MODULE 3

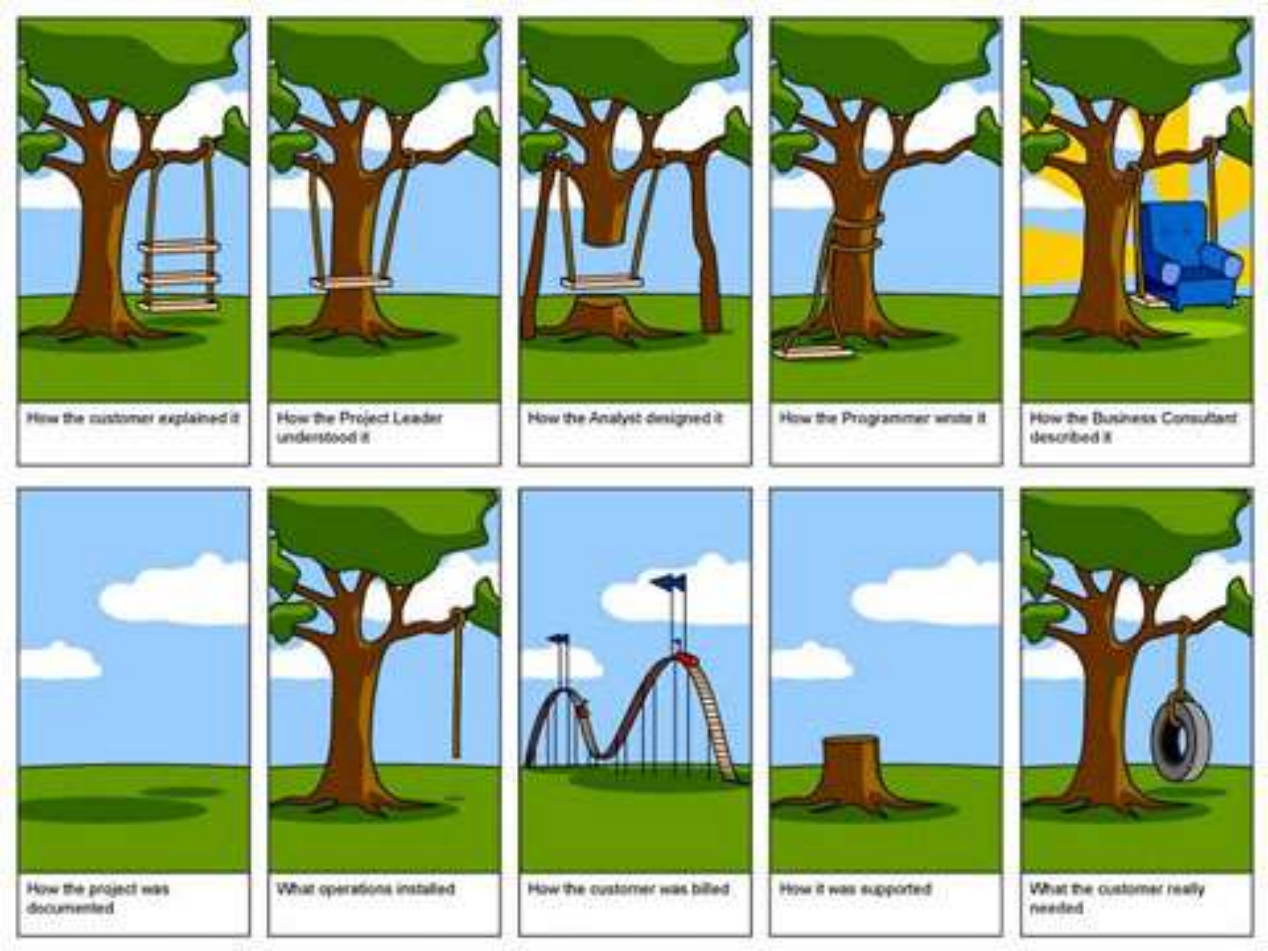
ARCHITECTURAL DESIGN

"ESTABLISHING THE OVERALL STRUCTURE OF A SOFTWARE SYSTEM"

2 CHAPTER OVERVIEW

- Understand the importance of Architectural Design
- Understand that different models may be required to document a software architecture
- Introduce architecture design process
  - **System Structure**
  - **Control Models**
  - **Modular decomposition**
  - **Domain Specific architectural**

3


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

4 SOFTWARE ARCHITECTURE

- *Architectural design* :It is the design process for
  - identifying the sub-systems making up a system and
  - establishing a framework for sub-system control and the communication between sub-systems.
- The output of this design process is a description of the *Software Architecture*

5 WHY ARCHITECTURE DESIGN?

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

- analyze the effectiveness of the design in meeting its stated requirements,
- consider various architectural alternatives (convergence) at a stage when making design changes is still relatively easy, and
- reduce the risks associated with the construction of the software.

6 WHY IS ARCHITECTURE IMPORTANT?

- Representations of software architecture are an enabler for communication between all parties (stakeholders) interested in the development of a CBS.
- The architecture highlights early design decisions that will have a profound impact on all software engineering work that follows and, as important, on the ultimate success of the system as an operational entity.
- Architecture "constitutes a relatively small, intellectually graspable model" of how the system is structured and how its components work together"

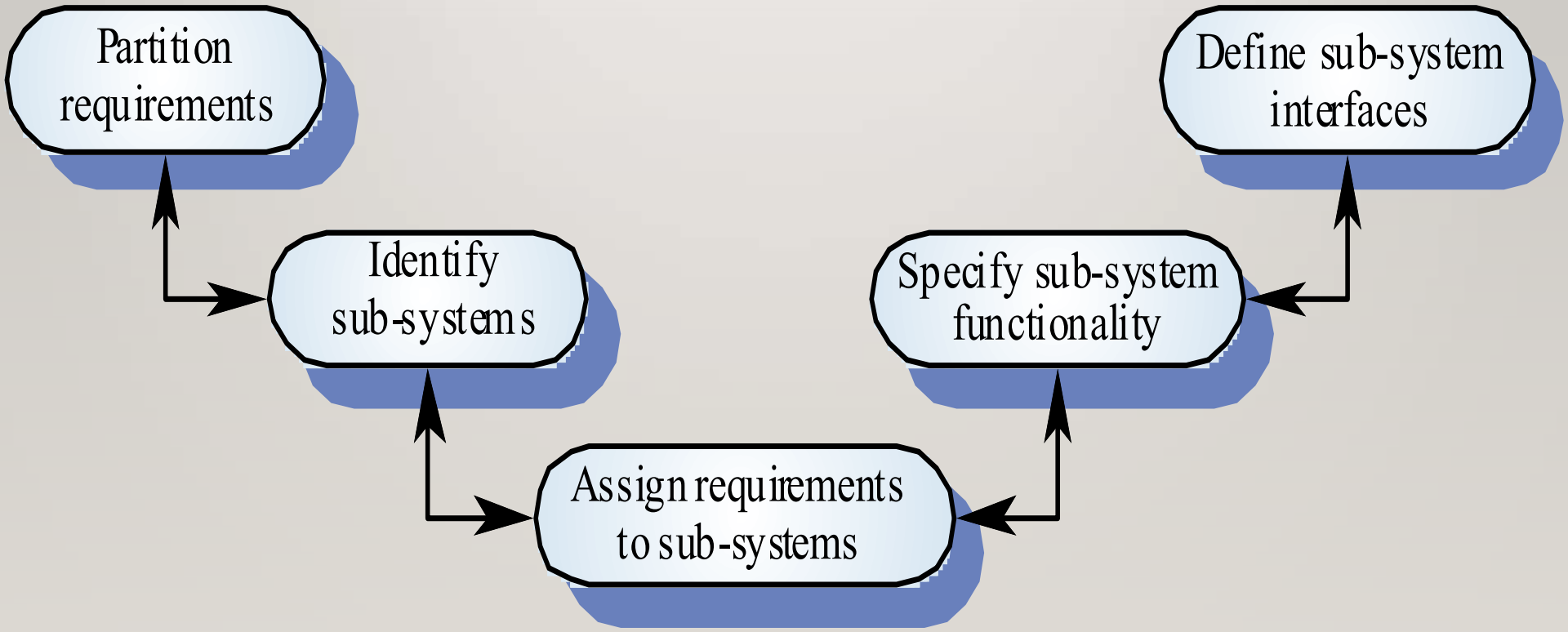
7 ADVANTAGES OF EXPLICIT ARCHITECTURE

- Stakeholder communication
  - Architecture may be used as a focus of discussion by system stakeholders
- System analysis
  - Means that analysis of whether the system can meet its non-functional requirements is possible
- Large-scale reuse
  - The architecture may be reusable across a range of systems

8 ARCHITECTURAL DESIGN PROCESS

- System structuring
  - The system is decomposed into several principal sub-systems and communications between these sub-systems are identified
- Control modelling
  - A model of the control relationships between the different parts of the system is established
- Modular decomposition
  - The identified sub-systems are decomposed into modules

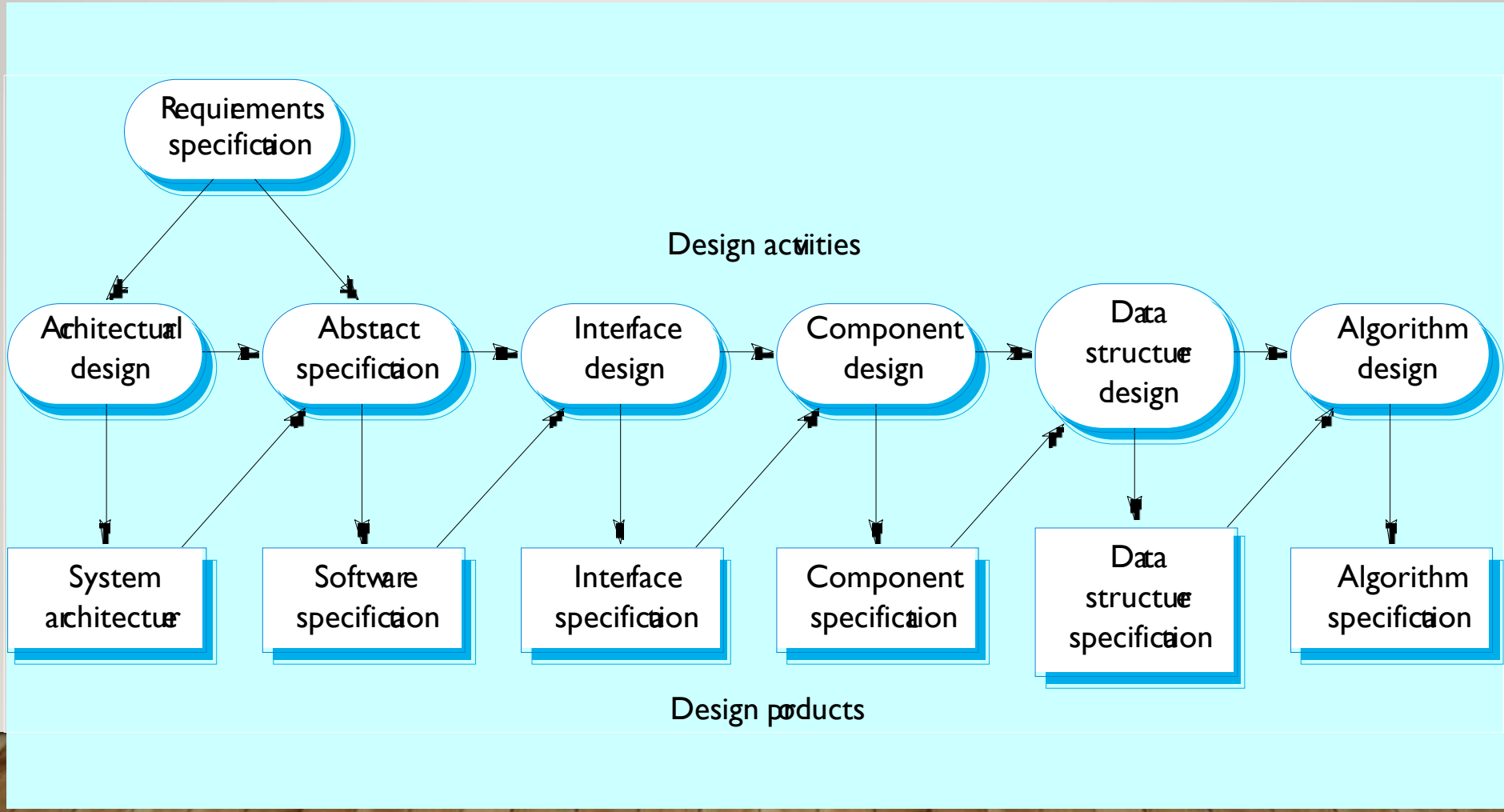
9 THE SYSTEM DESIGN PROCESS



10 THE SYSTEM DESIGN PROCESS

- Partition requirements
  - Organise requirements into related groups
- Identify sub-systems
  - Identify a set of sub-systems which collectively can meet the system requirements
- Assign requirements to sub-systems
  - Causes particular problems when COTS are integrated
- Specify sub-system functionality
- Define sub-system interfaces
  - Critical activity for parallel sub-system development

11 THE SOFTWARE DESIGN PROCESS



12 SYSTEM DESIGN PROBLEMS

- Requirements partitioning to hardware, software and human components may involve a lot of negotiation
- Difficult design problems are often assumed to be readily solved using software
- Hardware platforms may be inappropriate for software requirements so software must compensate for this

13 SUB-SYSTEMS AND MODULES

- A SUB-SYSTEM is a system in its own right whose operation is independent of the services provided by other sub-systems.
- A MODULE is a system component that provides services to other components but would not normally be considered as a separate system

DATA DESIGN –AT ARCHITECTURAL LEVEL …

- Design of one or more databases to support the application architecture
- Design of methods for 'mining' the content of multiple databases
  - navigate through existing databases in an attempt to extract appropriate business-level information
  - Design of a data warehouse — a large, independent database that has access to the data that are stored in databases that serve the set of applications required by a business

DATA DESIGN--AT THE COMPONENT LEVEL …

- refine data objects and develop a set of data abstractions
- implement data object attributes as one or more data structures
- review data structures to ensure that appropriate relationships have been established
- simplify data structures as required

16 DATA DESIGN—COMPONENT LEVEL

1. The ~~systematic analysis principles applied~~ to function and behavior should also be applied to data.
2. All data structures and the operations to be performed on each should be identified.
3. A data dictionary should be established and used to define both data and program design.
4. Low level data design decisions should be deferred until late in the design process.

A DESIGN—COMPONENT LEVEL CONT'D

5. The representation of data structure should be known only to those modules that must make direct use of the data contained within the structure.
6. A library of useful data structures and the operations that may be applied to them should be developed.
7. A software design and programming language should support the specification and realization of abstract data types.

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.

19

ARCHITECTURAL MODELS

- Different architectural models may be produced during the design process
- Each model presents different perspectives on the architecture

20 ARCHITECTURAL MODELS

- Static structural model
  - that shows the major system components
- Dynamic process model
  - that shows the process structure of the system
- Interface model
  - that defines sub-system interfaces
- Relationships model
  - such as a data-flow model defines the flow of data

SYSTEM DESIGN PROCESS

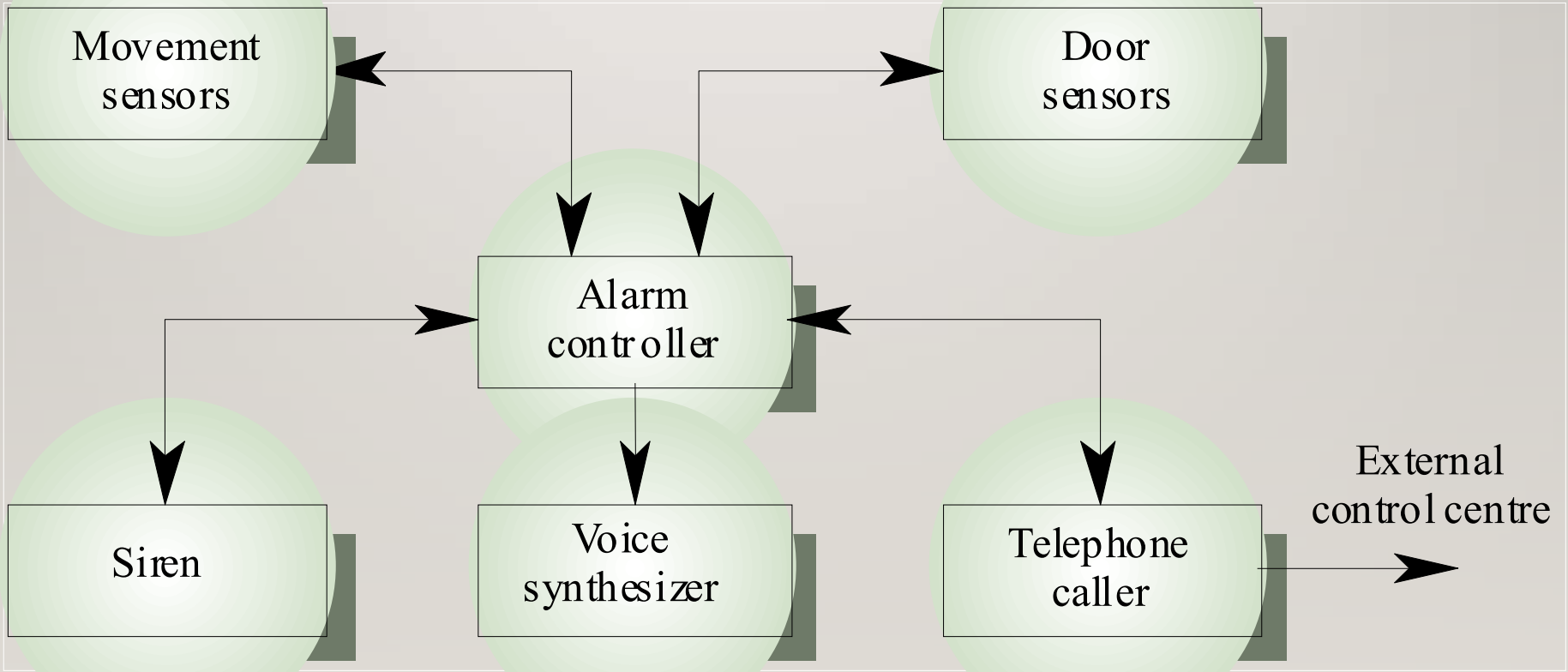
- Different types of software architecture covering
  - System Structure
  - Control Models
  - Modular decomposition
  - Domain Specific architectural

22 SYSTEM STRUCTURE MODELLING

- An architectural model presents an abstract view of the sub-systems making up a system
- May include major information flows between sub-systems
- Usually presented as a block diagram
- May identify different types of functional component in the model

23

INTRUDER ALARM SYSTEM



COMPONENT TYPES IN ALARM SYSTEM

- Sensor
  - Movement sensor, door sensor
- Actuator
  - Siren
- Communication
  - Telephone caller
- Co-ordination
  - Alarm controller
- Interface
  - Voice synthesizer

25

SYSTEM STRUCTURING

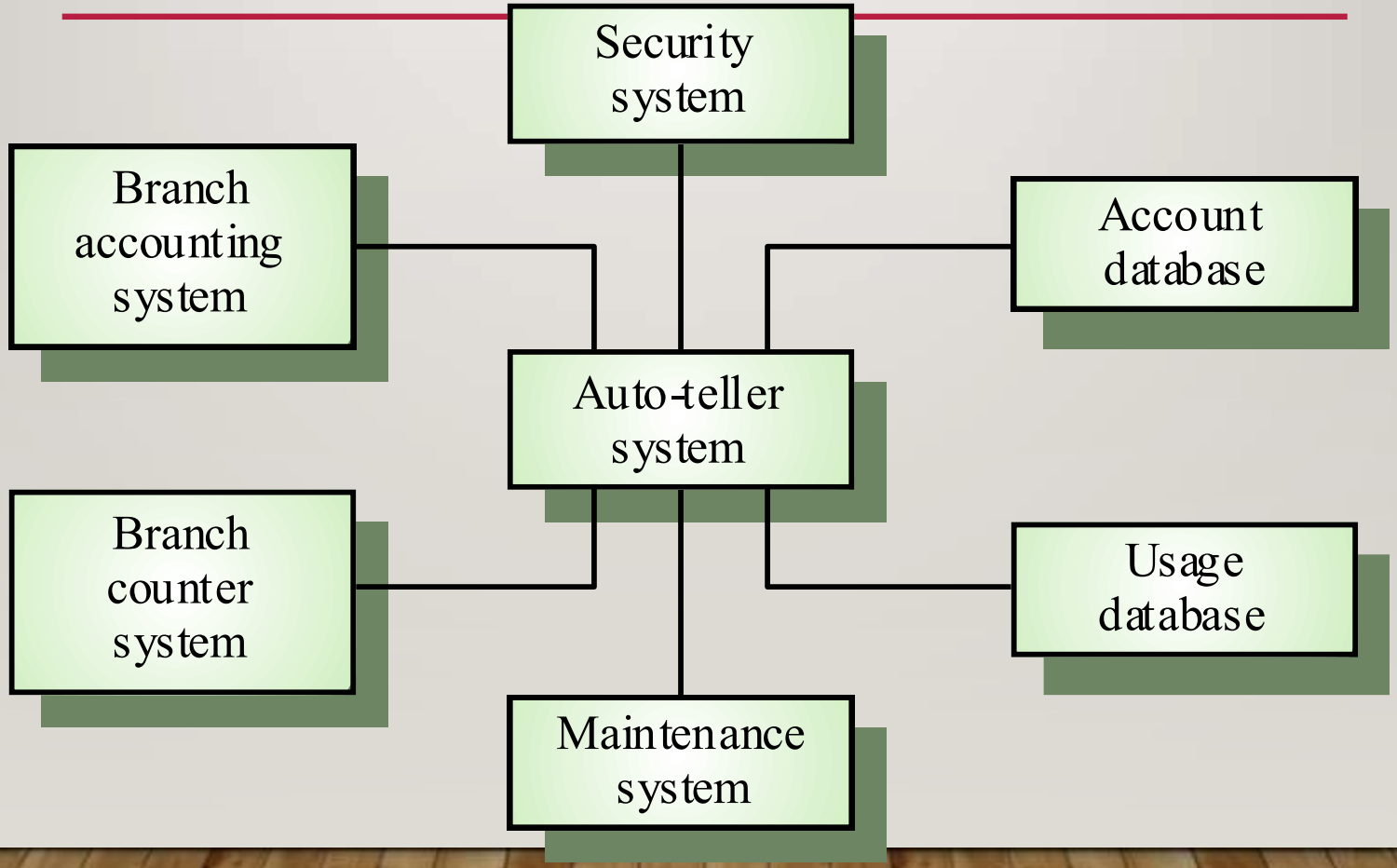
- Concerned with decomposing the system into interacting sub-systems
- The architectural design is normally expressed as a block diagram presenting an overview of the system structure
- More specific models showing how sub-systems share data, are distributed and interface with each other may also be developed

