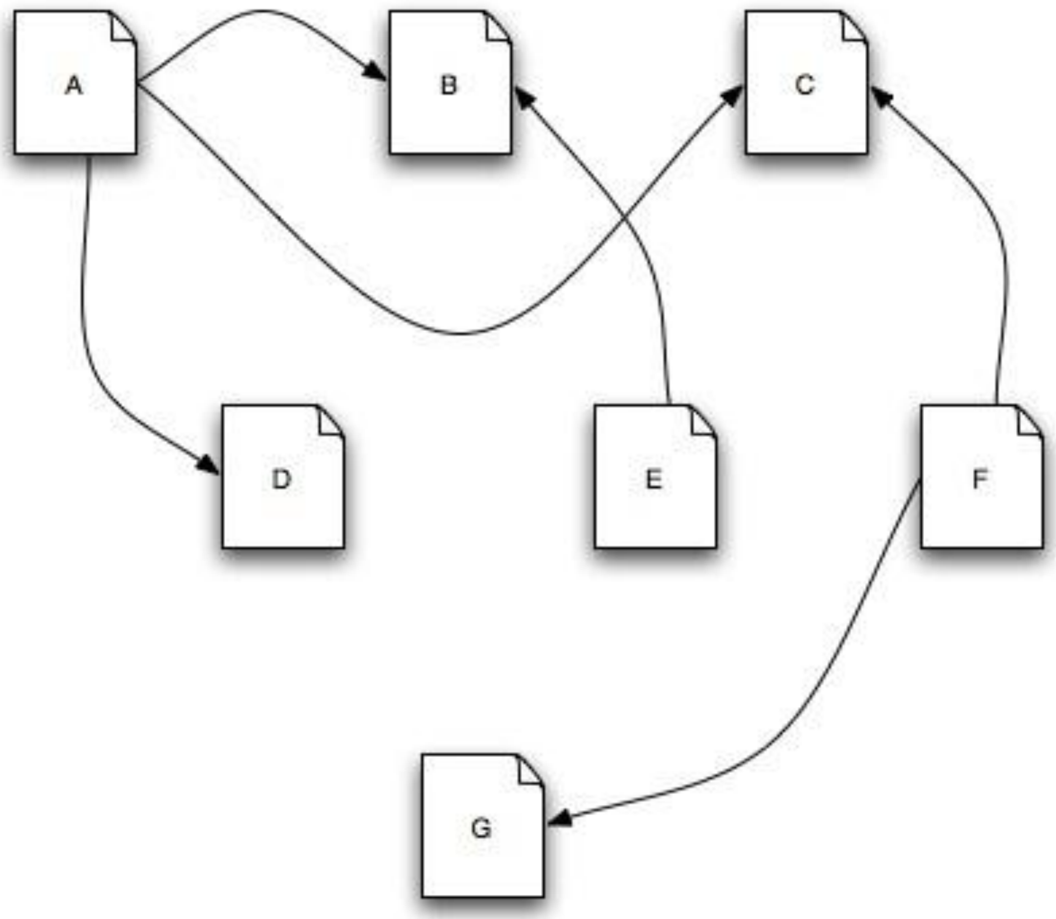
卓越 潜力

- Giving preeminence to architecture offers the potential for
  - Intellectual control
  - Conceptual integrity
  - Effective basis for knowledge reuse
  - Realizing experience, designs, and code
  - Effective project communication
  - Management of a set of variant systems
- Limited-term focus on architecture will not yield significant benefits!

Architecture in Action: WWW

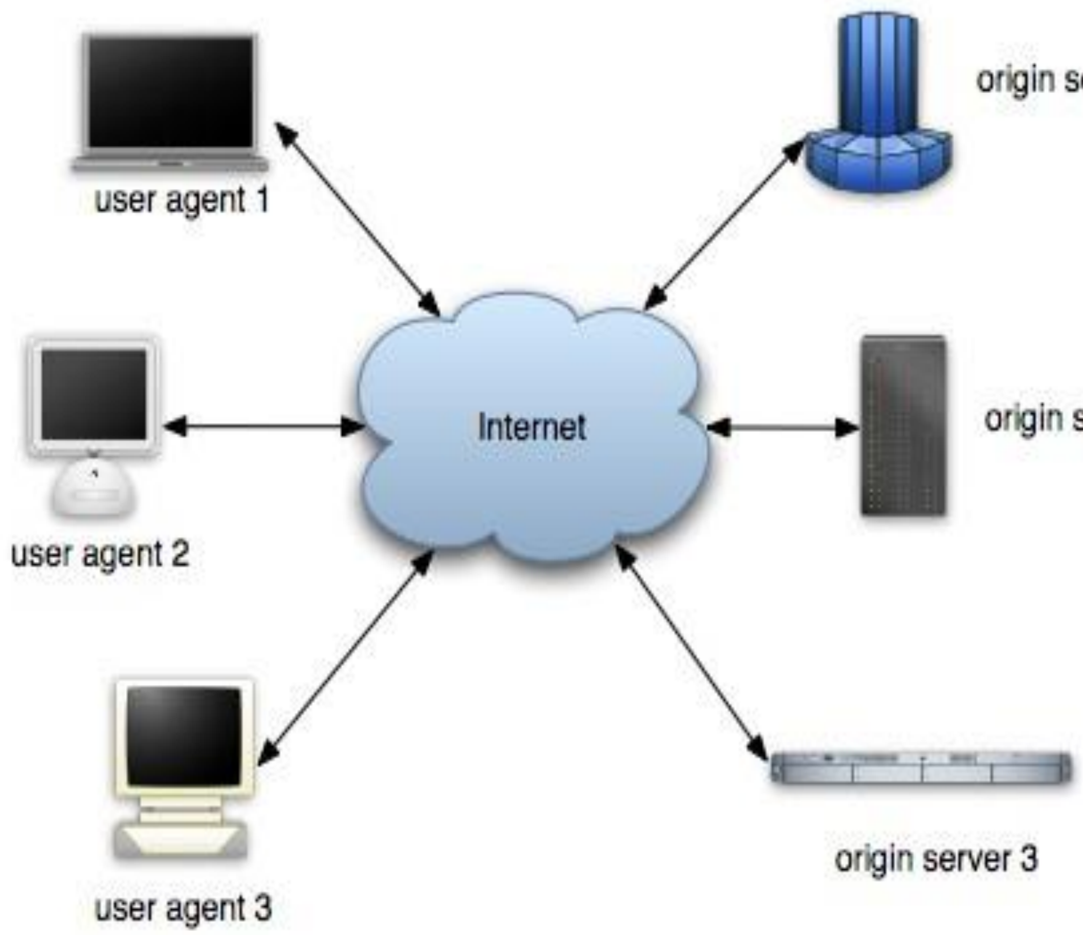
中国科学技术大学
University of Science and Technology of China

