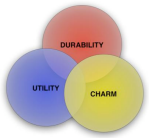
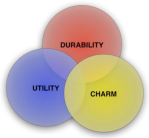


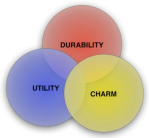
Class 5 SSW565

Gregg Vesonder
Stevens Institute of Technology



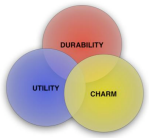
Roadmap

- Logbook
- More architecture: ilities
  - Performance
  - Concurrency
  - Security
- Laws of Simplicity
- Service Oriented Architectures
- Reading this week: none
- Reading next week: Starr & Zimmerman (moved to next class)



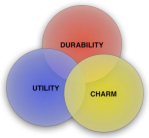
Key Dates

- Thursday class June 18th - MidTerm
- Thursday class June 25th
- June 29th logbooks due
- Final July 20th



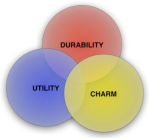
Clarifications & Additions BitTorrent

- Peer to peer
- Each client supplies pieces of data (64K - 1M) it receives to other clients
- Clients can do almost anything: preparing the material, tracking it, distributing it



BitTorrent - Sharing Files

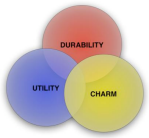
- First create a "torrent" containing meta data and the tracker which coordinates downloads among peers (this is the central point - the BitTorrent software can be downloaded from many sites)
- Private BitTorrent trackers can require registration. Can track activity
- Trackers do not have to list the metadata - the list of BitTorrent files that have to be downloaded for a given artifact, but many do
- Amazon, World of Warcraft, … use BitTorrents
- http://www.bittorrent.com/



Protocols

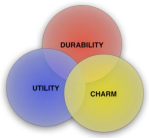
A protocol determines the format and order of messages exchanged between two or more communicating entities. As well as the actions taken on the transmission and/or receipt of a message or other event

Kurose and Ross(2003)



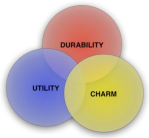
Logbook

- SAC, Simple Accessible Code or Simple, Accessible, Comprehensible



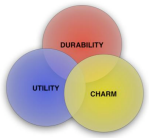
Tiers and Sessions

- Probably the most popular architecture used in Enterprise Applications:
  - Involve persistent data
  - Usually a lot of data gigs to terabytes
  - Many people access it concurrently
  - Many interface screens for different uses and different user populations
  - Lots of batch processing
  - Complex data base schema
  - Performance is an issue
- Good platform to discuss ilities at the same time



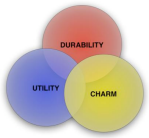
History of Tiers and Sessions

- From 1 tier (big iron + thin client with specialized software)
- To two tier with (Powerbuilder|VB|Delphi) fat clients and database
- To multi tier or layer with UI + domain logic + database -- popularized by the web
- Tier usually refers to that tier being on dedicated hardware, multiple layers can share hardware



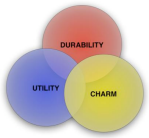
Layer Pros and Cons

- Pros:
  - Understand layer as a coherent whole w/o understanding other layers
  - Ease of substitution of alternate implementations
  - Minimize dependencies between layers
  - Locus for standardization
  - Great for reuse
- Cons:
  - Layers can encapsulate some but not all things well - cascading layers
  - Extra layers can harm performance
  - Often hard to define layers in terms of responsibility that maximizes the benefits



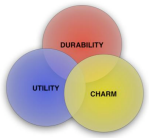
Back to Layers

- The three layers: presentation, domain logic and data source
  - Presentation how to handle interaction between user and software: rich client graphics UI or a web browser or enhanced web browser (application command line/menu based for same application would have 2 forms of presentations - menu based for the novice and command line for the expert)
  - Data source logic is database but also transaction monitors, other applications
  - Domain logic (Business "logic") calculations based on inputs, stored data, validation, …
- Separation is tricky - choose most appropriate, but do have some kind of separation
- Also a dependency issue -- domain and data source should NEVER be dependent on presentation



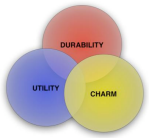
Where to Process Layers

- Client? Desktop? Server?
- Simplest - run everything on servers with a web browser
  - But what about responsiveness and disconnected operations?
- Running a rich client means running presentation on client, web interface on server
  - PERL documentation
- With HTML, every decision needs a roundtrip
- Web presentation if you can and rich client if you must
- Domain Logic:
  - Placing it on the desktop means you have more units to upgrade and maintain
  - Splitting across desktop and server is worst because you will have to track where everything is



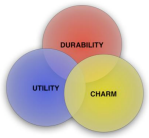
Performance Engineering

- Response time is the amount of clock time required to respond to a request
- Latency is the minimum amount of time to receive any response - time for the system to respond after hitting carriage return or a ping. Application programmers usually cannot control this- reason for minimizing remote procedure calls
- Throughput of a system is the number of requests that can be processed in a specified time interval
- Performance is the degree to which a software system or component meets its objectives for timeliness (other concerns too - footprint, memory usage, …)
- Load is a measure of how much stress a system is under - how many users are logged in, how many processes -- used as a context for other measures such as response time - response time is 0.5 sec with 10 users, 2 sec with 20 users.