26 CONTEXT MODELS OF ANALYSIS

- Context models are used to illustrate the boundaries of a system
- Social and organisational concerns may affect the decision on where to position system boundaries
- Architectural models show the a system and its relationship with other systems

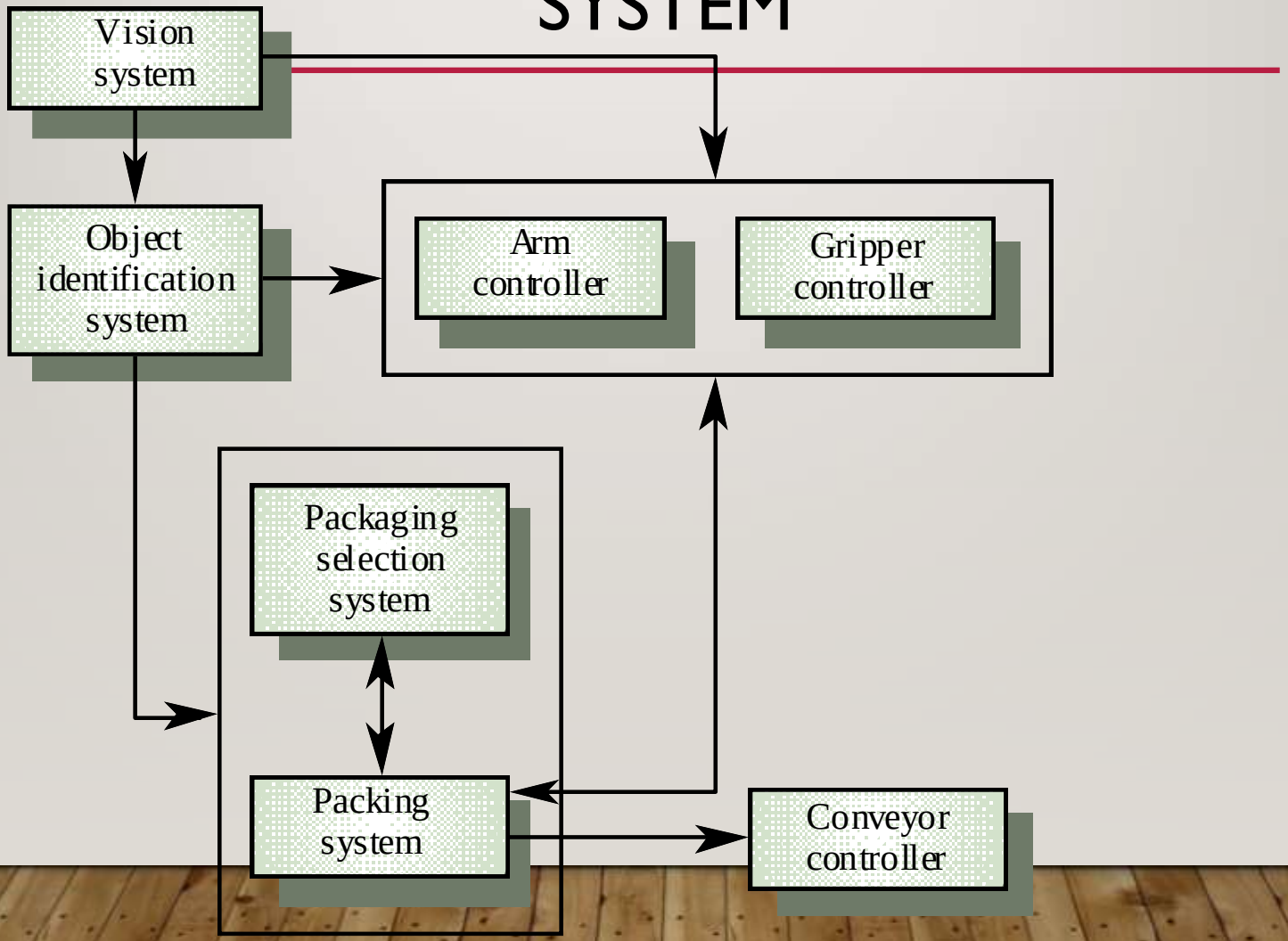
27

THE CONTEXT OF AN ATM SYSTEM



28

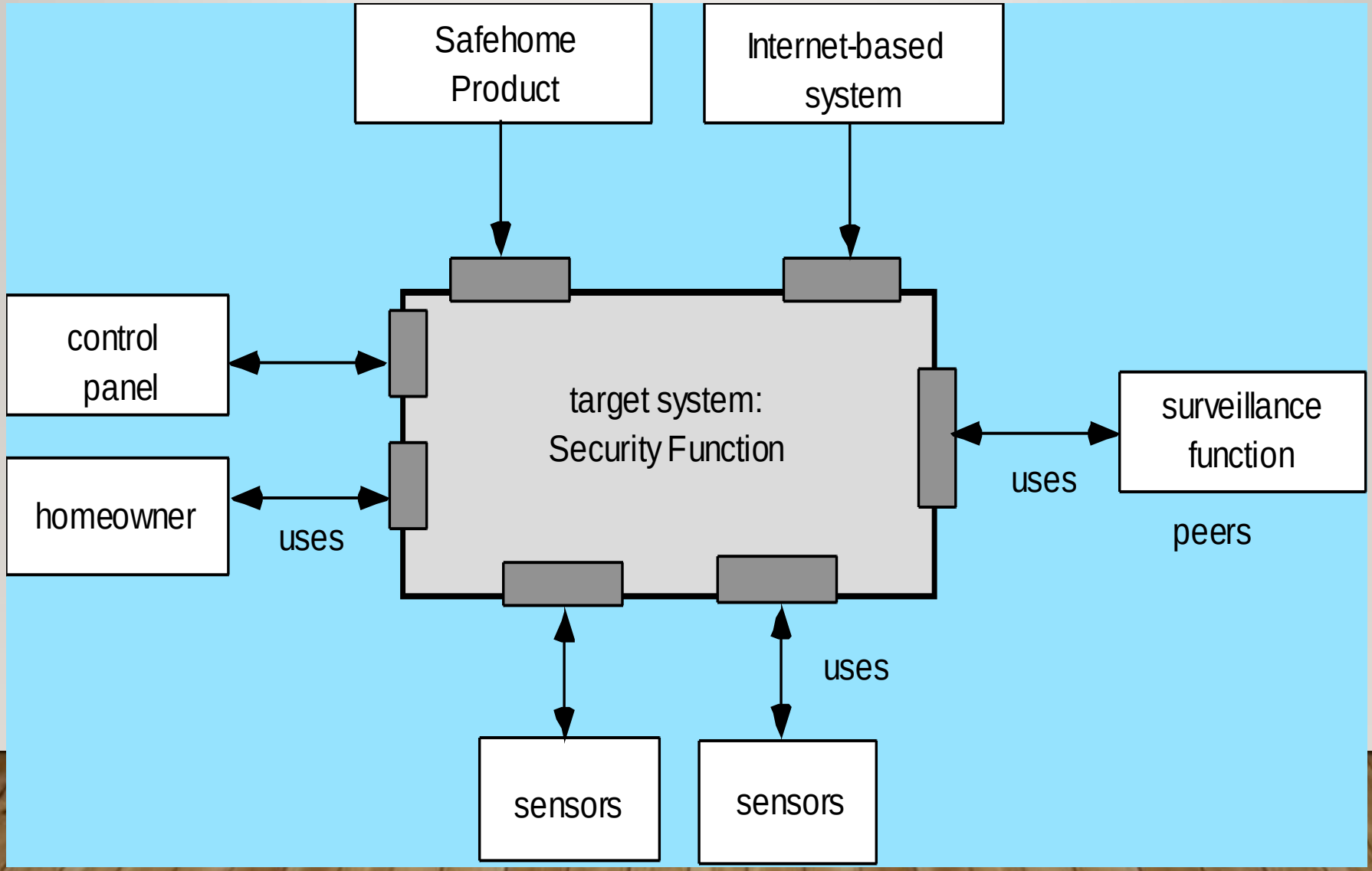
PACKING ROBOT CONTROL SYSTEM



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 CONTEXT



ARCHETYPES

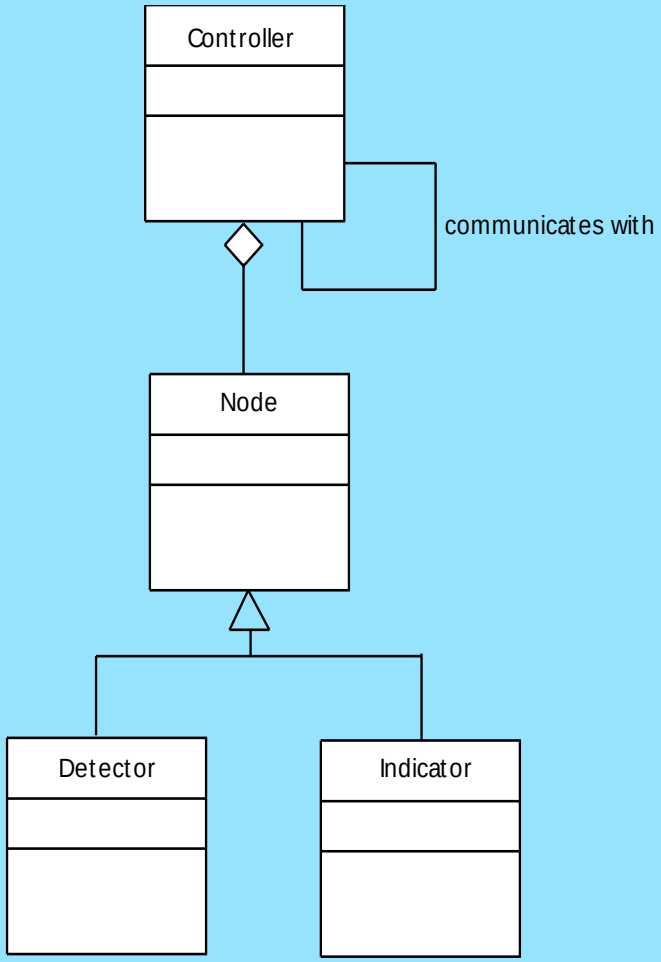


Figure 10.7 UML relationships for SafeHome security function archetypes (adapted from [BOS00])

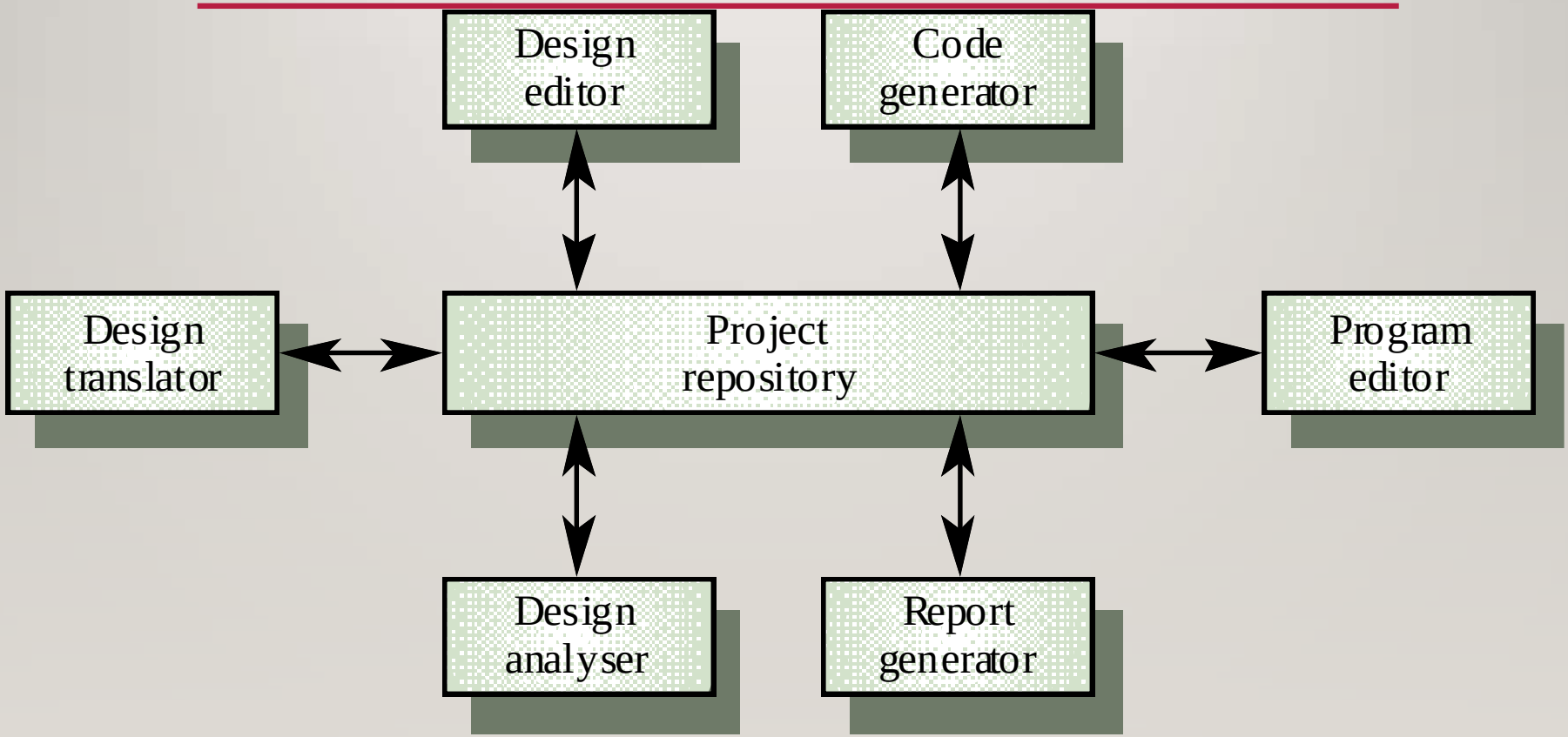
SYSTEM STRUCTURING

- Specific Models of the System Structuring
  - 10.1.1 The Repository Model
  - 10.1.2 The Client-Server Model
  - 10.1.3 The abstract Machine Model

33 10.1.1 THE REPOSITORY MODEL

- Sub-systems must exchange data. This may be done in two ways:
  - Shared data is held in a central database or repository and may be accessed by all sub-systems
  - Each sub-system maintains its own database and passes data explicitly to other sub-systems
- When large amounts of data are to be shared, the repository model of sharing is most commonly used

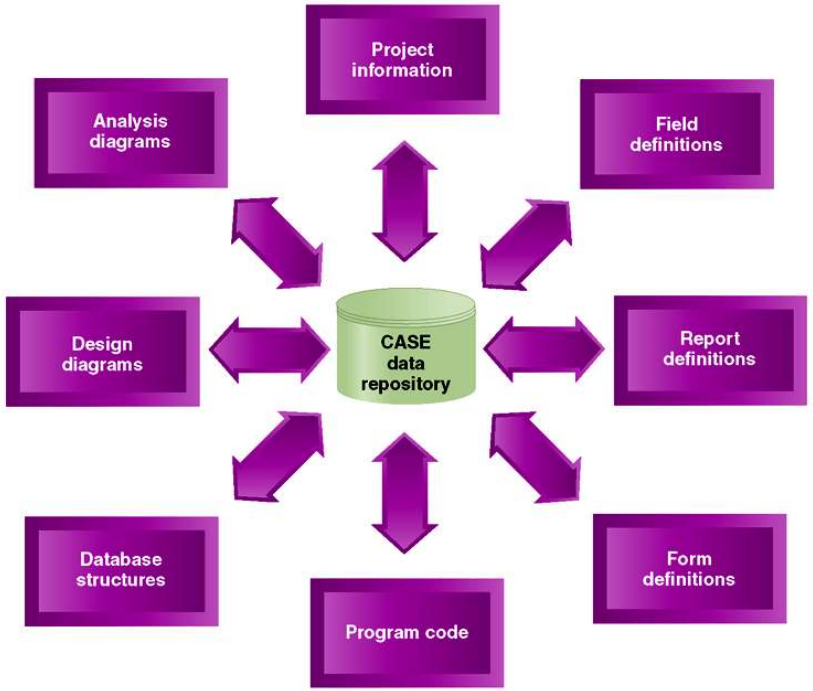
34 CASE TOOLSET ARCHITECTURE



SYSTEM DEVELOPMENT INFORMATION STORED IN THE CASE REPOSITORY

FIGURE 9-6

System development information stored in the CASE repository.



PROJECT MANAGEMENT: COORDINATING THE PROJECT

- Coordinating Project Teams
  - Project schedule - coordinating ongoing work
- The Project Team at RMO
  - As project team grows – structure may change
- Coordinating Information
  - CASE tools and central repository
  - Team communication and information coordination
  - Track open items and unresolved issues

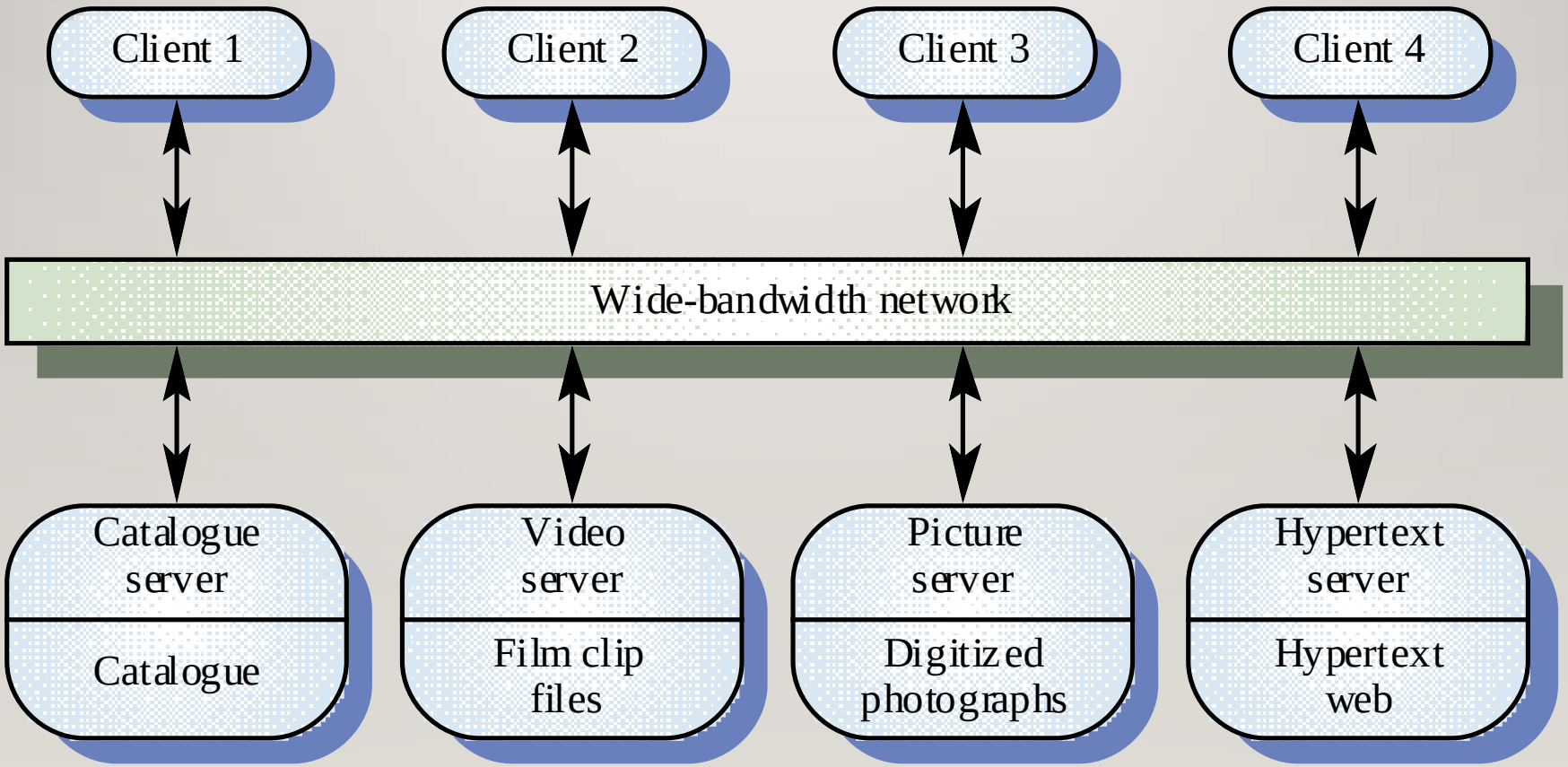
REPOSITORY MODEL CHARACTERISTICS

- Advantages
  - Efficient way to share large amounts of data
  - Sub-systems need not be concerned with HOW data is produced in Centralised management
  - Sharing model is published as the repository schema
- Disadvantages
  - Sub-systems must agree on a repository data model. Inevitably a compromise
  - Data evolution is difficult and expensive
  - No scope for specific management policies
  - Difficult to distribute efficiently

