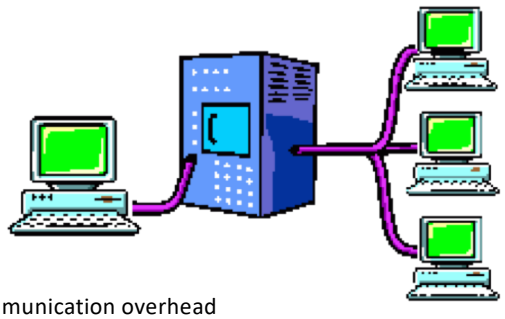
IT Systems Evolution

Monolithic Systems

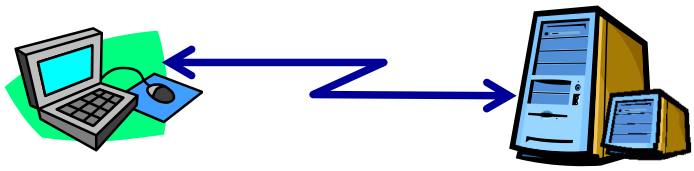
- One program or system does everything
  - exemplified by mainframe systems



  - no communication overhead
  - maximal coupling of components
  - throughput dependent on hardware performance
  - doesn't scale well

Client/Server (2-Tier)

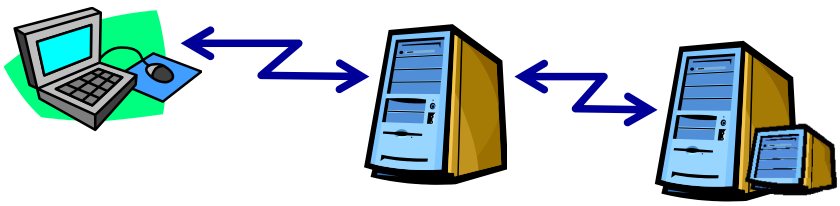
- Server provides back end services
  - typically PC client to Unix/Linux server



- business logic distributed between client and server
- server often no more that a database server
- small communication overhead
- server performance is system bottleneck
- expensive administrative overheads if fat client needs updating

3-Tier Architecture

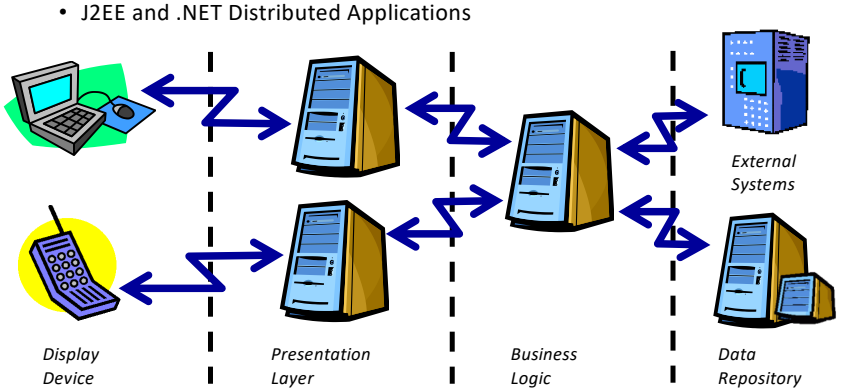
- Middleware server provides business logic
  - typically web browser client, Unix/Windows middleware



- separate backend server provides database access
- thin client is simply a presentation device
- large communication overhead
- scalable by adding additional middleware servers
- database performance is system bottleneck