- This is the Web



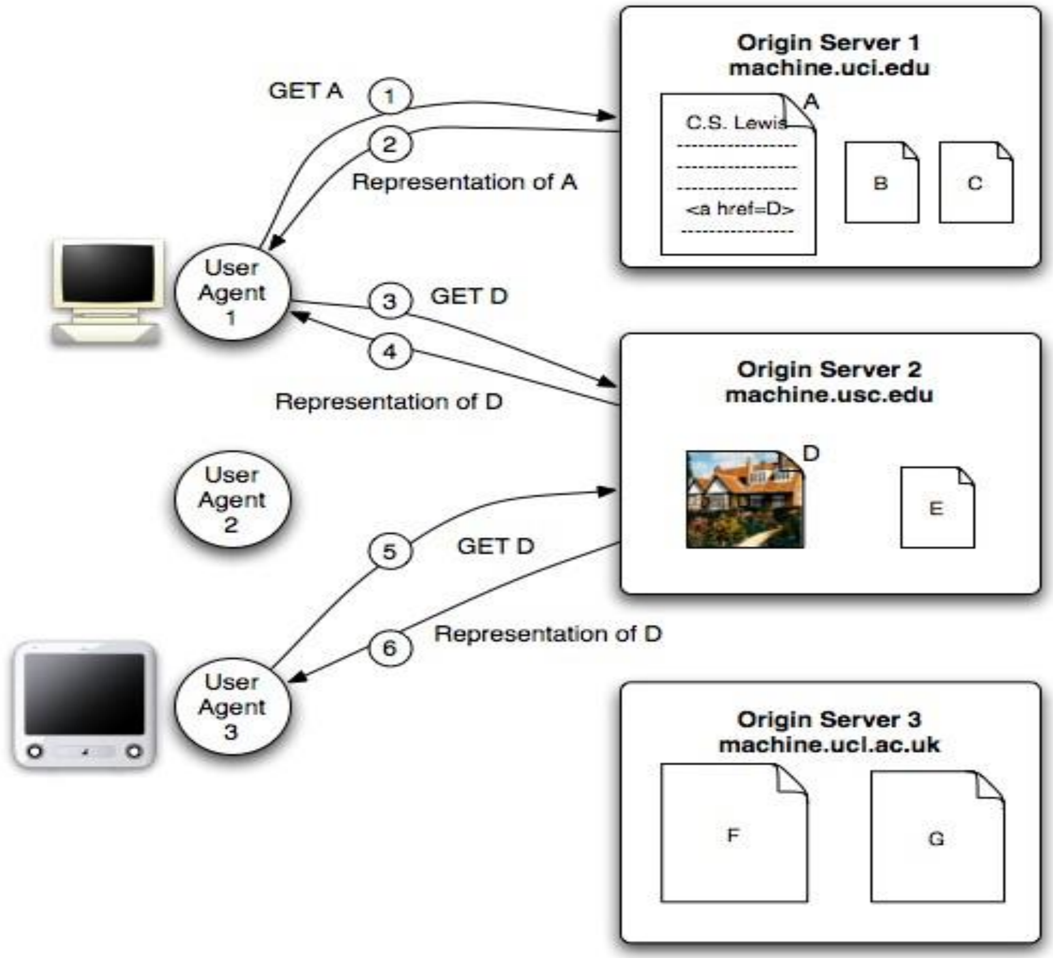
Architecture in Action: WWW

- So is this



Architecture in Action: WWW

- And this



WWW in a (Big) Nutshell

中国科学技术大学
University of Science and Technology of China

- The Web is a collection of resources, each of which has a unique name known as a “uniform resource
locator” (URL)
- Each resource denotes, informally, some information.
- URI’s can be used to determine the identity of a machine on the Internet, known as an origin server, where the value of the resource may be ascertained.
- Communication is initiated by clients, known as user agents, who make requests to servers.
  - Web browsers are common instances of user agents.

Web是资源的集合，每个资源都有一个唯一的名称，称为统一资源定位器(URL)
每种资源非正式地表示一些信息。
URI可以用来确定Internet上的一台机器(称为源服务器)的身份，可以确定资源的值。
通信是由客户机(称为用户代理)发起的，它向服务器发出请求。
Web浏览器是用户代理的常见实例。

WWW in a (Big) Nutshell (cont'd)

中国科学技术大学 University of Science and Technology of China

- Resources can be manipulated through their representations.
  - HTML is a very common representation language used on the Web.
- All communication between user agents and origin servers must be performed by a simple, generic protocol (HTTP), which offers the command methods GET, POST, etc.
- Communication between user agents and origin servers is fully self-contained. 独立的
  - (So-called "stateless interactions")

资源可以通过它们的表示来操纵。HTML是一种非常常用的Web表示语言。用户代理和源服务器之间的所有通信都必须通过一个简单的通用协议(HTTP)执行，该协议提供了GET、POST等命令方法。用户代理和源服务器之间的通信是完全自包含的。(所谓的无状态交互)

WWW's Architecture

- Architecture of the Web is wholly separate from the code
- There is no one piece of code that implements the architecture.
- There are multiple pieces of code that implement the various components of the architecture.
  - E.g., different Web browsers

WWW’s Architecture (cont’d)

- Stylistic constraints of the Web’s architectural style are not apparent in the code
  - The effects of the constraints are evident in the Web
- One of the world’s most successful applications is only understood adequately from an architectural vantage point.