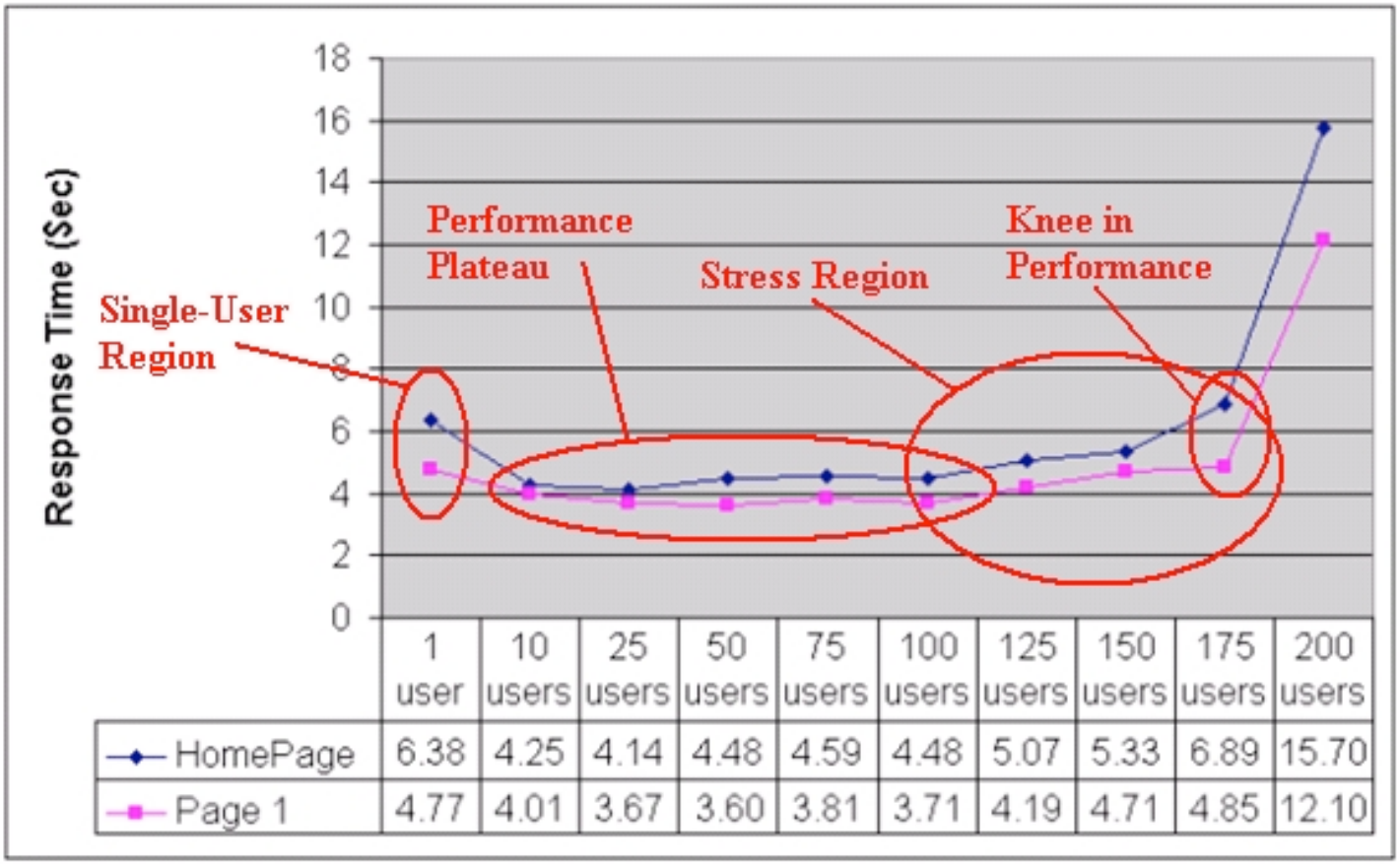
More Performance Terms

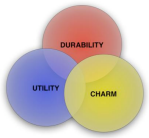
- Load sensitivity is an expression of how the response time varies with load
- Greater sensitivity results in degradation, we will see curves soon
- Efficiency is performance divided by resources
- Capacity is maximum effective throughput or load
- Scalability is the ability of a system to continue to meet its response time of throughput objectives as the demand increases and how adding resources affects performance
  - Vertical scalability more power to a single server, horizontal scalability adding more servers
  - Scalability in enterprise apps is usually more important than capacity or efficiency
  - Where the knee of the curve hits you changing from linear to exponential (usually due to a resource such as cpu, disk, network, sockets, threads, …) IMPORTANT TO KNOW WHERE THE KNEE IS AND WHERE YOUR TARGET LOAD FALLS