10.1.2 THE CLIENT-SERVER ARCHITECTURE MODEL

- Distributed system model which shows how data and processing is distributed across a range of components
- Set of stand-alone servers which provide specific services such as printing, data management, etc.
- Set of clients which call on these services
- Network which allows clients to access servers

39 FILM AND PICTURE LIBRARY



40 CLIENT-SERVER ARCHITECTURE

- Client-Server divides programs into two types
- Server – manages information system resources or provides well defined services for client
- Client – communicates with server to request resources or services
- Advantage – Deployment flexibility
  - Location, scalability, maintainability
- Disadvantage – Potential performance, security, and reliability issues from network communication

INTERACTION AMONG CLIENT, SERVER, AND A SERVICE-RELATED DATA STORE

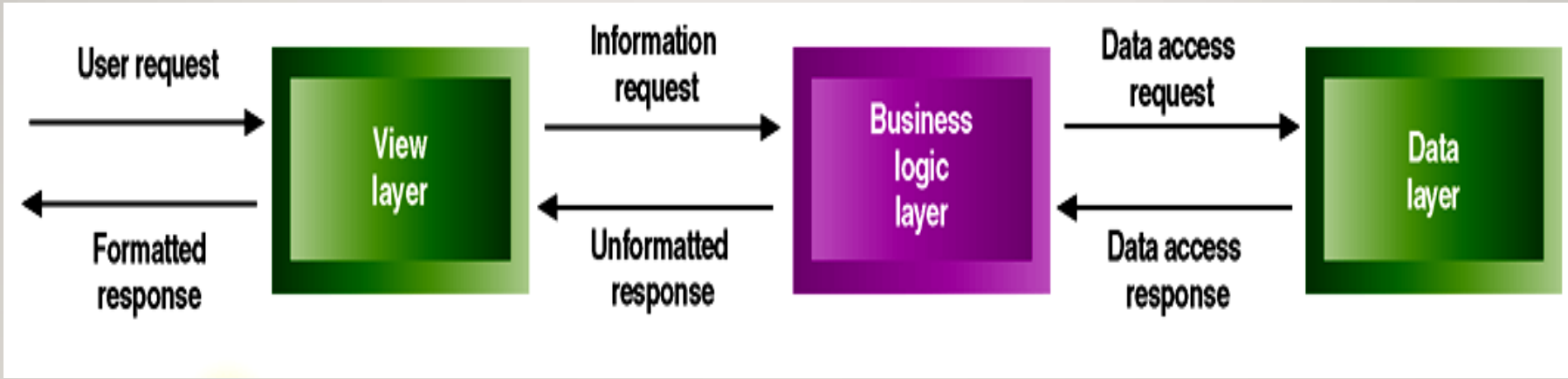
42 CLIENT-SERVER ARCHITECTURAL PROCESS

- Decompose application into client and server programs, modules, or objects
  - Identify resources or services that can be centrally managed by independent software units
- Determine which clients and servers will execute on which computer systems
- Describe communication protocols and networks that connect clients and servers

THREE-LAYER CLIENT-SERVER ARCHITECTURE

- Layers can reside on one processor or be distributed to multiple processors
- Data layer – manages stored data in databases
- Business logic layer – implements rules and procedures of business processing
- View layer – accepts user input and formats and displays processing results

44 THREE-LAYER ARCHITECTURE



MIDDLEWARE

- Aspect of distributed computing
- Connects parts of an application and enables requests and data to pass between them
- Teleprocessing monitors, transaction processing modules, object request brokers (ORBs)
- Designers reply on standard frameworks and protocols incorporated into middleware

INTERNET AND WEB-BASED APPLICATION ARCHITECTURE

- Web is complex example of client-server architecture
- Can use Web protocols and browsers as application interfaces
- Benefits
  - Accessibility
  - Low-cost communication
  - Widely implemented standards

47

NEGATIVE ASPECTS OF INTERNET APPLICATION DELIVERY

- Breaches of security
- Fluctuating reliability of network throughput
- Slow, throughput speeds to home users
- Volatile, changing standards

NETWORK DESIGN

- Integrate network needs of new system with existing network infrastructure
- Describe processing activity and network connectivity at each system location
- Describe communications protocols and middleware that connects layers
- Ensure that network capacity is sufficient
  - Data size per access type and average
  - Peak number of access per minute or hour

49

NETWORK DIAGRAM FOR TSN CUSTOMER SUPPORT SYSTEM

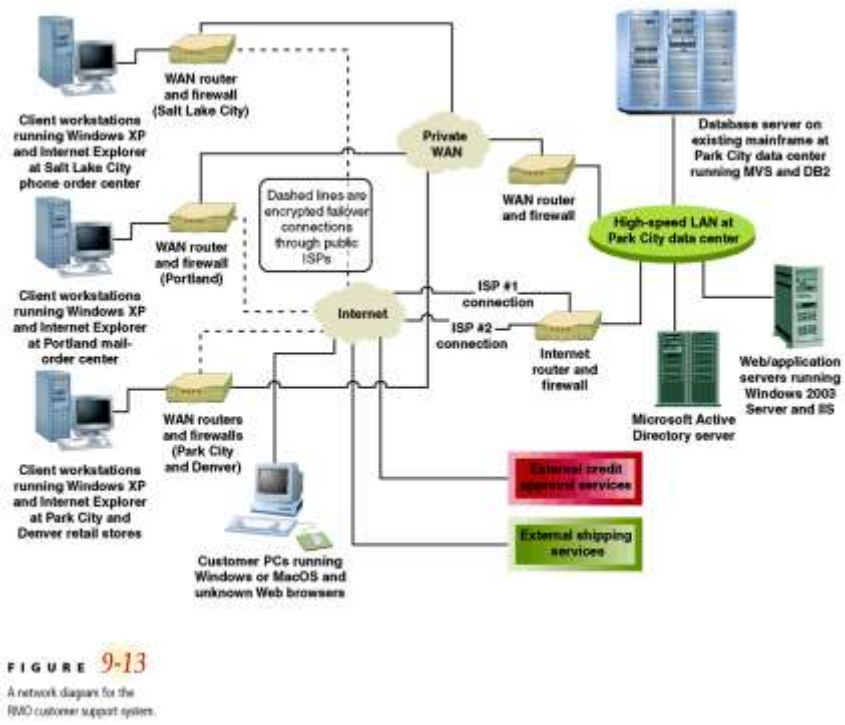


FIGURE 9-13
A network diagram for the RMO customer support system.

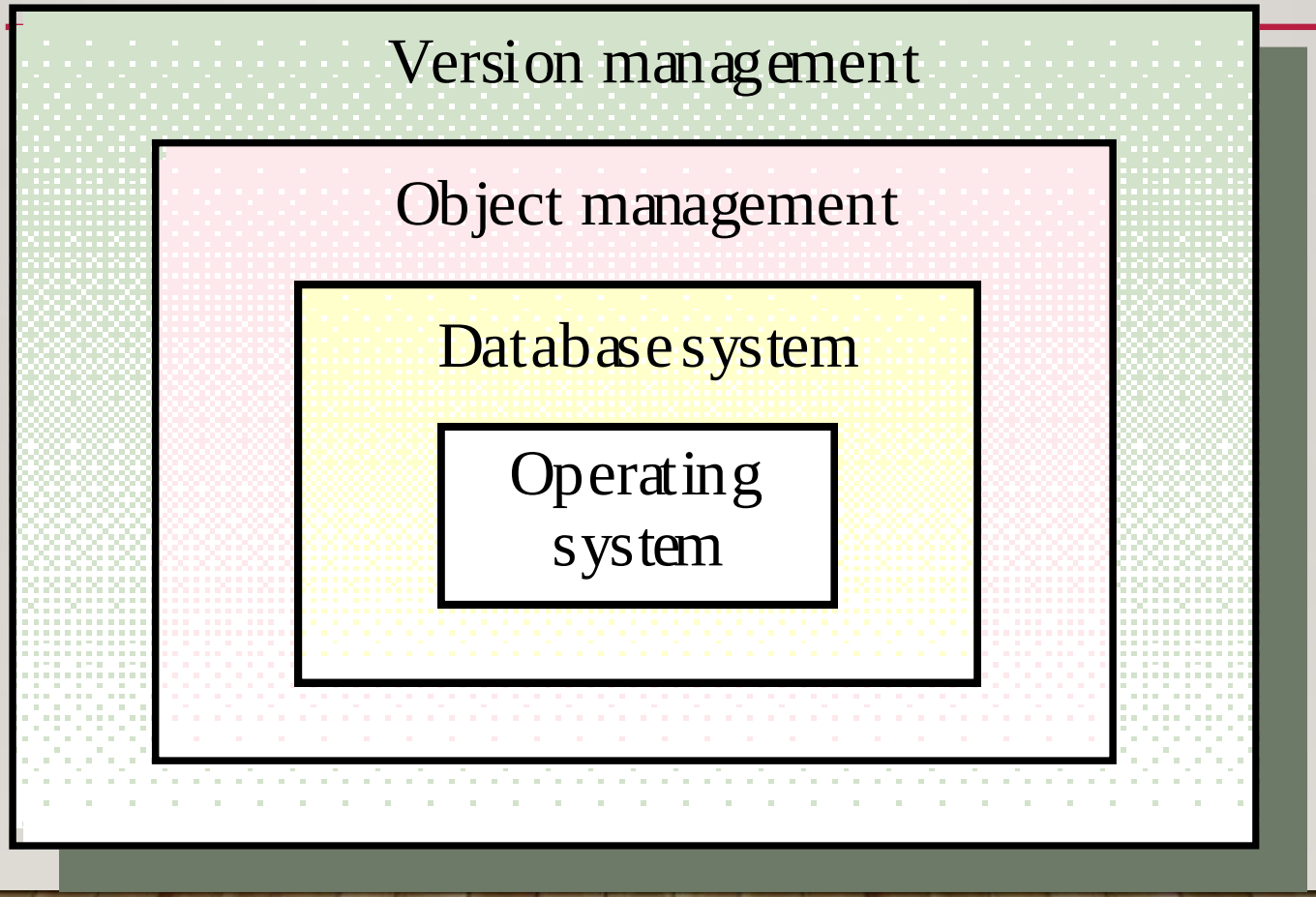
CLIENT-SERVER CHARACTERISTICS

- Advantages
  - Distribution of data is straightforward
  - Makes effective use of networked systems. May require cheaper hardware
  - Easy to add new servers or upgrade existing servers
- Disadvantages
  - No shared data model so sub-systems use different data organisation. data interchange may be inefficient
  - Redundant management in each server
  - No central register of names and services - it may be hard to find out what servers and services are available

10.1.3 ABSTRACT MACHINE MODEL

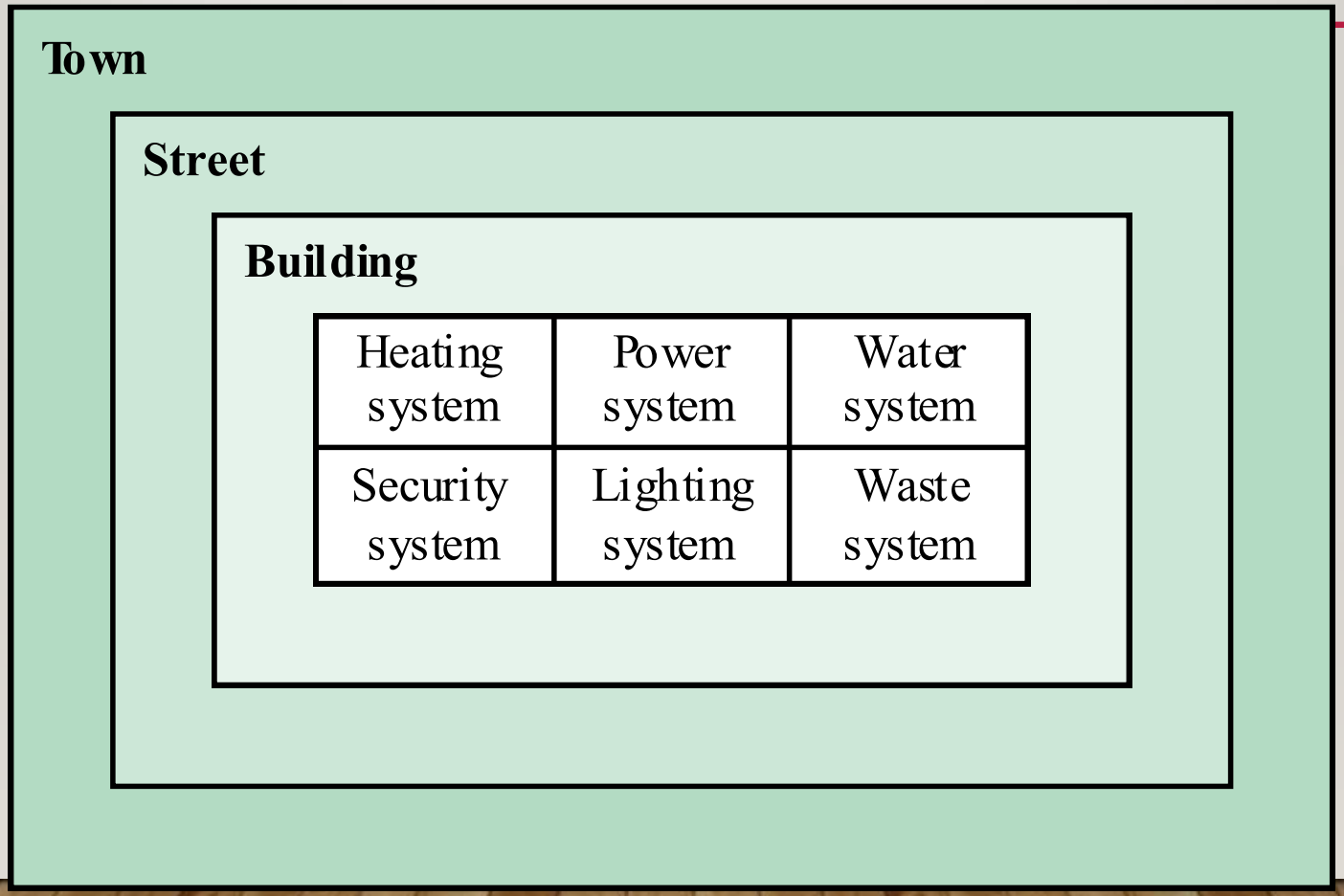
- Used to model the interfacing of sub-systems
- Organises the system into a set of layers (or abstract machines) each of which provide a set of services
- Supports the incremental development of sub-systems in different layers. When a layer interface changes, only the adjacent layer is affected
- However, often difficult to structure systems in this way

52 VERSION MANAGEMENT SYSTEM

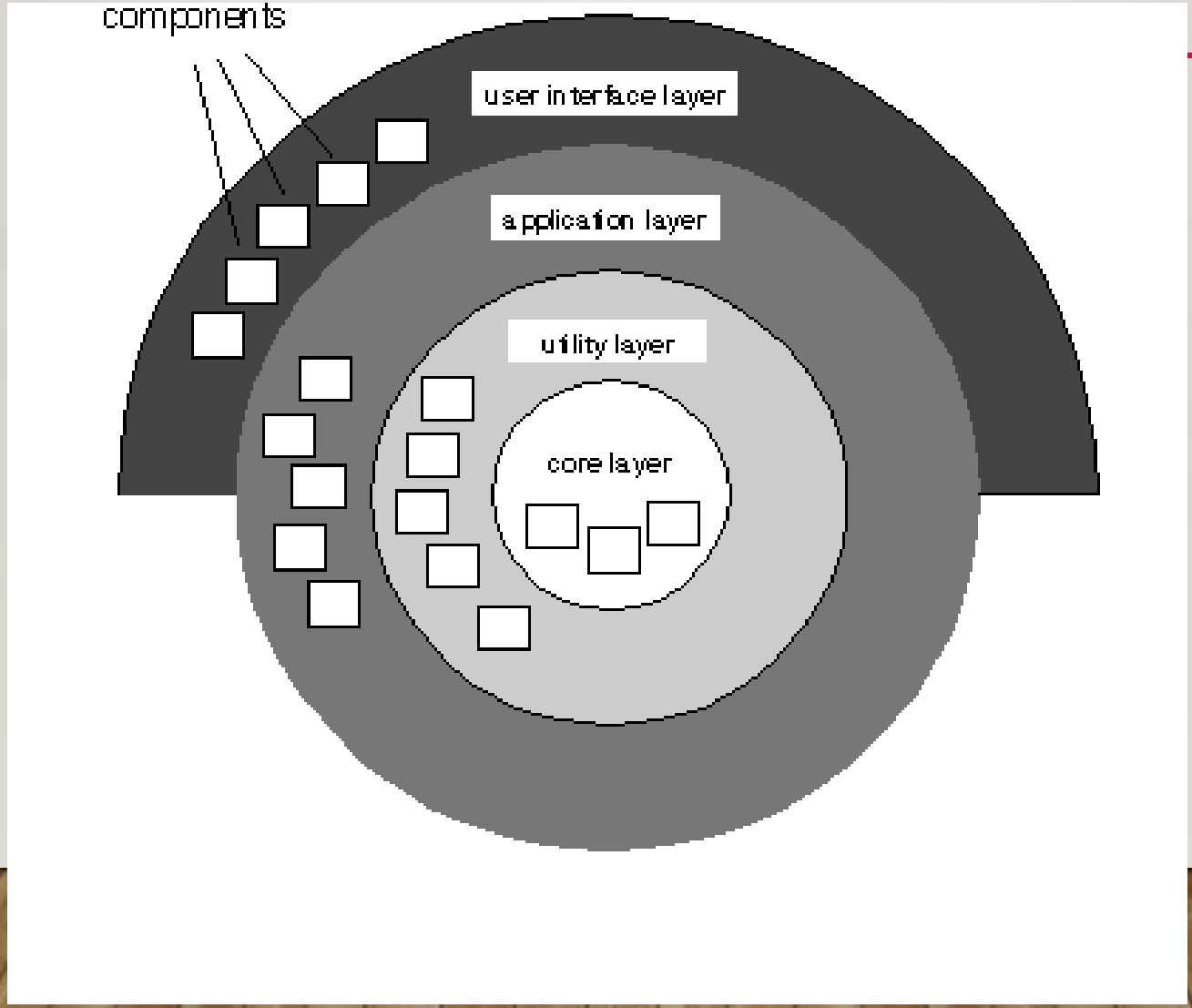


53

SYSTEM HIERARCHIES



LAYERED ARCHITECTURE



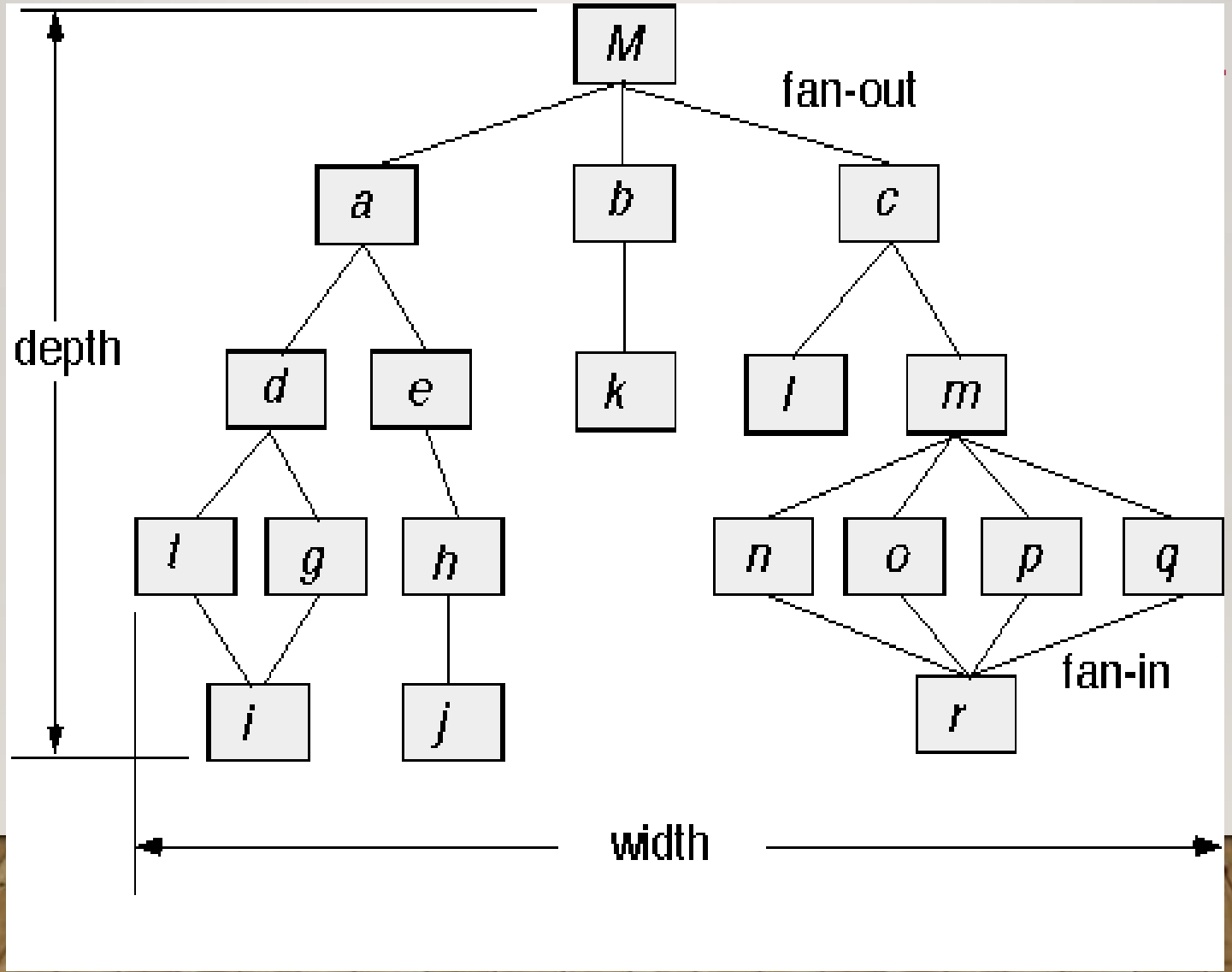
10.2 CONTROL MODELS

- Are concerned with the flow of control between sub-systems.
- Distinct from the system decomposition model
- Centralised control
  - One sub-system has overall responsibility for control and starts and stops other sub-systems
- Event-based control
  - Each sub-system can respond to externally generated events from other sub-systems or the system's environment