Architecture in Action (2): @ small scale

- Unix command line

```
ls invoices |  grep -e august  | sort
```

- Application architecture can be understood based on very few rules 应用程序架构可以通过很少的规则来理解
- Applications can be composed by non-programmers
  - Akin to Lego blocks
- A simple architectural concept that can be comprehended and applied by a broad audience

Software Product Line

DSSA 领域特定的架构　重要概念：可重用

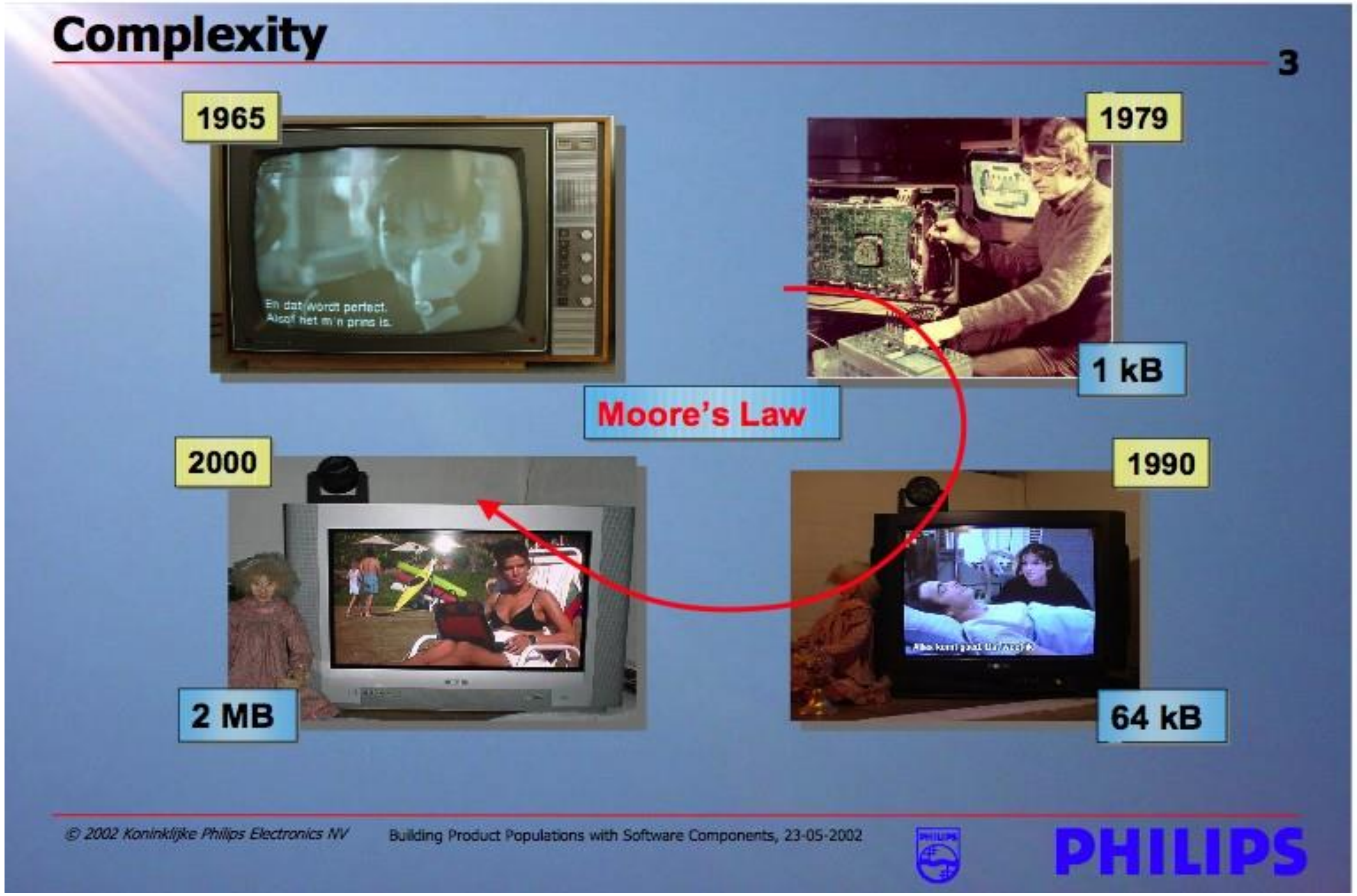
- Motivating example
  - A consumer is interested in a 35-inch HDTV with a built-in DVD player for the North American market.

Such a device might contain upwards of a million lines of embedded software.

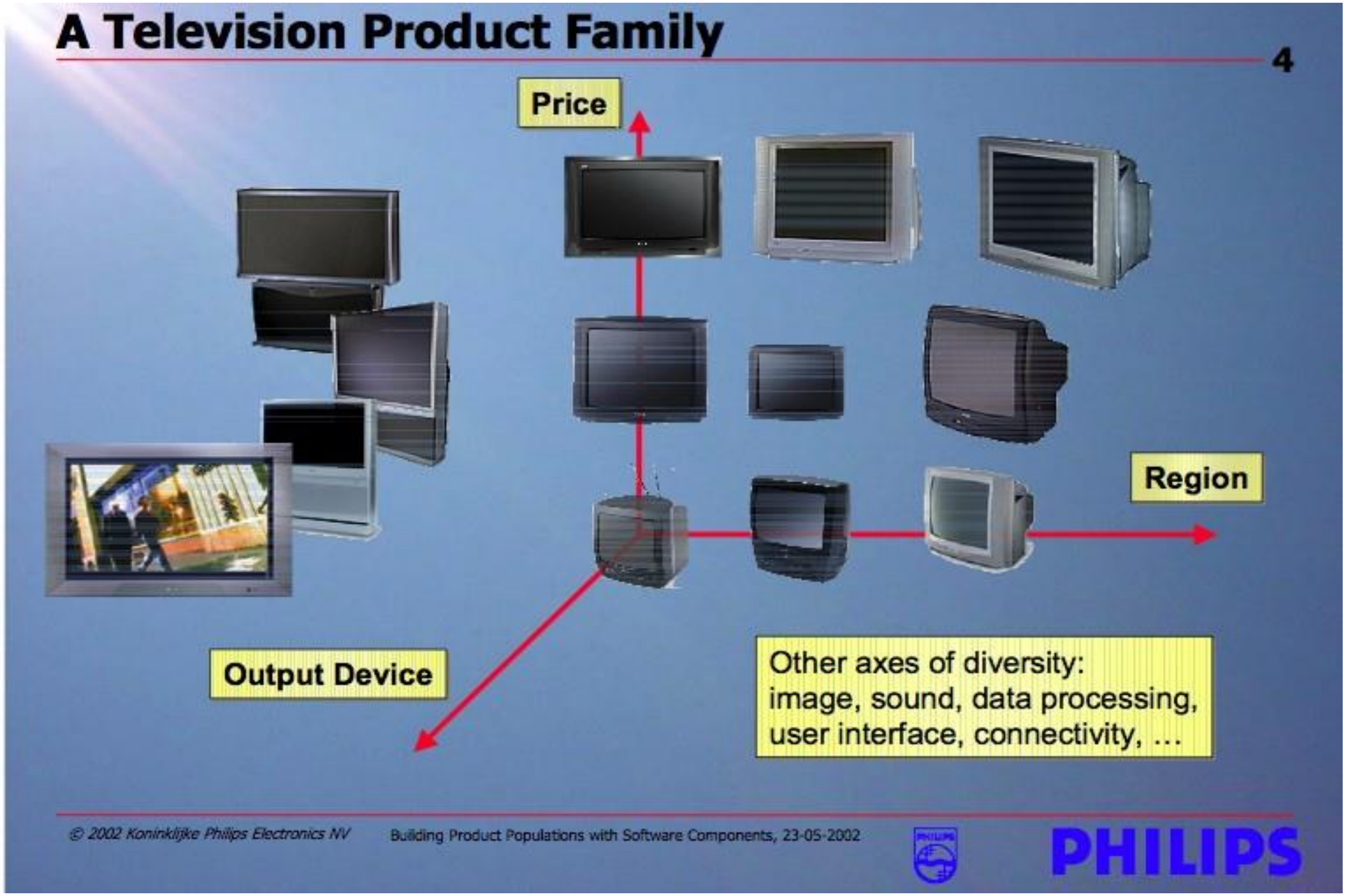
This particular television/DVD player will be very similar to a 35-inch HDTV without the DVD player, and also to a 35-inch HDTV with a built-in DVD player for the European market, where the TV must be able to handle PAL or SECAM encoded broadcasts, rather than North America’s NTSC format.