Example Performance Curve

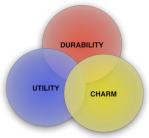
http://www.awprofessional.com/articles/article.asp?p=391645&seqNum=1&rl=1





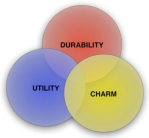
Performance Failures

- Consequences: damaged customer relations -> project failure
- "…performance problems are most often due to fundamental architecture or design factors rather than inefficient coding." (Smith and Williams, 2001)
- The "Fix it later" strategy is due to myths and a blasé attitude caused by Moore's Law



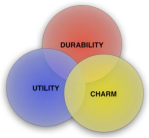
Performance Myths

- You need something to measure before you can begin to manage performance
  - You can begin estimating right away and then tune with the data - you do want to know where the dragons lie
  - If you are relying on hardware to solve performance, you should verify it is a cost effective solution
  - (in the dark ages, when I was young, performance was always an issue and is now, but it is not treated with the necessary rigor)
- Managing performance takes too much time, it will delay project completion
  - Level of effort should depend on level of risk
- Performance models are complex and expensive to construct
  - End to end performance testing may not be possible, performance models may be only option



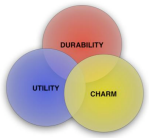
Proactive Performance Management

- Do it early
- Have expertise:
  - Someone on project responsible for performance engineering
  - Corporate performance engineering group
  - Identify, track and manage the performance risks



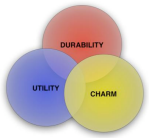
Performance Strategies

- Simple-Model strategy - start with simplest performance model (do something)
- Best and worst case strategies, upper and lower bounds of performance
  - Then adapt to precision as more data/information becomes available
- Distributed systems are tricky
- If performance is the risk, get serious and do rigorous software/hardware performance engineering



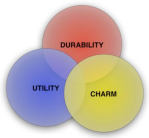
Final Words On Performance

- Often the easiest and most tempting and most used solution is to get the system running and then optimize it
- However sometimes early architecture decisions affect performance in ways that are very difficult to optimize away
- For example, one thing to avoid is remote calls
- But often you will find yourself in a state where performance is sacrificed for understandability
- A significant change in configuration can affect your whole performance world
- It is also tempting to rely on the inevitable performance boosts, unfortunately so do the OS folks and other tool developers



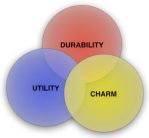
Concurrency

- Doing things in parallel rather than doing them serially, one at a time
- Difficult because it is hard to enumerate all possibilities and therefore hard to test
- Transaction mangers handle some of it, but issues arise because data often spans transactions.



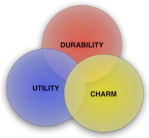
Concurrency: Lost Updates

- A user, X, checks out a file from a source code control system and begins to edit it. Simultaneously Y checks out the same file, changes it and checks it in again before X checks it in.
  - Results in an inconsistent read - when you read 2 things that are correct but not at the same time
- Balance between correctness and liveness - how much concurrent activity can go on
- To understand we have to consider execution context



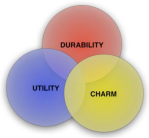
Execution Context

- Request is a single call from the outside world that software works on and may send back a response
- Session is a long running interaction between a client and a server, may be single or multiple requests
- Processes, threads (share resources) and isolated threads (no shared memory)
- Transaction combines several requests that the client wants treated as a single request



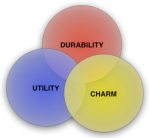
Issues

- Concurrency issues occur when more than one active agent (process, thread) has access to same piece of data
  - Isolation remedy - partition data so that any part can only be accessed by one active agent, e.g. file locks - can have shared reads
  - Immutability remedy - recognize data that does not change - relax concurrency concerns
  - Recognize applications that only read data



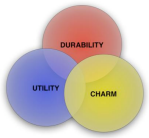
Concurrency Control

- Mutable data that cannot be isolated
  - Optimistic locking - both can access data but first one finished changes are accepted second and on will raise a conflict that needs to be resolved
  - Pessimistic locking - first one gets the data and no one else can access until they are through (check in, check out)
  - Optimistic lock is about conflict detection and pessimistic lock is about conflict prevention
  - Decision to choose between two is based on frequency and severity of conflicts. If low, pick optimistic locks.