10.2 CONTROL MODELS

- 10.2.1 Centralized Control
  - The Call-return model
  - The Manager model
- 10.2.2 Event-driven-Systems
  - Broadcast models
  - Interrupt-driven models

CALL AND RETURN ARCHITECTURE



CALL AND RETURN ARCHITECTURE

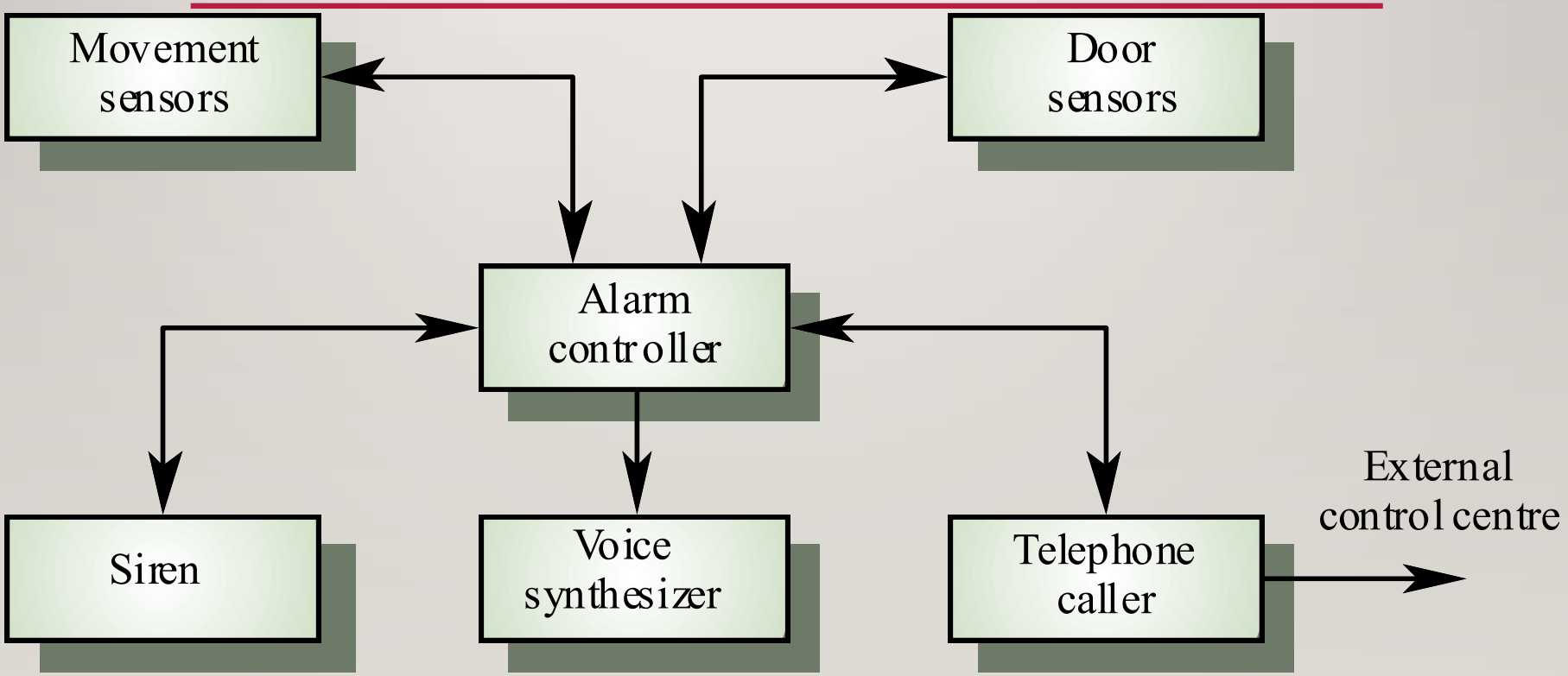
- Similar to top-down subroutine model where control starts at the top of a subroutine hierarchy and through subroutine calls, passes to the lower levels, and return control to its parent.
- Applicable only to sequential systems.
- This model may be used at module level to control functions or objects.
- This model is embedded in programming languages such as Ada, Pascal, c

THE MANAGER MODEL

- One system component is designated as a system manager and controls the starting, stopping and coordination of other system processes.
- Applicable to concurrent systems where a subsystem or module can execute in parallel with other processes.
- Usually implemented as CASE statement.

60

INTRUDER ALARM SYSTEM



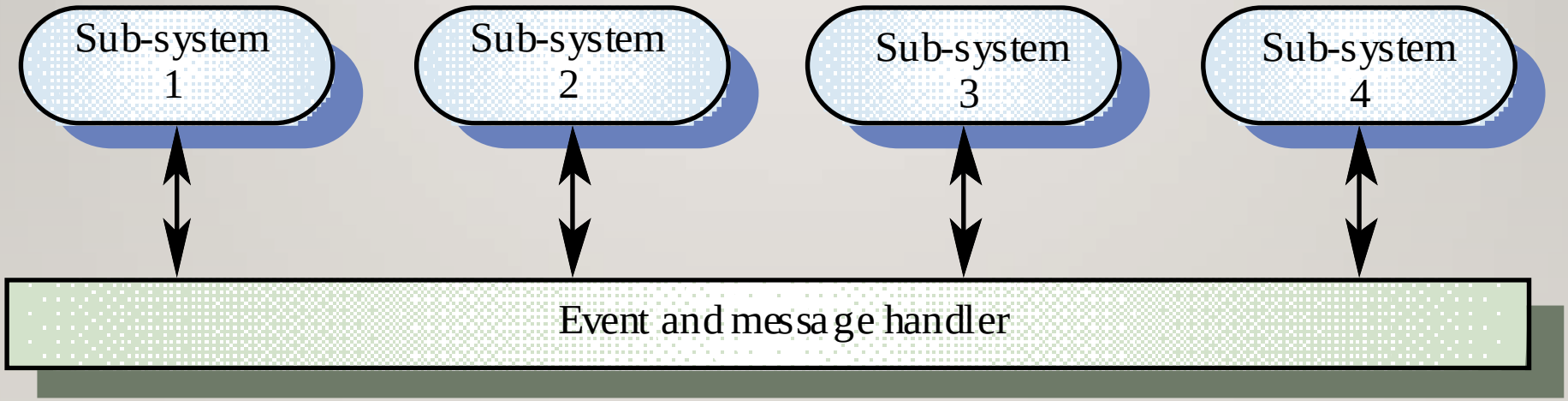
61 10.2.2 EVENT-DRIVEN SYSTEMS

- Driven by externally generated events where the timing of the event is out with the control of the sub-systems which process the event
- Two principal event-driven models
  - **Broadcast models**. An event is broadcast to all sub-systems. Any sub-system which can handle the event may do so
  - **Interrupt-driven models.** Used in real-time systems where interrupts are detected by an interrupt handler and passed to some other component for processing
- Other event driven models include spreadsheets and production systems

62 10.2.2.1 BROADCAST MODEL

- Effective in integrating sub-systems on different computers in a network
- Sub-systems register an interest in specific events. When these occur, control is transferred to the sub-system which can handle the event
- Control policy is not embedded in the event and message handler. Sub-systems decide on events of interest to them
- However, sub-systems don't know if or when an event will be handled

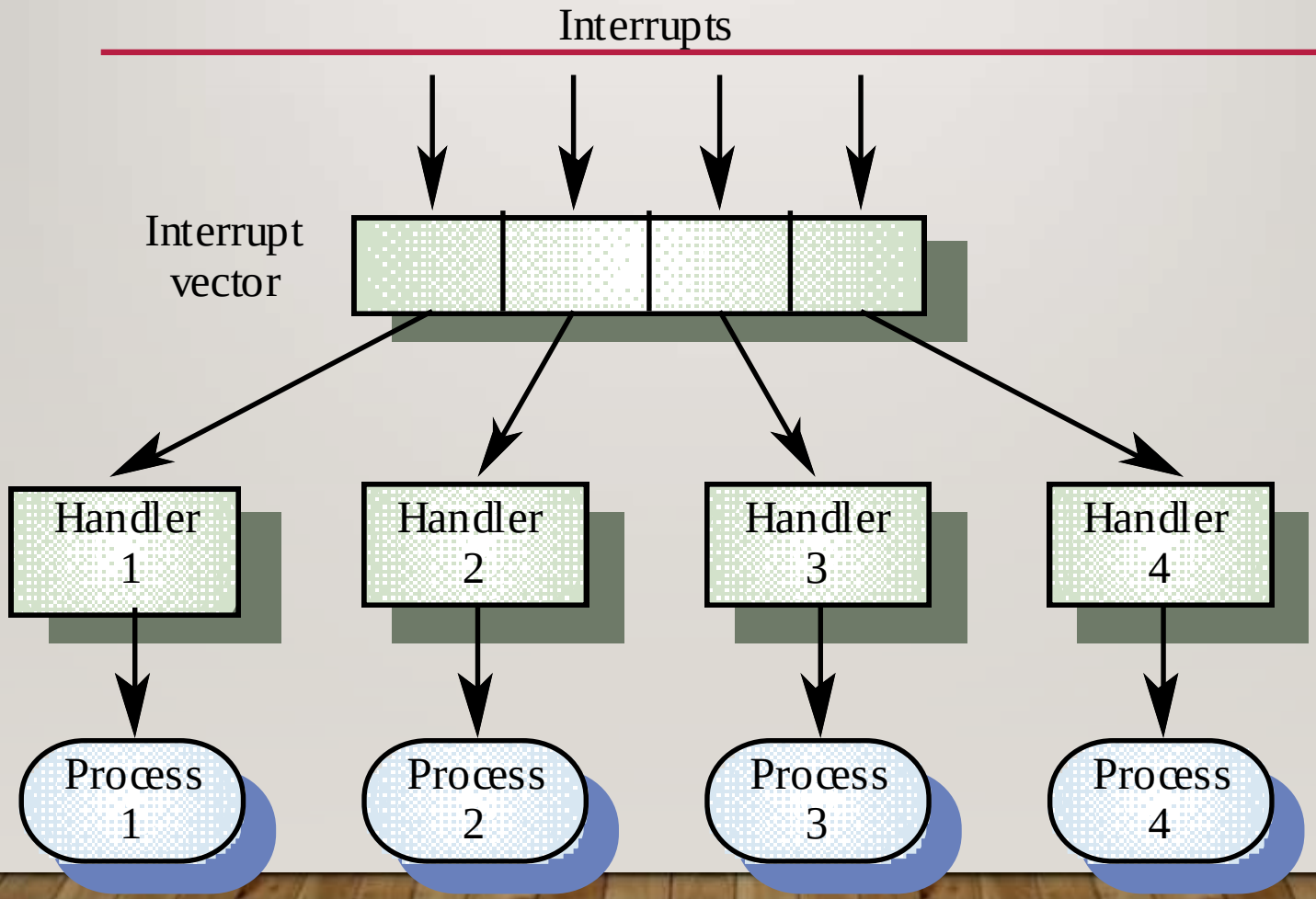
SELECTIVE BROADCASTING



10.2.2.2 INTERRUPT-DRIVEN SYSTEMS

- Used in real-time systems where fast response to an event is essential
- There are known interrupt types with a handler defined for each type
- Each type is associated with a memory location and a hardware switch causes transfer to its handler
- Allows fast response but complex to program and difficult to validate

65 INTERRUPT-DRIVEN CONTROL



66 10.3 MODULAR DECOMPOSITION

- Another structural level where sub-systems are decomposed into modules
- Two modular decomposition models covered
  - An object model where the system is decomposed into interacting objects
  - A data-flow model where the system is decomposed into functional modules which transform inputs to outputs. Also known as the pipeline model
- If possible, decisions about concurrency should be delayed until modules are implemented

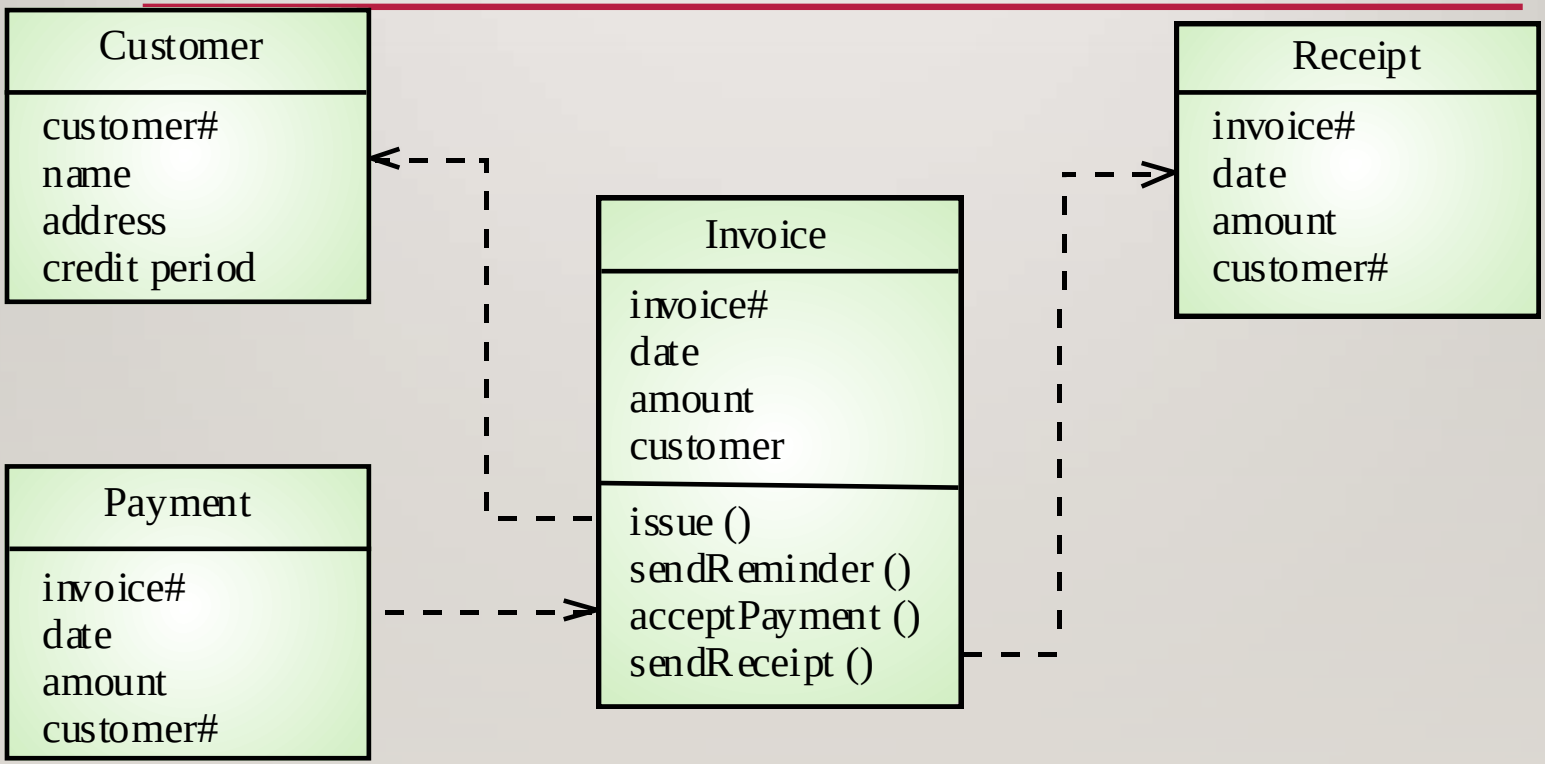
10.3 MODULAR DECOMPOSITION

- 10.3.1 Object Models
- 10.3.2 Data-Flow models

10.3.1 OBJECT MODELS

- Structure the system into a set of loosely coupled objects with well-defined interfaces
- Object-oriented decomposition is concerned with identifying object classes, their attributes and operations
- When implemented, objects are created from these classes and some control model used to coordinate object operations

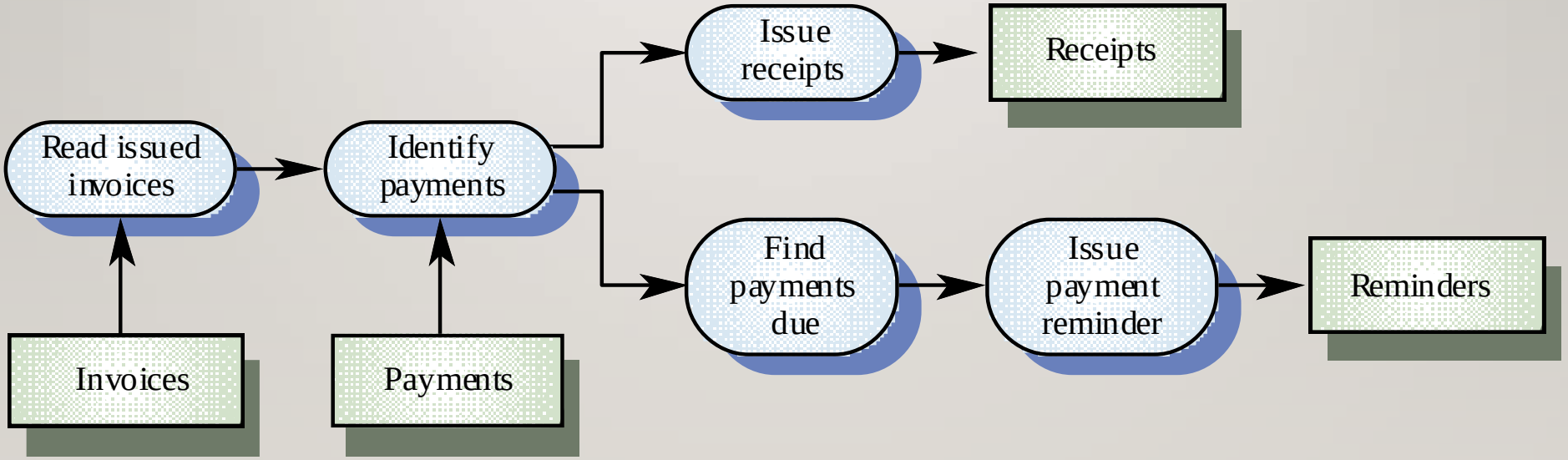
69 INVOICE PROCESSING SYSTEM



10.3.2 DATA-FLOW MODELS

- Functional transformations process their inputs to produce outputs
- May be referred to as a pipe and filter model (as in UNIX shell)
- Variants of this approach are very common. When transformations are sequential, this is a batch sequential model which is extensively used in data processing systems
- Not really suitable for interactive systems

71 INVOICE PROCESSING SYSTEM



ARCHITECTURAL PATTERNS

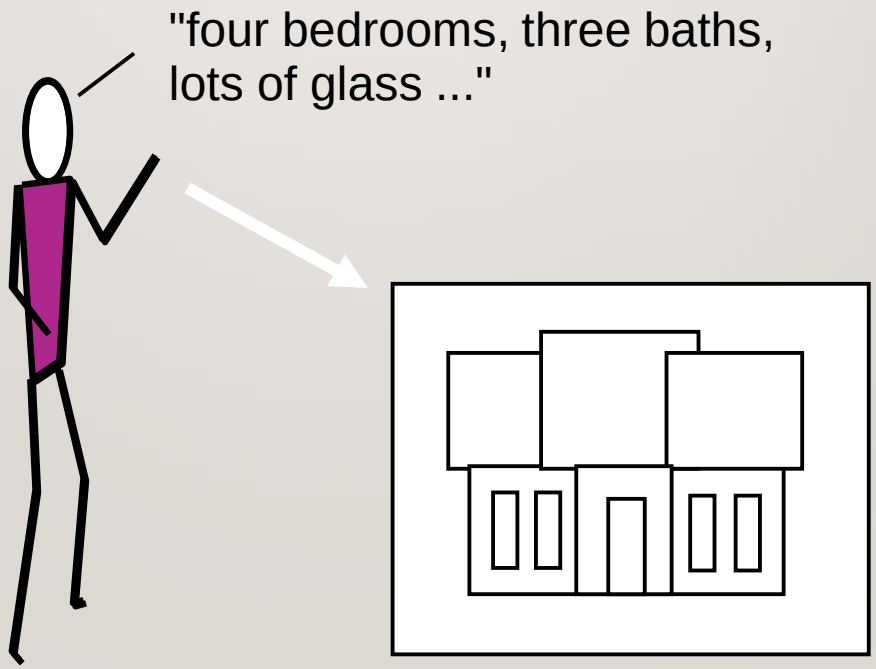
- Concurrency—applications must handle multiple tasks in a manner that simulates parallelism
  - *operating system process management* pattern
  - *task scheduler* pattern
- Persistence—Data persists if it survives past the execution of the process that created it. Two patterns are common:
  - a *database management system* pattern that applies the storage and retrieval capability of a DBMS to the application architecture
  - an *application level persistence* pattern that builds persistence features into the application architecture
- Distribution— the manner in which systems or components within systems communicate with one another in a distributed environment
  - A *broker* acts as a 'middle-man' between the client component and a server component.

