BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Software Engineering

Topic 4
Design Engineering

BINUS UNIVERSITY ONLINE LEARNING

Acknowledgement

These slides have been adapted from
Pressman, R.S. (2015). *Software Engineering : A Practioner's Approach. 8th ed*. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157. Chapter 12, 13, 14, 15, 16, 17, and 18

People
Innovation
Excellence

Learning Objectives

LO2 : Explain the software engineering practices and business environment

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Contents

SOFTWARE ARCHITECTURE

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Architecture

The architecture is not the operational software. Rather, it is a representation that enables a software engineer to:

(1) analyze the effectiveness of the design in meeting its stated requirements,

(2) consider architectural alternatives at a stage when making design changes is still relatively easy, and

(3) reduce the risks associated with the construction of the software.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Software Architecture

Architectural Styles

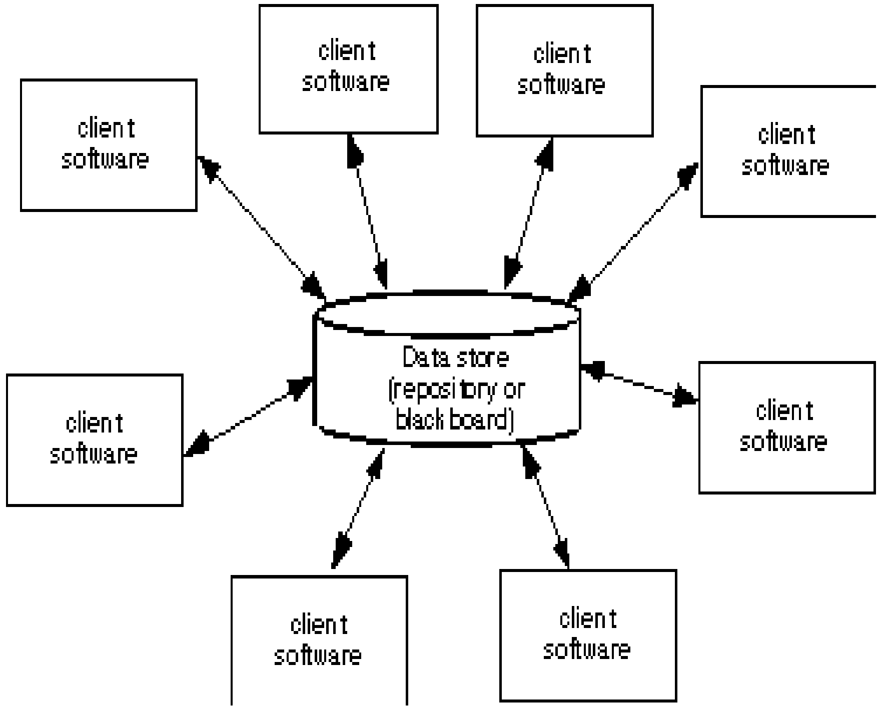
Each style describes a system category that encompasses: (1) a **set of components** (e.g., a database, computational modules) that perform a function required by a system, (2) a **set of connectors** that enable “communication, coordination and cooperation” among components, (3) **constraints** that define how components can be integrated to form the system, and (4) **semantic models** that enable a designer to understand the overall properties of a system by analyzing the known properties of its constituent parts.

- **Data-centered architectures**
- **Data flow architectures**
- **Call and return architectures**
- **Object-oriented architectures**
- **Layered architectures**