These closely related televisions will similarly each have a million or more lines of code embedded within them.

Growing Sophistication of Consumer Devices


Complexity
3
1965
1979
1 kB
Moore's Law
2000
1990
2 MB
64 kB
© 2002 Koninklijke Philips Electronics NV
Building Product Populations with Software Components, 23-05-2002
PHILIPS

Families of Related Products



A Television Product Family

4

Other axes of diversity:
image, sound, data processing,
user interface, connectivity, …

The Necessity and Benefit of PLs

- Building each of these TVs from scratch would likely put Philips out of business

- Reusing structure, behaviors, and component implementations is increasingly important to successful business practice
  - It simplifies the software development task
  - It reduces the development time and cost
  - it improves the overall system reliability

- Recognizing and exploiting commonality and variability across products

Reuse as the Big Win

- Architecture: reuse of
  - Ideas
  - Knowledge
  - Patterns
  - Engineering guidance
  - Well-worn 平凡的 experience
- Product families: reuse of
  - Structure
  - Behaviors
  - Implementations
  - Test suites…

Architect

What is an architect?

What is Software Architect

中国科学技术大学
University of Science and Technology of China

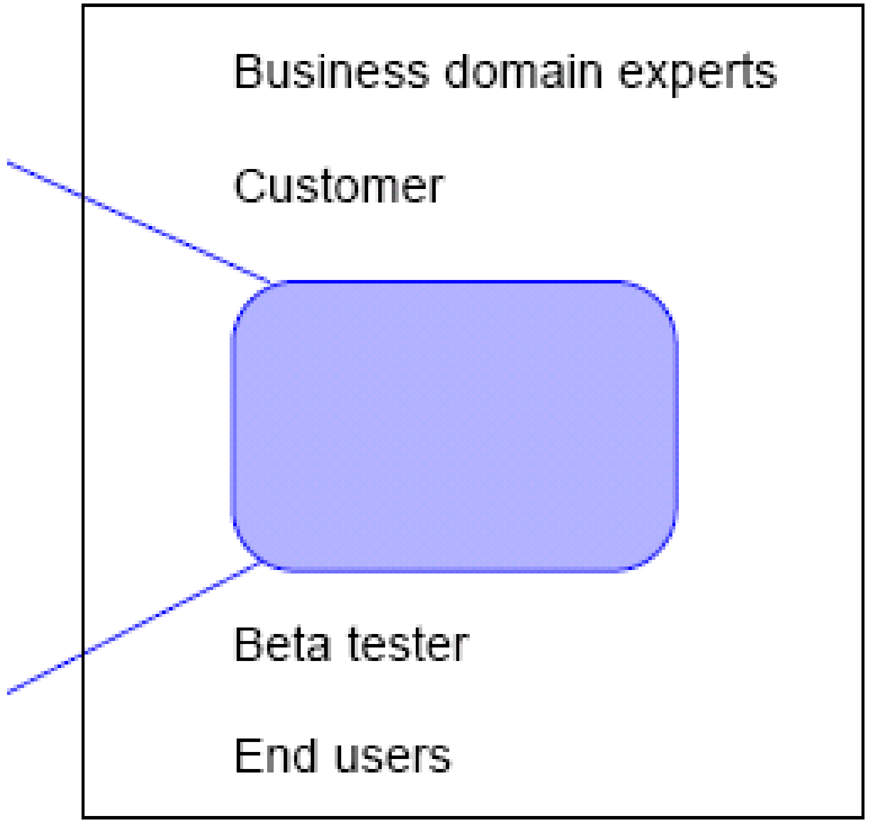
- Circa 25 BCE, Vitruvius described the role of an architect as:
  - *The ideal architect should be a man of letters,* 作家
  - *a mathematician, familiar with historical studies,*
  - *a diligent of philosophy,*
  - *acquainted with music,*
  - *not ignorant of medicine,*
  - *learned in the responses of Juri sconsultis,*
  - *familiar with astronomy and astronomical calculations*

The Architect’s Role

- The architect:
  - Visualizes the behavior of the system
  - Creates the blueprint for the system
  - Defines the way in which the elements of the system work together
  - Distinguishes between functional and nonfunctional system requirements
  - Is responsible for integrating 整合 non-functional requirements into the system

Development Team

- The architect is a member of a development team.
  - Team leads
  - Enterprise modeler
  - Architect
  - System designers
  - Developers
  - Testers
  - QA staff
  - Configuration experts