73 ANALYZING ARCHITECTURAL DESIGN

1. Collect scenarios.
2. Elicit requirements, constraints, and environment description.
3. Describe the architectural styles/patterns that have been chosen to address the scenarios and requirements:
   - module view
   - process view
   - data flow view
4. Evaluate quality attributes by considering each attribute in isolation.
5. Identify the sensitivity of quality attributes to various architectural attributes for a specific architectural style.
6. Critique candidate architectures (developed in step 3) using the sensitivity analysis conducted in step 5.

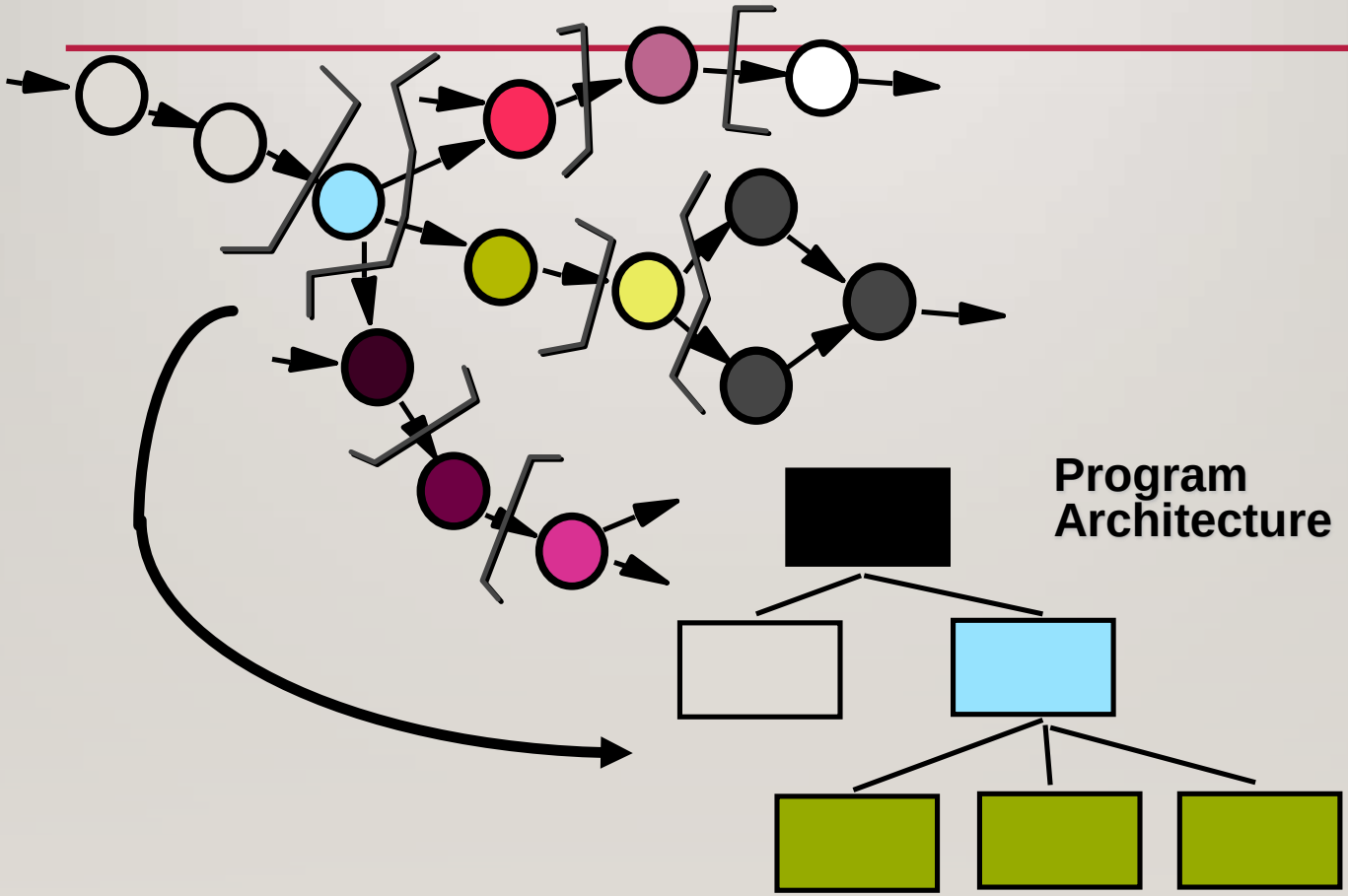
N ARCHITECTURAL DESIGN METHOD

customer requirements



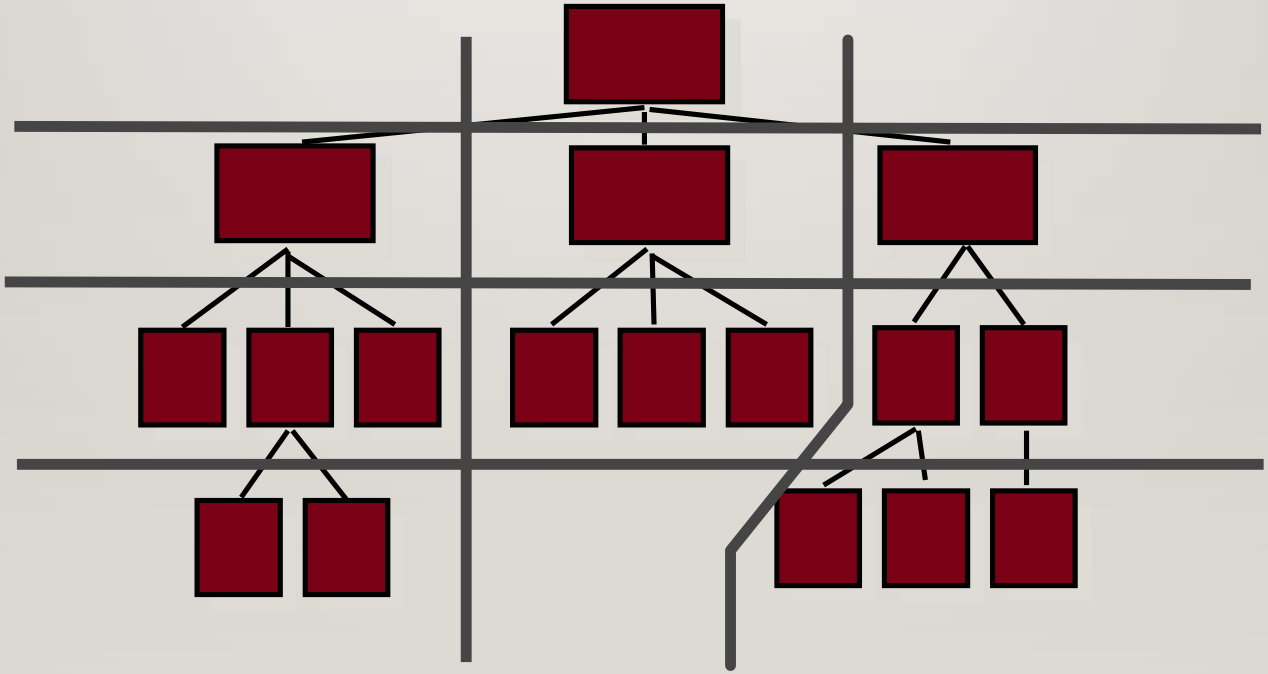
architectural design

DERIVING PROGRAM ARCHITECTURE



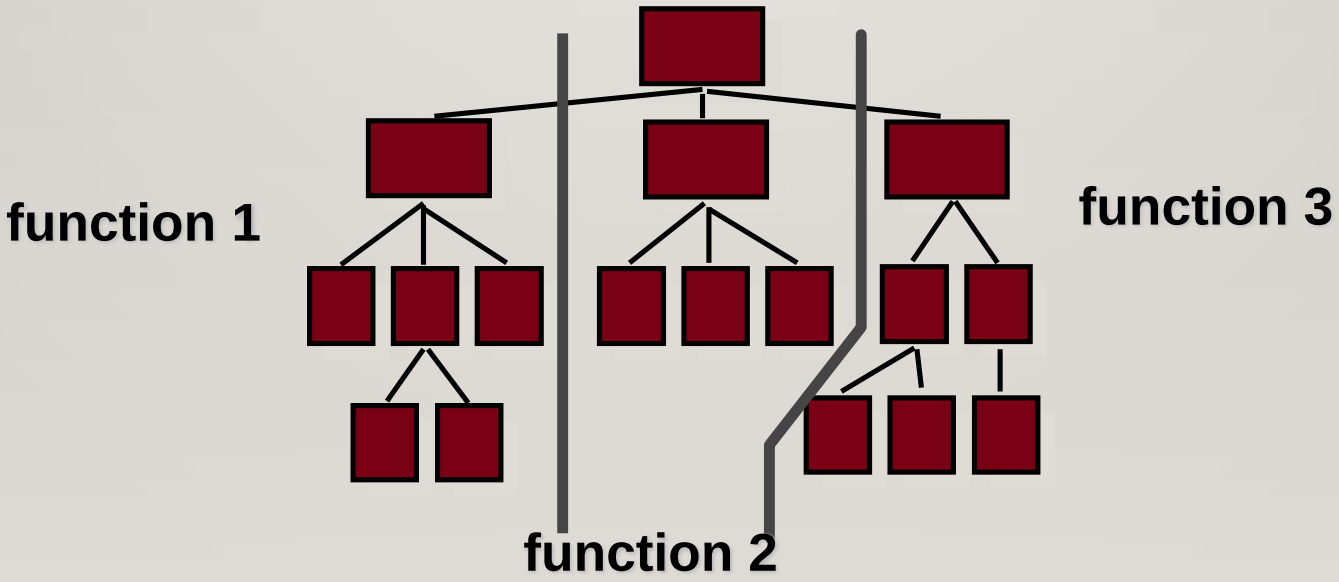
PARTITIONING THE ARCHITECTURE

- "horizontal" and "vertical" partitioning are required



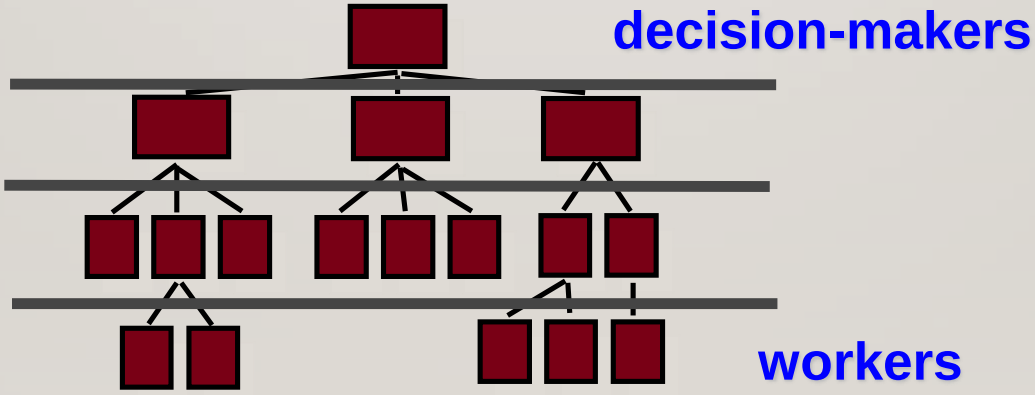
VERTICAL PARTITIONING

- define separate branches of the module hierarchy for each major function
- use control modules to coordinate communication between functions



HORIZONTAL PARTITIONING: FACTORING

- design so that decision making and work are stratified
- decision making modules should reside at the top of the architecture



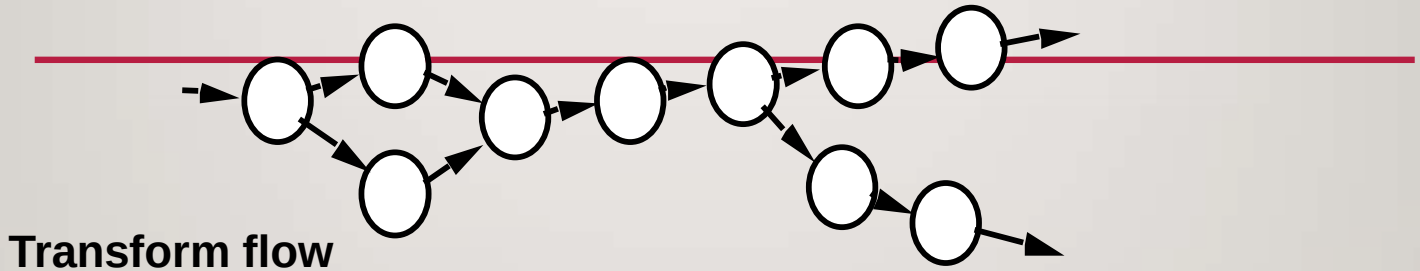
WHY PARTITIONED ARCHITECTURE?

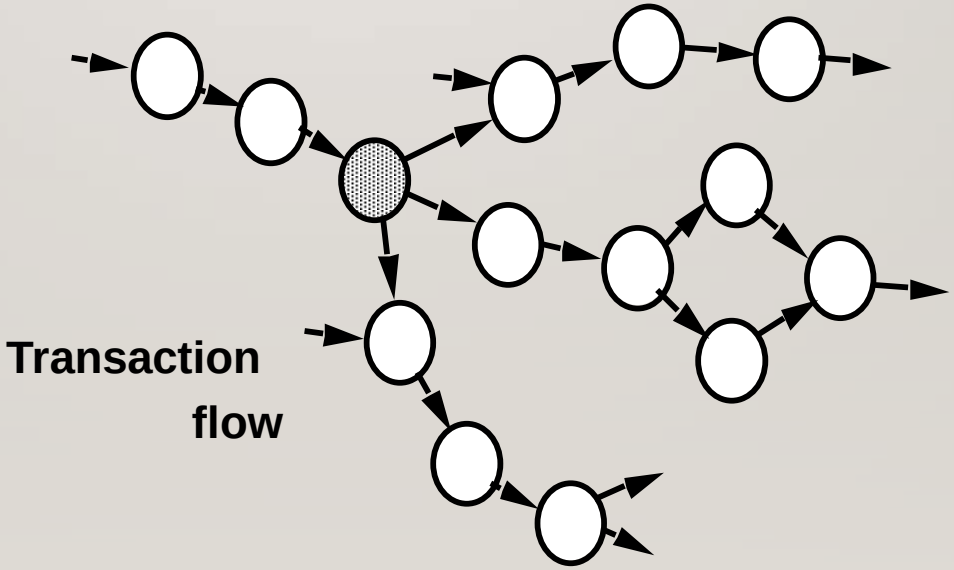
- results in software that is easier to test
- leads to software that is easier to maintain
- results in propagation of fewer side effects
- results in software that is easier to extend

STRUCTURED DESIGN

- objective: to derive a program architecture that is partitioned
- approach:
  - the DFD is mapped into a program architecture
  - the PSPEC and STD are used to indicate the content of each module
- notation: structure chart

FLOW CHARACTERISTICS





GENERAL MAPPING APPROACH

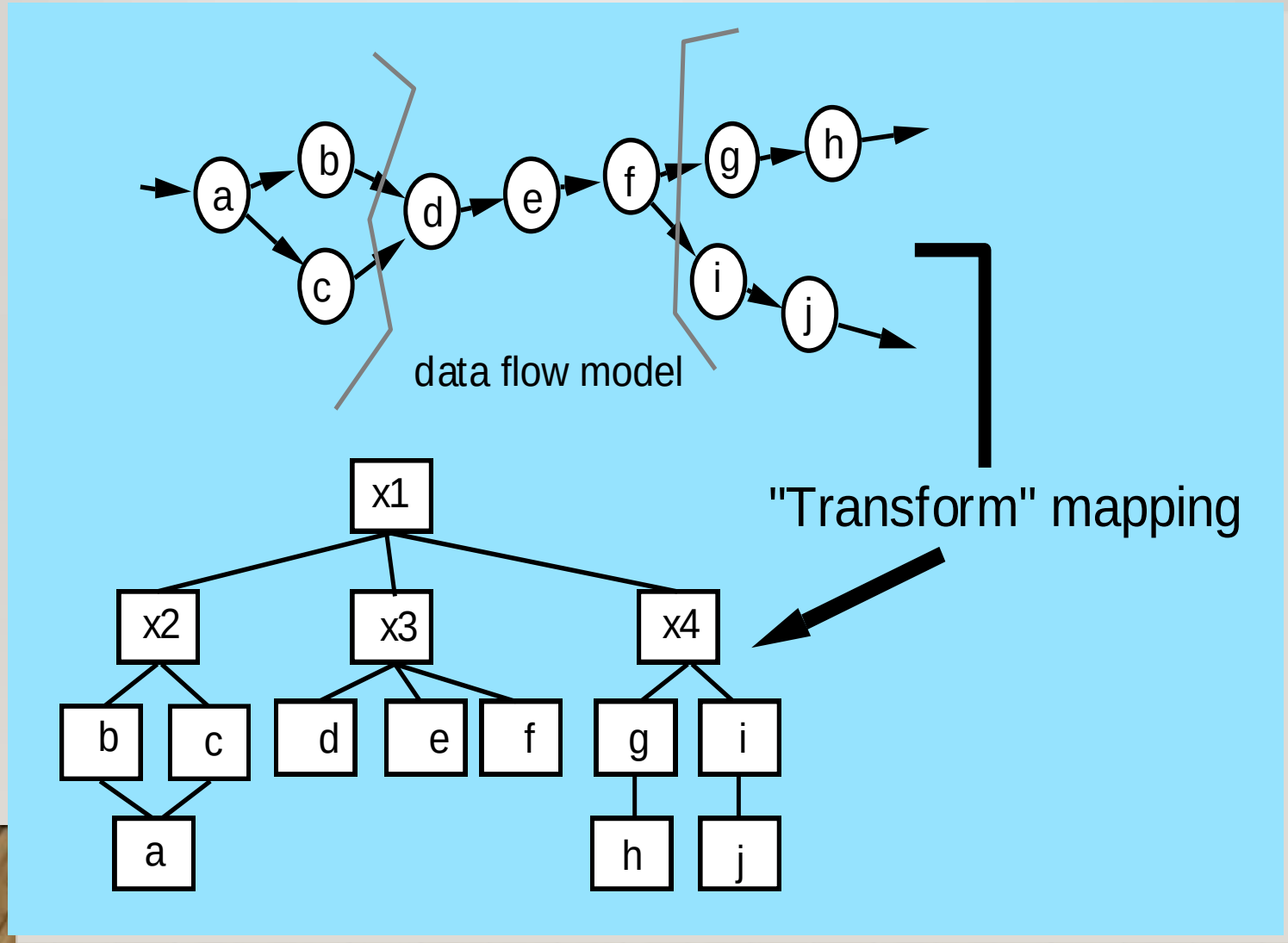
1. Isolate incoming and outgoing flow boundaries; for transaction flows, isolate the transaction center