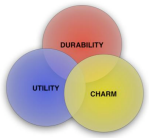
Deadlocks

- Issue with pessimistic reads - neither can make progress until other completes: X edits A and Y then requests A edit Y requests B edit X needs to edit B in order to edit A. -- complex chain
  - Select a victim that has to discard work - but since they can become very complex deadlock detection is difficult
  - Use timeouts
  - Force users to acquire all their locks at session initiation
  - If X has lock and Y tries to get it, Y automatically becomes victim
  - Having a deadlock proof scheme relies on knowing all scenarios - difficult



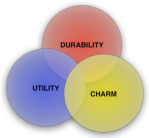
Transactions

- Usually handles concurrency in the Enterprise
- ACID properties:
  - Atomicity each step in sequence must complete successfully or all work must roll back. Partial completion is not permitted
  - Consistency - system resources must be in a consistent, noncorrupt state at start and end
  - Isolation - result of an individual transaction must not be visible to other open transactions until that transaction commits successfully
  - Durability - a committed transaction must be made permanent - survive all crashes



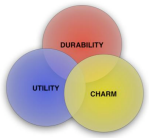
State

- Values of variables, data
- Some sessions are inherently stateful and what should you do
  - Not loved -- stateful sessions consume server resources
  - But necessary -- shopping cart
- Session state versus record data - long-term persistent data stored in database



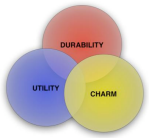
Session State Solutions

- Store data on client (cookies)
- Server session state
  - Hold in memory between requests
  - Store in database - involves parsing data into tables and fields
- Client has bandwidth issues, all data has to be transported with each request
  - Small amounts of data make this easier, but there is still security issues on client
- Server side
  - Session isolation - each session has own data hard to master in database approach -- also issue with using it quickly in memory on server or at client data is there, but for database have to access it
  - Session migration versus server affinity (role playing games)



Final Words on State

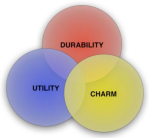
- What happens when a user cancels in the middle --clean everything out! Easiest for client approach
- Database approach can cope with client crashing, server crashing



LoS

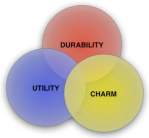
- Reduce
- Organize
- Time
- Learn
- Difference
- Context

- EMOTION
- TRUST
- FAILURE
- The one
  - Away
  - Open
  - Power



EMOTION

- MORE!
- Simplicity looks cheap
  - Individual differences
- "Form follows function and feeling follows form!"
- Email and :-) -> ☺
- Blinging (marking) nude electronics
  - Protection - enlarge or protect simple surfaces

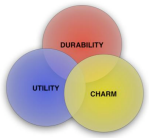


http://www.decalgirl.com

http://www.letscrystalit.com/

nvouspc.com

alienware.com

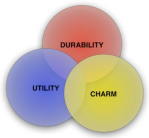


EMOTION 2

- Animism, anthropomorphism - naming of cars, computers (Shintoism & Miyazaki)

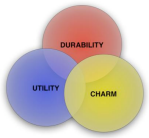
imdb.com

http://movies.lovetoknow.com/wiki/Fantasia



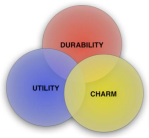
Emotion - 3

- "Perhaps this is the fundamental distinction between pure art and pure design. While great art makes you wonder, great design makes things clear."
- "Achieving clarity isn't difficult. … The true challenge is achieving comfort."
- ROE - Return on Emotion



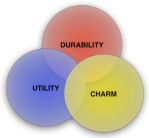
TRUST

- The best interface is none
  - TiVo Suggestions
  - Social filtering
  - Expert filtering - chef's discretion
- The power of undo
- The fear of "trust me" - trust but verify :-), opps that was EMOTION!



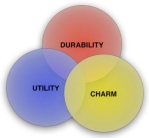
HOW MUCH DO YOU NEED TO KNOW ABOUT A SYSTEM?

HOW MUCH DOES THE SYSTEM KNOW ABOUT YOU?

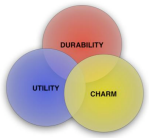


FAILURE

- ROF - return on failure - even when you fail to simplify, you learn – value to the journey

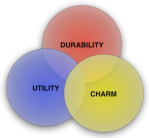


Simplicity is about subtracting the obvious and adding the meaningful

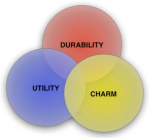


Ten

- Away: More appears like less by simply moving it far, far away
- Open: Openness simplifies complexity
- Power: Use less, gain more
  - Axiom of Design: More constraints, better solutions are revealed

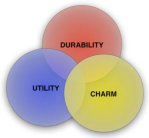


What's on your shelf?