N-Tier Architecture

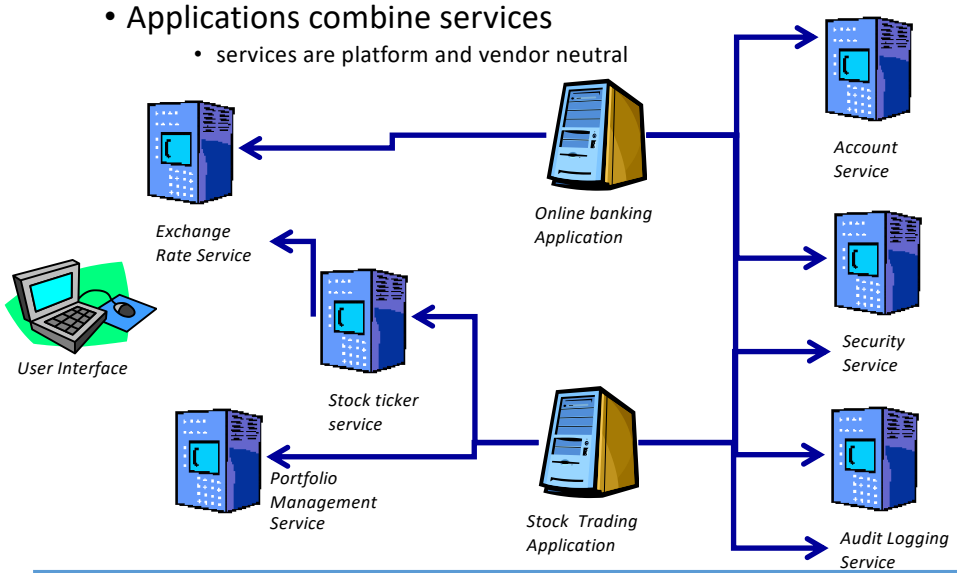
- Many specialised tiers
  - J2EE and .NET Distributed Applications



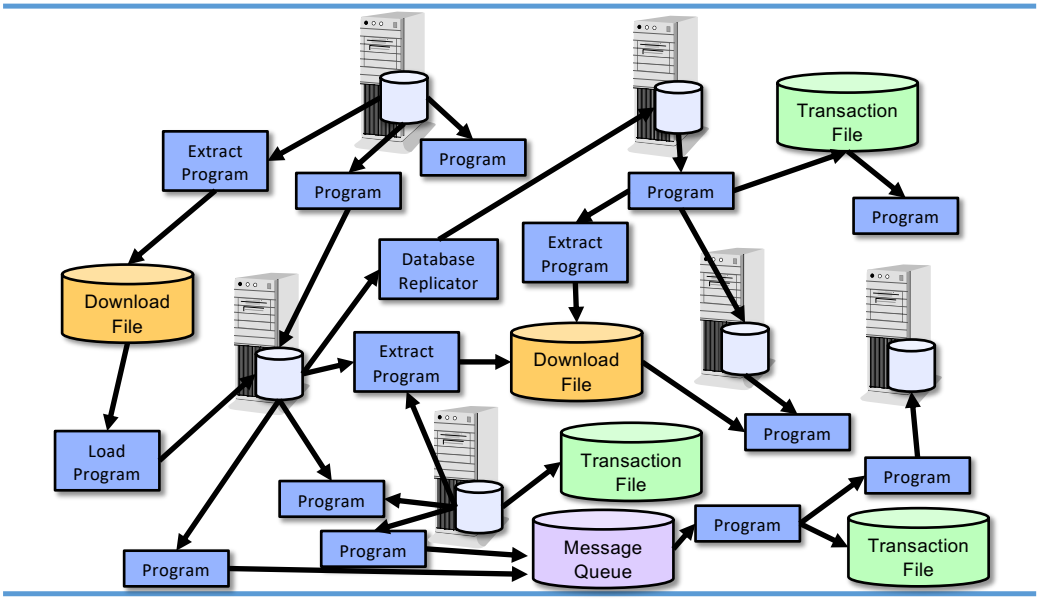
  - very large communication overhead
  - backend network performance is often the bottleneck
  - different presentation servers for client devices