2. Working from the boundary outward, map DFD transforms into corresponding modules

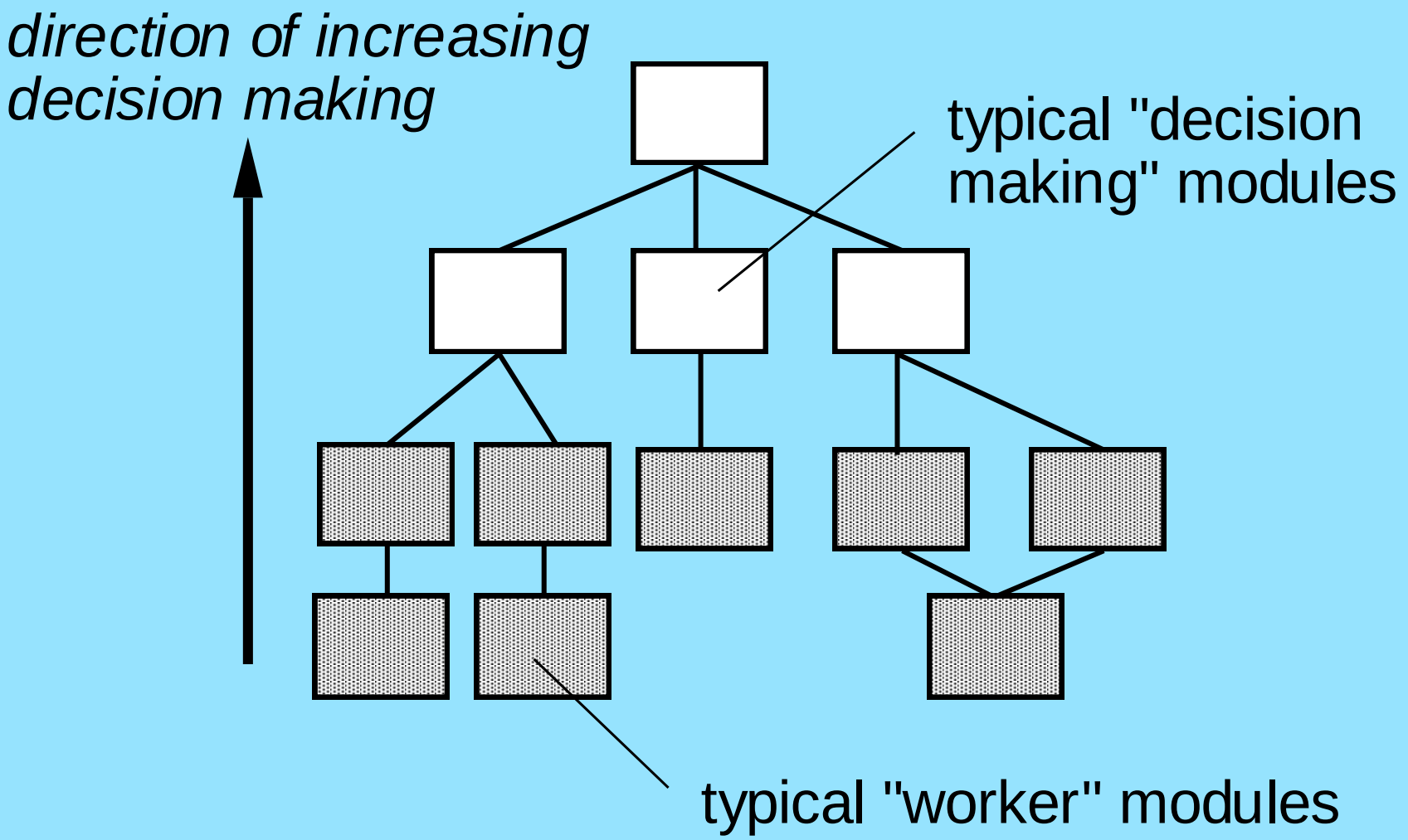
3. Add control modules as required

4. Refine the resultant program structure using effective modularity concepts

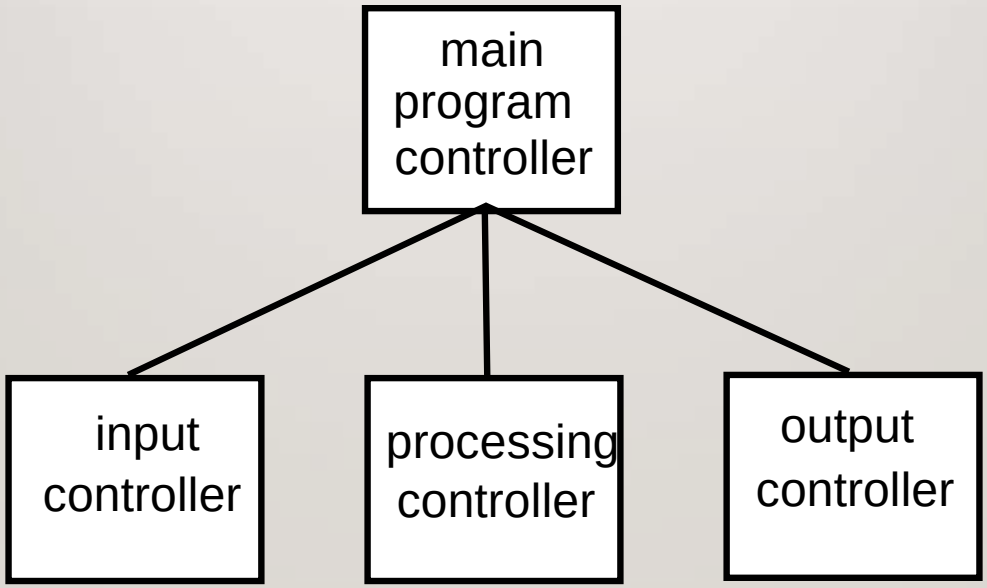
TRANSFORM MAPPING



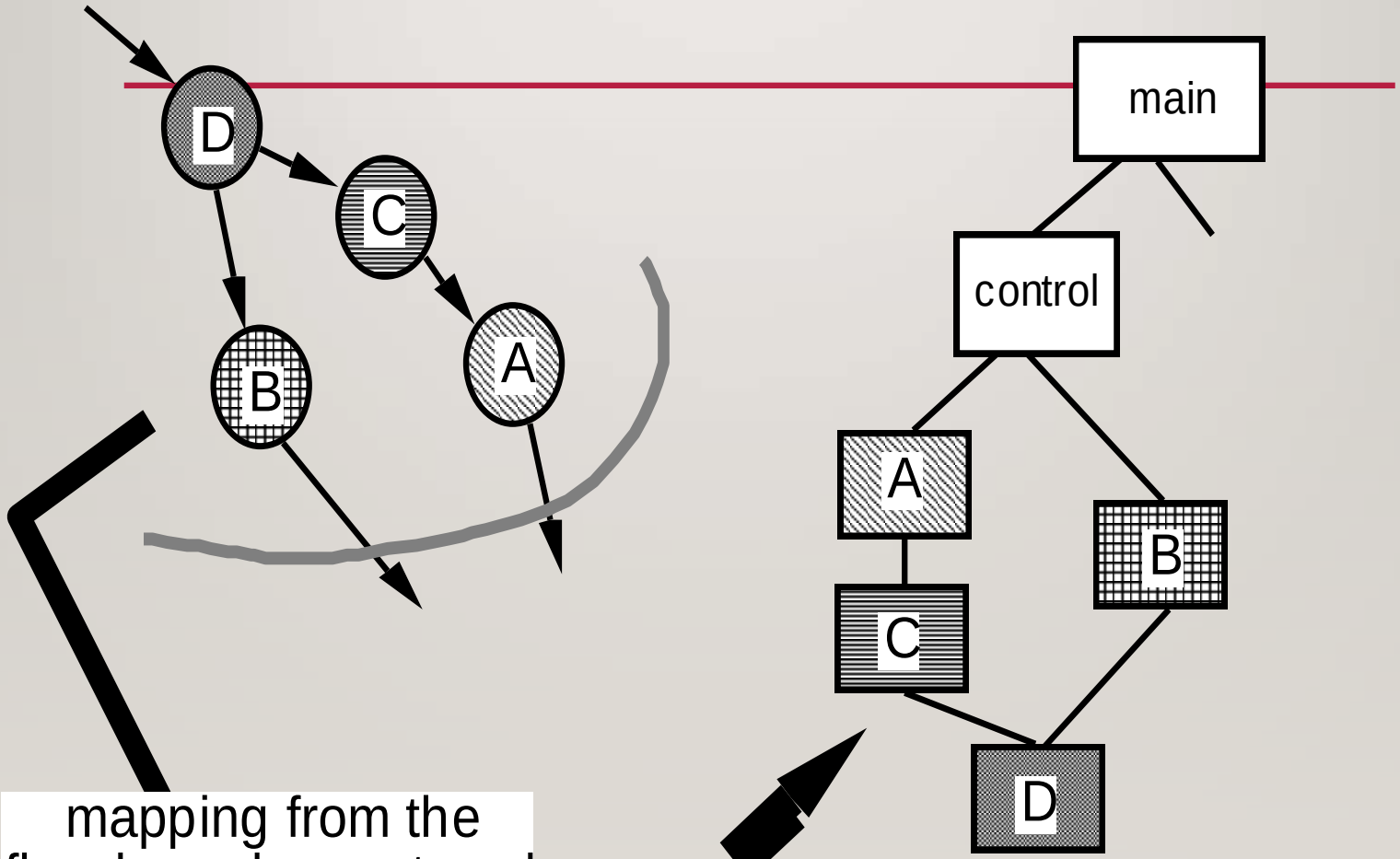
FACTORING


direction of increasing decision making
typical "decision making" modules
typical "worker" modules

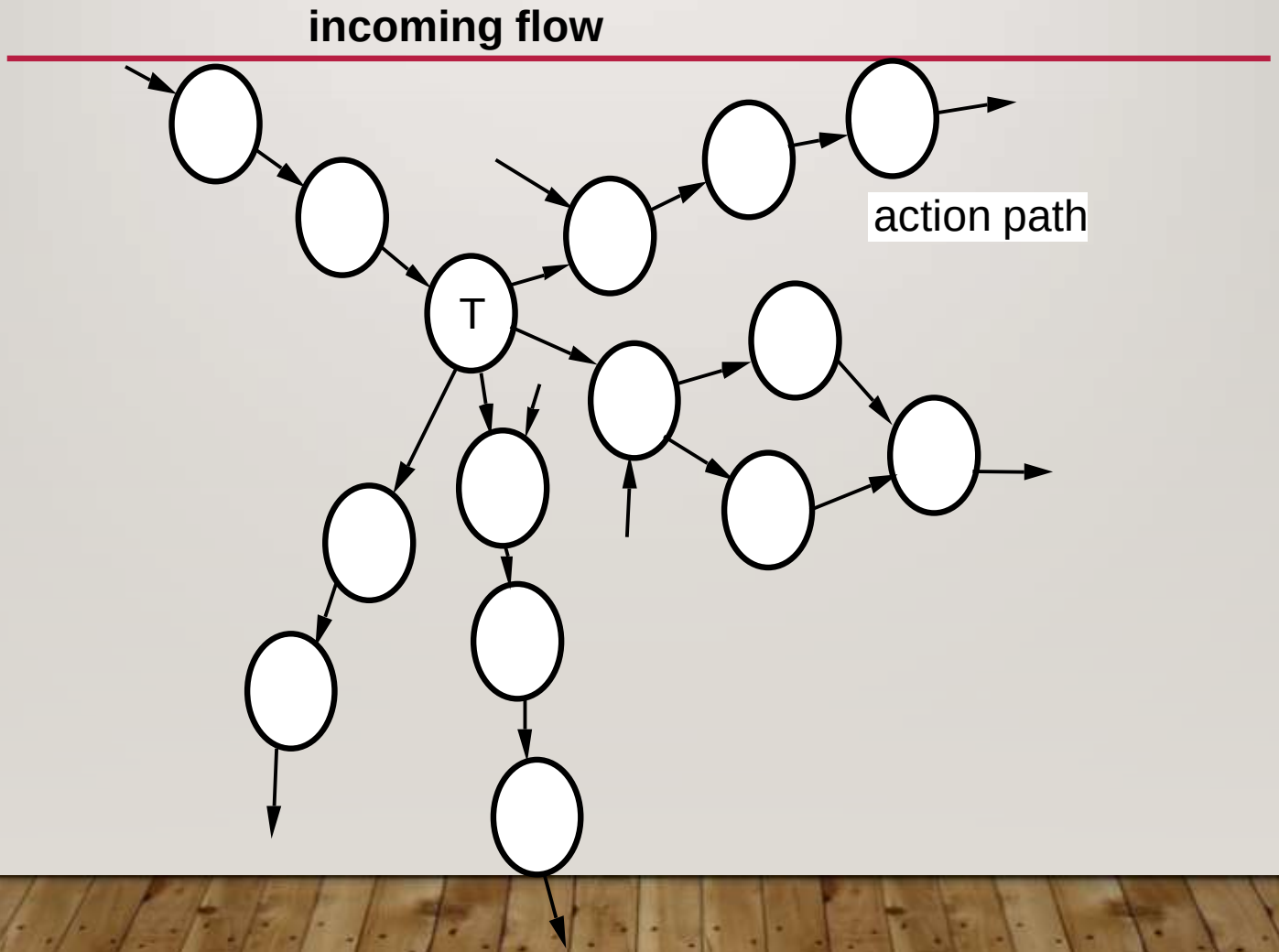
FIRST LEVEL FACTORING



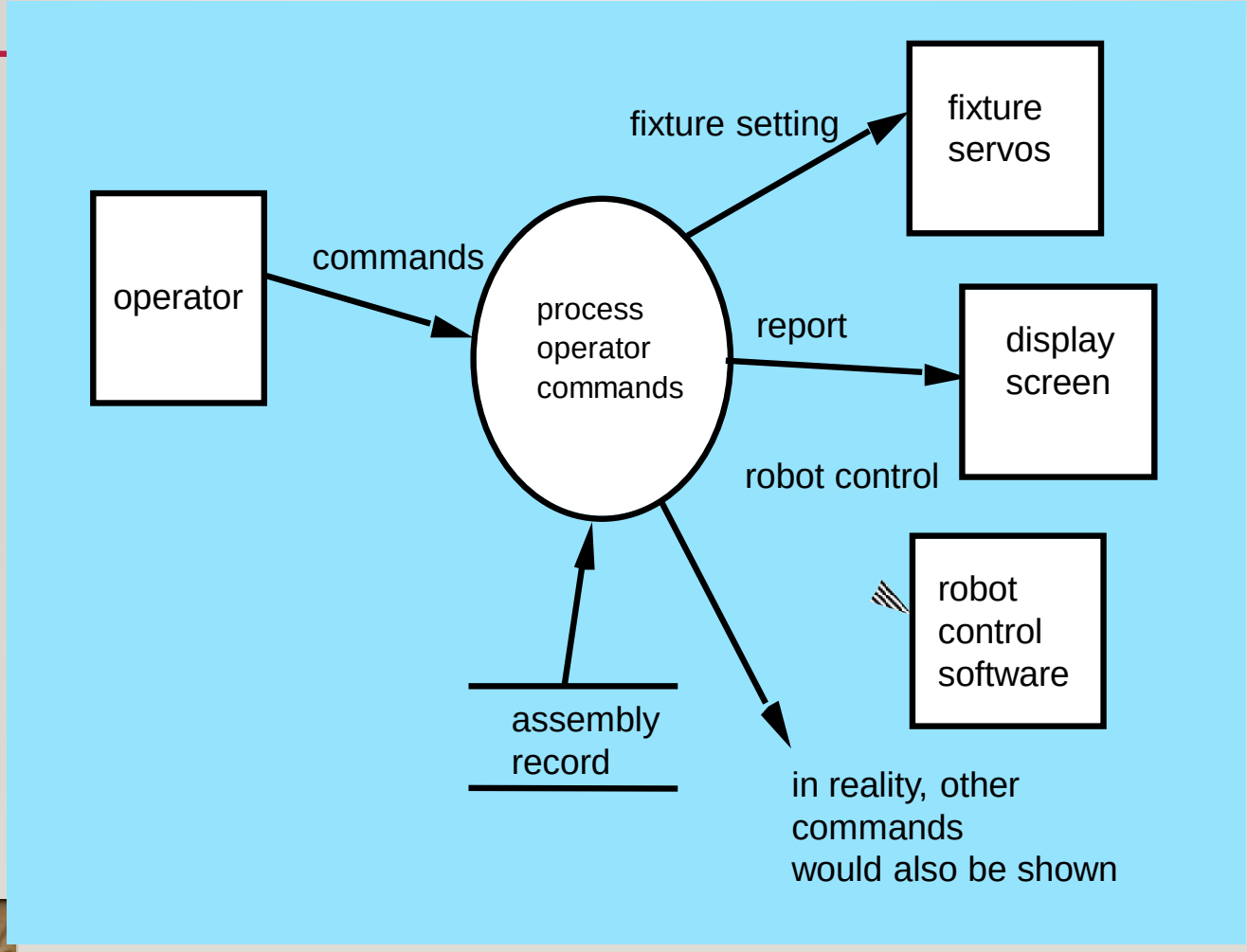
SECOND LEVEL MAPPING



TRANSACTION FLOW



TRANSACTION EXAMPLE



REFINING THE ANALYSIS MODEL

1. write an English language processing narrative for the level 01 flow model
2. apply noun/verb parse to isolate processes, data items, store and entities
3. develop level 02 and 03 flow models
4. create corresponding data dictionary entries
5. refine flow models as appropriate

... now, we're ready to begin design!

DERIVING LEVEL I

Processing narrative for " process operator commands"

90

Process operator command software reads operator commands from the cell operator. An error message is displayed for invalid commands. The command type is determined for valid commands and appropriate action is taken. When fixture commands are encountered, fixture status is analyzed and a fixture setting is output to the fixture servos. When a report is selected, the assembly record file is read and a report is generated and displayed on the operator display screen. When robot control switches are selected, control values are sent to the robot control system.

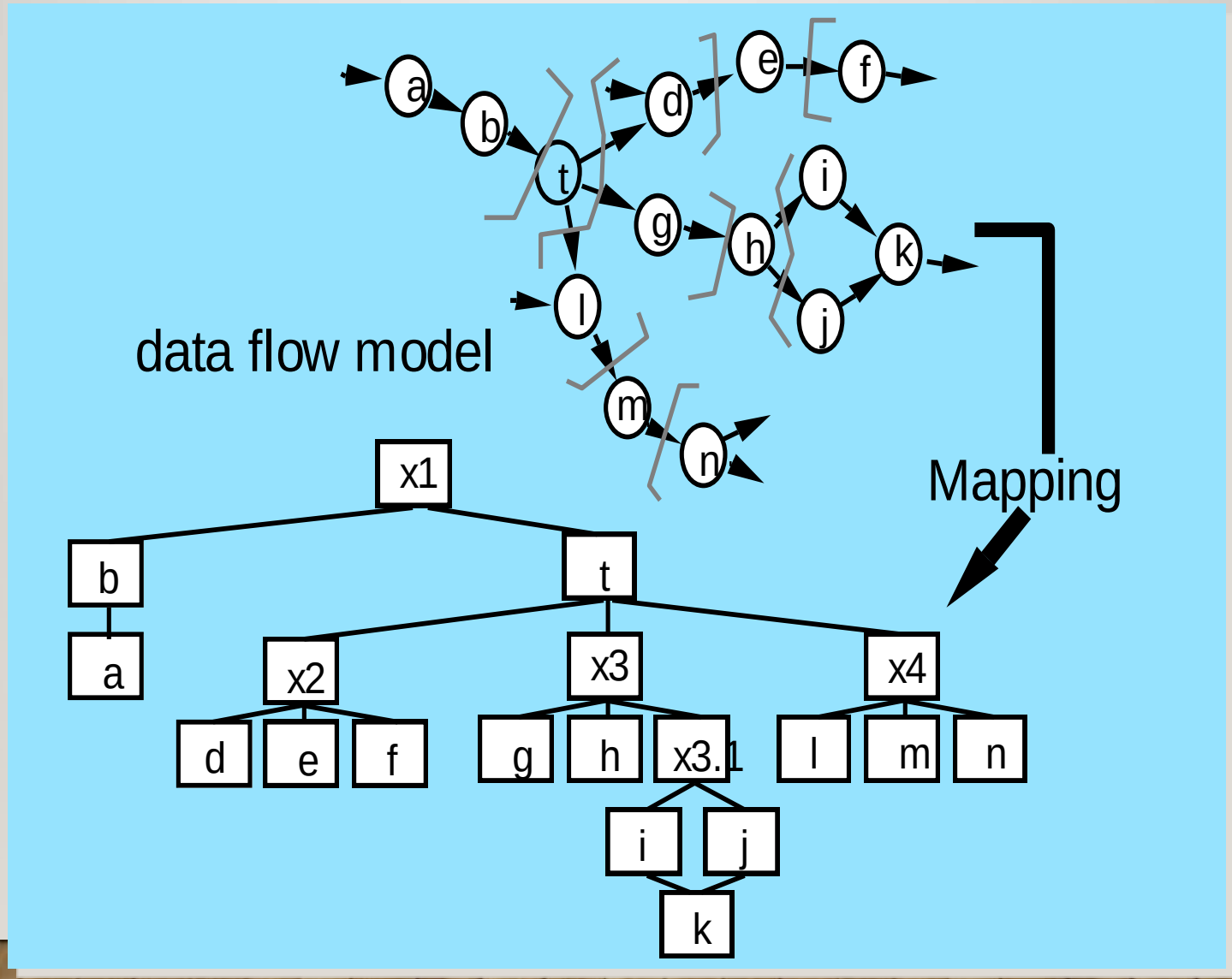
noun-verb parse

Process operator command software reads operator *commands* from the cell *operator*. An *error message* is displayed for *invalid commands*. The *command type* is determined for *valid commands* and appropriate action is taken. When *fixture commands* are encountered, *fixture status* is analyzed and a *fixture setting* is output to the *fixture servos* When a *report* is selected, the *assembly record file* is read and a report is generated and displayed on the operator *display screen*. When *robot control switches* are selected, *control values* are sent to the *robot control system.*