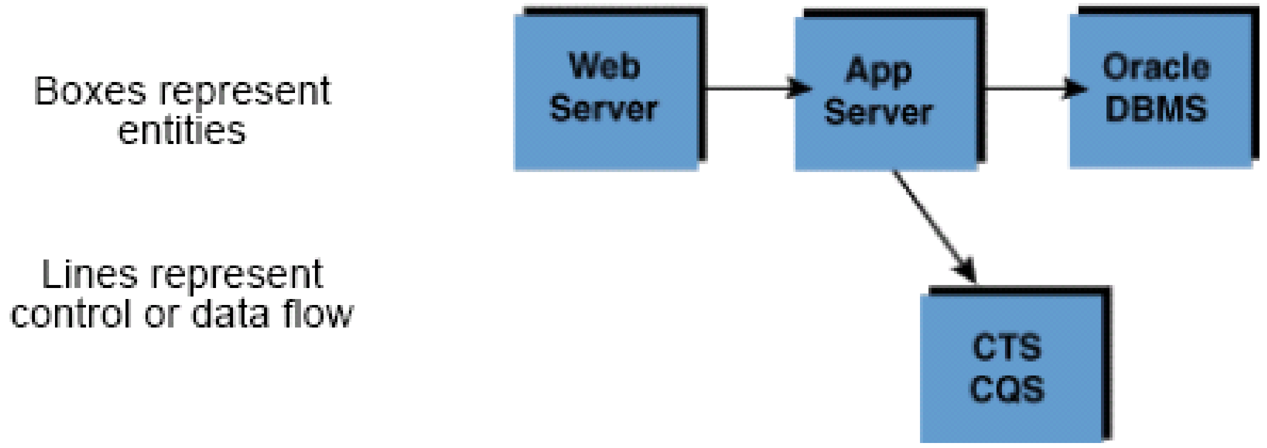


What Architect Do

- Gather System Requirements
- Translate requirement to architecture design
- Implement proof of concept, define development environment
- Develop Architecture Module
- Communicate Architecture Design
- Be the technical expert
- Mentor/help Developers

Defining Architecture

- Architecture refers to an abstract representation of a system's components and behaviors.
- Architecture does not contain details about implementation.
- Architectures are best represented graphically.



Technical Skills

- Requirement gathering/Management
- Modeling and analysis methodology 方法学
- Full Software Development Life Cycle
- Modern architectural technologies, such as Java EE and .NET
- In depth knowledge of programming languages
- Network, Security, hardware platforms
- Database

None Technical Skills

- Facilitation/consulting 引导
- Communication/Presentation/Sales skill
- Mentoring
- Domain Knowledge of area you working on
- Leadership
- Organizational politics
- Business acumen/Strategy 敏锐

What is a the best Architect

- The best architects are
  - good technologists,
  - and command respect in the technical community,
  - 战略家
    but also are good strategists, organizational politicians (in the best sense of the word), consultants and leaders

Things to remember

1. Focus on people, not technology or techniques.

The greatest architecture model in the world has little value if you can't find a way to work with developers effectively and convince them to use it.

2. Keep it simple.

The simpler the architecture, the greater the chance that it will be understood and actually followedby developers.

Things to remember

3. Work iteratively and incrementally.

Your architecture will evolve over time due to new requirements, new technological choices, and greater understanding amongst your enterprise architecture team.

4. Roll up your sleeves.

Developers won't respect you, and therefore won't accept your architecture, if you aren't willing to get actively involved in their project efforts.

Things to remember

5. Build it before you talk about it.

Everything works in diagrams but can fail miserably in practice.
Just like in the RUP, you should prove your application architecture via technical prototyping; there is no reason why you can't do the same at the enterprise level.

6. Look at the whole picture.

This is a primary skill of enterprise architects, which is one of the reasons why a multi-view approach is so important to you

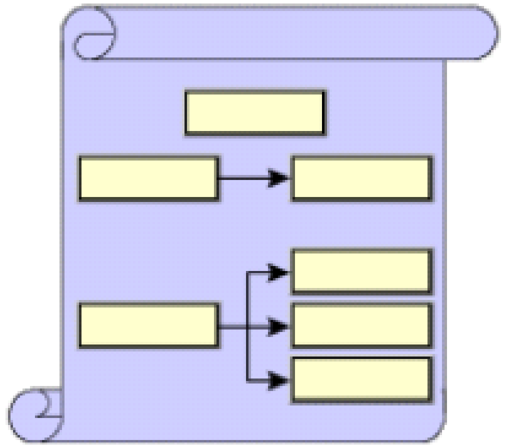
Things to remember

7. Make architecture attractive to your customers.

If your enterprise architecture artifacts aren't easy to understand, to access, and to work with, your customers (developers and senior managers) will very likely ignore your work.

