Architectural Design

Idaho State University | Computer Science

Isaac Griffith

CS 3321
Department of Computer Science
Idaho State University

ROAR

Topics covered

- Architectural design decisions
- Architectural views

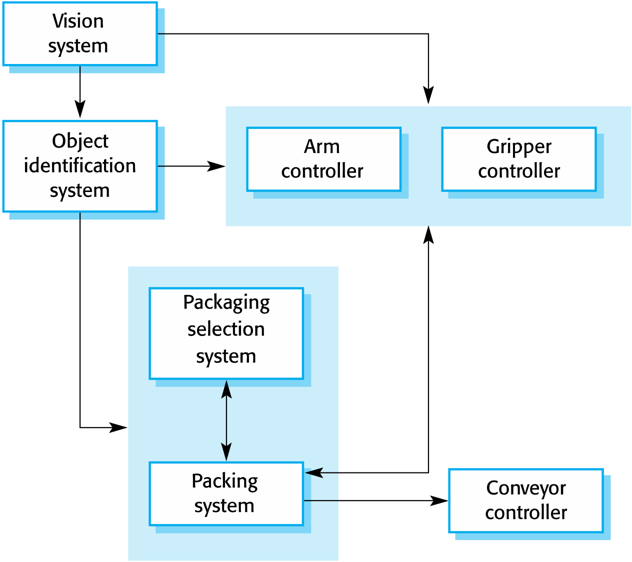
Architectural design

- Architectural design is concerned with understanding how a software system should be organized and designing the overall structure of that system.
- Architectural design is the critical link between design and requirements engineering, as it identifies the main structural components in a system and the relationships between them.
- The output of the architectural design process is an architectural model that describes how the system is organized as a set of communicating components.

Agility and architecture

- It is generally accepted that an early stage of agile processes is to design an overall systems architecture.
- Refactoring the system architecture is usually expensive because it affects so many components in the system

Packing robot architecture



Architectural abstraction

- Architecture in the small is concerned with the architecture of individual programs. At this level, we are concerned with the way that an individual program is decomposed into components.
- Architecture in the large is concerned with the architecture of complex enterprise systems that include other systems, programs, and program components. These enterprise systems are distributed over different computers, which may be owned and managed by different companies.

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
  - Product-line architectures may be developed.

Architectural representations

- Simple, informal block diagrams showing entities and relationships are the most frequently used method for documenting software architectures.
- But these have been criticized because they lack semantics, do not show the types of relationships between entities nor the visible properties of entities in the architecture.
- Depends on the use of architectural models.The requirements for model semantics depends on how the models are used.

Box and line diagrams

- Very abstract - they do not show the nature of component relationships nor the externally visible properties of the sub-systems.
- However, useful for communication with stakeholders and for project planning.

Use of architectural models

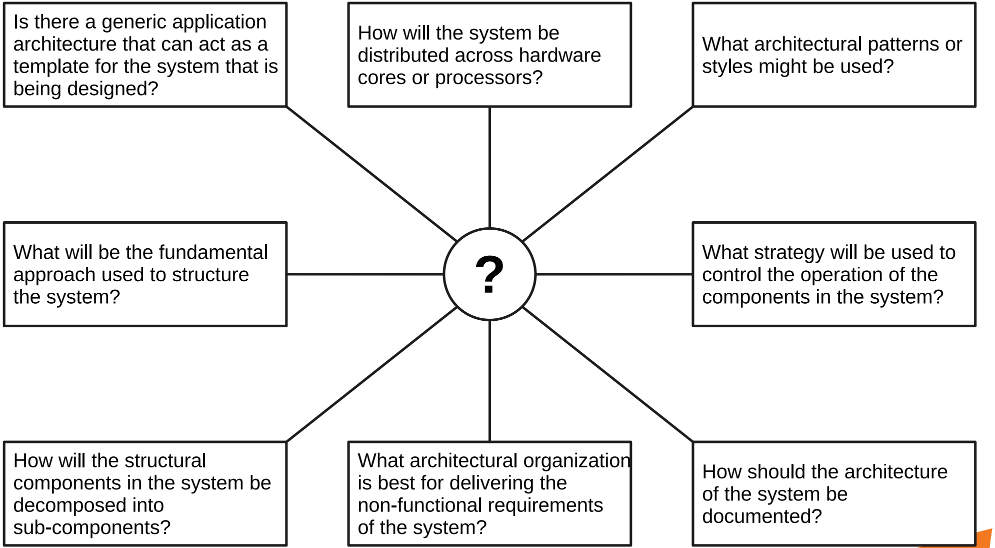
- As a way of facilitating discussion about the system design
  - A high-level architectural view of a system is useful for communication with system stakeholders and project planning because it is not cluttered with detail. Stakeholders can relate to it and understand an abstract view of the system. They can then discuss the system as a whole without being confused by detail.
- As a way of documenting an architecture that has been designed
  - The aim here is to produce a complete system model that shows the different components in a system, their interfaces and their connections.