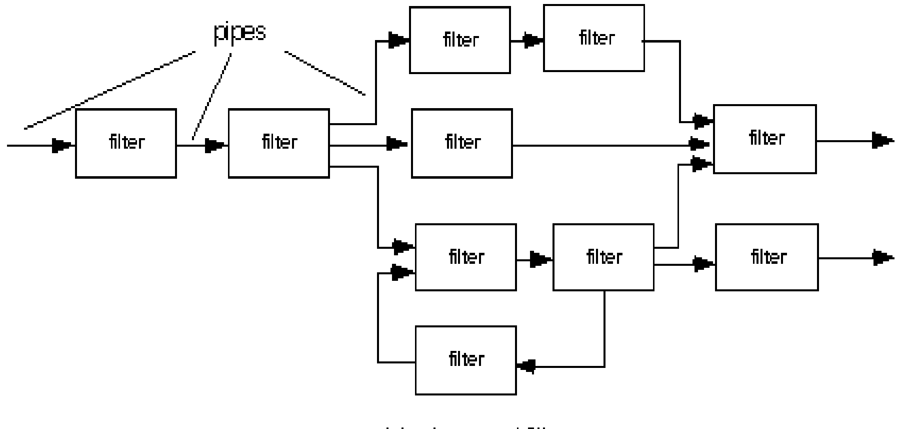
People
Innovation
Excellence

Software Architecture

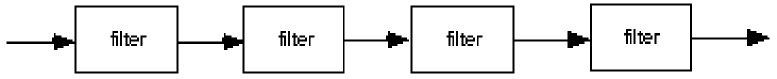
Data Centered Architecture



Data Flow Architecture



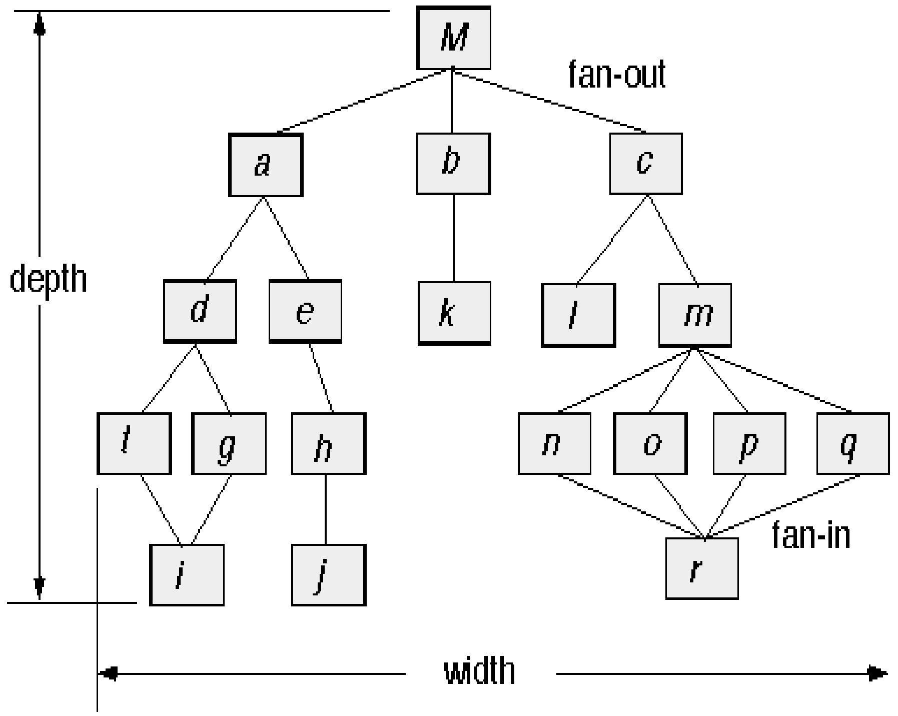
(a) pipes and filters



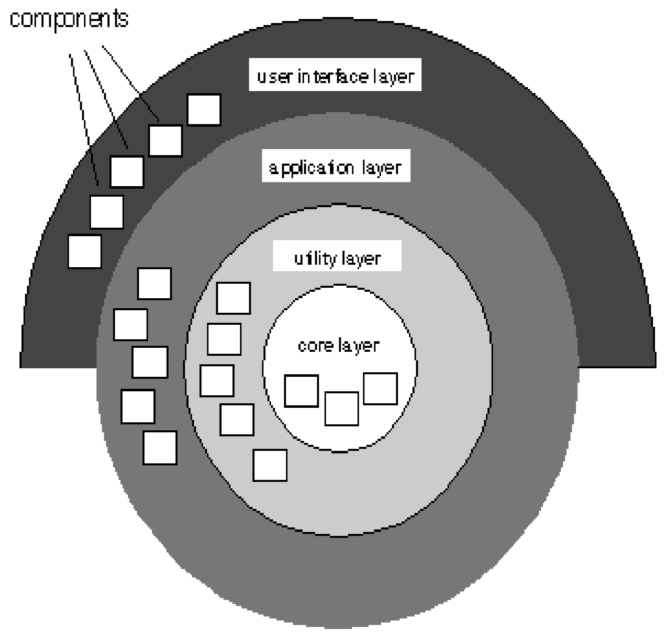
(b) batch sequential

Software Architecture

Call and Return Architecture



Layered Architecture



ARCHITECTURAL DESIGN

BINUS UNIVERSITY ONLINE LEARNING

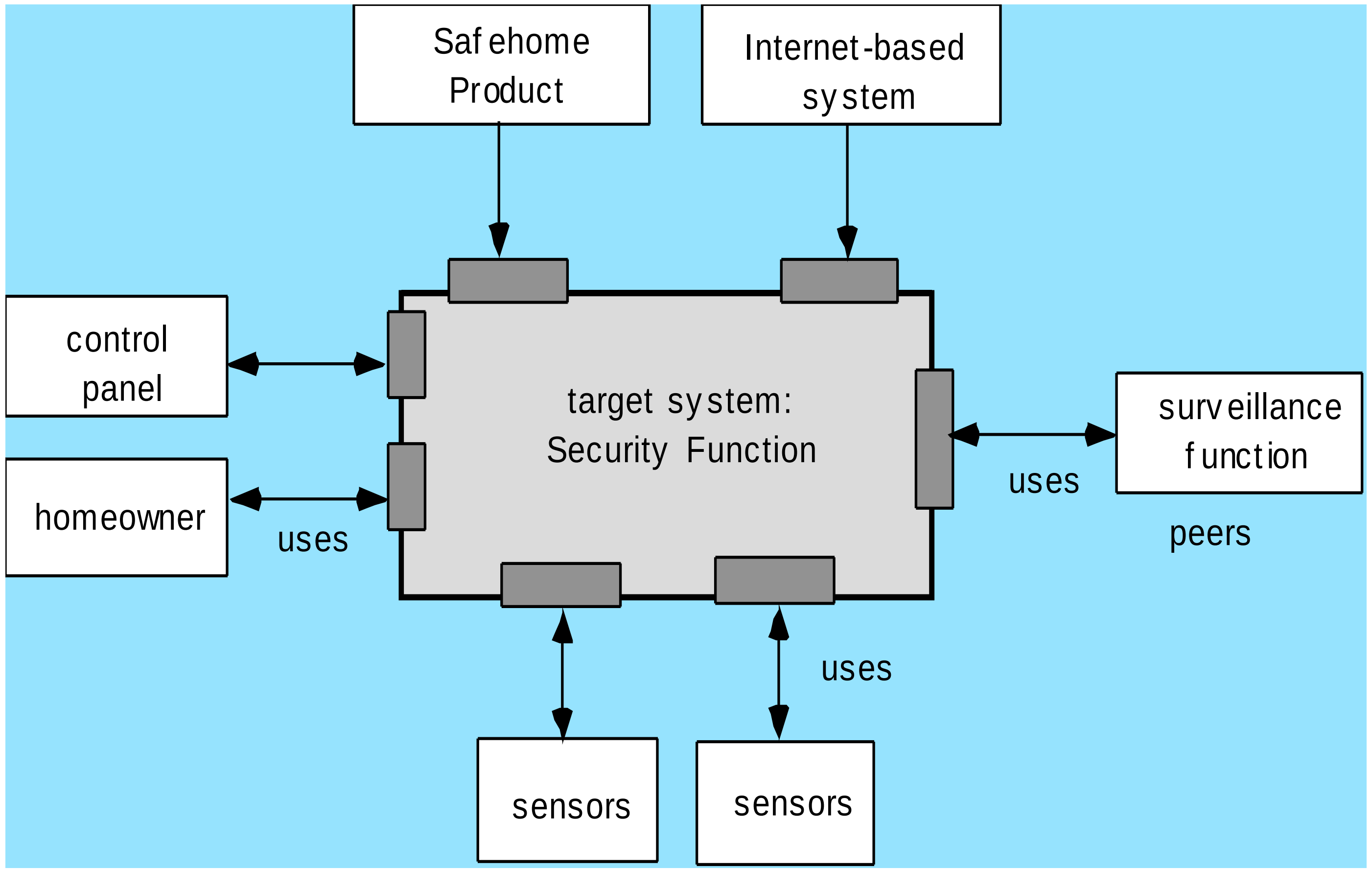
People Innovation Excellence

Architectural Design

- The software must be placed into context
    - the design should define the external entities (other systems, devices, people) that the software interacts with and the nature of the interaction
- A set of architectural archetypes should be identified
    - An *archetype* is an abstraction (similar to a class) that represents one element of system behavior
- The designer specifies the structure of the system by defining and refining software components that implement each archetype