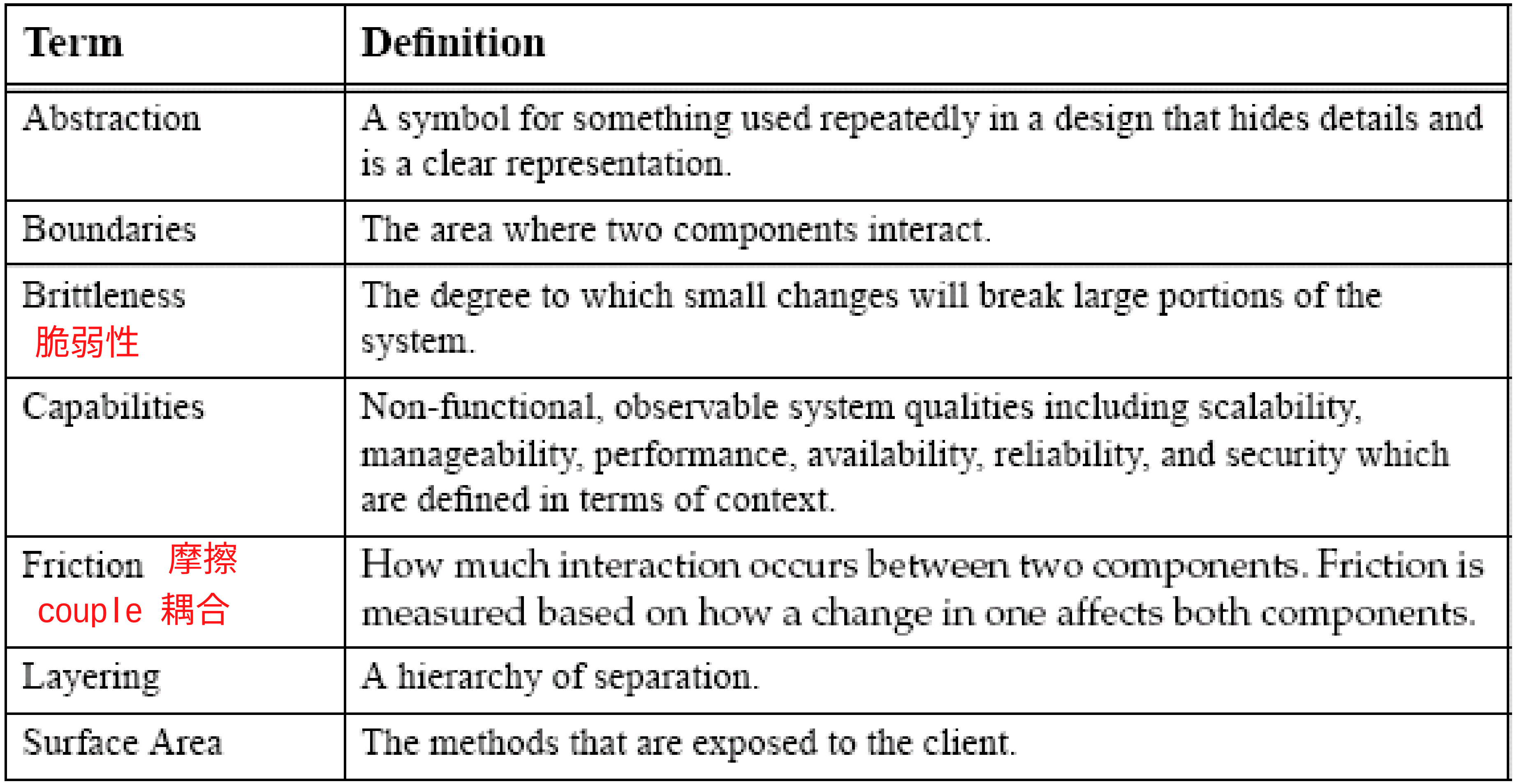
The Art of Architecture

- An architect communicates the design of the system to other members of the team.
- Defining an architecture is a creative process.
- The creative process can have positive and negative aspects.
- Architects try to balance creativity with science in the form of models, frameworks, and patterns.



Defining Architecture Terminology

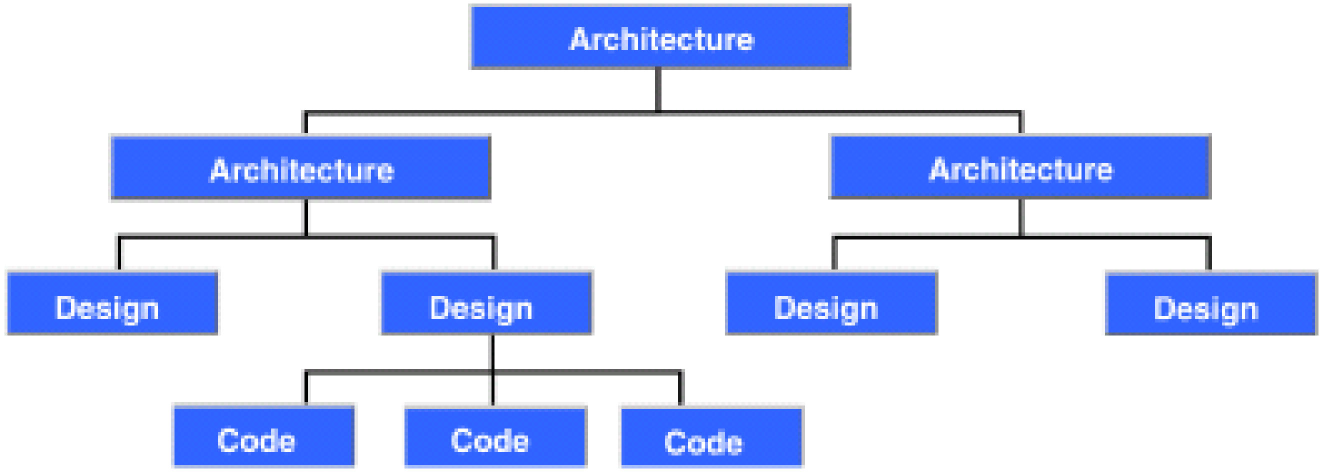
中国科学技术大学 University of Science and Technology of China



Term	Definition
Abstraction	A symbol for something used repeatedly in a design that hides details and is a clear representation.
Boundaries	The area where two components interact.
Brittleness 脆弱性	The degree to which small changes will break large portions of the system.
Capabilities	Non-functional, observable system qualities including scalability, manageability, performance, availability, reliability, and security which are defined in terms of context.
Friction 摩擦 couple 耦合	How much interaction occurs between two components. Friction is measured based on how a change in one affects both components.
Layering	A hierarchy of separation.
Surface Area	The methods that are exposed to the client.

Architecture and Design

- Key difference is in the level of details.
- Architecture operates at a high level of abstraction.
- Design operates at a low level of abstraction.



Architectural Building Blocks

- Capabilities and design goals 非功能需求
- Process and artifacts
- Communication mechanisms

To build an architecture, the architect uses the following:

- Architectural fundamentals
- Experience:
  - Best practices
  - Frameworks, patterns, and idioms

Capabilities and Design Goals

Capabilities:

- Are *non-functional*, *observable* system qualities that affect the long term viability of your architecture.
- Depend on the context of your system.
- Can be specified, measured, and prioritized.
  - Example of specifying *reliability* — No more than 1 failure per 1000 attempts or 99.9 percent.

Design goals:

- Are *non-functional*, *non-observable* system qualities designed into a system to *achieve* the capabilities.

Process and Artifacts

- A process consists of an orderly sequence of steps from initiation to delivery to deployment.
- The purpose of a process is to:
  - Address *assumptions*
  - Minimize the impact of *risks*
  - Identify *constraints*
- Artifacts result from the process.
- UML is used to communicate the architecture and design to other members of the team, both present and future.

