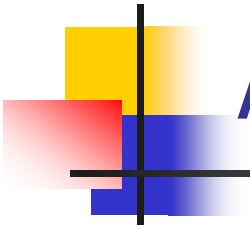
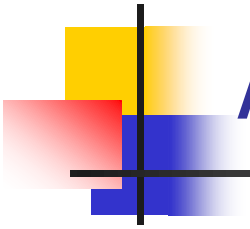


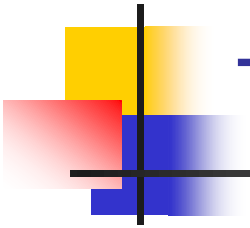
Analysis and Design Discipline contd

- Early elaboration focus on creating an initial architecture for the system (**Define a candidate Architecture**)
  - Provide starting point for main analysis work
- What if the architecture already exists?
  - Then focus on refining



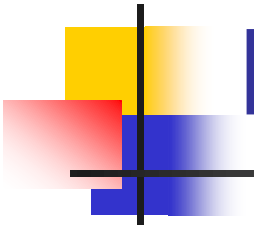
Analysis and Design Discipline contd

- Success of iterative and incremental process is predicated on early identification of the *system's architectural modules*
  - Modules - highly cohesive with minimal coupling
- Recall: Plan and control to avoid degeneration into "ad-hock hacking"



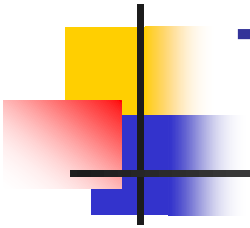
Teams Ignore Architecture Importance

- Why?
  - In a rush to get product out quickly, or
  - They don’t believe architecting gives them any real value
- Rush to code is disastrous



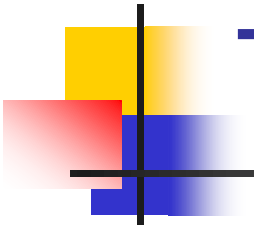
Need for Architecture

- It's development is a complicated issue
- System Architecture is developed interactively during elaboration
- Note:
  - Architecture of the proposed system does not appear in a flash
    - Exploration of the use cases



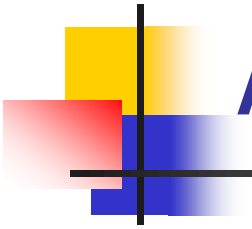
The Architecture Team

- Chief architect, assisted by a small team
- Team's Main Activities:
  - Definition of the architecture of the sw
  - Assessment of technical risks of the project
  - Definition of the order and content of successive iterations
    - Planning of each iteration



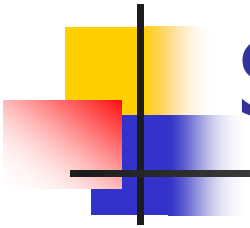
Team Activities contd.

- Provide consulting to various design, implementation, integration, and QA teams
- Assist in providing future market directions



Architectural Design Phase

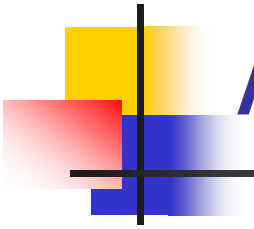
- Recall: The spec doc is a *contract* between developer and client for delivery of the software product
- Specs list all requirements the software product must satisfy
  - Specs handed to system architects and designers to develop modules of the system's architecture and its internal workings
  - What's the approach of some developers?



Some SW Development Approach

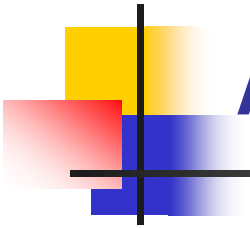
- **The Software Development Process**

1) Order the T-shirts for the Development team
2) Announce availability
3) Write the code
4) Write the manual
5) Hire a Product Manager
6) Spec the software (Writing the specs after the code helps to ensure that the software meets the specifications)
7) Ship
8) Test (the customers are a big help here)
9) Identify bugs as potential enhancements
10) Announce the upgrade program



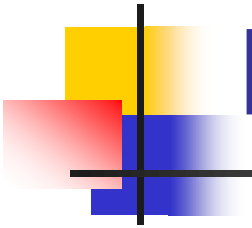
Architectural Design Phase

- Description of the systems in terms of its modules is called *architectural design*
- Includes decisions about the solution strategies
- Description of the internal workings of each module (use case) is detailed design
  - Develops detailed algorithms and data structures for each module