SOA

- Applications combine services
  - services are platform and vendor neutral

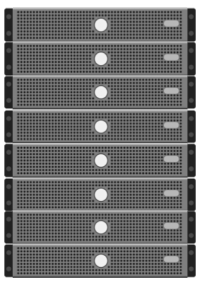


Infrastructure for SOA



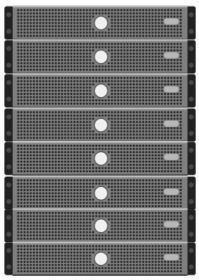
Infrastructure Technologies

- Servers
  - Windows
  - Linux



Infrastructure Technologies

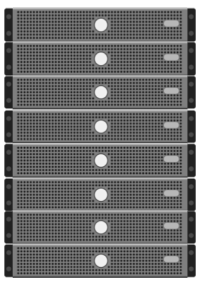
- Servers
  - Windows
  - Linux



- Mainframes

Infrastructure Technologies

- Servers
  - Windows
  - Linux

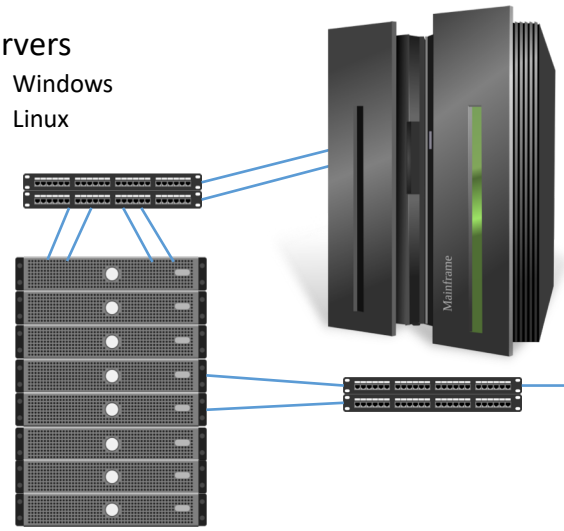


- Mainframes

- PCs & Mobile

Infrastructure Technologies

- Servers
  - Windows
  - Linux

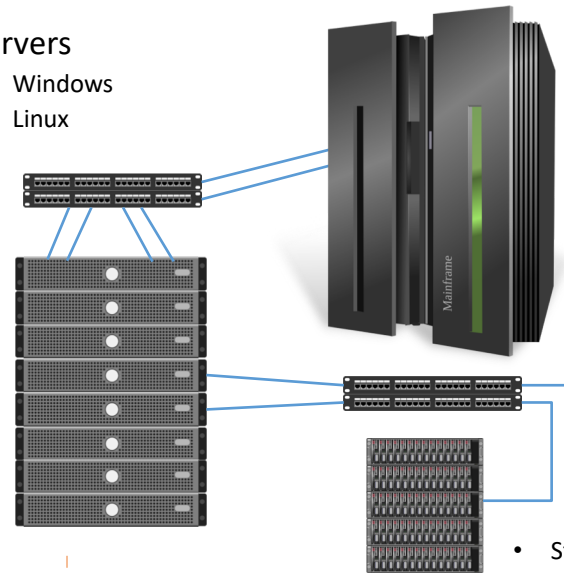


- Mainframes

- PCs & Mobile

Infrastructure Technologies

- Servers
  - Windows
  - Linux



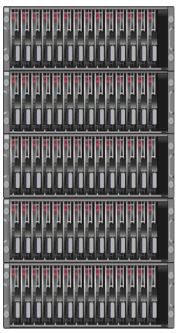
- Mainframes

- PCs & Mobile

- Storage Networks

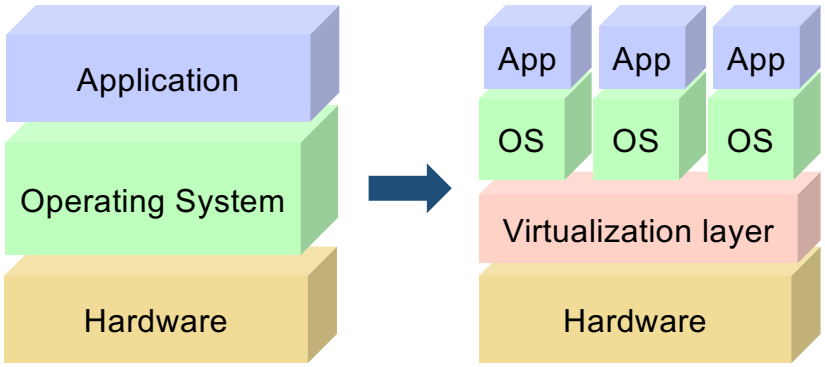
Enterprise Storage

- Local Disk
  - boot disks
- Storage Area Networks
  - aka SAN
  - linked centralized storage for servers
  - utilize fast, dedicated, fibre networks
- Network Attached Storage
  - aka NAS
  - used for less demanding performance scenarios
  - cheaper option
- Direct Attached Storage
  - DAS
  - used for virtual application servers