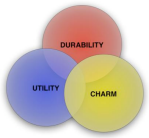


Reduce
Organize
Time
Learn
Differences
Context
Emotion
Trust
Failure
The One

Away
Open
Power

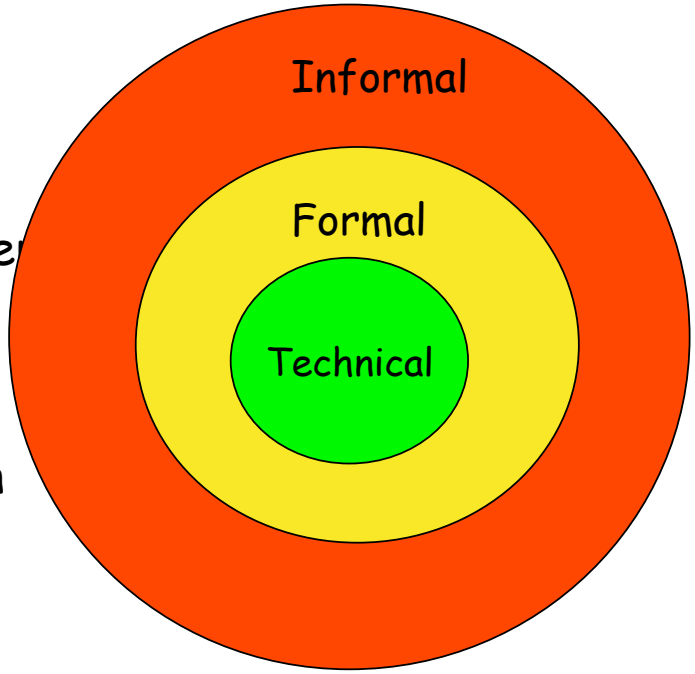


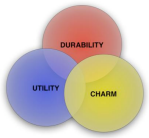
Simplicity is about subtracting the obvious and adding the meaningful



On Security

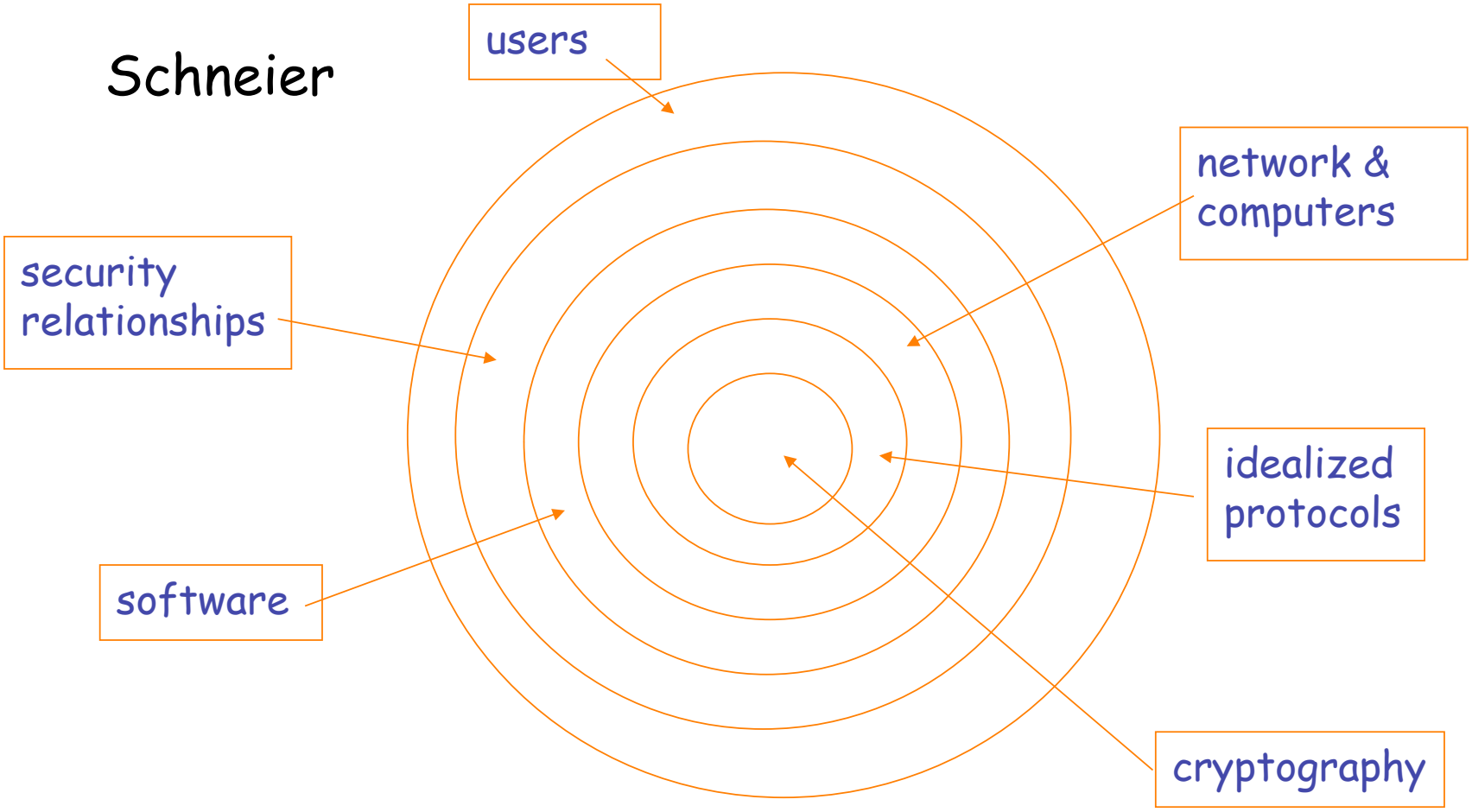
- Kaufman, et.al., "… how to communicate securely over an insecure medium"
- Garfinkel and Spafford(1991) "A computer is secure if you can depend on it and its software to behave as expected."
- Dhillon - "Coordination in three's"
- Schneier: prevention, detection, reaction