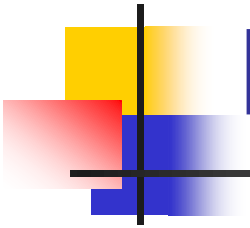
Architectural Design Phase

- Architectural design is concerned with
  - selection of a solution strategy
  - modularization of the system
- Solution strategy needs to resolve
  - client (user interface)
  - server (database) issues, as well as
  - middlewear needed to ‘glue’ client and server



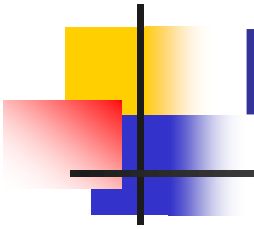
Role: Software Architect

- Role of the software architect:
  - Lead and coordinate technical activities and artifacts throughout the project
  - Establish the overall structure for each architectural view and decomposition of the views
  - Grouping of elements and the interfaces between major groupings
- In contrast to the other roles, the software architect's view is one of breadth rather than depth



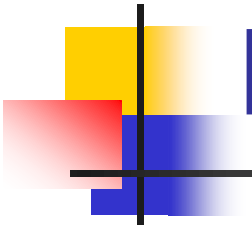
Role: Designer

- Must know use-case modeling techniques, system requirements, and software design techniques
- Designer's role defines the responsibilities, operations, attributes, and relationships of one or several classes
  - Determines adjustments to the implementation environment



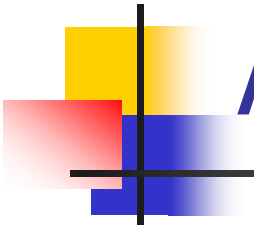
Role: Designer

- May have responsibility for
  - one or more design packages
  - Design of subsystems (and classes)



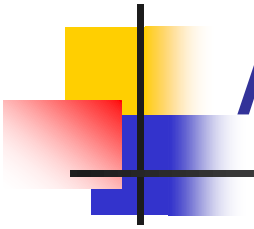
Designer Requirements

- Designer must have a solid working knowledge of
  - Use-case modeling techniques
  - System requirements
  - Software design techniques, including OO analysis and design techniques, (and UML)
  - Technologies with which the system will be implemented
- Designer must understand the architecture of the system as represented in the software Architecture Document



Analysis and Design

- Analysis and Design starts with the use-case model and supplementary specs from the Requirement discipline
  - Ends with the design model
- Design activities are centered on the architecture
- Production and validation of the architecture is the main focus of early design iterations

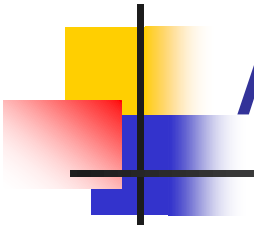


Analysis and Design

- Architecture:
  - an important vehicle for developing a good design and for increasing the quality of models

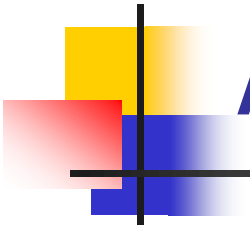
Analysis vs Design

Analysis	Design
Focuses on understanding the problem	Focuses on understanding the solution
Idealized design	Close to real code
Behavior	Operations and attributes
	Object life cycle
Functional requirements	Non-Functional requirements
A small model	a large model



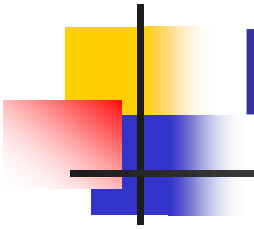
Analysis vs Design

- Difference between analysis and design are ones of focus and emphasis
- Analysis goal:
  - Understand the problem and begin to develop a visual model of the system..
- Analysis focuses on translating functional requirements into software concepts



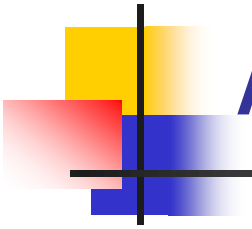
Analysis

- The idea:
  - Get a rough ‘cut’ of the objects that comprises your system
  - focusing on behavior