Architectural Design

Architectural Context

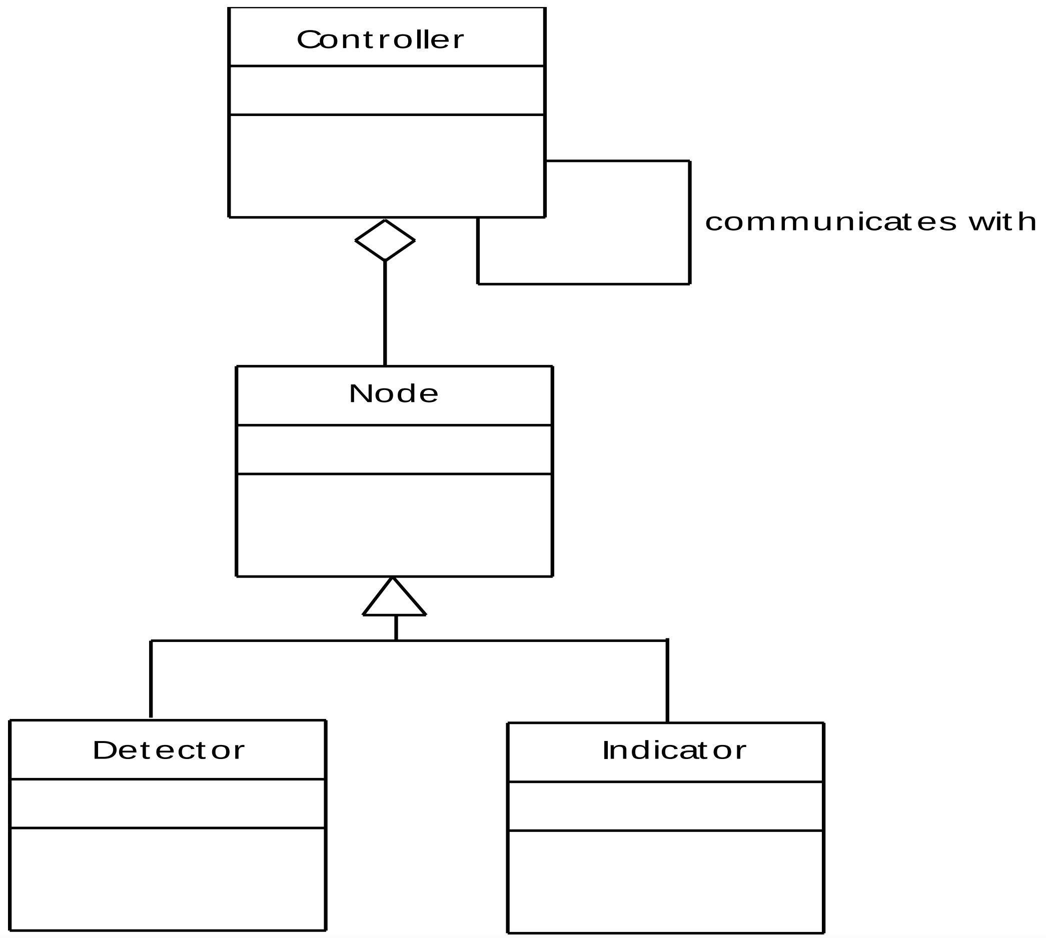


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Architectural Design

Archetypes

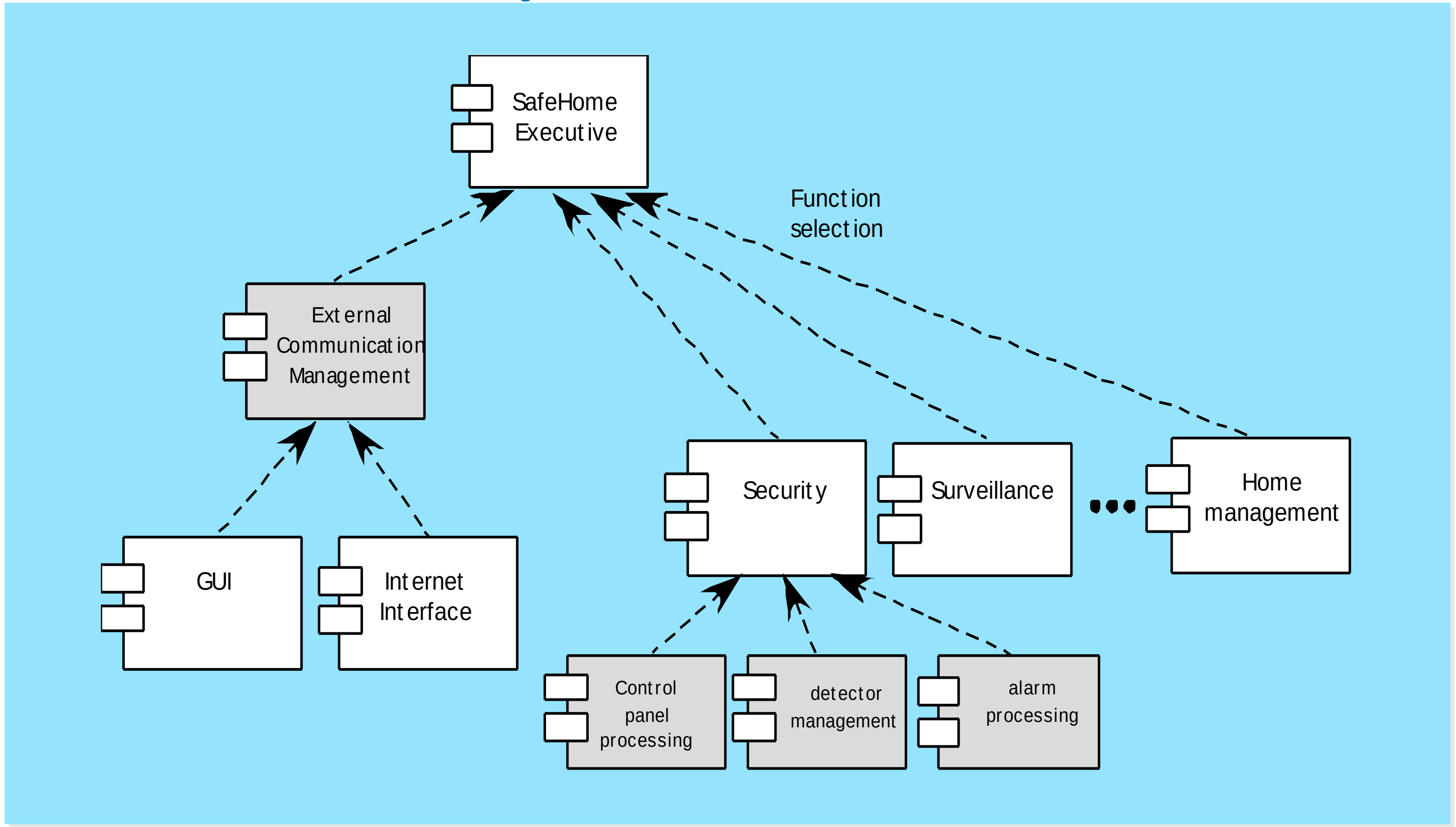


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Architectural Design

Component Structure

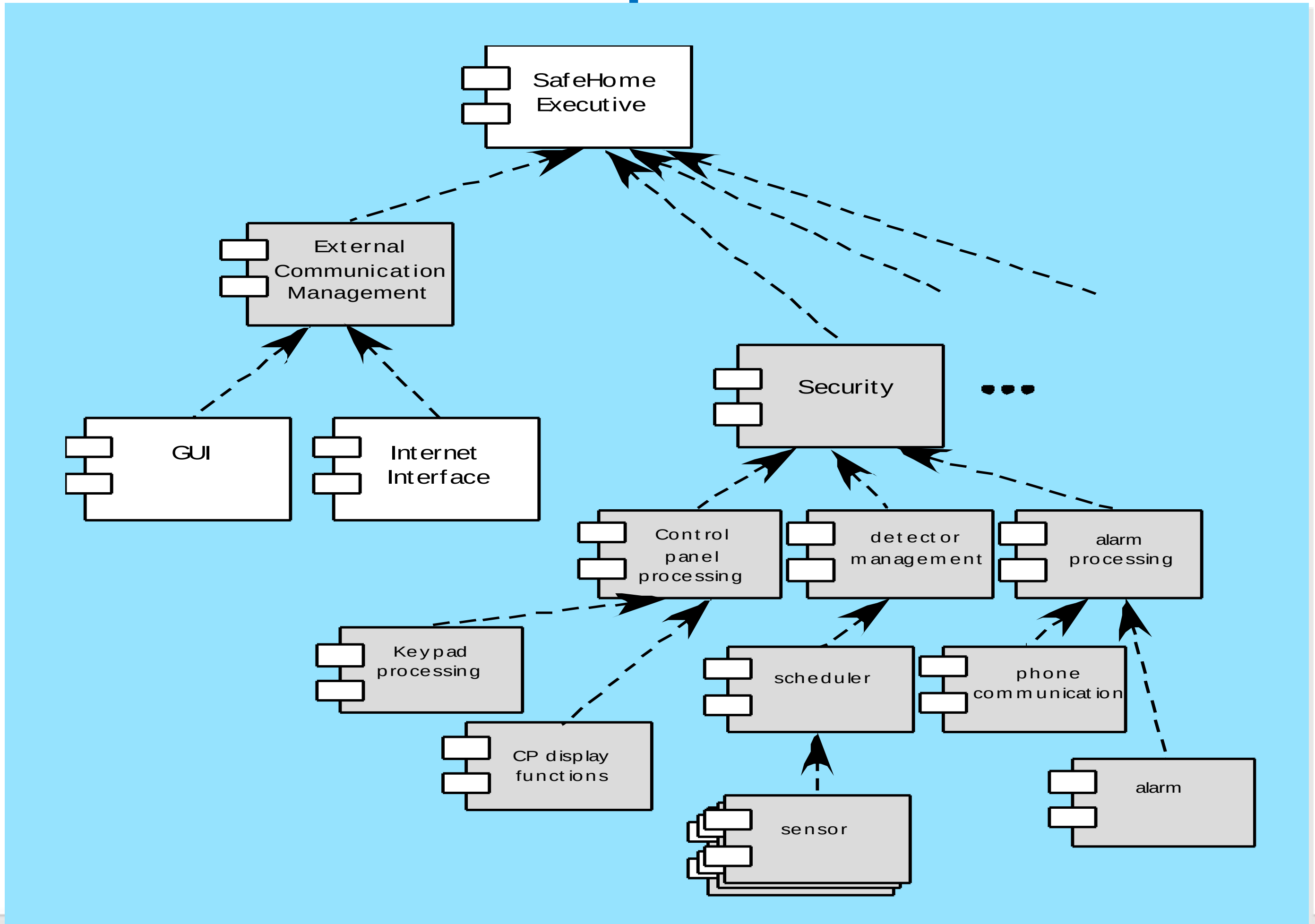


BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Architectural Design

Refined Component Structure



COMPONEN LEVEL DESIGN

BINUS UNIVERSITY ONLINE LEARNING

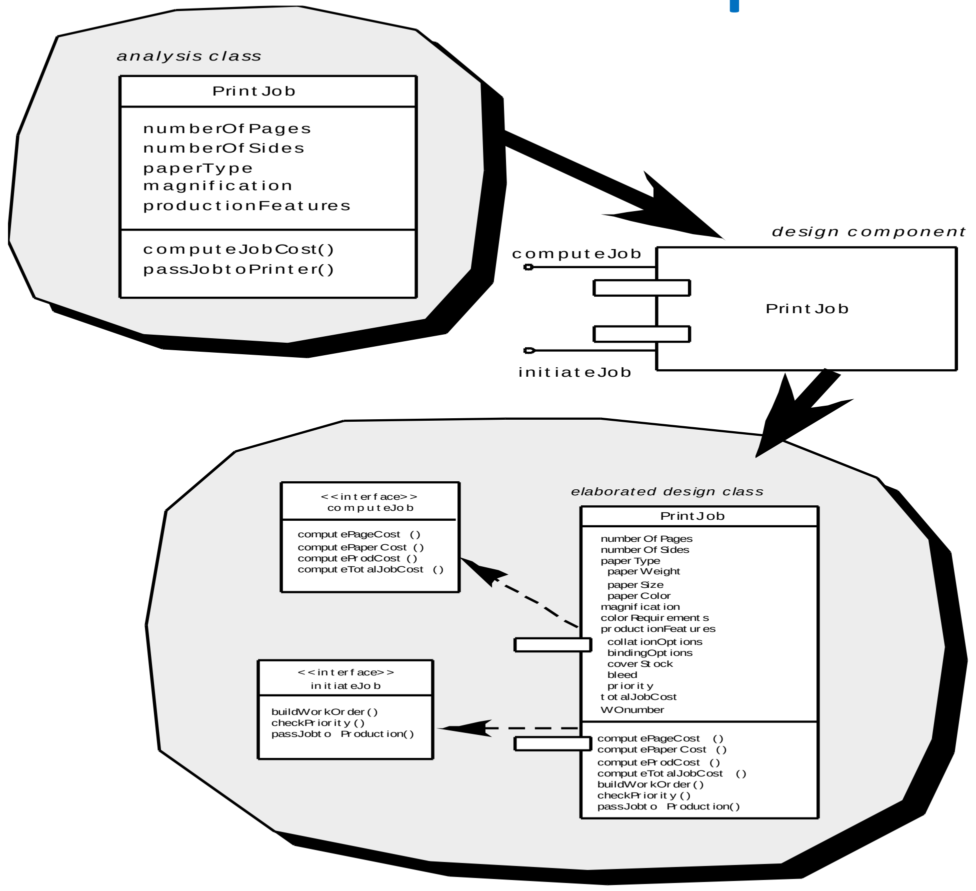
People Innovation Excellence

Component Level Design

- *OMG Unified Modeling Language Specification* [OMG01] defines a component as
  - "… a modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces."
- *OO view:* a component contains a set of collaborating classes
- *Conventional view:* a component contains processing logic, the internal data structures that are required to implement the processing logic, and an interface that enables the component to be invoked and data to be passed to it.

