Software Architecture

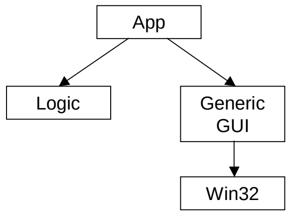
Introduction

Overview

- General Introduction
  - definitions
  - importance
  - context
- Parnas KWIC case study
- General types of systems architecture
  - monolithic
  - client/server
  - 3-tiered
- grocerygateway.com case study

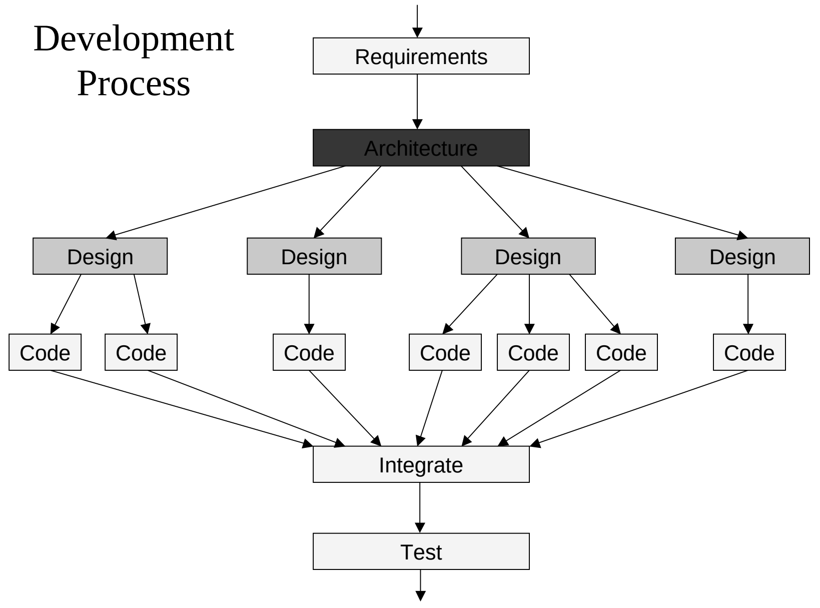
Definition

- A “software architecture” is the structure (or structures) of a system, which comprise
  - software components,
  - the externally visible properties of those components,
  - and the relationships among them.



- Architecture defines “components”
  - an abstraction
  - suppresses details not pertinent to its interactions with other components
- An architecture can comprise more than one structure
  - modular structure (calls/uses)
  - process structure (invokes, communicates with, synchronises with)
  - physical structure (libraries, DLL’s, processors)
  - inheritance structure (inherits)
  - …

Development Process



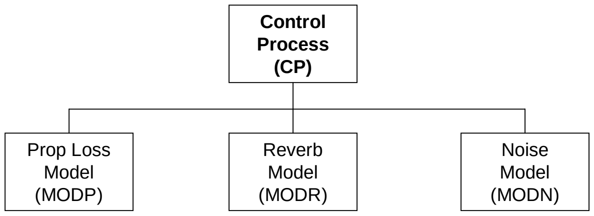
Why is architecture important?

- Manifests early design decision
  - most difficult to get correct and hardest to change
  - defines constraints on the implementation
  - inhibits or enables quality attributes
- Defines a work-breakdown structure
  - organization (especially important for long-distance development)
  - estimation
- A vehicle for stakeholder communication
  - an architecture is the earliest artefact that enables the priorities among competing concerns to be analysed
- Reviewable
  - architectural errors are vastly more expensive to fix once a system has been coded
- Can serve as a basis for training new developers

Architecture process steps

- create the business case
- understand the requirements
- create the architecture
- represent and communicate the architecture
- evaluate the architecture
- implement based on the architecture
  - ensuring conformance
- enhance/maintain based on the architecture
  - ensuring conformance

How do we describe an architecture?



- What is the nature of the components?
- What is the nature of the links?
- Does the layout have any significance?
- How does it operate at runtime
  - Dataflow
  - Control flow
- Can we evaluate this architecture?

Functionality & Quality Attributes

- Functionality usually takes 1st place during development.
- Systems are more frequently re-designed not because they are functionally deficient, but rather because
  - They are difficult to maintain
  - Difficult to port
  - Won't scale
  - Too slow
  - Too insecure
  - Not fault tolerant

Architectural Means of Achieving Quality

- Two questions
  - What structure shall I employ to
    - Assign workers
    - Derive a work breakdown
    - Exploit pre-packaged components
    - Plan for modification
  - What structure shall I employ so that the system, at runtime, fulfills its behavioral and quality attributes.