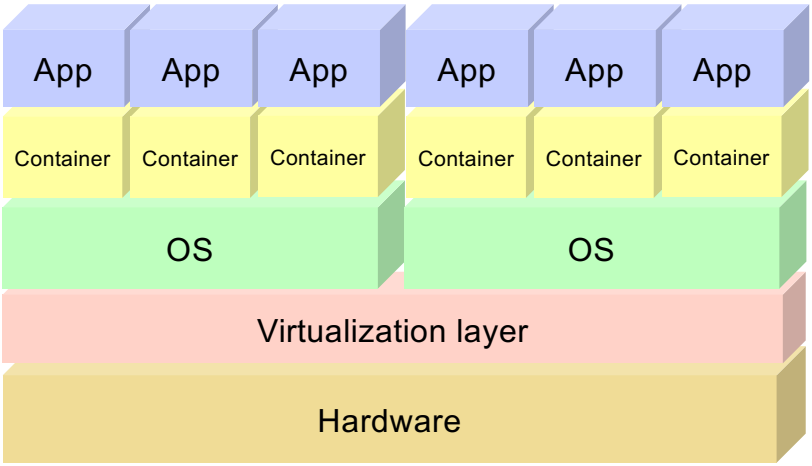
Virtualisation Technologies

- Server Virtualization
  - creation of software version of server
  - allowing hardware to run multiple OS instances



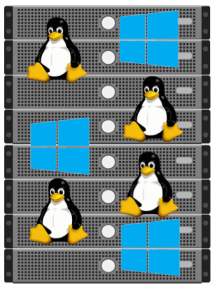
Virtualisation Technologies

- Containers used for deployment
  - such as Docker



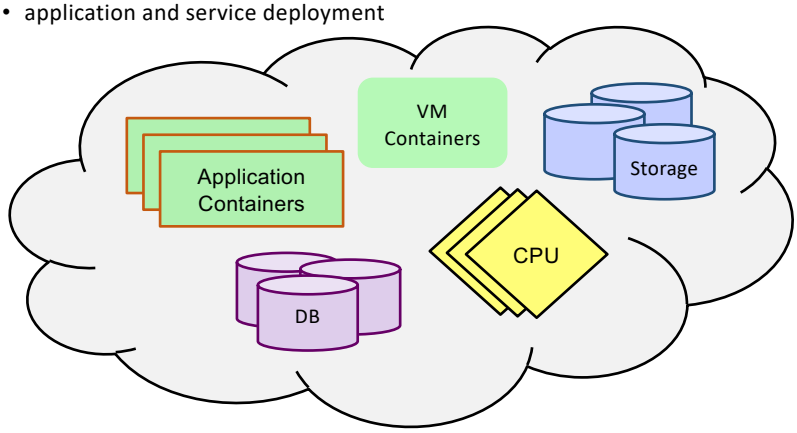
Virtualisation Technologies

- Virtualization is key to
  - reducing server sprawl
  - improving plant manageability
  - dynamic load balancing
  - extending self-service (infrastructure on demand)
  - allowing users to get computing capacity when they need it
  - letting users release it when they don't need it
  - transparent maintenance operations
  - encouraging innovation



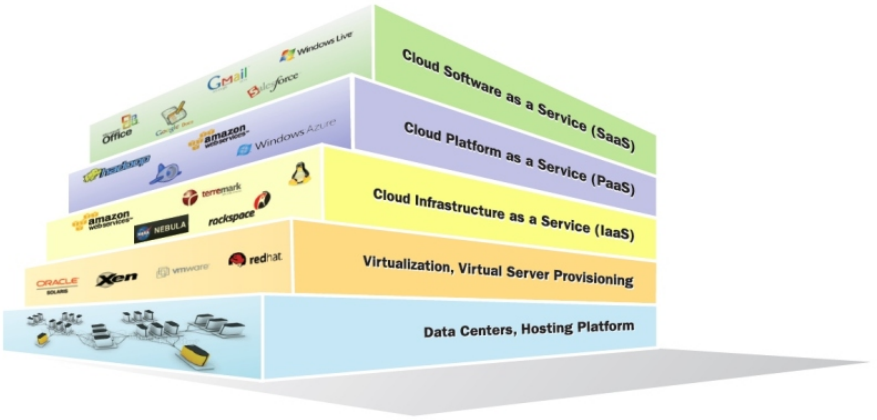
Cloud Technologies

- Provides a shared environment for
  - computation, storage, databases
  - application and service deployment