Architectural design decisions

Architectural design decisions

- Architectural design is a creative process so the process differs depending on the type of system being developed.
- However, a number of common decisions span all design processes and these decisions affect the non-functional characteristics of the system.

Architectural design decisions



Is there a generic application architecture that can act as a template for the system that is being designed?

How will the system be distributed across hardware cores or processors?

What architectural patterns or styles might be used?

What will be the fundamental approach used to structure the system?

What strategy will be used to control the operation of the components in the system?

How will the structural components in the system be decomposed into sub-components?

What architectural organization is best for delivering the non-functional requirements of the system?

How should the architecture of the system be documented?

Architectural reuse

- Systems in the same domain often have similar architectures that reflect domain concepts.
- Application product lines are built around a core architecture with variants that satisfy particular customer requirements.
- The architecture of a system may be designed around one of more architectural patterns or ‘styles’.
  - These capture the essence of an architecture and can be instantiated in different ways.

Architecture characteristics

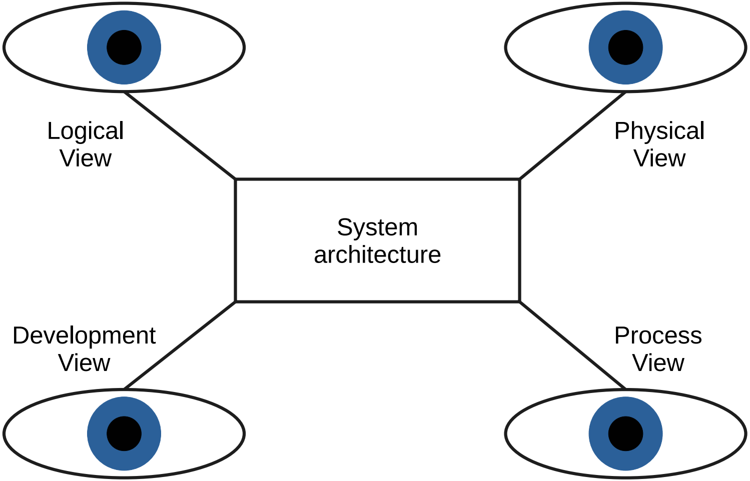
- Performance
  - Localize critical operations and minimize communications. Use large rather than fine-grain components.
- Security
  - Use a layered architecture with critical assets in the inner layers.
- Safety
  - Localize safety-critical features in a small number of sub-systems.
- Availability
  - Include redundant components and mechanisms for fault tolerance.
- Maintainability
  - Use fine-grain, replaceable components.

Architectural views

Architectural views

- What views or perspectives are useful when designing and documenting a system's architecture?
- What notations should be used for describing architectural models?
- Each architectural model only shows one view or perspective of the system.
  - It might show how a system is decomposed into modules, how the run-time processes interact or the different ways in which system components are distributed across a network. For both design and documentation, you usually need to present multiple views of the software architecture.

Architectural views



4+1 view architecture

- A logical view, which shows the key abstractions in the system as objects or object classes.
- A process view, which shows how, at run-time, the system is composed of interacting processes.
- A development view, which shows how the software is decomposed for development.
- A physical view, which shows the system hardware and how software components are distributed across the processors in the system.
- Related using use cases or scenarios (+1)

Representing architectural views

- Some people argue that the Unified Modeling Language (UML) is an appropriate notation for describing and documenting system architectures
- I disagree with this as I do not think that the UML includes abstractions appropriate for high-level system description.
- Architectural description languages (ADLs) have been developed but are not widely used

Architectural patterns

Architectural patterns

- Patterns are a means of representing, sharing and reusing knowledge.
- An architectural pattern is a stylized description of good design practice, which has been tried and tested in different environments.
- Patterns should include information about when they are and when the are not useful.
- Patterns may be represented using tabular and graphical descriptions.