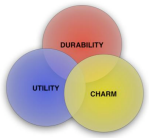




More on Security

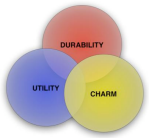
Schneier





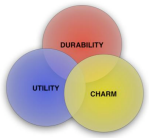
On Privacy

- Complying with a person's desires when it comes to handling "his or her" personal information. … the right of individuals to determine if, when, how and to what extent data about themselves will be collected, stored, transmitted, used and shared with others. - Cannon
- "True Names" - Vernor Vinge
- Anonymity is related



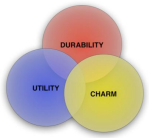
At Risk

- Money
- Information/Data
- Information/Data integrity
- Time and other resources (computational)
- Privacy
- Confidentiality
- Availability
- Others(?)
- (some vocabulary)



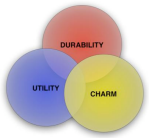
Threats

- A threat is a specific means to realize a risk
- A threat model is a collection of threats for a specific environment
- A vulnerability is a systematic artifact that exposes the user, data, or system to a threat
  - E.g., buffer overflow
- Sources of vulnerability:
  - Bad software (hardware)
  - Bad design/requirements
  - Bad policy/configuration
  - System misuse



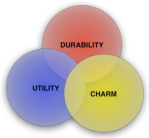
Adversaries

- An adversary is any entity trying to circumvent the security infrastructure
  - Curious and clueless (script kiddies)
  - Casual attackers seeking to understand systems
  - Folks with axe to grind
  - Malicious groups of sophisticated users
  - Competitors
  - Governments
  - Creative employees - how about you?



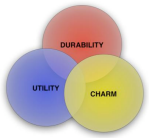
Attacks

- Attack is an attempt to exploit vulnerabilities
- Kinds of Attacks:
  - Passive (eavesdropping)
  - Active (password guessing)
  - Denial of Service (DoS)
    - Distributed denial of service (many points)
- Compromise occurs when an attack succeeds - usually involves taking over resources



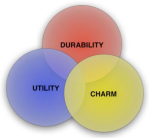
Participants

- Participants are expected system entities
  - Computers, agents, people, enterprises, …
    - Which are often referred to as servers, clients, users, entities, hosts, routers
  - Security is defined with respect to these entities because every party may have a unique view
- A trusted third party
  - Trusted by all parties for some things
  - Often as introducer, arbiter, authenticator



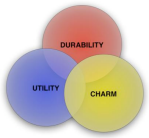
Trust

- Trust refers to the degree to which an entity is expected to behave
  - What not to do - don't expose passwords
  - What to do - obtain permission ..
- A trust model describes, for a particular environment, who is trusted to do what?
- We make these decisions each day!