Cloud Technologies

- Cloud can have multiple meanings and service models
  - numerous internal and external providers

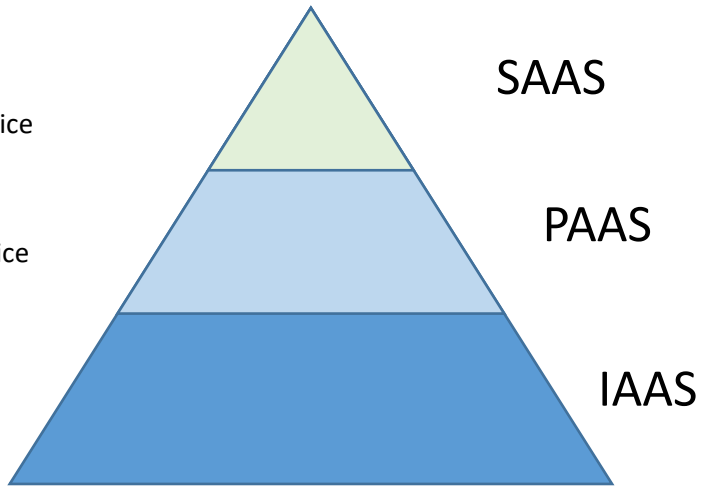


Cloud

- A paradigm allowing on demand (network) access to shared computing resources
- A model for managing, storing and processing data "online" via internet technologies
- Flexible
  - "Use it when you need it"
- Scalable
  - Elasticity

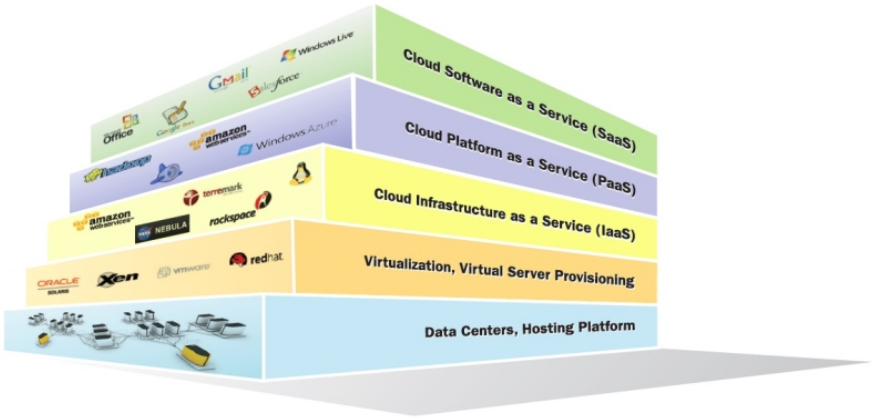
Cloud Delivery Models

- "…as a service"
- SAAS
  - Software as a service
- PAAS
  - Platform as a service
- IAAS
  - Infrastructure as as service



Cloud Delivery Models

- Numerous internal and external providers



Data

- Critical part of enterprise infrastructure
  - key to all activities / operations
- Data may be sourced externally
  - such as market data and client information
- Most data generated internally
  - such as trade related information
- Data retention
  - historical data also a significant aspect
- Big challenges
  - quality & veracity, volume, velocity, variety, availability

What Data?

- Reference Data
  - foundation of all commercial data
  - Clients / Counter Parties / Legal Entities
  - Currencies
  - Calendars
  - Product Data
  - Employee & Alumni

- Market Data Feeds
  - real-time data and information related to pricing
  - market prices
  - economic data
  - news
  - legal data
  - government data

What Data?

- Transaction Data
  - the commercial activity of the firm
  - cover trades, legal contracts, agreements, loans, cash balances

- Derived Data
  - calculated data based on firm activity
  - e.g. financial P&L, Desk Risk measures etc.
  - customer intelligence, predictive analysis
  - compliance detection

- Management & Corporate data
  - management reporting
  - public & regulatory filings

What Data?