Two Structures

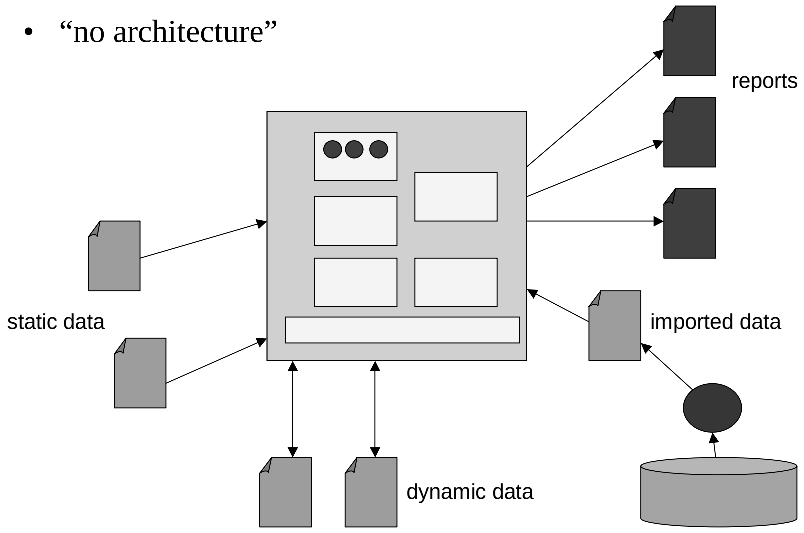
- Modular structure
  - Purely static
  - Disappears at run-time
- Structures that survive through execution
  - E.g., pipes, processes, networks, …
- Both views need to be considered (not the same)

System Architecture Choices

- Monolithic
  - 1 large program, imports/exports data
- Client/Server
  - collection of clients, updates database
  - “fat client”
- 3-tiered (n-tiered)
  - collection of clients, 1 mid-tier process for “business rules”
  - “thin client”
- Peer-to-Peer
  - distributed collection of servents/clervers

Monolithic Systems

- "no architecture"

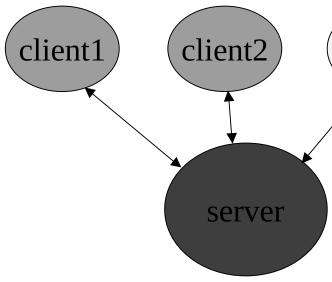


Examples

- Most programs you deal with day-to-day
  - word processing
  - spreadsheets
  - powerpoint
  - e-mail (?)
  - inexpensive accounting packages
  - development environments
  - compilers
  - most games
    - (not *Combat Flight Simulator*)
- Large, corporate batch systems
  - payroll
  - reports
  - Descartes route planning

Client/Server

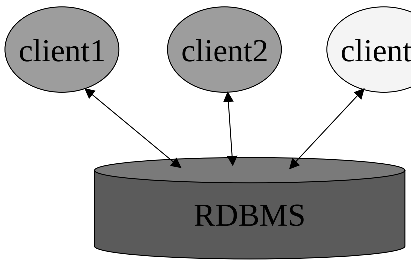
- In general, any application where multiple clients connect to a single server.



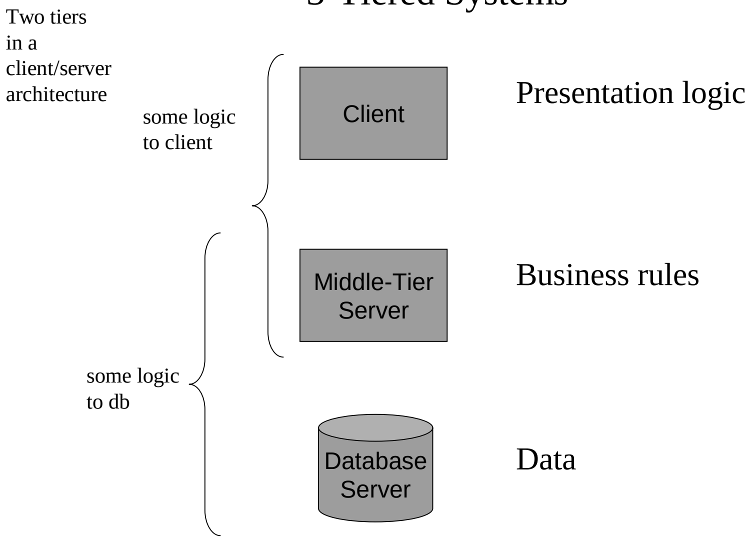
- one client program (most typical)
  or
- multiple client programs

Relational Databases

- Most common client/server program is where the server is a relational database server.
  - warning: some use the term client/server to refer to this usage exclusively (we won't).