Communication Mechanisms

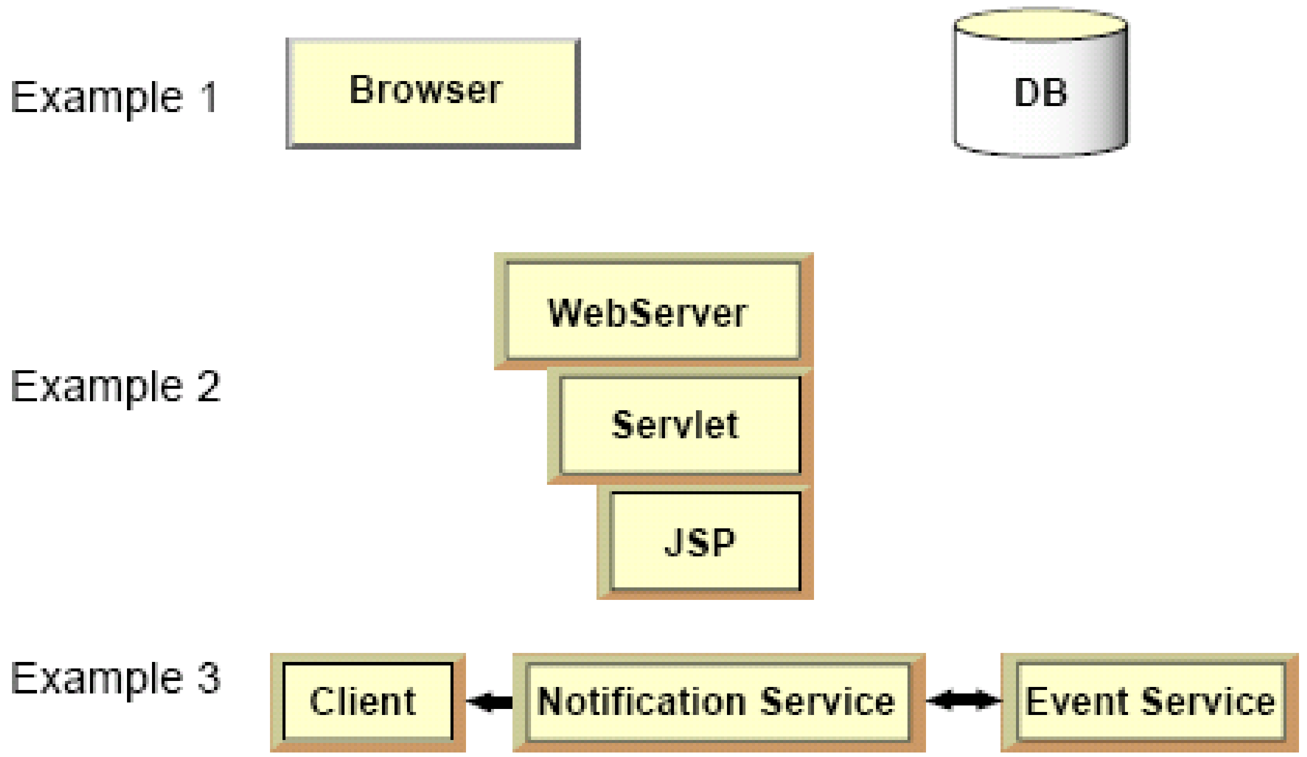
- HTTP, HTTPS
- RMI
- IIOP
- Messaging
- Transactions (ACID)

Architectural Fundamentals

- Top-down design approach
- Identifying layers, tiers, and services
- Establishing service-based architectures
- Abstraction

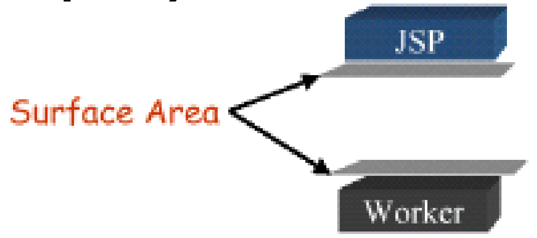
Abstraction

- Abstraction refers to creating symbols that hide design details:

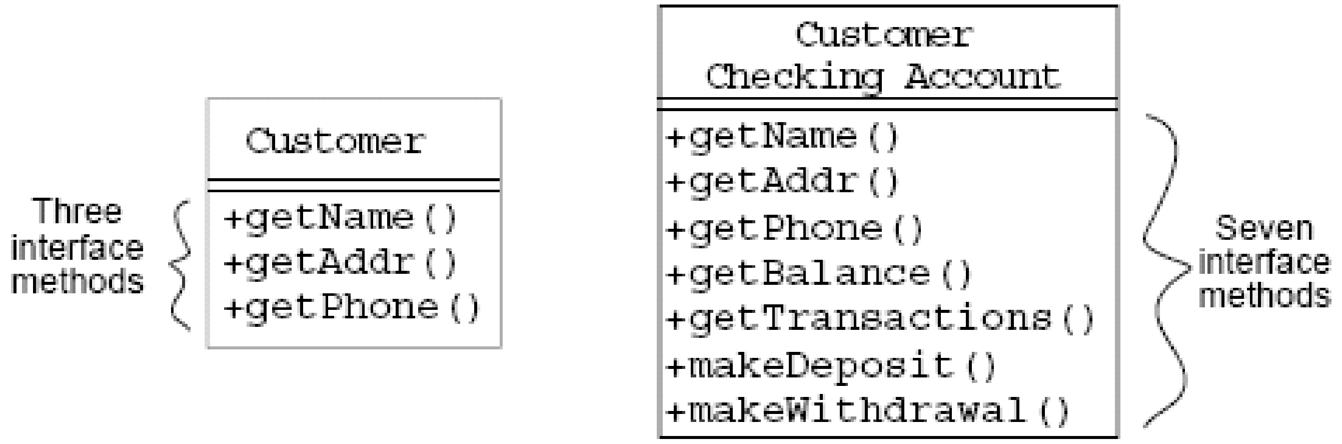


Abstraction – Surface Area

- Well-defined modules interact with one another in simple, well-defined ways.