Design Goal

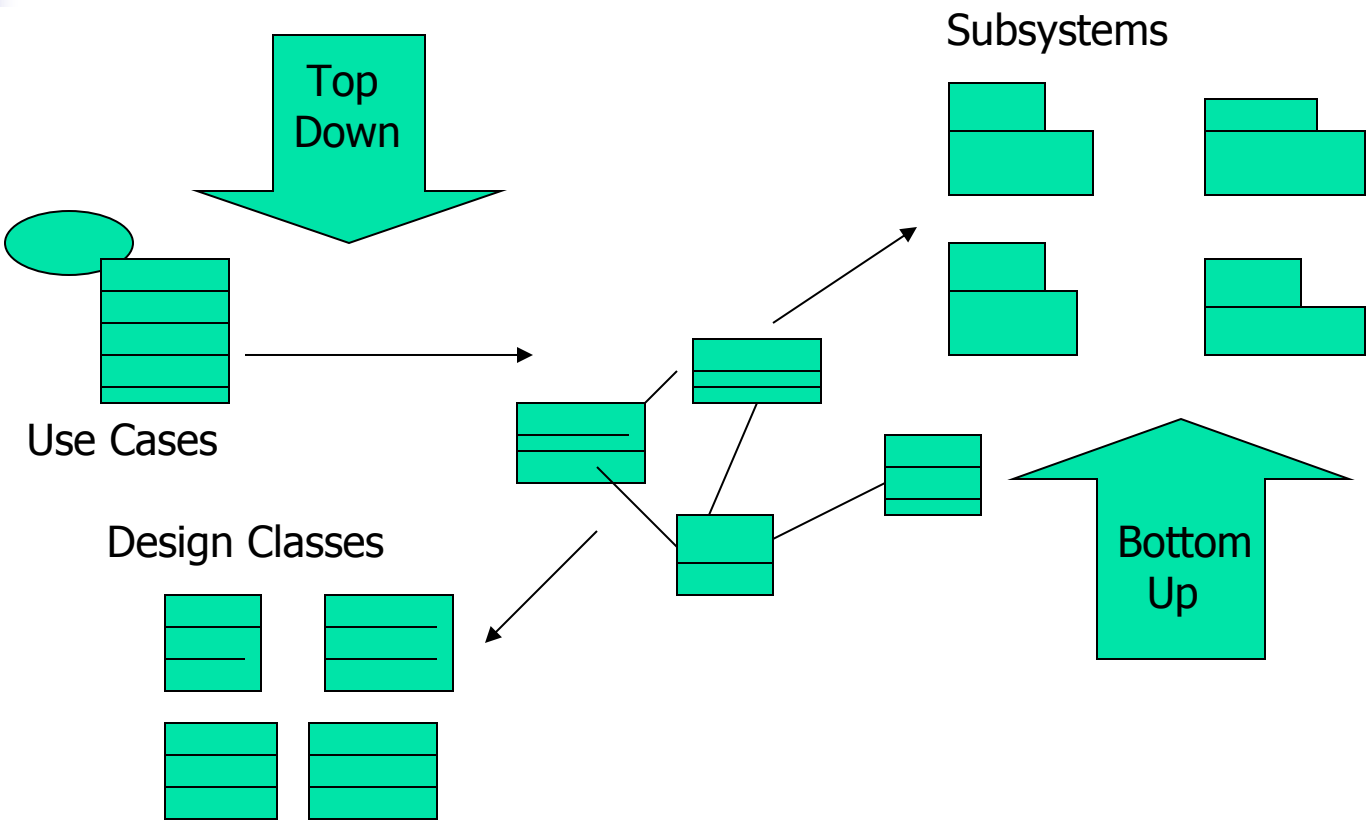
- Refine the model through seamless transition to coding
- The power of design:
  - Creates a metaphor for the real world
    - Change the nature of the problem, making it easier to solve

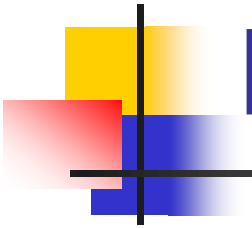


Analysis and Design

- Use case defines a middle level
- Analysis classes are not defined in a top-down or bottom-up pattern, they are in the middle
- Defining subsystems is moving up and defining classes is moving down