3-Tiered Systems



GroceryGateway.com

Case Study: Distributed Internet Application Design

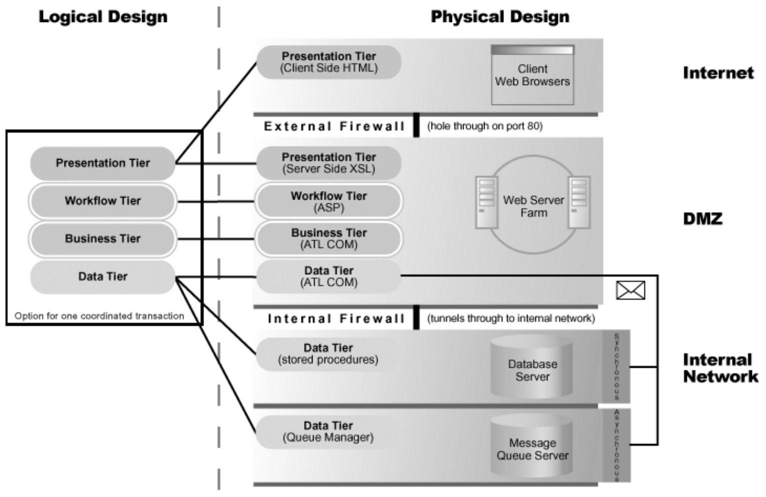
Introducing Grocery Gateway

- Founded in late 1997 out of a basement
- Over $70 million in private financing in the last 18 months
  - Leading Canadian venture capital and institutional investors
- Over 60,000 registered customers
- Employee base has grown from 30 to over 400 in the past twelve months
- Software Development Group is 25 people divided into four different teams:
  - web development,
  - server components,
  - database development,
  - and software test

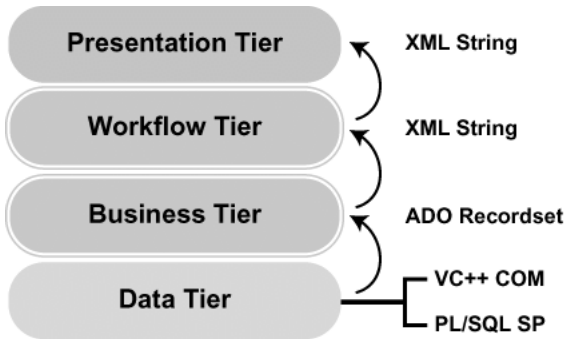
What they do

- Sell products over the Internet and we deliver them directly to your door
- Use groceries to initiate the relationship and create the pipeline to your home
- Leverage this pipeline to introduce complimentary products
- Setting the standards in:
  - Online merchandising
  - Single item picking/packing and home delivery operations
  - Systems integration
  - Customer service

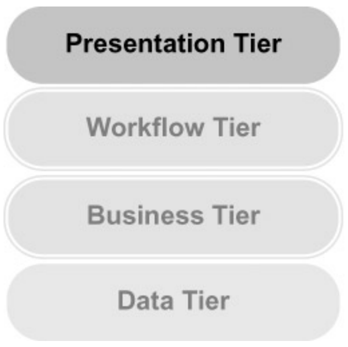
GG: Logical and Physical Overview



Application Design

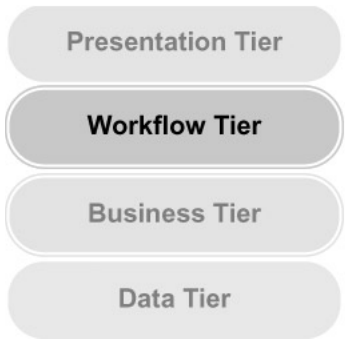


GG: Presentation Tier



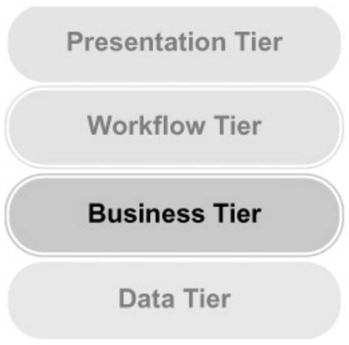
- A closer look…
- Transform XML strings into presentation format (e.g., HTML)
- Communicates with workflow tier – accepts XML string

GG: Workflow Tier



- A closer look…
- Manages shared data cache
- Manage user context data (preferences, personalization)
- Prepare working data for transactions (e.g., HTML form to XML)
- Communicates with presentation tier – handles calls and publishes data
- Communicates with business tier – get data from business tier, executes transactions from business tier

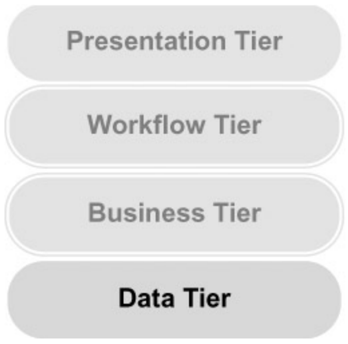
GG: Business Tier



- A closer look…
- Enforces business rules
- Converts ADO Recordsets to XML strings
- May handle exceptions raised by data tier
- Logs outcome of transactions
- Enforces security requirements
- Communicates with the workflow tier – sends XML strings
- Communicates with data tier – manages one or more transactions with data tier components

Architecture Intro. SE II 25

GG: Data Tier



- A closer look…
- Executes data transactions
- Enforces ACID constraints
- Data stores include databases, and the messaging sub-system
- Communicates with the business tier – passes Active Data Objects (ADO) recordsets
- Implementation is stateless COM objects and stored procedures

Architecture Intro. SE II 26

Some of the design challenges they encountered…

- Performance Monitoring
  - Where is the scalability bottleneck?
  - How to improve the response time (performance) for a feature?
- Run-time Monitoring
  - How to confirm that the application is working?
  - If something isn't working, how to figure out where the problem is?