- Client & User Activity
  - every customer and user action
  - e.g. customer interactions, website activity, app usage
  - as well as Financial Adviser activity

- Productivity & Communications
  - employee and client communication and records
  - 100 billion emails in archive
  - IM, voice calls & recordings
  - videos and webcasts

- Log & Audit Data
- Infrastructure & Security Data
  - entitlements, technology assets

What Data?

- Not forgetting ...
  - all of the internal disks in firm computers
  - plus disks connected to VMs
- Daily Backups
  - need to satisfy regulatory requirements
  - duplicate copies are sent offsite for use in disaster recovery scenarios
- And more ...
  - fraud detection evidence
  - system dependencies / configurations
  - data flows etc.

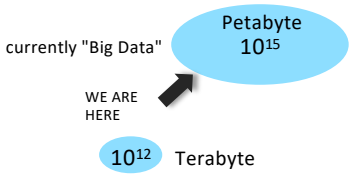
Data Volume

- Volume – its "Big"

10^{12} Terabyte

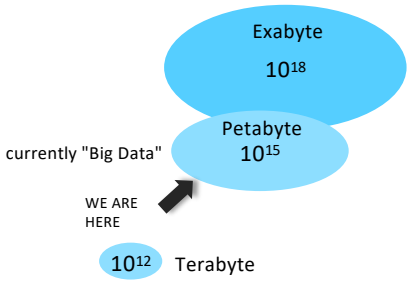
Data Volume

- Volume – its "Big"



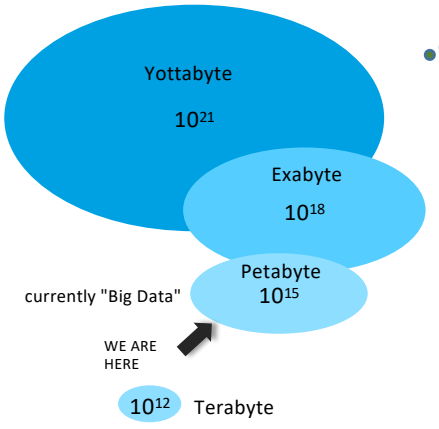
Data Volume

- Volume – its "Big"



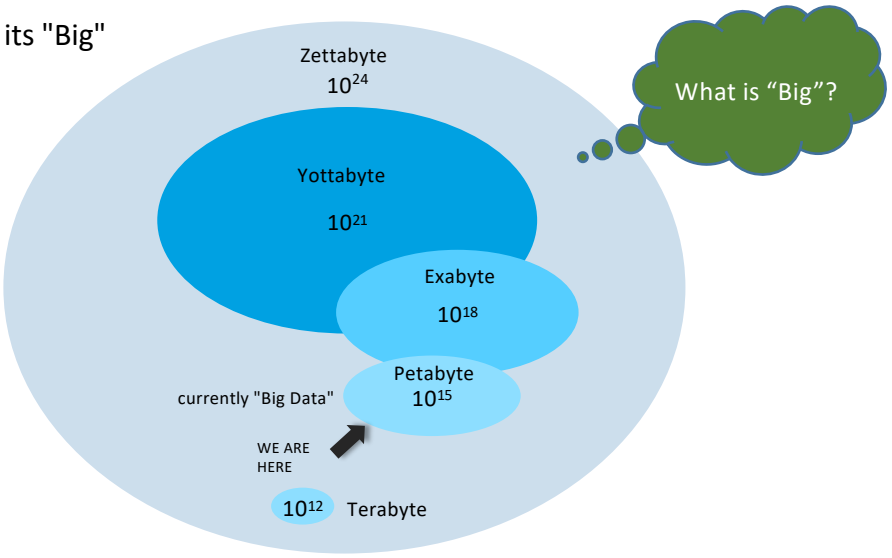
Data Volume

- Volume – its "Big"



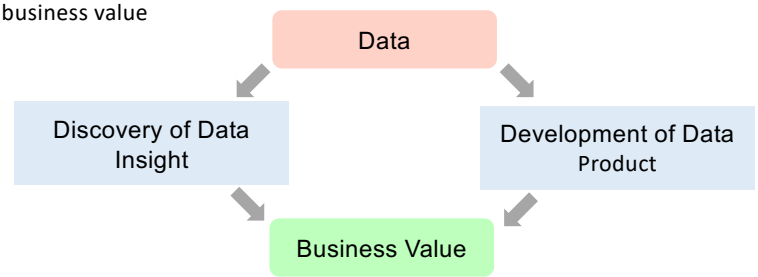
Data Volume

- Volume – its "Big"