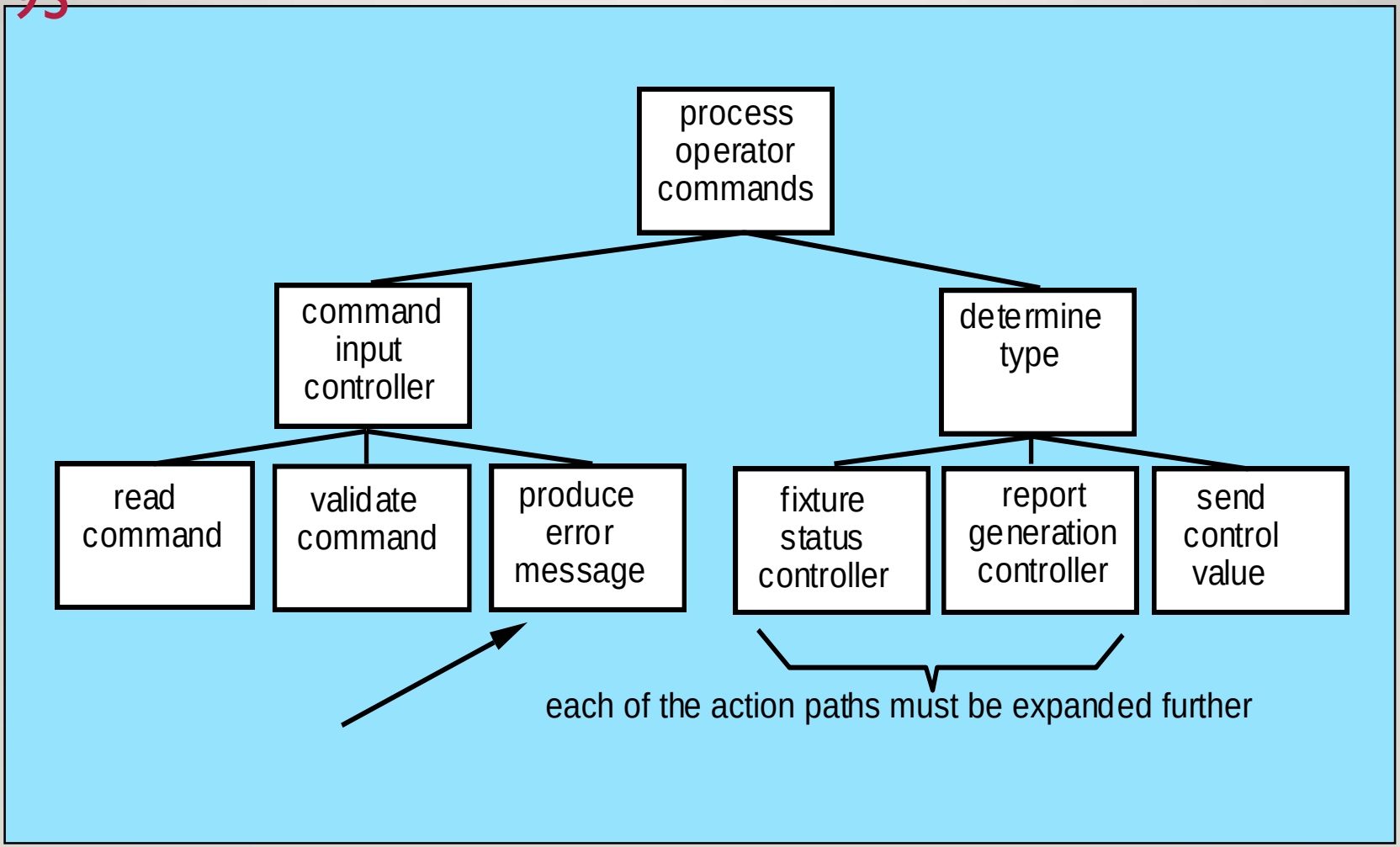
TRANSACTION MAPPING PRINCIPLES

- isolate the incoming flow path
- define each of the action paths by looking for the "spokes of the wheel"
- assess the flow on each action path
- define the dispatch and control structure
- map each action path flow individually

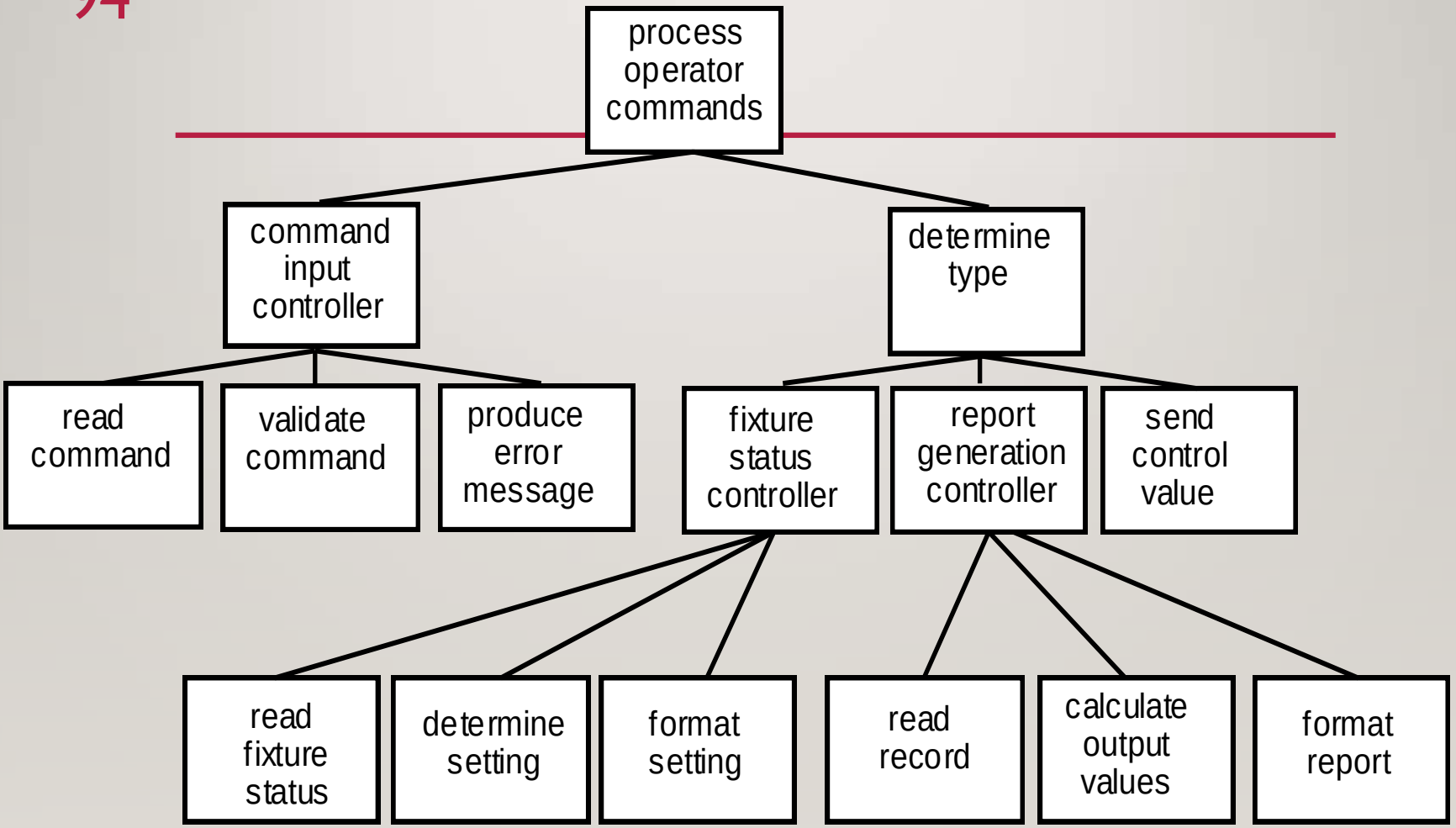
TRANSACTION MAPPING



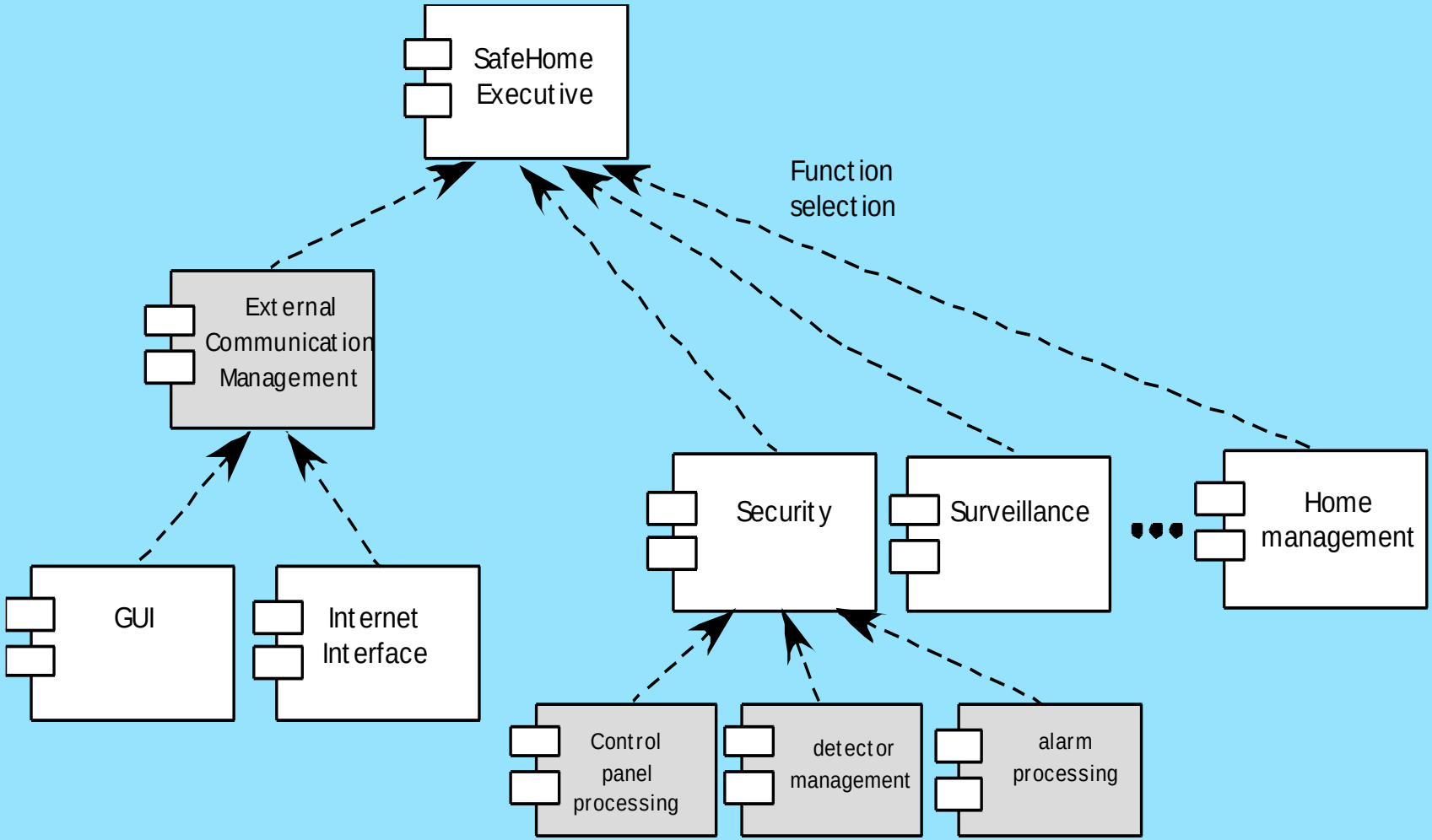
MAP THE FLOW MODEL



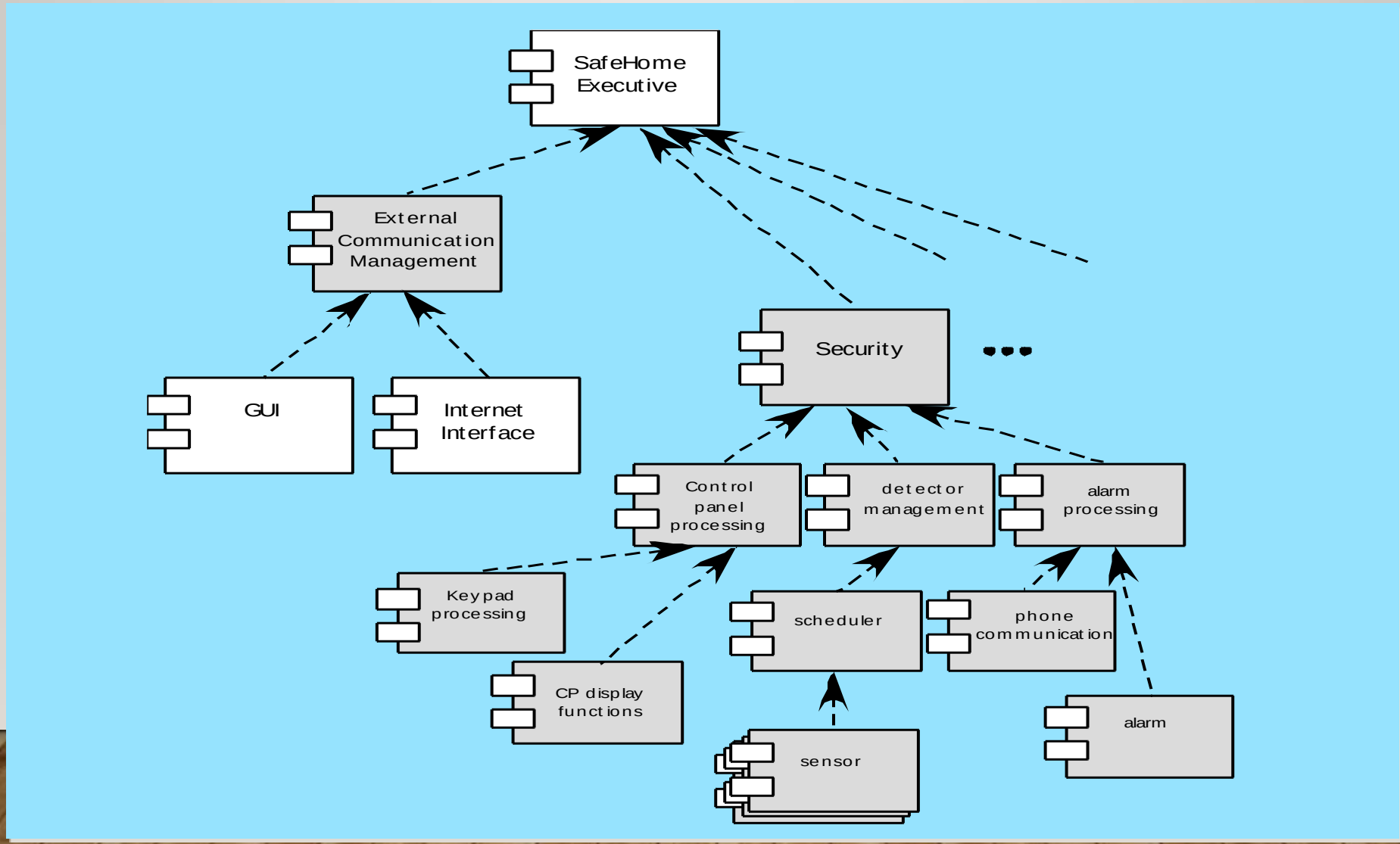
REFINING THE STRUCTURE CHART



COMPONENT STRUCTURE



REFINED COMPONENT STRUCTURE



QUERIES
?

THANK YOU

Module 3

USER INTERFACE DESIGN

DEPT OF ISE, DSCE

User interface design

Objectives

- To suggest some general design principles for user interface design
- To explain different interaction styles and their use
- To explain when to use graphical and textual information presentation
- To explain the principal activities in the user interface design process
- To introduce usability attributes and approaches to system evaluation

Topics covered

- Design issues
- The user interface design process
- User analysis
- User interface prototyping
- Interface evaluation

The user interface

- User interfaces should be designed to match the skills, experience and expectations of its anticipated users.
- System users often judge a system by its interface rather than its functionality.
- A poorly designed interface can cause a user to make catastrophic errors.
- Poor user interface design is the reason why so many software systems are never used.

Human factors in interface design

- Limited short-term memory
  - People can instantaneously remember about 7 items of information. If you present more than this, they are more liable to make mistakes.
- People make mistakes
  - When people make mistakes and systems go wrong, inappropriate alarms and messages can increase stress and hence the likelihood of more mistakes.
- People are different
  - People have a wide range of physical capabilities. Designers should not just design for their own capabilities.
- People have different interaction preferences
  - Some like pictures, some like text.

UI design principles

- UI design must take account of the needs, experience and capabilities of the system users.
- Designers should be aware of people's physical and mental limitations (e.g. limited short-term memory) and should recognise that people make mistakes.
- UI design principles underlie interface designs although not all principles are applicable to all designs.

Design principles

- User familiarity
  - The interface should be based on user-oriented terms and concepts rather than computer concepts. For example, an office system should use concepts such as letters, documents, folders etc. rather than directories, file identifiers, etc.
- Consistency
  - The system should display an appropriate level of consistency. Commands and menus should have the same format, command punctuation should be similar, etc.
- Minimal surprise
  - If a command operates in a known way, the user should be able to predict the operation of comparable commands

Design principles

- Recoverability
  - The system should provide some resilience to user errors and allow the user to recover from errors. This might include an undo facility, confirmation of destructive actions, 'soft' deletes, etc.
- User guidance
  - Some user guidance such as help systems, on-line manuals, etc. should be supplied
- User diversity
  - Interaction facilities for different types of user should be supported. For example, some users have seeing difficulties and so larger text should be available

Design issues in UIs

- Two problems must be addressed in interactive systems design
  - How should information from the user be provided to the computer system?
  - How should information from the computer system be presented to the user?
- User interaction and information presentation may be integrated through a coherent framework such as a user interface metaphor.

Interaction styles

- Direct manipulation
- Menu selection
- Form fill-in
- Command language
- Natural language

Interaction styles

Multiple user interfaces

LIBSYS interaction

- Document search
  - Users need to be able to use the search facilities to find the documents that they need.
- Document request
  - Users request that a document be delivered to their machine or to a server for printing.

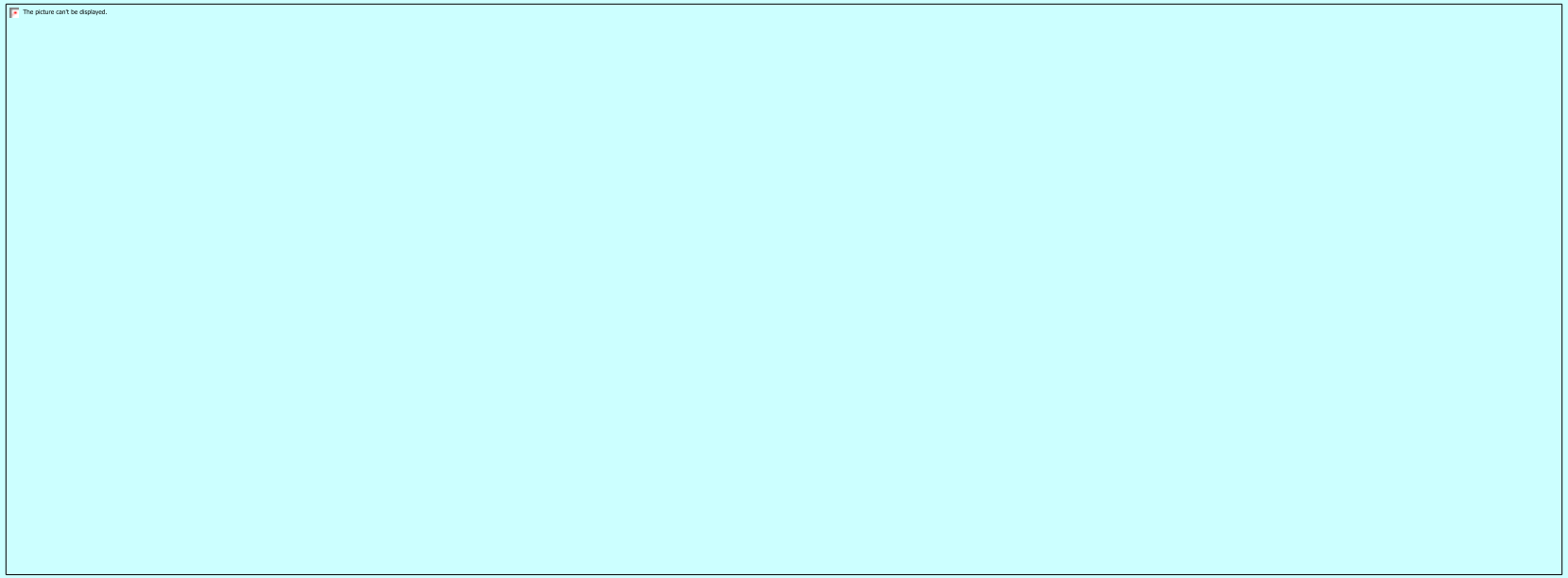
Web-based interfaces

- Many web-based systems have interfaces based on web forms.
- Form field can be menus, free text input, radio buttons, etc.
- In the LIBSYS example, users make a choice of where to search from a menu and type the search phrase into a free text field.