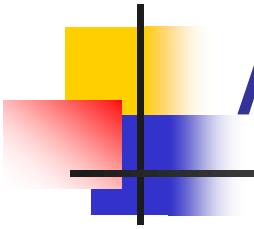
Analysis and Design





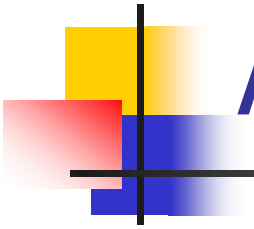
Review: A & D

- Analysis and Design is Architecture-Centric
- A system's architecture:
  - A primary artifact for conceptualizing, constructing, managing, and evolving the system
- Benefits:
  - Intellectual control over objects
  - Manages complexity and maintains system integrity
  - Provides an effective basis for large-scale reuse
  - Provides a basis for project management
  - Helps with component-based development



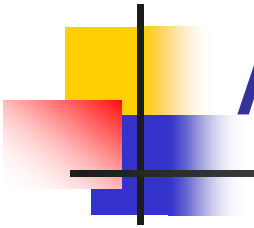
Architecture

- Software architecture encompasses a set of significant decisions about the organization of a software system
  - Selection of the structural elements and their interfaces
  - Behavior as specified in collaborations among those elements
  - Composition of the structural and behavioral elements
  - Architectural style that guides the organization



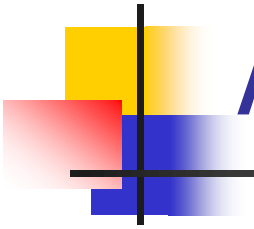
Architecture

- A concept that's easy to understand but difficult to define precisely …
  - Difficult to draw sharp line between design and architecture
  - one aspect of design that concentrates on some specific features



Architecture

- There's more to architecture than just structure…
- IEEE Working Group on architecture:
  - "the highest-level concept of a system in its environment"
- Encompasses the "fit" with system integrity - economical constraints, aesthetic concerns, style
- Not limited to an inward focus
  - takes into consideration the system as a whole

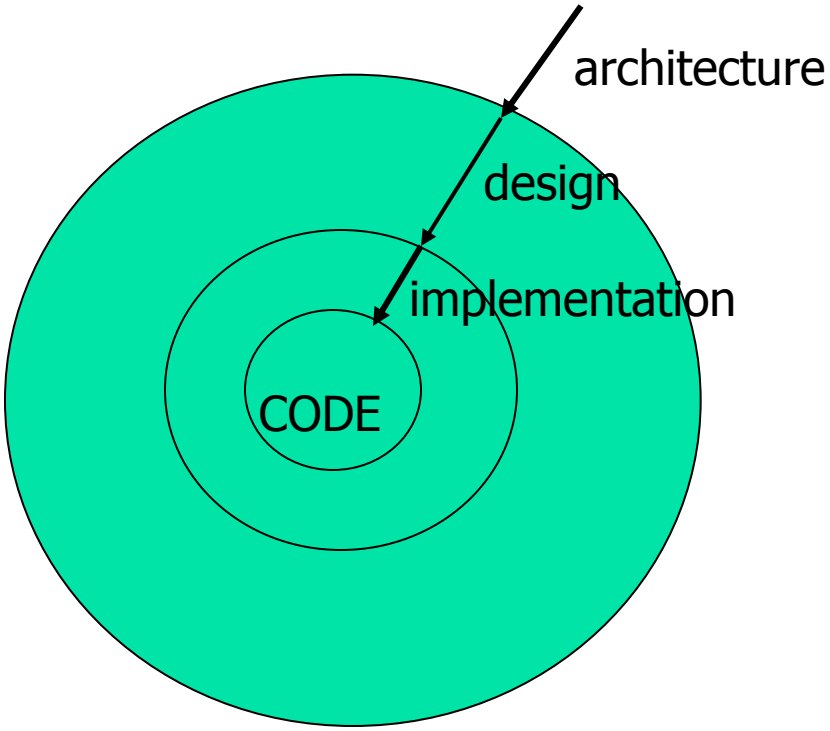


Architecture

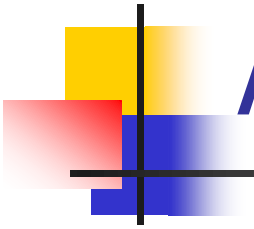
- In the RUP, the architecture of a software system is the organization of structure of the system's significant components interacting through interfaces with components composed of successively smaller components and interfaces