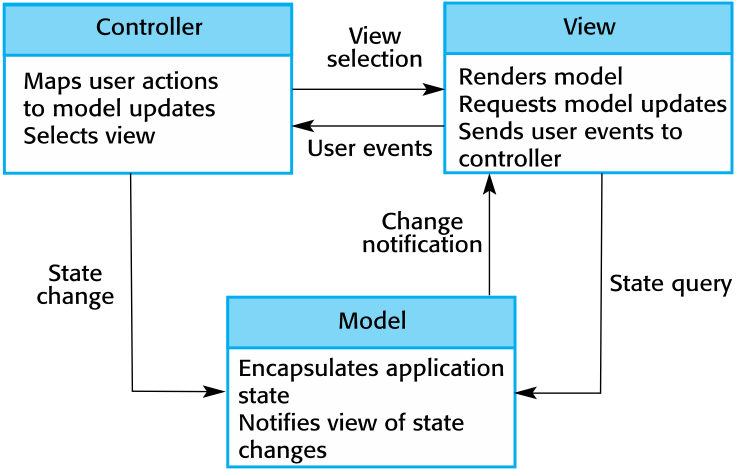
Model-View-Controller (MVC)

- **Description**
  - Separates presentation and interaction from the system data. The system is structured into three logical components that interact with each other. The Model component manages the system data and associated operations on that data. The View component defines and manages how the data is presented to the user. The Controller component manages user interaction (e.g., key presses, mouse clicks, etc.) and passes these interactions to the View and the Model. See Figure 6.3.
- **Example**
  - Figure 6.4 shows the architecture of a web-based application system organized using the MVC pattern.
- **When used**
  - Used when there are multiple ways to view and interact with data. Also used when the future requirements for interaction and presentation of data are unknown.

Model-View-Controller (MVC)

- **Advantages**
  - Allows the data to change independently of its representation and vice versa. Supports presentation of the same data in different ways with changes made in one representation shown in all of them.
- **Disadvantages**
  - Can involve additional code and code complexity when the data model and interactions are simple.

The organization of the MVC



Controller

Maps user actions to model updates
Selects view

View selection

User events

View

Renders model
Requests model updates
Sends user events to controller

Change notification

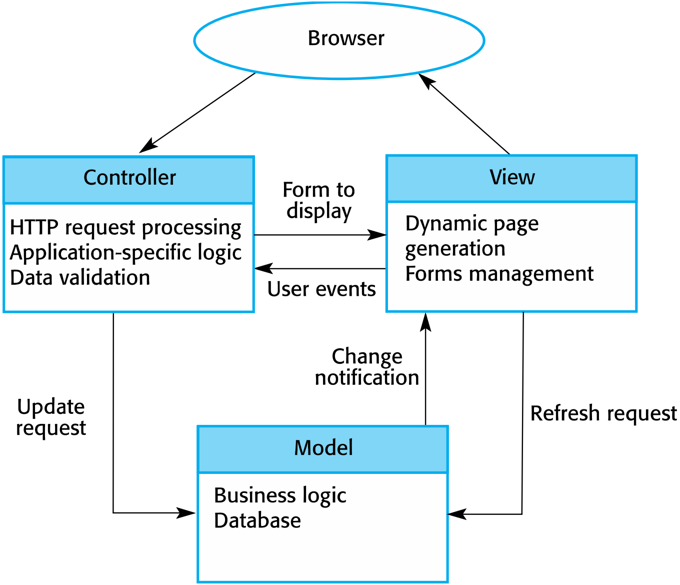
State change

State query

Model

Encapsulates application state
Notifies view of state changes

WebApp using MVC



Layered architecture

- Used to model the interfacing of sub-systems.
- Organizes 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 artificial to structure systems in this way.