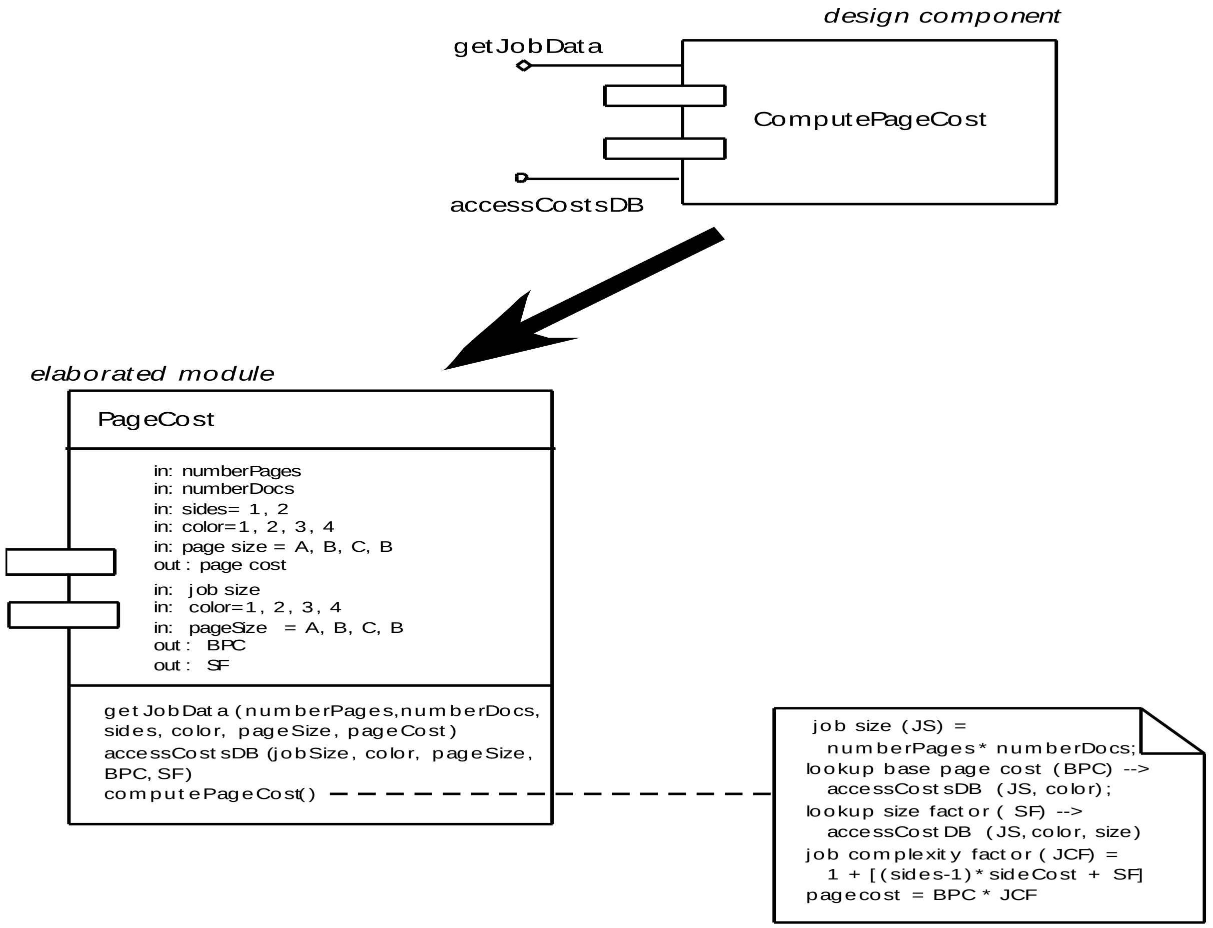
Component Level Design

OO Component



Component Level Design

Conventional Component



```
in: numberPages
in: numberDocs
in: sides= 1, 2
in: color=1, 2, 3, 4
in: page size = A, B, C, B
out : page cost
in:  job size
in:  color=1, 2, 3, 4
in:  pageSize  = A, B, C, B
out :  BPC
out :  SF
```

```
getJobData (numberPages,numberDocs,
sides, color, pageSize, pageCost)
accessCostsDB (jobSize, color, pageSize,
BPC, SF)
computePageCost()
```

```
 job size (JS) =
   numberPages* numberDocs;
lookup base page cost (BPC) -->
   accessCostsDB (JS, color);
lookup size factor ( SF) -->
   accessCostDB (JS, color, size)
job complexity factor ( JCF) =
   1 + [(sides-1)* sideCost + SF]
pagecost = BPC * JCF
```

People
Innovation
Excellence

Component Level Design

Cohesion

- Conventional view:
  - the “single-mindedness” of a module
- OO view:
  - *cohesion* implies that a component or class encapsulates only attributes and operations that are closely related to one another and to the class or component itself
- Levels of cohesion
  - Functional
  - Layer
  - Communicational
  - Sequential
  - Procedural
  - Temporal
  - utility

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Component Level Design

Coupling

- Conventional view:
  - The degree to which a component is connected to other components and to the external world
- OO view:
  - a qualitative measure of the degree to which classes are connected to one another
- Level of coupling
  - Content
  - Common
  - Control
  - Stamp
  - Data
  - Routine call
  - Type use
  - Inclusion or import
  - External

People
Innovation
Excellence

Component Level Design

Component Design for WebApps

- WebApp component is
  - (1) a well-defined cohesive function that manipulates content or provides computational or data processing for an end-user, or
  - (2) a cohesive package of content and functionality that provides end-user with some required capability.
- Therefore, component-level design for WebApps often incorporates elements of content design and functional design.

Component Level Design

Component-Based Development

- When faced with the possibility of reuse, the software team asks:
  - Are commercial off-the-shelf (COTS) components available to implement the requirement?
  - Are internally-developed reusable components available to implement the requirement?
  - Are the interfaces for available components compatible within the architecture of the system to be built?
- At the same time, they are faced with the following impediments to reuse ...

BINUS UNIVERSITY ONLINE LEARNING

Component Level Design

User Interface Design Models

- User model — a profile of all end users of the system
- Design model — a design realization of the user model
- Mental model (system perception) — the user’s mental image of what the interface is
- Implementation model — the interface “look and feel” coupled with supporting information that describe interface syntax and semantics

People
Innovation
Excellence

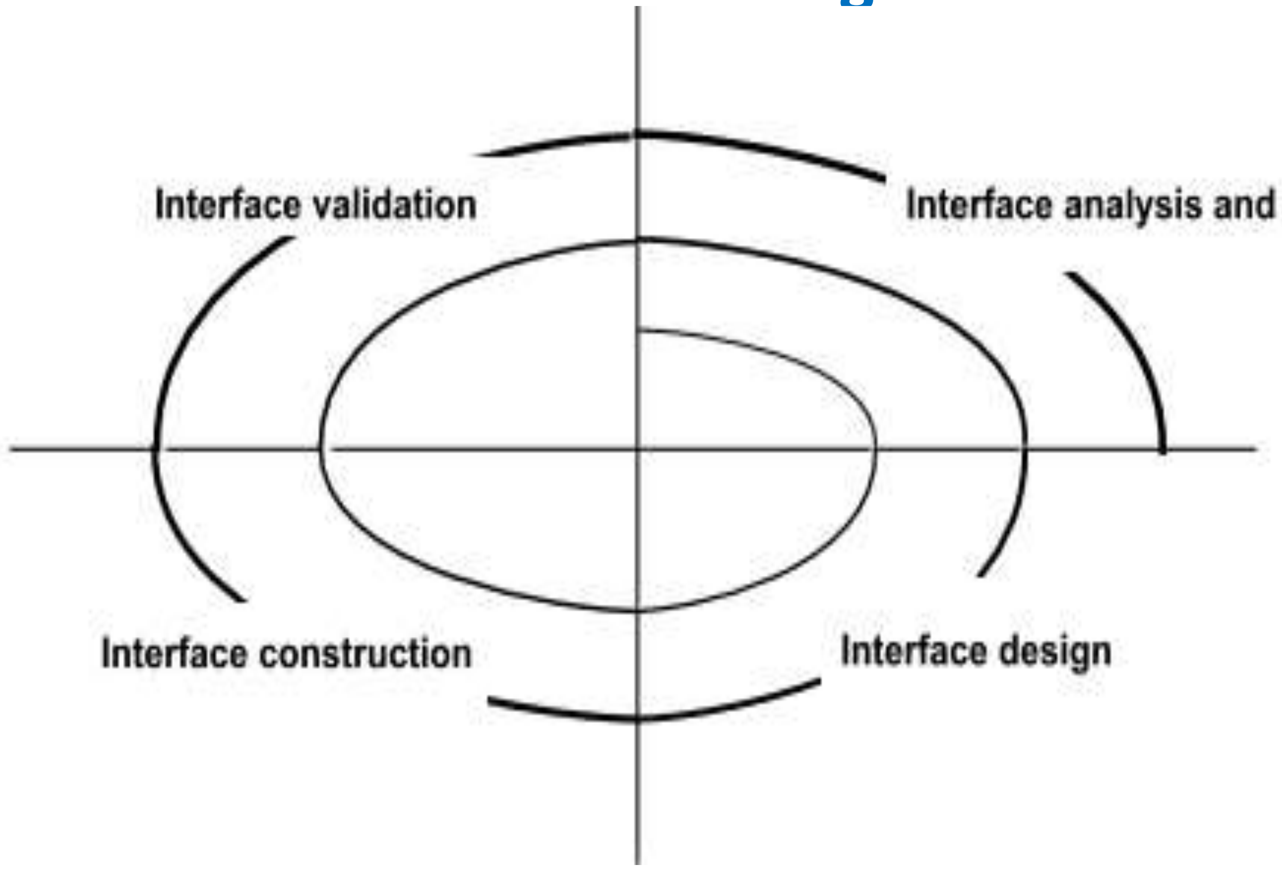
INTERFACE DESIGN

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Interface Design

User Interface Design Process



BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Interface Design

Interface Analysis

- Interface analysis means understanding
  - (1) the people (end-users) who will interact with the system through the interface;
  - (2) the tasks that end-users must perform to do their work,
  - (3) the content that is presented as part of the interface
  - (4) the environment in which these tasks will be conducted.