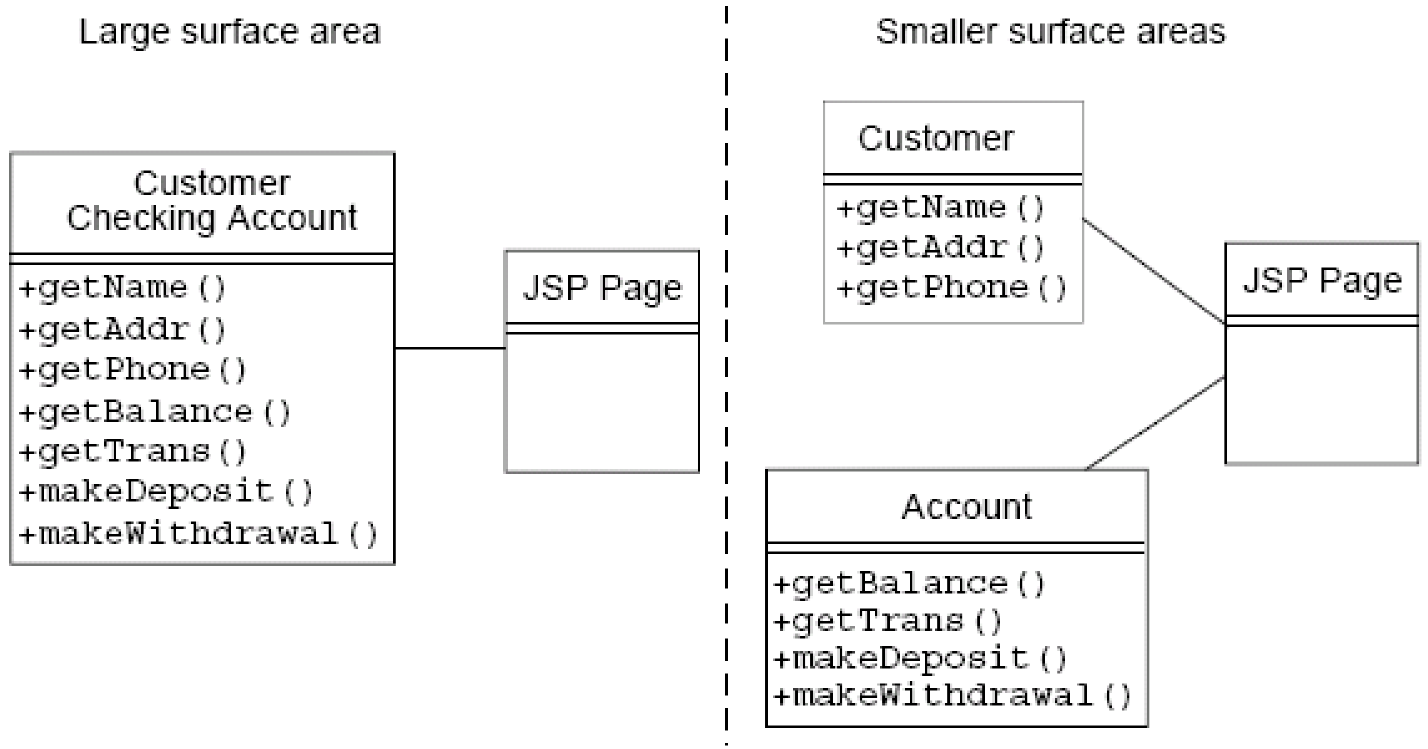


- Surface area is the different ways that modules interact.



Abstraction – Surface Area

- The greater the surface area, the more ways a change in one module can affect another.



Check Your Progress

- Define the role of an architect
- Define the term “architecture”
- Explain architectural terms such as abstraction, boundaries, brittleness, and capabilities
- List the differences between “architecture” and “design”
- Identify the fundamentals of system architecture
- Identify and define key architectural principles
- Explain the concept of abstraction, and how it is implemented in system architecture

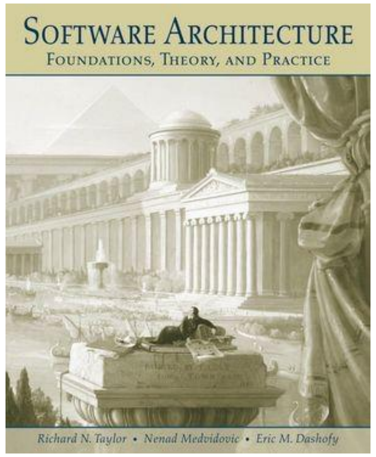
Think Beyond

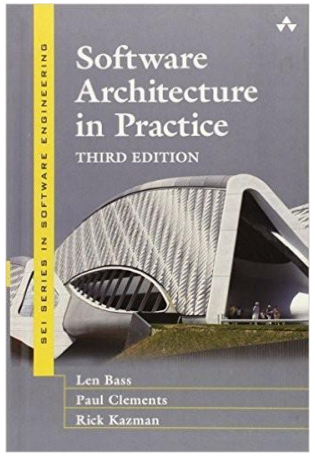
- The architect has many issues to consider, as discussed in this module.
- How absolute are these issues?
- Can you think of how they relate to each other?
- Can you think of other issues that should be considered?

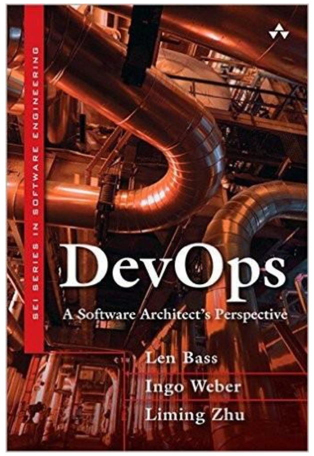
Conclusion

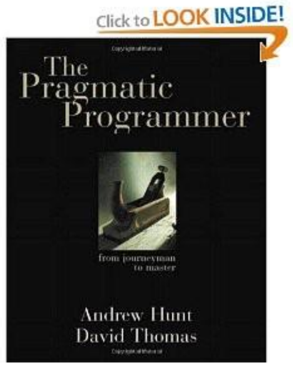
- How do I start
  - Become an excellent developer who knows why not just how
  - Understand the relationship of underline technical architecture and software systems
  - Understand current platform specific architecture
  - Know the domain you are working on, become an expert
  - Read, read, read, think, think, think

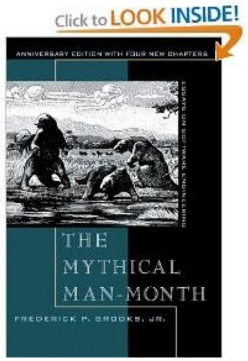
Books

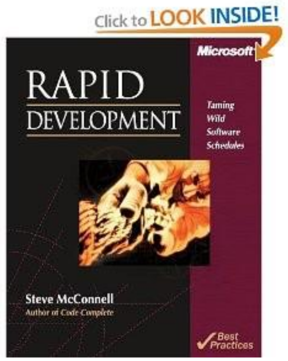

SOFTWARE ARCHITECTURE
FOUNDATIONS, THEORY, AND PRACTICE
Richard N. Taylor • Nenad Medvidovic • Eric M. Dashofy

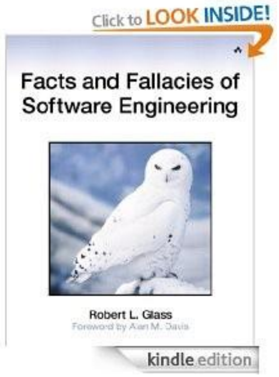

SEI SERIES IN SOFTWARE ENGINEERING
Software Architecture in Practice
THIRD EDITION
Len Bass
Paul Clements
Rick Kazman


SEI SERIES IN SOFTWARE ENGINEERING
DevOps
A Software Architect's Perspective
Len Bass
Ingo Weber
Liming Zhu


Click to LOOK INSIDE!
The Pragmatic Programmer
from journeyman to master
Andrew Hunt
David Thomas


Click to LOOK INSIDE!
ANNIVERSARY EDITION WITH FOUR NEW CHAPTERS
THE MYTHICAL MAN-MONTH
FREDERICK P. BROOKS, JR.


Click to LOOK INSIDE!
Microsoft
RAPID DEVELOPMENT
Taming Wild Software Schedules
Steve McConnell
Author of Code Complete
Best Practices


Click to LOOK INSIDE!
Facts and Fallacies of Software Engineering
Robert L. Glass
Foreword by Alan M. Davis
kindle edition