Data Science

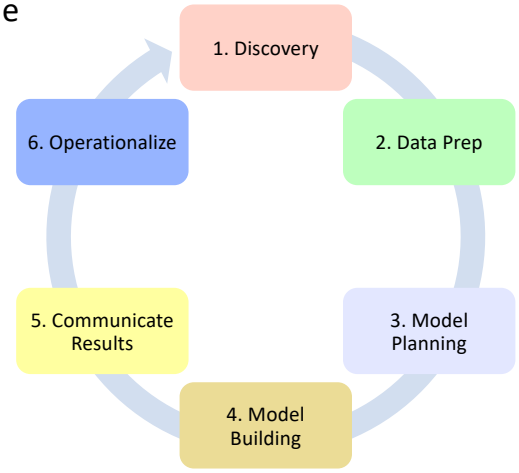
- Interdisciplinary field
  - aims to extract knowledge or insights
  - from data in various forms
  - to generate business value



- Combines tools and techniques from
  - mathematics and statistics
  - data analytics
  - machine learning

Data Analytics

- Process of examining data to help with decision making
- Data Analytics Lifecycle

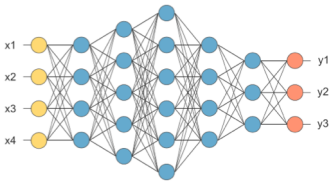


Data Analytics

- Variety of technologies available
- Python
  - using NumPy, SciPy, MatplotLib, pandas etc.
- Java / Scala
  - with Hadoop, Hive, Pig
- R
  - language focused on data analysis
- MATLAB

Machine Learning

- Typically uses training sets to generate
  - predictive or classification systems
- Common to use Neural Networks
  - long history in AI
- Python
  - TensorFlow (open source API from Google)
- Java / Scala
  - Apache Spark Machine Learning
- Big Data Analytics
  - typically combines Data Analytics and Machine Learning



Environments

- Modern Software is normally developed, tested and deployed in distinct hardware/software environments
  - Dev
    - Where the development is done
    - Enough for the developers to do their job
  - Test
    - Where testing takes place
    - Integration/UAT mostly - Dev env will allow for unit testing
    - Should be as similar as possible to Production
  - Production
    - Where the software runs
    - Should be highly protected
    - Access tightly controlled
    - Changes tightly managed

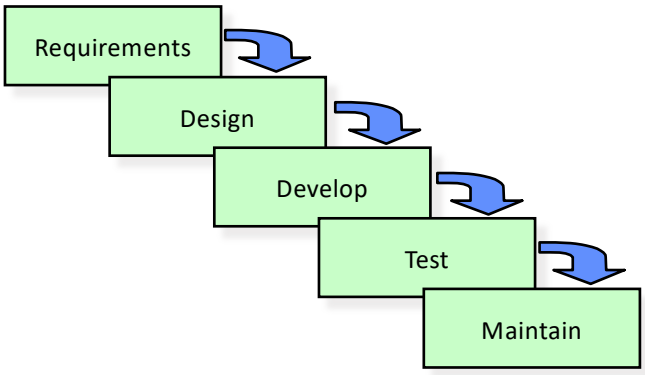
Code & Fix

- The default development life cycle
  - aka “code like hell and pray”
  - usually a sign that development is out of control
- Approach
  - start with a general idea of what is wanted
  - minimal analysis and design
  - code, refactor, debug until it works
  - good for practically no overheads

- Why it doesn’t work
  - no means of assessing progress (no project plan)
  - can’t gauge quality, security or risk
  - problems typically appear late in the cycle and are expensive to fix
  - end product is never truly known
- Code & Fix is dangerous on any but a tiny project

Waterfall Model

- Development flows through a series of phases
  - each phase must complete before next one starts
  - minimal feedback and iteration
  - formal or informal review at the end of each phase should determine whether to move onto the next phase