BINUS UNIVERSITY ONLINE LEARNING

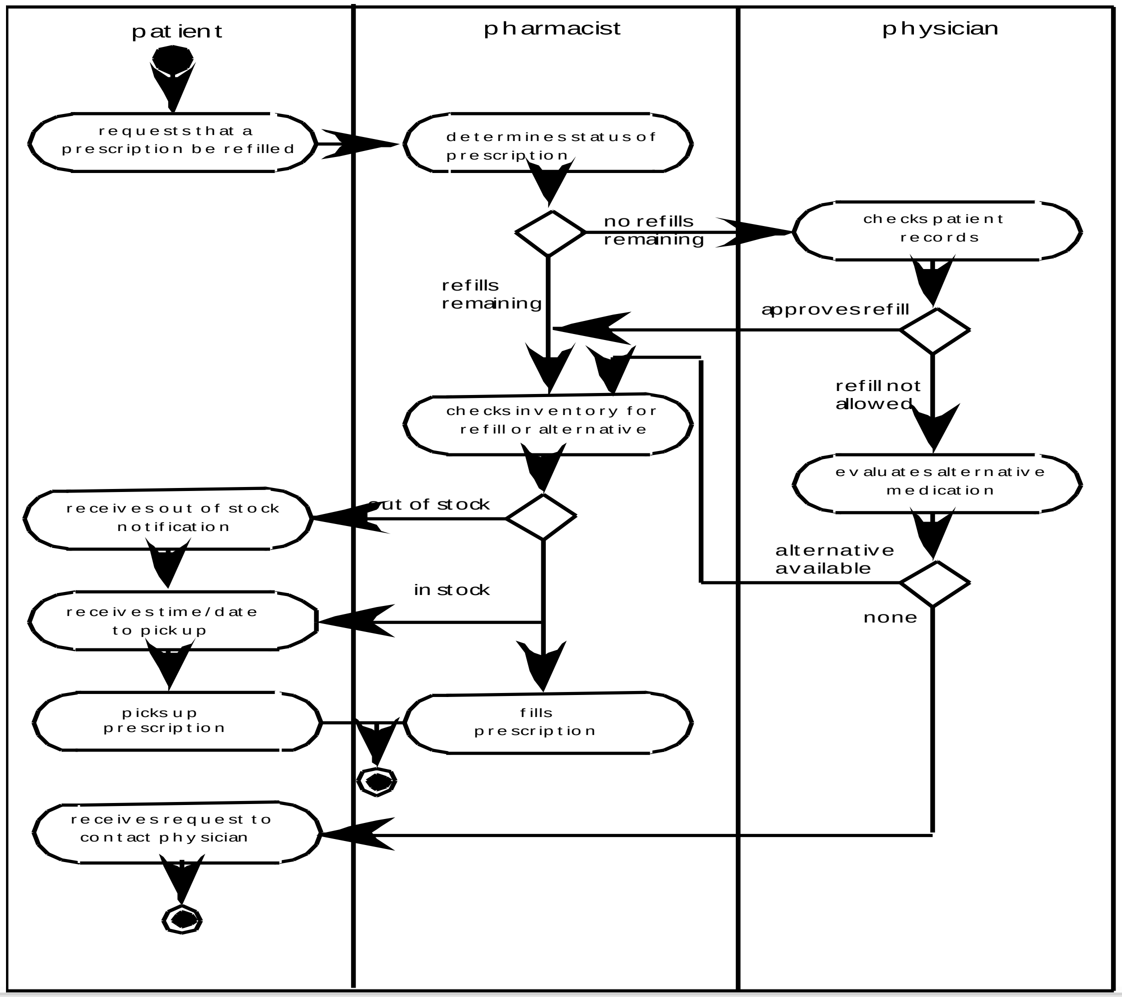
People Innovation Excellence

Task Analysis and Modeling

- Answers the following questions …
  - What work will the user perform in specific circumstances?
  - What tasks and subtasks will be performed as the user does the work?
  - What specific problem domain objects will the user manipulate as work is performed?
  - What is the sequence of work tasks—the workflow?
  - What is the hierarchy of tasks?
- Use-cases define basic interaction
- Task elaboration refines interactive tasks
- Object elaboration identifies interface objects (classes)
- Workflow analysis defines how a work process is completed when several people (and roles) are involved

Interface Design

Swimlane Diagram


patient
pharmacist
physician
requests that a prescription be refilled
determines status of prescription
no refills remaining
checks patient records
refills remaining
approves refill
refill not allowed
checks inventory for refill or alternative
evaluates alternative medication
out of stock
receives out of stock notification
alternative available
none
in stock
receives time/date to pick up
picks up prescription
fills prescription
receives request to contact physician

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

Interface Design

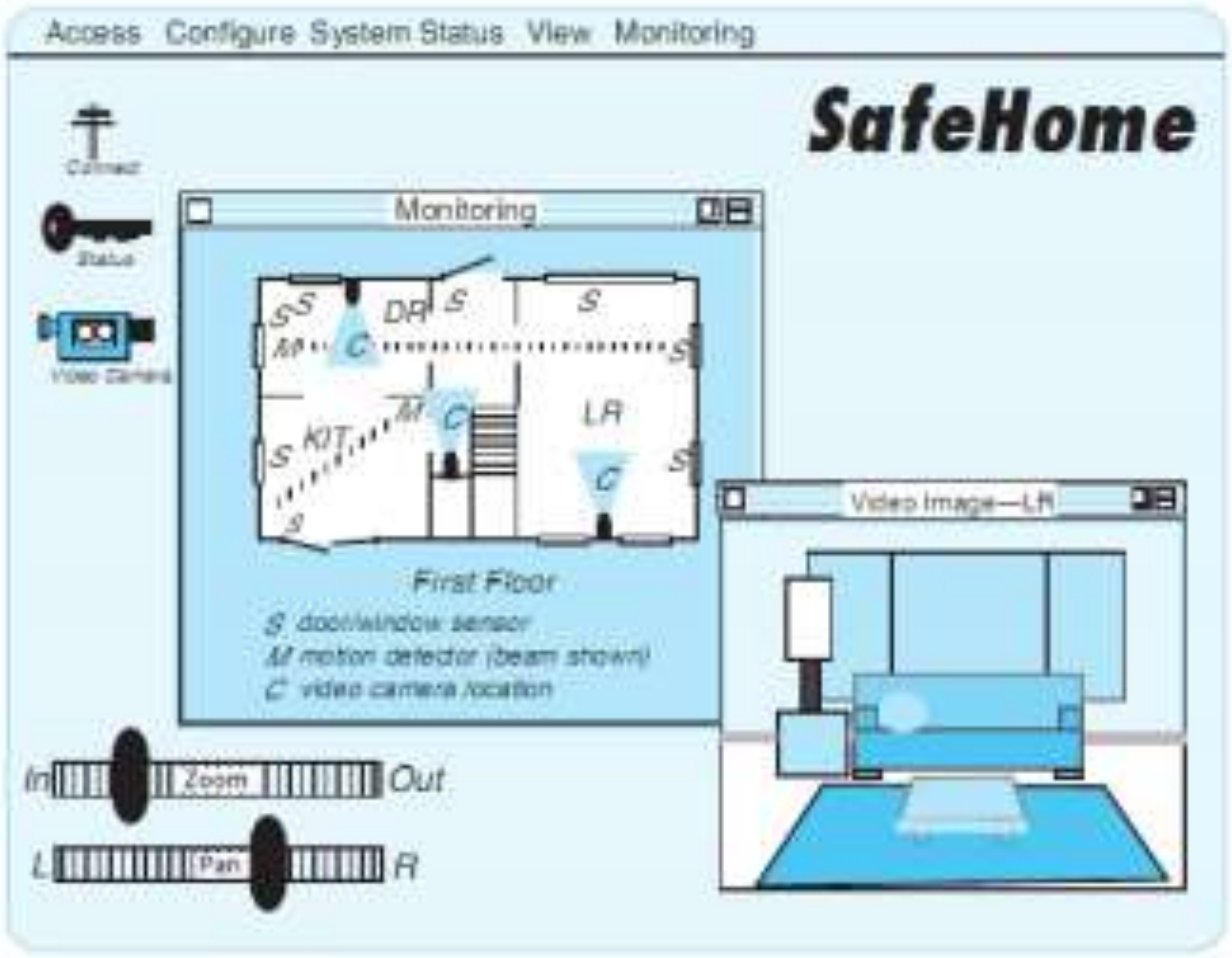
Interface Design Steps

- Using information developed during interface analysis, define interface objects and actions (operations).
- Define events (user actions) that will cause the state of the user interface to change. Model this behavior.
- Depict each interface state as it will actually look to the end-user.
- Indicate how the user interprets the state of the system from information provided through the interface.