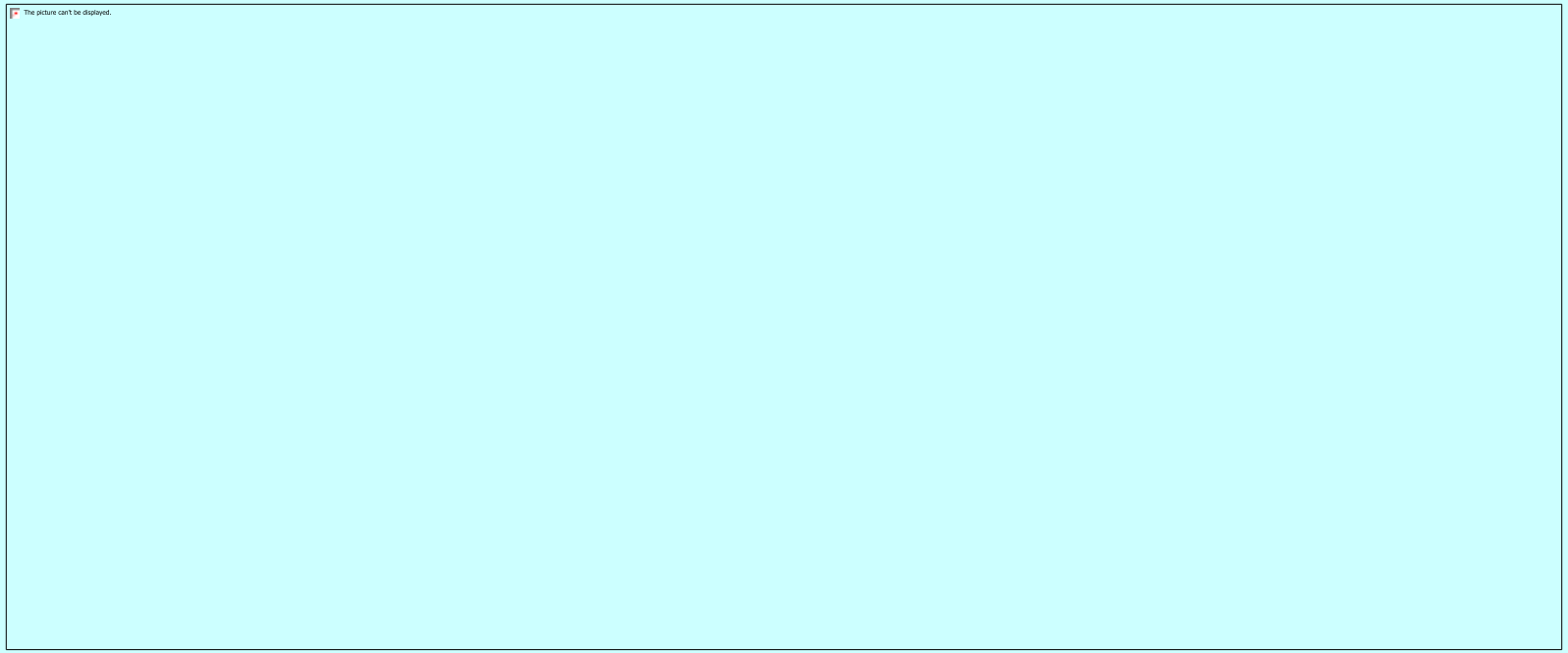
Information presentation

- Information presentation is concerned with presenting system information to system users.
- The information may be presented directly (e.g. text in a word processor) or may be transformed in some way for presentation (e.g. in some graphical form).
- The Model-View-Controller approach is a way of supporting multiple presentations of data.

Information presentation



Model-view-controller



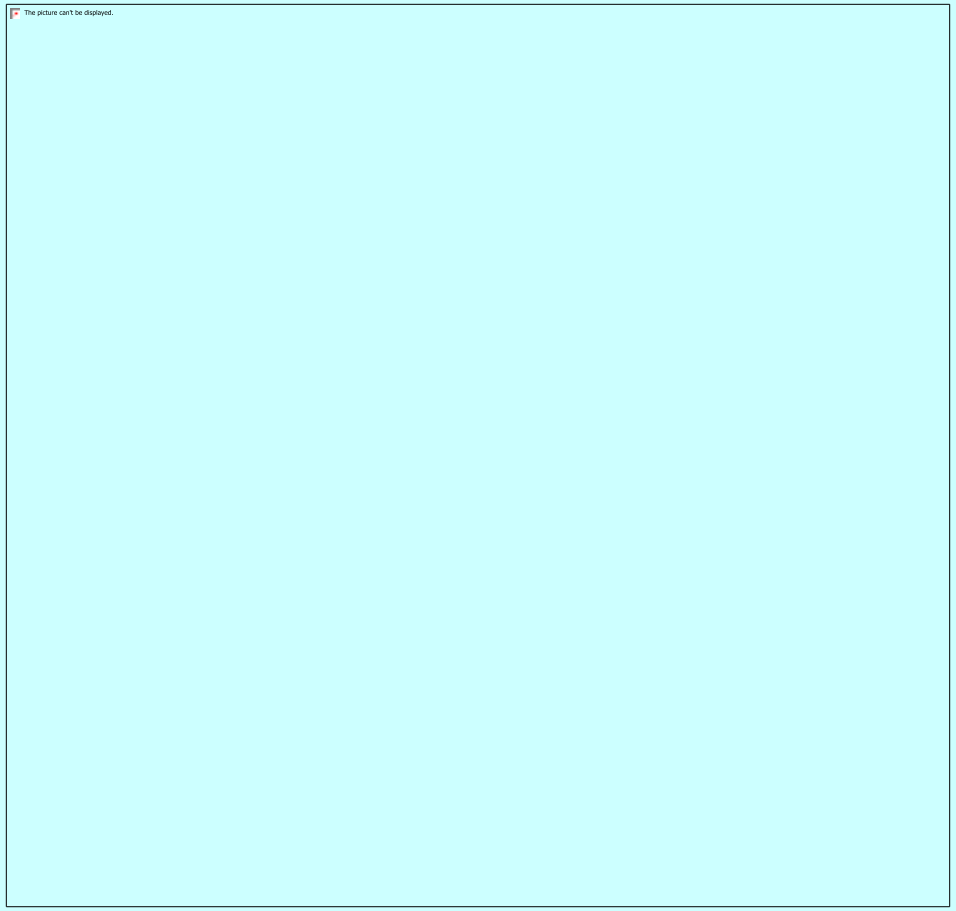
Information presentation

- Static information
  - Initialised at the beginning of a session. It does not change during the session.
  - May be either numeric or textual.
- Dynamic information
  - Changes during a session and the changes must be communicated to the system user.
  - May be either numeric or textual.

Information display factors

- Is the user interested in precise information or data relationships?
- How quickly do information values change? Must the change be indicated immediately?
- Must the user take some action in response to a change?
- Is there a direct manipulation interface?
- Is the information textual or numeric? Are relative values important?

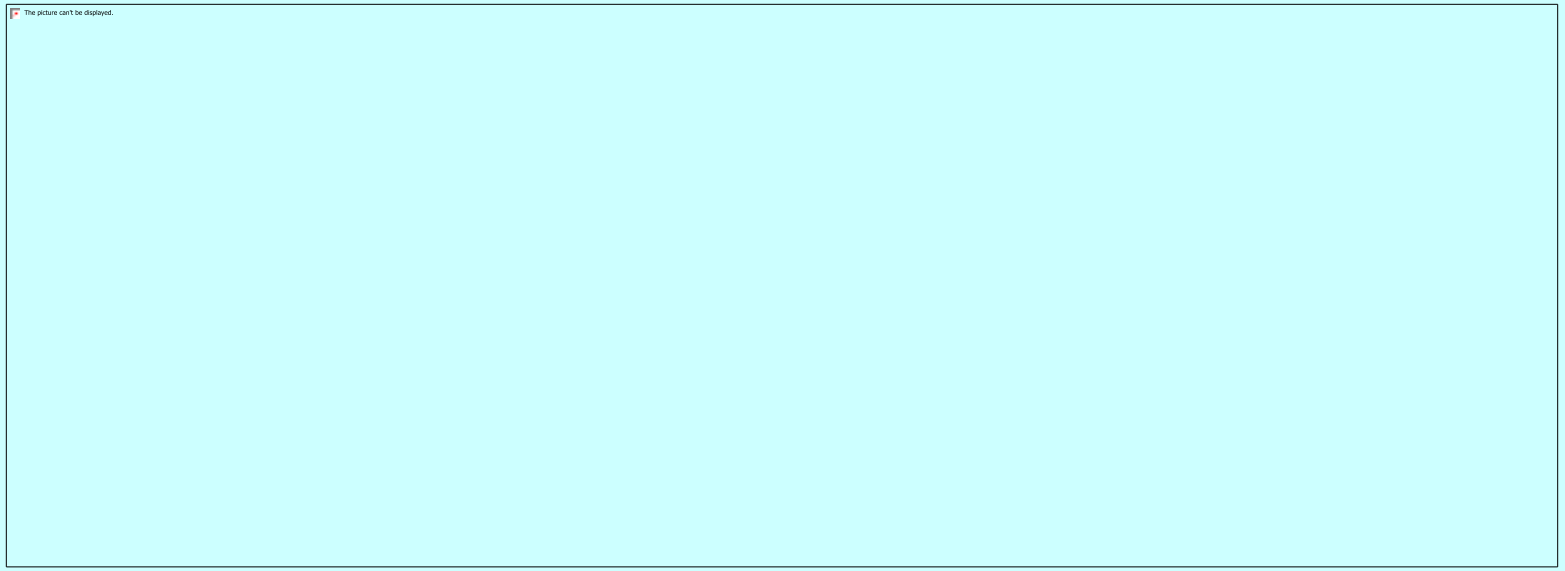
Alternative information presentations



Analogue or digital presentation?

- Digital presentation
  - Compact - takes up little screen space;
  - Precise values can be communicated.
- Analogue presentation
  - Easier to get an 'at a glance' impression of a value;
  - Possible to show relative values;
  - Easier to see exceptional data values.

Presentation methods



Displaying relative values

Data visualisation

- Concerned with techniques for displaying large amounts of information.
- Visualisation can reveal relationships between entities and trends in the data.
- Possible data visualisations are:
  - Weather information collected from a number of sources;
  - The state of a telephone network as a linked set of nodes;
  - Chemical plant visualised by showing pressures and temperatures in a linked set of tanks and pipes;
  - A model of a molecule displayed in 3 dimensions;
  - Web pages displayed as a hyperbolic tree.

Colour displays

- Colour adds an extra dimension to an interface and can help the user understand complex information structures.
- Colour can be used to highlight exceptional events.
- Common mistakes in the use of colour in interface design include:
  - The use of colour to communicate meaning;
  - The over-use of colour in the display.

Colour use guidelines

- Limit the number of colours used and be conservative in their use.
- Use colour change to show a change in system status.
- Use colour coding to support the task that users are trying to perform.
- Use colour coding in a thoughtful and consistent way.
- Be careful about colour pairings.

Error messages

- Error message design is critically important. Poor error messages can mean that a user rejects rather than accepts a system.
- Messages should be polite, concise, consistent and constructive.
- The background and experience of users should be the determining factor in message design.

Design factors in message wording

User error

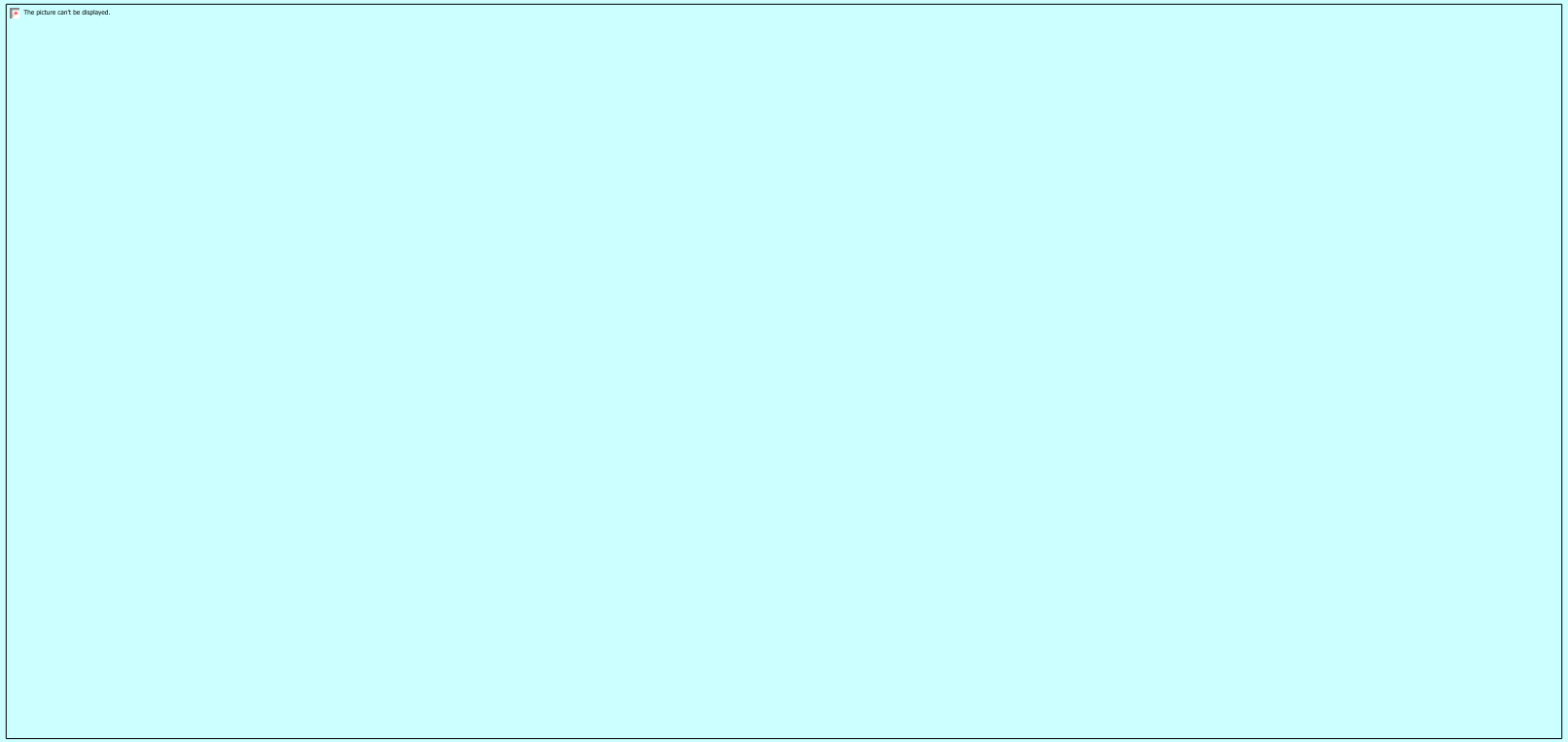
- Assume that a nurse misspells the name of a patient whose records he is trying to retrieve.

Good and bad message design

The UI design process

- UI design is an iterative process involving close liaisons between users and designers.
- The 3 core activities in this process are:
  - User analysis. Understand what the users will do with the system;
  - System prototyping. Develop a series of prototypes for experiment;
  - Interface evaluation. Experiment with these prototypes with users.

The design process



User analysis

- If you don't understand what the users want to do with a system, you have no realistic prospect of designing an effective interface.
- User analyses have to be described in terms that users and other designers can understand.
- Scenarios where you describe typical episodes of use, are one way of describing these analyses.

User interaction scenario

Jane is a student of Religious Studies and is working on an essay on Indian architecture and how it has been influenced by religious practices. To help her understand this, she would like to access some pictures of details on notable buildings but can't find anything in her local library.

She approaches the subject librarian to discuss her needs and he suggests some search terms that might be used. He also suggests some libraries in New Delhi and London that might have this material so they log on to the library catalogues and do some searching using these terms. They find some source material and place a request for photocopies of the pictures with architectural detail to be posted directly to Jane.

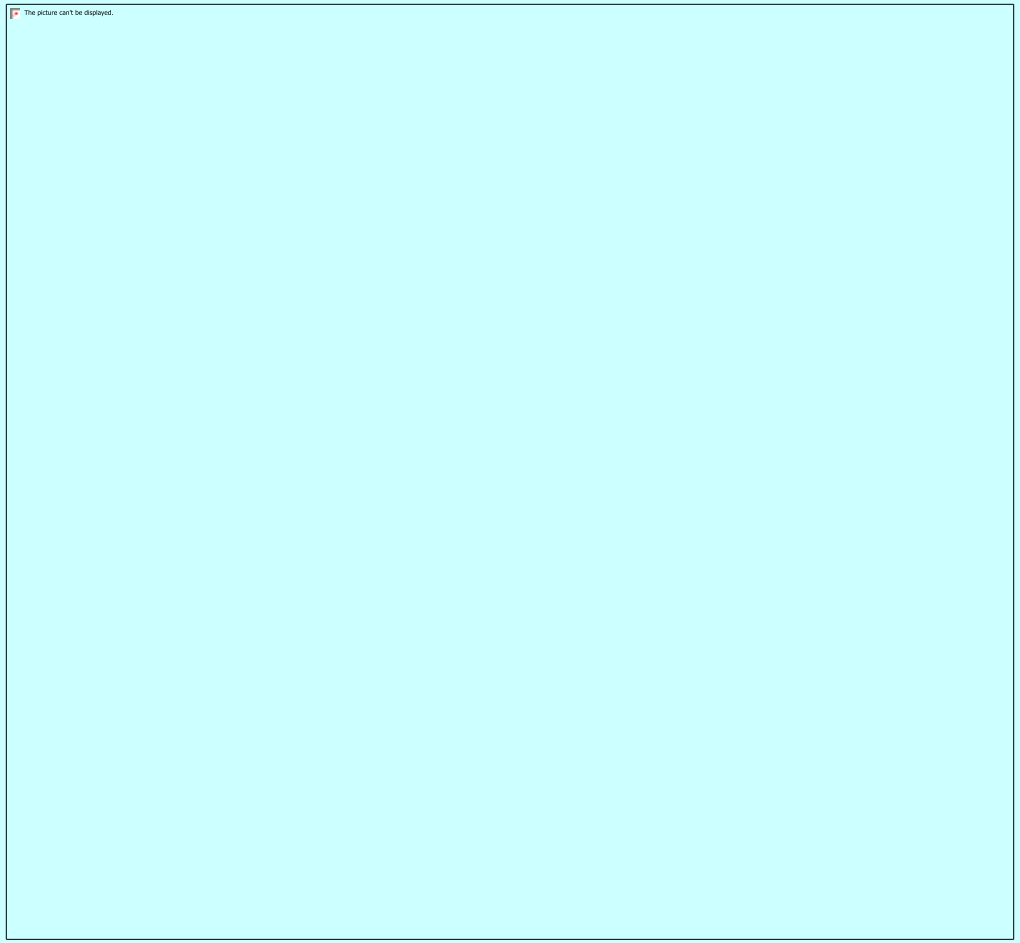
Requirements from the scenario

- Users may not be aware of appropriate search terms so need a way of helping them choose terms.
- Users have to be able to select collections to search.
- Users need to be able to carry out searches and request copies of relevant material.

Analysis techniques

- Task analysis
  - Models the steps involved in completing a task.
- Interviewing and questionnaires
  - Asks the users about the work they do.
- Ethnography
  - Observes the user at work.

Hierarchical task analysis



Interviewing

- Design semi-structured interviews based on open-ended questions.
- Users can then provide information that they think is essential; not just information that you have thought of collecting.
- Group interviews or focus groups allow users to discuss with each other what they do.

Ethnography

- Involves an external observer watching users at work and questioning them in an unscripted way about their work.
- Valuable because many user tasks are intuitive and they find these very difficult to describe and explain.
- Also helps understand the role of social and organisational influences on work.

Ethnographic records

Air traffic control involves a number of control ‘suites’ where the suites controlling adjacent sectors of airspace are physically located next to each other. Flights in a sector are represented by paper strips that are fitted into wooden racks in an order that reflects their position in the sector. If there are not enough slots in the rack (i.e. when the airspace is very busy), controllers spread the strips out on the desk in front of the rack.

When we were observing controllers, we noticed that controllers regularly glanced at the strip racks in the adjacent sector. We pointed this out to them and asked them why they did this. They replied that, if the adjacent controller has strips on their desk, then this meant that they would have a lot of flights entering their sector. They therefore tried to increase the speed of aircraft in the sector to ‘clear space’ for the incoming aircraft.

Insights from ethnography

- Controllers had to see all flights in a sector. Therefore, scrolling displays where flights disappeared off the top or bottom of the display should be avoided.
- The interface had to have some way of telling controllers how many flights were in adjacent sectors so that they could plan their workload.

User interface prototyping

- The aim of prototyping is to allow users to gain direct experience with the interface.
- Without such direct experience, it is impossible to judge the usability of an interface.
- Prototyping may be a two-stage process:
  - Early in the process, paper prototypes may be used;
  - The design is then refined and increasingly sophisticated automated prototypes are then developed.

Paper prototyping

- Work through scenarios using sketches of the interface.
- Use a storyboard to present a series of interactions with the system.
- Paper prototyping is an effective way of getting user reactions to a design proposal.

Prototyping techniques

- Script-driven prototyping
  - Develop a set of scripts and screens using a tool such as Macromedia Director. When the user interacts with these, the screen changes to the next display.
- Visual programming
  - Use a language designed for rapid development such as Visual Basic. See Chapter 17.
- Internet-based prototyping
  - Use a web browser and associated scripts.

User interface evaluation

- Some evaluation of a user interface design should be carried out to assess its suitability.
- Full scale evaluation is very expensive and impractical for most systems.
- Ideally, an interface should be evaluated against a usability specification. However, it is rare for such specifications to be produced.

Usability attributes

The picture can't be displayed.

Simple evaluation techniques

- Questionnaires for user feedback.
- Video recording of system use and subsequent tape evaluation.
- Instrumentation of code to collect information about facility use and user errors.
- The provision of code in the software to collect on-line user feedback.

Key points

- User interface design principles should help guide the design of user interfaces.
- Interaction styles include direct manipulation, menu systems form fill-in, command languages and natural language.
- Graphical displays should be used to present trends and approximate values. Digital displays when precision is required.
- Colour should be used sparingly and consistently.

Key points

- The user interface design process involves user analysis, system prototyping and prototype evaluation.
- The aim of user analysis is to sensitise designers to the ways in which users actually work.
- UI prototyping should be a staged process with early paper prototypes used as a basis for automated prototypes of the interface.
- The goals of UI evaluation are to obtain feedback on how to improve the interface design and to assess if the interface meets its usability requirements.