Architecture Constrains

- Architecture involves a set of strategic design decisions, rules or patterns that constrain design and construction

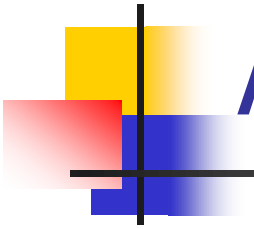


Architecture decisions are the most fundamental decisions and changing them has significant ripple effects



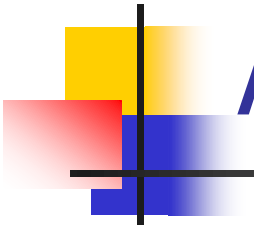
Architecture Constrains

- Architecture $\rightarrow$ initial set of constraints placed on the system
  - These are the most important constraints
  - They constitute the fundamental decisions about the design
- Architecture puts a framework around the design



Architect's Job

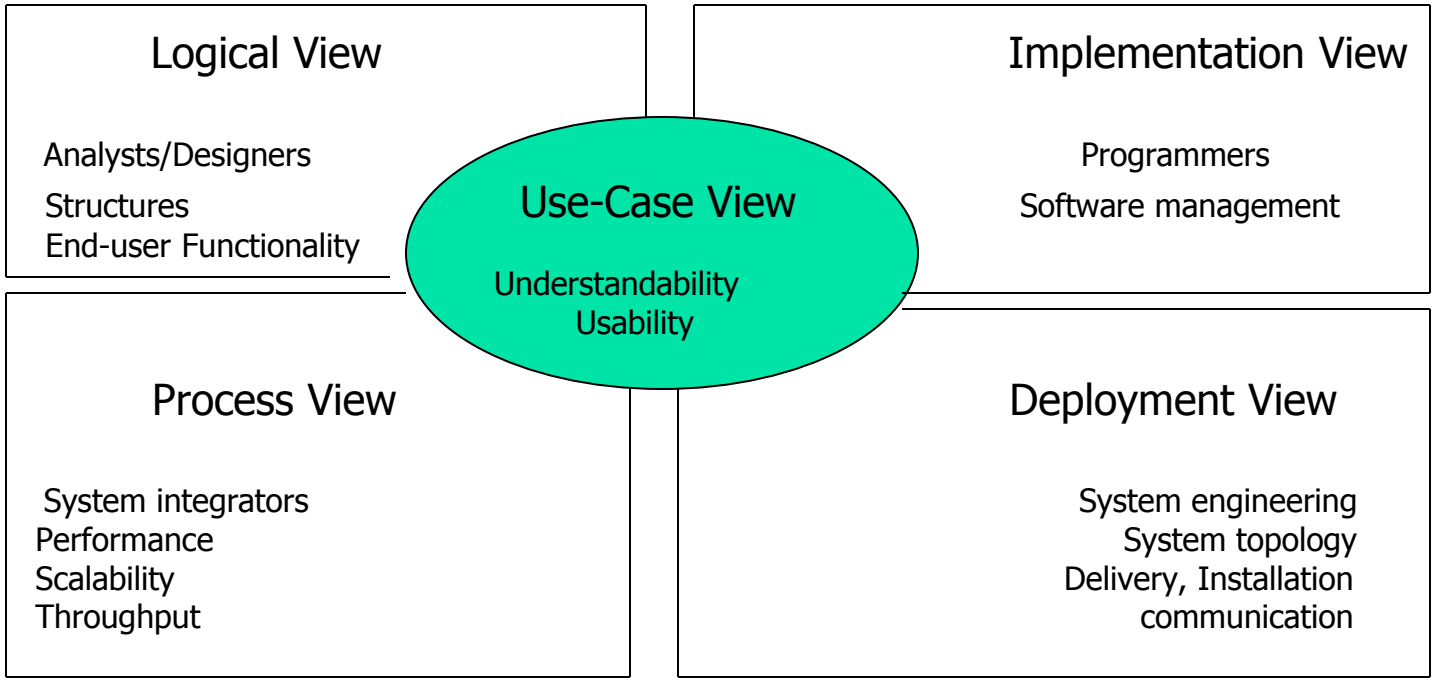
- Eliminate unnecessary creativity
  - eliminated as you get closer to code
  - Architecture constrains design, which constrains implementation
- Why?
  - During implementation, for example, creativity can be spent elsewhere