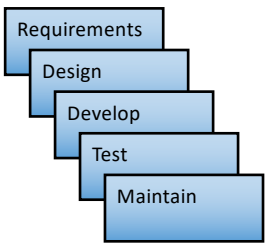


Waterfall Model

- Good for
  - projects where requirements are stable
  - implementation using well understood technical methodologies
  - tackling complexity in an orderly fashion
  - providing structure to projects
  - technically weak or inexperienced staff
- Problems
  - lacks flexibility, especially when dealing with incomplete or changing requirements
  - requires solutions to be fully understood (analyzed) and designed before coding can begin
  - tends to increase risk
  - delays problem identification
    - little indication that the system will work until late in the development life cycle
  - minimal feedback and iteration

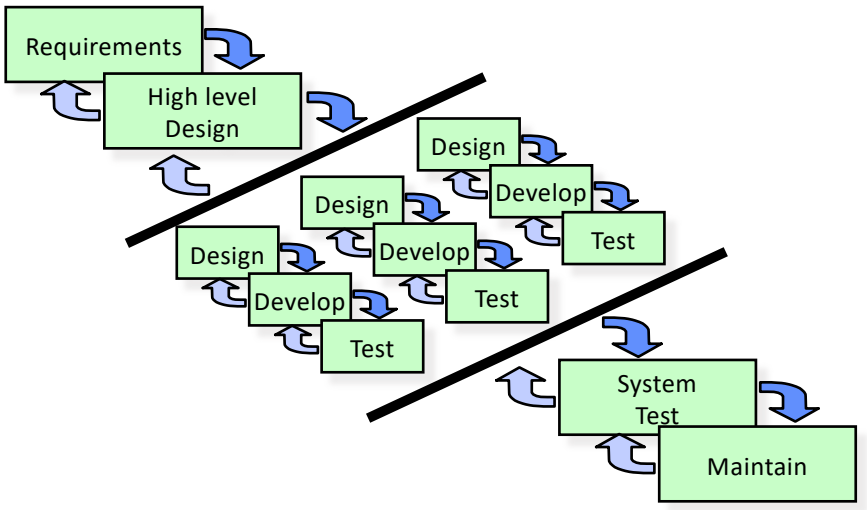
Sashimi Waterfall

- Overlapping development phases
  - next phase may be started before current one completes
- Advantages
  - allows exploration of next phase
  - can have several phases running at once
  - good if being done by one team
- Problems
  - can end up with everything happening together
  - milestones can be ambiguous
  - may not be easy with separate teams



Waterfall with Subprojects

- Retains top level control but allows flexibility



Iterative and Agile SDLC Models

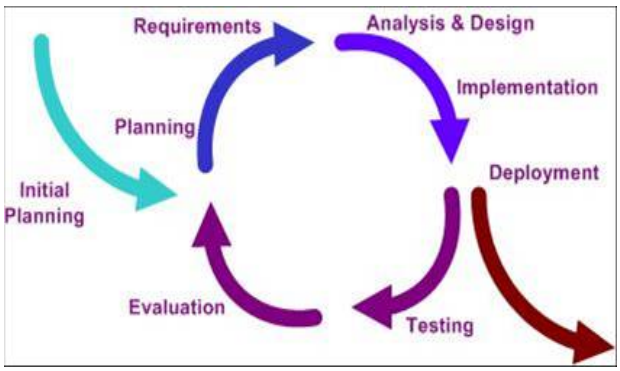
- Reflect more pragmatic approach to development
  - requirements not clearly understood
  - requirements change as business changes
  - successive iterations refine solution
- Iterative models
  - spiral model
  - staged delivery
  - evolutionary prototyping
  - time boxing
  - test driven development
  - extreme programming
  - Rational unified process

Why Do We Need Agile Development?

- Ziv's Uncertainty Principle
  - uncertainty is inherent and inevitable in software development processes and products
- Humphrey's Requirements Uncertainty Principle
  - the requirements will not be fully known until the users have had time to use the system
- Wegner's Lemma
  - it is not possible to completely specify an interactive system

Iterative Model

- A flexible, non-linear, approach to development
  - implementation converges towards final system
  - requirements may evolve with prototypes
  - each cycle has some analysis, design code & test
  - each cycle delivers something of use to the client



Spiral Model