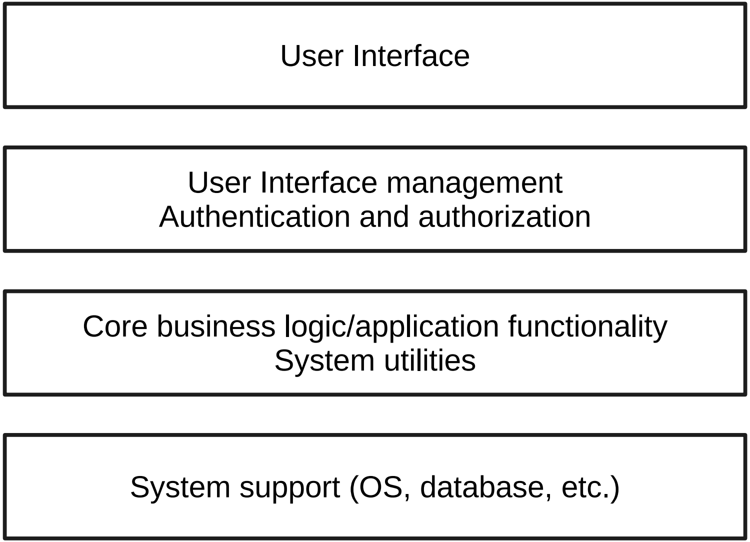
The Layered architecture pattern

- **Description**
  - Organizes the system into layers with related functionality associated with each layer. A layer provides services to the layer above it so the lowest-level layers represent core services that are likely to be used throughout the system. See Figure 6.6.
- **Example**
  - A layered model of a system for sharing copyright documents held in different libraries, as shown in Figure 6.7.
- **When used**
  - Used when building new facilities on top of existing systems; when the development is spread across several teams with each team responsibility for a layer of functionality; when there is a requirement for multi-level security.

The Layered architecture pattern

- **Advantages**
  - Allows replacement of entire layers so long as the interface is maintained. Redundant facilities (e.g., authentication) can be provided in each layer to increase the dependability of the system.
- **Disadvantages**
  - In practice, providing a clean separation between layers is often difficult and a high-level layer may have to interact directly with lower-level layers rather than through the layer immediately below it. Performance can be a problem because of multiple levels of interpretation of a service request as it is processed at each layer.

A generic layered architecture



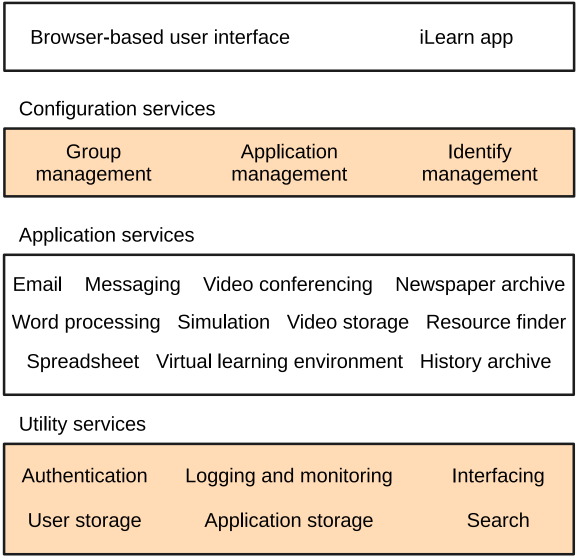
User Interface

User Interface management
Authentication and authorization

Core business logic/application functionality
System utilities

System support (OS, database, etc.)

iLearn architecture



Browser-based user interface   iLearn app

Configuration services

Group management   Application management   Identify management

Application services

Email   Messaging   Video conferencing   Newspaper archive

Word processing   Simulation   Video storage   Resource finder

Spreadsheet   Virtual learning environment   History archive

Utility services

Authentication   Logging and monitoring   Interfacing

User storage   Application storage   Search

Key points

- A software architecture is a description of how a software system is organized.
- Architectural design decisions include decisions on the type of application, the distribution of the system, the architectural styles to be used.
- Architectures may be documented from several different perspectives or views such as a conceptual view, a logical view, a process view, and a development view.

Are there any questions?