People
Innovation
Excellence

Interface Design

Preliminary screen layout



WEB AND MOBILE INTERFACE DESIGN

Interface Design Principles - I

- **Anticipation**—A WebApp should be designed so that it anticipates the use's next move.
- **Communication**—The interface should communicate the status of any activity initiated by the user
- **Consistency**—The use of navigation controls, menus, icons, and aesthetics (e.g., color, shape, layout)
- **Controlled autonomy**—The interface should facilitate user movement throughout the WebApp, but it should do so in a manner that enforces navigation conventions that have been established for the application.
- **Efficiency**—The design of the WebApp and its interface should optimize the user's work efficiency, not the efficiency of the Web engineer who designs and builds it or the client-server environment that executes it.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Interface Design Principles - II

- **Focus**—The WebApp interface (and the content it presents) should stay focused on the user task(s) at hand.
- **Fitt's Law**—"The time to acquire a target is a function of the distance to and size of the target."
- **Human interface objects**—A vast library of reusable human interface objects has been developed for WebApps.
- **Latency reduction**—The WebApp should use multi-tasking in a way that lets the user proceed with work as if the operation has been completed.
- **Learnability**— A WebApp interface should be designed to minimize learning time, and once learned, to minimize relearning required when the WebApp is revisited.

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

WebApp and Mobile Interface Design

Interface Design Principles-III

- **Maintain work product integrity**—A work product (e.g., a form completed by the user, a user specified list) must be automatically saved so that it will not be lost if an error occurs.
- **Readability**—All information presented through the interface should be readable by young and old.
- **Track state**—When appropriate, the state of the user interaction should be tracked and stored so that a user can logoff and return later to pick up where she left off.
- **Visible navigation**—A well-designed WebApp interface provides "the illusion that users are in the same place, with the work brought to them."

Interface Design Workflow-I

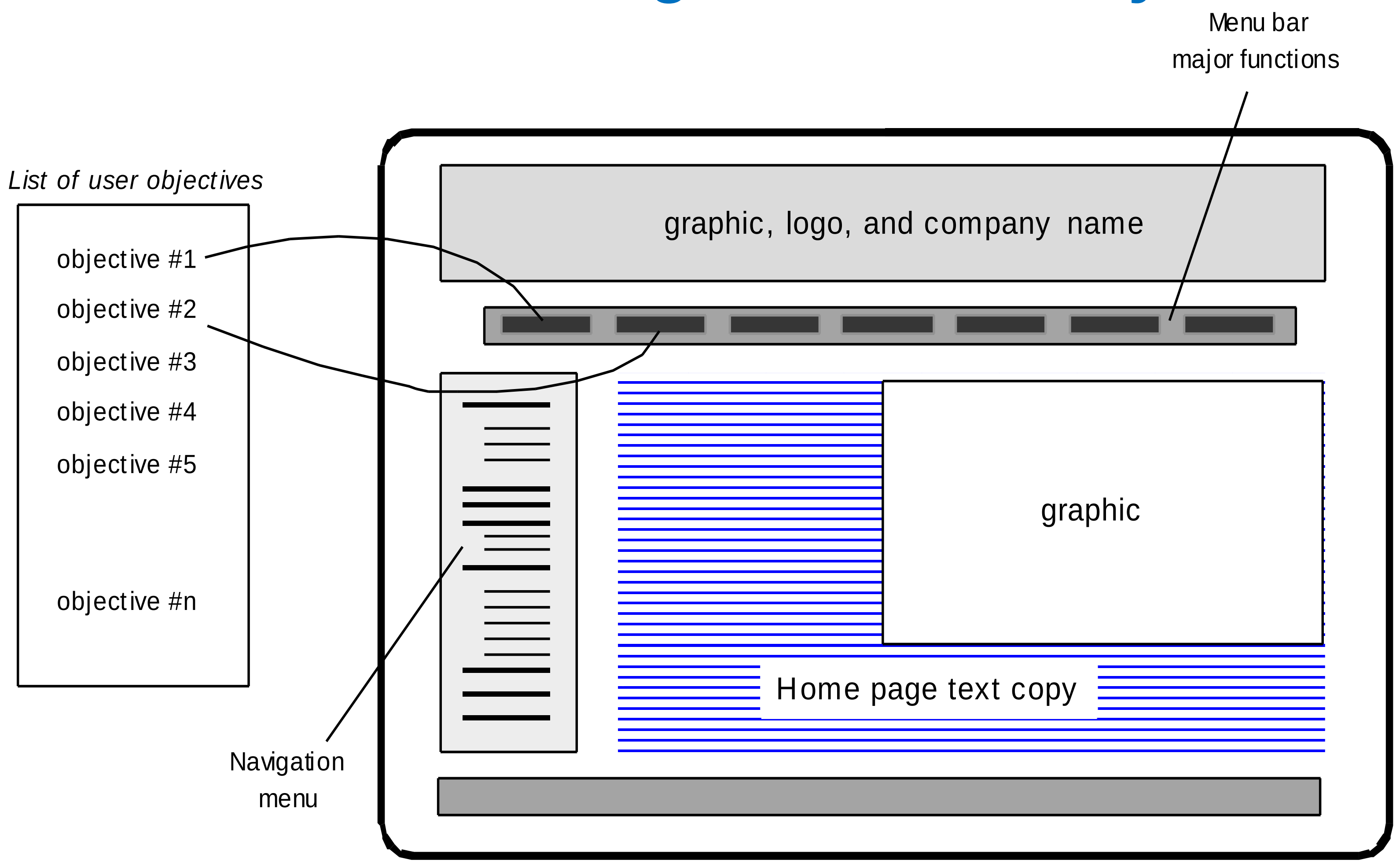
- Review information contained in the analysis model and refine as required.
- Develop a rough sketch of the WebApp interface layout.
- Map user objectives into specific interface actions.
- Define a set of user tasks that are associated with each action.
- Storyboard screen images for each interface action.
- Refine interface layout and storyboards using input from aesthetic design.

Interface Design Workflow-II

- Identify user interface objects that are required to implement the interface.
- Develop a procedural representation of the user’s interaction with the interface.
- Develop a behavioral representation of the interface.
- Describe the interface layout for each state.
- Refine and review the interface design model.

WebApp and Mobile Interface Design

The interface design evaluation cycle



DESIGN PATTERNS

Design Patterns

Effective Patterns

- Coplien [Cop05] characterizes an effective design pattern in the following way:
  - *It solves a problem*: Patterns capture solutions, not just abstract principles or strategies.
  - *It is a proven concept*: Patterns capture solutions with a track record, not theories or speculation.
  - *The solution isn't obvious*: Many problem-solving techniques (such as software design paradigms or methods) try to derive solutions from first principles. The best patterns *generate* a solution to a problem indirectly--a necessary approach for the most difficult problems of design.
  - *It describes a relationship*: Patterns don't just describe modules, but describe deeper system structures and mechanisms.
  - *The pattern has a significant human component (minimize human intervention).* All software serves human comfort or quality of life; the best patterns explicitly appeal to aesthetics and utility.

Design Patterns

Generative Patterns

- *Generative patterns* describe an important and repeatable aspect of a system and then provide us with a way to build that aspect within a system of forces that are unique to a given context.

- A collection of generative design patterns could be used to “generate” an application or computer-based system whose architecture enables it to adapt to change.