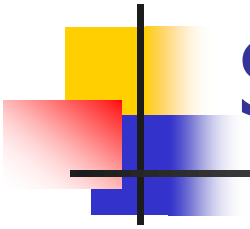


Architecture View

- Logical View:
  - Addresses functional requirements
  - Captured in class diagrams containing classes and relationships representing key abstractions

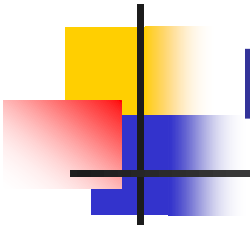
Rational's Architecture "4+1" View





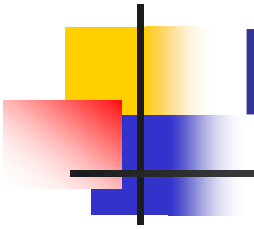
Software Architecture "4+1" View

- Architecture is many things to different people…
  - On a particular project, there are usually multiple stakeholders, each with his/her own concerns and view of the system to be developed
- The goal:
  - Provide each of the stakeholders with a view of the system that addresses his/her concerns, and suppresses the other details …RUPs "4+1" view



Rational's Architecture "4+1" View

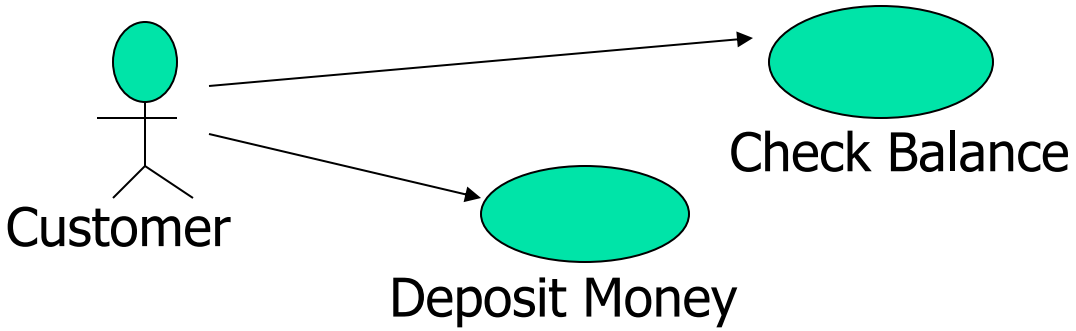
- An architectural view - a simplified description (abstraction) of a system from a particular perspective, covering particular concerns, while omitting entities not relevant to this perspective
- Not all systems require all views



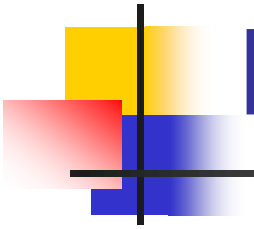
Review

- Analysis and Design is Use-Case Driven
- Benefits of use cases
  - Concise, simple, and understandable by a wide range of stakeholders
  - Help with synchronization

Review: Use-Case

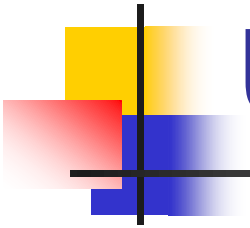


- It's one method of organizing your requirements
  - Instead of bulleted list of requirements, organize them in a way that tells how someone may use the system



Review Use Case

- Use of Use-Case
  - Make requirement more complete and consistent
  - Make for better understanding the importance of a requirement from a user's perspective
  - Shows common thread through the system when it performs certain tasks
    - Use cases - the thread that define the behavior performed by the system



Use-Case Realization (Later!!!)

- Describes how a particular use case is realized within the design, in terms of collaborating objects
- A use-case realization ties together the use cases from the use-case model with the classes and relationships of the design model
  - We'll revisit after we (re)visit classes !!!!!!