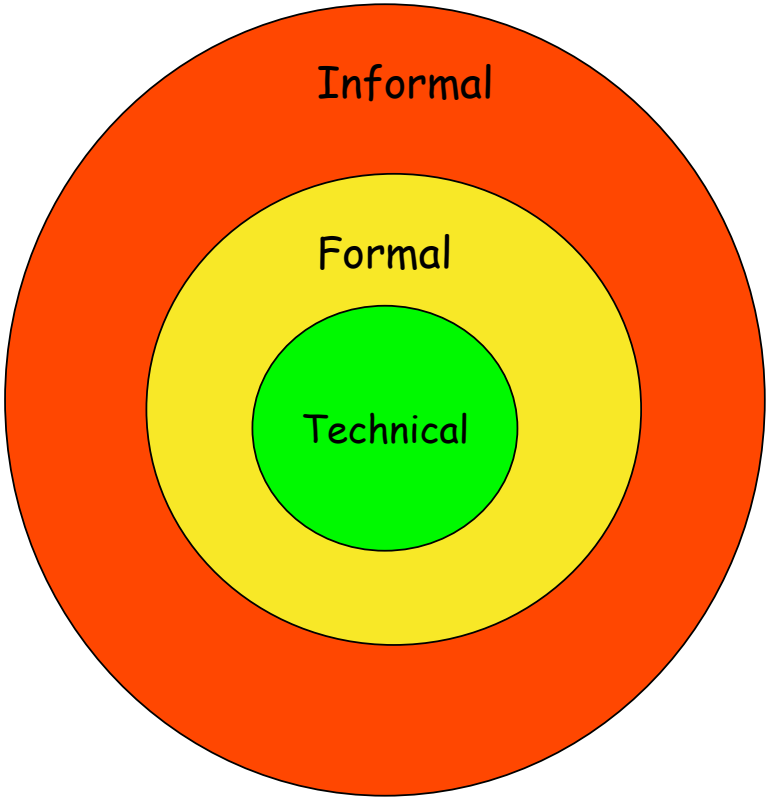


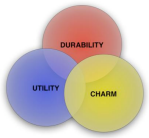
Security Model

- Combination of trust and threat models that address set of perceived risks:
  - Security requirements
  - Every design must have security model
  - Most common flaw is lack of a coherent security model
- Like most ilities it is very hard to retrofit a security model
- And note we are not talking about software alone



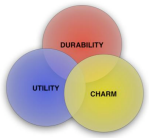
Coordination in 3's





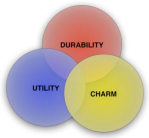
SOA: Service Oriented Architectures

- (OASIS) SOA is a paradigm for organizing and utilizing distributed capabilities that may be under control of different ownership domains
  - Any given need may require the combining of numerous capabilities and any single capability may address multiple needs
- Visibility, interaction and effect are key concepts
  - Visibility - refers to the ability of those with needs and those with capabilities to see each other
  - Interaction is using the capability (service)
  - Results in real world effects -- an act not an object
- Service combines these ideas:
  - The capability to do work for another
  - The description and specification of the work offered
  - The actual offer to do work
  - Services are the mechanism that brings needs and capabilities together

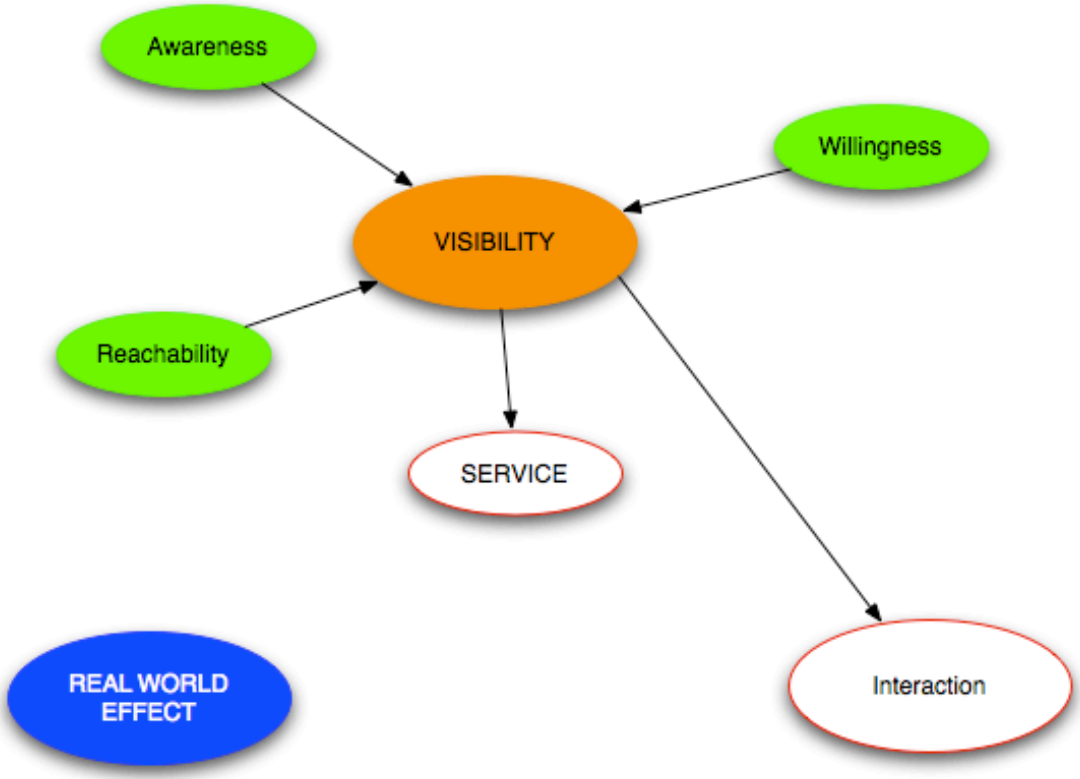


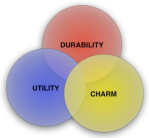
SOA -2

- SOA purpose is to assemble solutions that promote reuse, growth and interoperability (interfaces!)
- Service Participants:
  - Service providers
  - Service consumers
- Service description contains information necessary to interact with the service and describes the service in terms of inputs, outputs and associated semantics
- SOA is usually implemented using Web Services
- Central focus of SOA is the task or business function managed by delegation
- Service awareness can either be pushed or pulled