Describing a Pattern

- ***Pattern name***—describes the essence of the pattern in a short but expressive name
- ***Problem***—describes the problem that the pattern addresses
- ***Motivation***—provides an example of the problem
- ***Context***—describes the environment in which the problem resides including application domain
- ***Forces***—lists the system of forces that affect the manner in which the problem must be solved; includes a discussion of limitation and constraints that must be considered
- ***Solution***—provides a detailed description of the solution proposed for the problem
- ***Intent***—describes the pattern and what it does
- ***Collaborations***—describes how other patterns contribute to the solution
- ***Consequences***—describes the potential trade-offs that must be considered when the pattern is implemented and the consequences of using the pattern
- ***Implementation***—identifies special issues that should be considered when implementing the pattern
- ***Known uses***—provides examples of actual uses of the design pattern in real applications
- ***Related patterns***—cross-references related design patterns

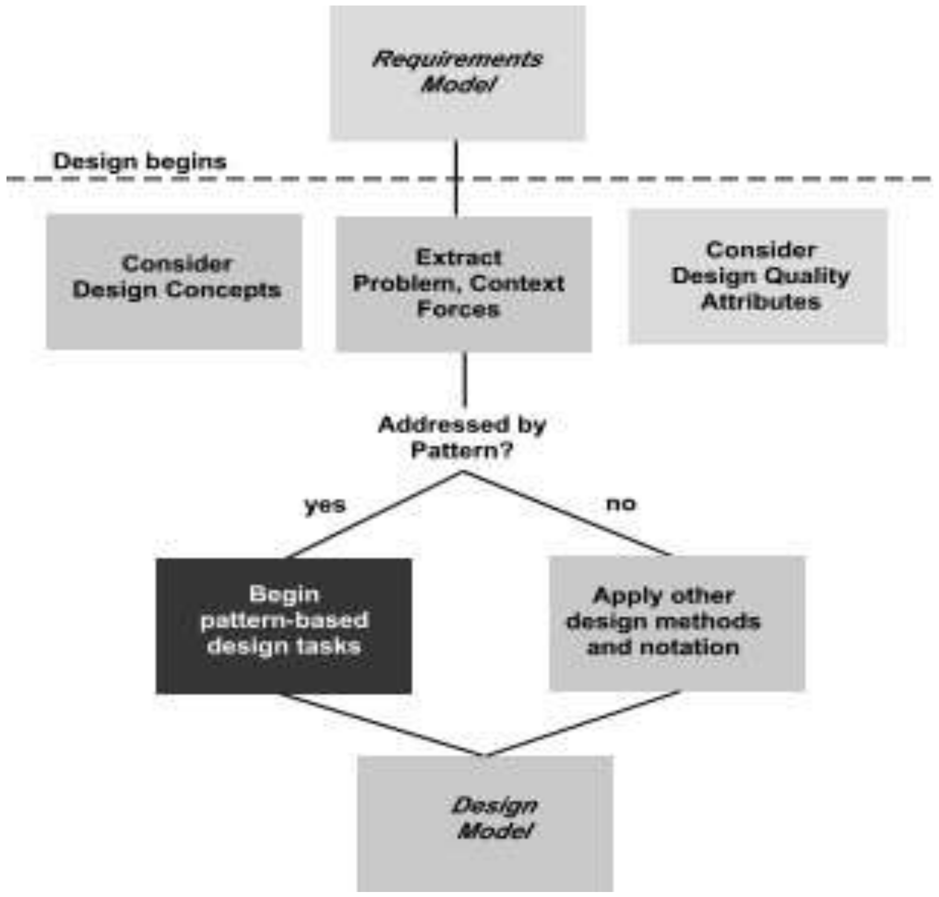
Design Patterns

Pattern-Based Design

- A software designer begins with a requirements model (either explicit or implied) that presents an abstract representation of the system.
- The requirements model describes the problem set, establishes the context, and identifies the system of forces that hold sway.
- Then …

Design Patterns

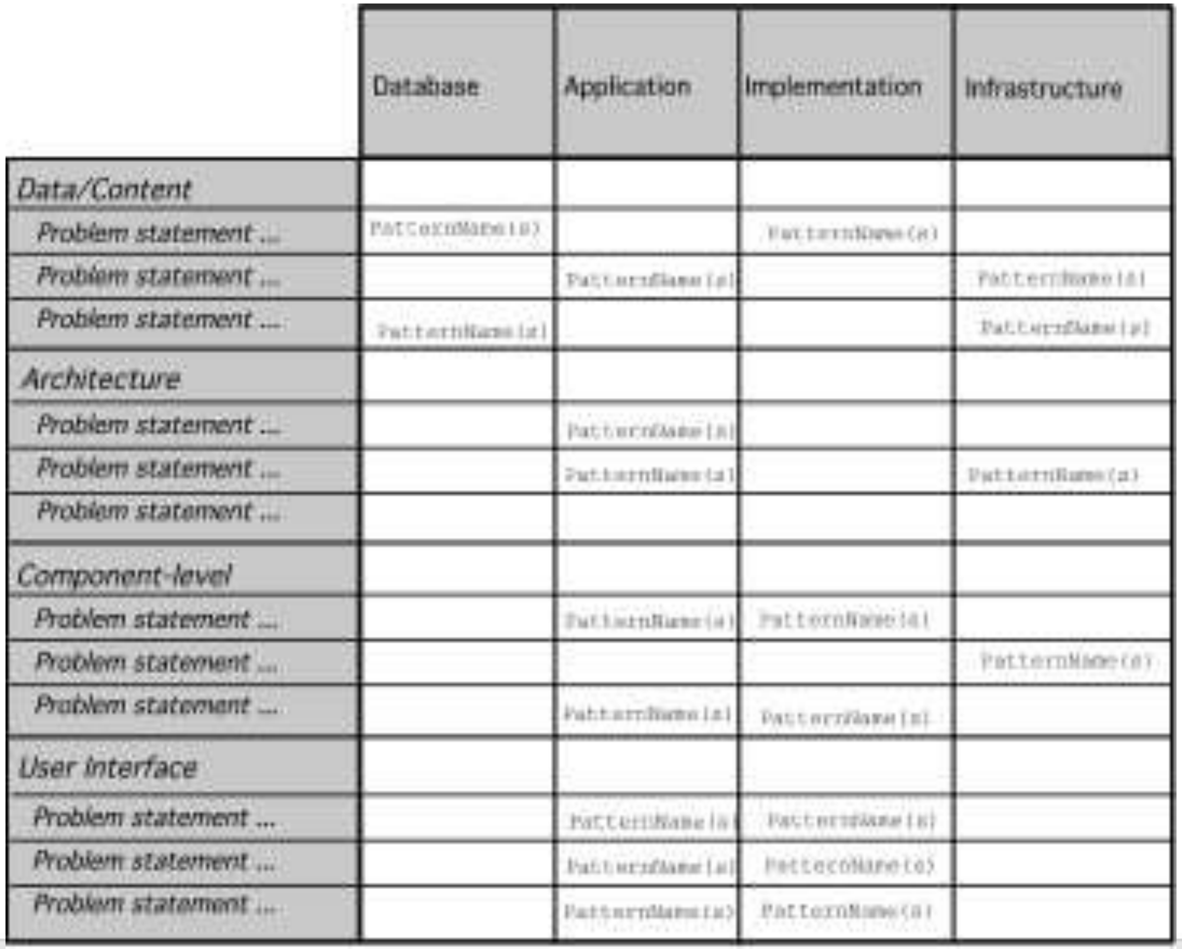
Pattern-based design in context



Requirements Model
Design begins
Consider Design Concepts
Extract Problem, Context Forces
Consider Design Quality Attributes
Addressed by Pattern?
yes
no
Begin pattern-based design tasks
Apply other design methods and notation
Design Model

Design Patterns

Pattern Organizing Table



	Database	Application	Implementation	Infrastructure
Data/Content				
Problem statement ...	PatternName(s)		PatternName(s)	
Problem statement ...		PatternName(s)		PatternName(s)
Problem statement ...	PatternName(s)			PatternName(s)
Architecture				
Problem statement ...		PatternName(s)		
Problem statement ...		PatternName(s)		PatternName(s)
Problem statement ...				
Component-level				
Problem statement ...		PatternName(s)	PatternName(s)	
Problem statement ...				PatternName(s)
Problem statement ...		PatternName(s)	PatternName(s)	
User interface				
Problem statement ...		PatternName(s)	PatternName(s)	
Problem statement ...		PatternName(s)	PatternName(s)	
Problem statement ...		PatternName(s)	PatternName(s)	

Architectural Patterns

- Example: every house (and every architectural style for houses) employs a **Kitchen** pattern.
- The **Kitchen** pattern and patterns it collaborates with address problems associated with the storage and preparation of food, the tools required to accomplish these tasks, and rules for placement of these tools relative to workflow in the room.
- In addition, the pattern might address problems associated with counter tops, lighting, wall switches, a central island, flooring, and so on.
- Obviously, there is more than a single design for a kitchen, often dictated by the context and system of forces. But every design can be conceived within the context of the ‘solution’ suggested by the **Kitchen** pattern.

BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

WebApps Design

Design & WebApp Quality

- Security
  - Rebuff external attacks
  - Exclude unauthorized access
  - Ensure the privacy of users/customers
- Availability
  - the measure of the percentage of time that a WebApp is available for use
- Scalability
  - Can the WebApp and the systems with which it is interfaced handle significant variation in user or transaction volume
- Time to Market

WEBAPPS DESIGN

Quality Dimensions for End-Users

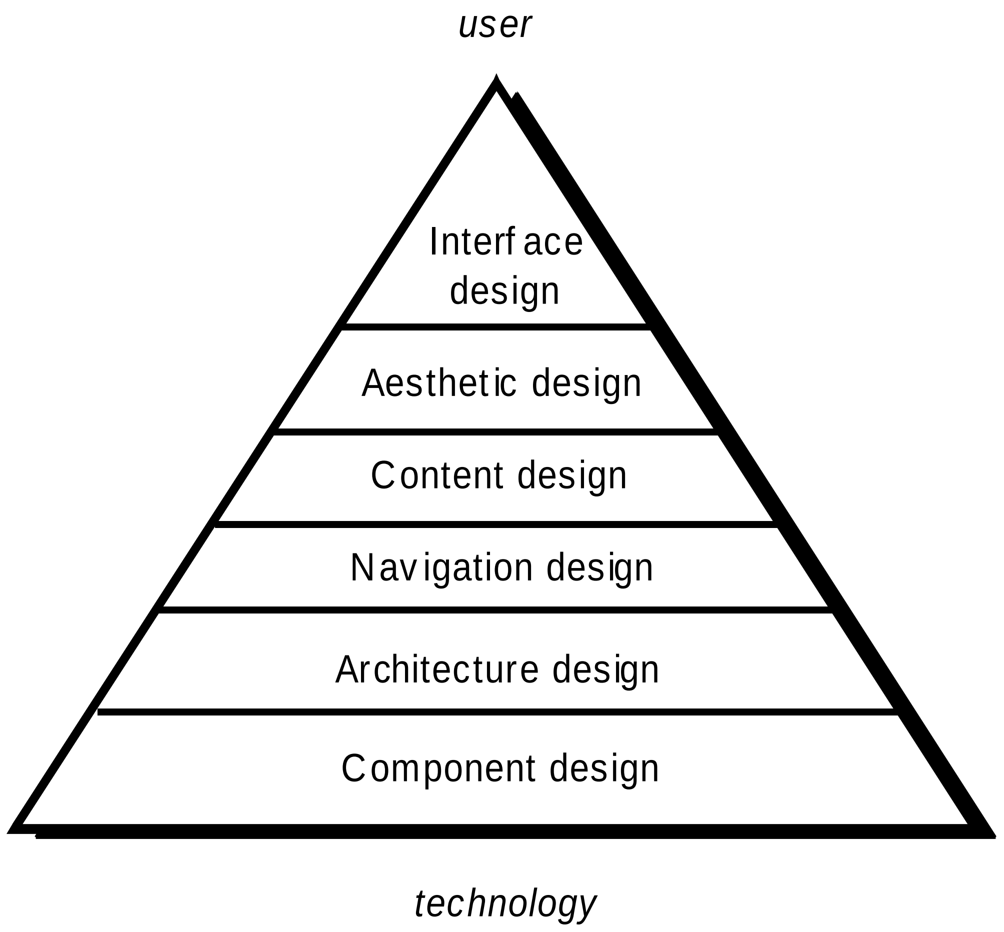
- *Time*
  - How much has a Web site changed since the last upgrade?
  - How do you highlight the parts that have changed?
- *Structural*
  - How well do all of the parts of the Web site hold together.
  - Are all links inside and outside the Web site working?
  - Do all of the images work?
  - Are there parts of the Web site that are not connected?
- *Content*
  - Does the content of critical pages match what is supposed to be there?
  - Do key phrases exist continually in highly-changeable pages?
  - Do critical pages maintain quality content from version to version?
  - What about dynamically generated HTML pages?

Quality Dimensions for End-Users

- *Accuracy and Consistency*
  - Are today's copies of the pages downloaded the same as yesterday's? Close enough?
  - Is the data presented accurate enough? How do you know?
- *Response Time and Latency*
  - Does the Web site server respond to a browser request within certain parameters?
  - In an E-commerce context, how is the end to end response time after a SUBMIT?
  - Are there parts of a site that are so slow the user declines to continue working on it?
- *Performance*
  - Is the Browser-Web-Web site-Web-Browser connection quick enough?
  - How does the performance vary by time of day, by load and usage?
  - Is performance adequate for E-commerce applications?

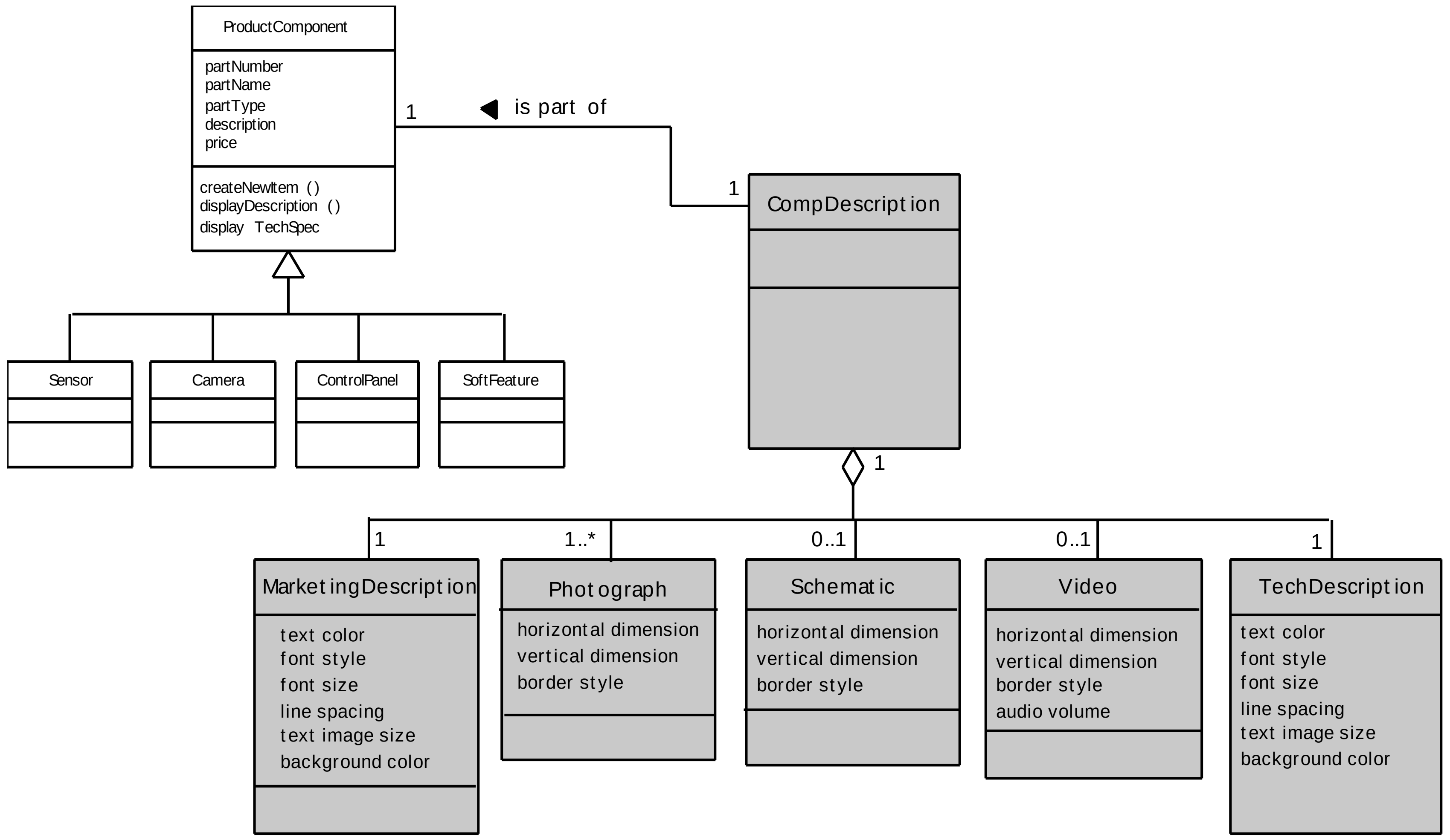
WebApps Design

Design Pyramid for WebApps



WebApps Design

Design representation of content objects



MOBILE APPS DESIGN

BINUS UNIVERSITY ONLINE LEARNING

MobileApp Design

Development Considerations

- Like all computing devices, mobile platforms are differentiated by the software they deliver – a combination of operating system (e.g., Android or iOS) and a small subset of the hundreds of thousands of MobileApps that provide a very wide range of functionality.
- New tools allow individuals with little formal training to create and sell apps alongside other apps developed by large teams of software developers.

People
Innovation
Excellence

MobileApp Design

Technical Considerations

- Multiple hardware and software platforms
- Many development frameworks and programming languages
- Many app store with different rules and tools
- Very short development cycles
- UI limitations and complexities of interaction with sensors and cameras
- Effective use of context
- Power management
- Security and privacy models and policies
- Computational and storage limitations
- Application that depend on external services
- Testing complexity

MobileApp Design

Best Practices

- Identify your audience
- Design for context of use
- There is a fine line between simlicity and laziness
- Use the platform as an advantage
- Make scrollbars and selection highlighting more salient
- Increase discoverability of advanced functionality
- Use clear and consistent labels
- Clever icons should never be developed at the expense of user under standing
- Support user expectations for personalization
- Long scrolling forms trump multiple screens on mobile device

References

- Pressman, R.S. (2015). *Software Engineering : A Practioner's Approach. 8th ed*. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157.
- Guide to the Software Engineering Body of Knowledge, SWEBOK, https://www.computer.org/education/bodies-of-knowledge/software-engineering
- UML Specification, https://www.omg.org/spec/UML/About-UML/
- Introduction to Software Architecture, http://www.youtube.com/watch?v=x30DcBfCJRI
- Component-based game engine, http://www.youtube.com/watch?v=_K4Mc3t9Rtc
- software design pattern, http://www.youtube.com/watch?v=ehGl_V61WJw

Q & A

BINUS UNIVERSITY ONLINE LEARNING

People Innovation Excellence

Thank You