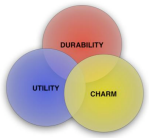
Visibility (OASIS, figure 4)





Service Interaction

- Information model is about information (duh) and requires consistent interpretation of strings and other tokens, requiring knowledge of the structure (syntax) and meaning (semantics).
  - Encryption service - information to decrypt/encrypt
  - Database service - requests to query or modify
- Behavior model - knowledge of actions permitted and process or time-oriented aspects of the interaction, described by actions on, responses to and timing dependencies (including sequences of actions).
  - Action model deals with the actions
  - Process model deals with temporal aspects including ordering
  - Note: when services are combined this gets funky and one discusses orchestration (a primary service invokes other services) choreography (services interacting with each other and maintaining state with no one service in control -- collaboration)



Service Description

- Information needed to use the service. SOA has a large amount of documentation and descriptions from a variety of customer perspectives. The information model is a key component of this.

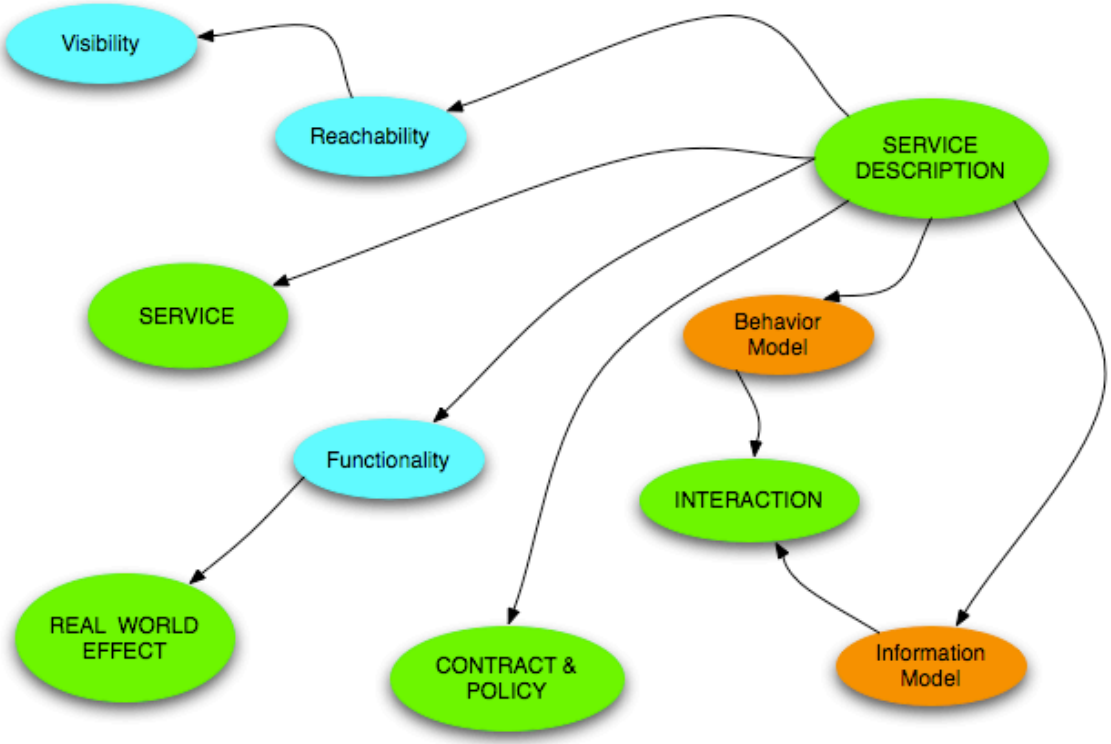
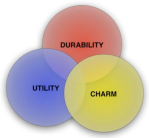
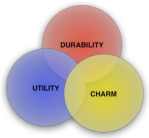


Figure 8 OASIS



SOA Implementation (WEB)

- XML used extensively
  - XML schema (DTD -> XSD)
  - XSLT for transformation
  - XQuery
  - XPATH
  - SOAP vs. REST
- WSDL (Web Services Description Language) used for service description
- And much more later in Design if time



References

- Fowler, Martin, Patterns of enterprise application architecture, Addison-Wesley, 2003.
- OASIS, "Reference Model for Service Oriented Architecture," http://www.oasis-open.org
- Wikipedia on BitTorrents
- Kurose, J.F. and Ross, K.W. Computer Networking: 2nd Edition, Addison-Wesley, 2003. ISBN:0-201-97699-4
- G. Dhillon, Principles of Information Systems Security, Wiley, 2006, ISBN 0-471-45056-1