

Class 5 CIS 573

Gregg Vesonder
University of Pennsylvania
Penn Engineering - Computer & Information Science
©2009 Gregg Vesonder

Roadmap

- Survey
- Continue on 3
- Case Study
- OO Design
- Quality
- Readings this class Somerville 27-29, **Andersson 3-4**
- Readings next class: Mid Term
- Readings next week - posted on wiki Sunday

Critical Dates

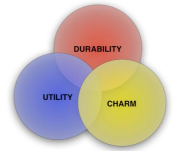
- Every class project review
- July 23rd Mid Term
- August 6th log books due
- August 11th project presentations
- August 13th Final

Teams

- Team 1 - Klein Keane, Beck, Buchman, Richardson, Nunez
- Team 2- Wilmarth, Caputo, Xiang, Francis, Nanda
- Team 3- Noronha, Fang, Huang
- Team 4-Whitehead, Liu, Ratnakar

Project Reports

- Presentation each class
 - Green, yellow, red -simplified model + gaps
 - Current pressing issues
 - What was done since last class
 - What will be done before next class
 - Gaps



Log Book

- On a license plate: Source code is free speech!

Case Study

1. List the non-functional requirements and the associated stakeholder(s) responsible for them
2. From the text and appendices, sketch out the scenarios, indicating any steps you may have added for completeness and continuity (using any notation):
 1. Becoming a subscriber (both Torsen and Shah versions)
 2. Ordering a book
 3. Leaving the service
3. Provide an overall critique of the "plan" for Saley and Tse listing elements of risk, issues, analysis of his resource estimate, roles and responsibilities and schedule. Draw on all of the information in the document for your evaluation.
4. How would you proceed? Which software process model would you use? How would you assign the current managers roles and responsibilities? Do you agree with Padalka? How would you acquire additional resources? Would you use or add to internal resources, external resources or a combination of internal and external resources?

Some Preliminaries

- **Object** (state (variables), behavior (methods))
 - Instance, instance variables, instantiated, encapsulation
- **Message** - everything an object can do is represented by its message interface
- **Class** - software blueprint including common elements of objects that need not be repeated
 - Class variables
- **Inheritance**
 - States and methods, override
- Data containing instances and function containing classes
- Polymorphism (overloading, method based; overriding, inheritance based)

OO

- Traditional techniques focus on **functions** of the system
- OO focuses on **identifying and interrelating the objects** that play a role in the system
- Convergence to the UML, Unified Modeling Language, Booch, Jacobson and Rumbaugh
- Heuristic thoughts- keep objects simple and each method should send messages to objects of a very limited set of classes (more when we explore OO metrics)
 - **Cohesion and coupling**

OO Analysis and Design

- OO Analysis = Requirements analysis + Domain class selection
 - Product = Complete requirements document + domain class model + basic sequence diagrams
 - Domain classes obtained via use cases -> sequence diagrams and brainstorming/editing process
 - Use domain classes to organize requirements
 - OO Design = all other activities except coding
 - Product = complete detailed design ready for coding
- Eric Braude, Software Design, John Wiley, 2004.

"Schools" of OO

- **European school**, influenced by the Scandinavian school of Programming, OO analysis and design is modeling real world objects both animate and inanimate
- **American school**, OO focuses on data abstraction and component reuse - identifying reusable components and building an inheritance hierarchy.
 - "What matters is not how closely we model today's reality but how extensible and reusable our software is"

OO Viewpoints

- **Modeling (European) viewpoint** - conceptual model of some part of a real or imaginary world.
 - Each object has identity, is unique
 - Objects have substance, properties that hold and can be discovered
 - Objects are implementations of abstract data types
 - Mutable state, variables of abstract data type
 - Operators to modify or inspect the state
 - Only way to access object
 - Interface to object
 - Object = identity + variables + operators or
 - Object = identity + state + behavior

OO Viewpoints - 2

- **Philosophical view** - objects as existential abstractions, the unifying notion underlying all computation
 - Beginning and end to objects
 - Eternal objects, e.g., integers
 - Not instantiated, cannot be changed
- **Software Engineering view** - data abstractions encapsulating data and operations
 - Object based languages encapsulates abstract data types in modules whereas
 - Object oriented also includes inheritance

OO Viewpoints - 3

- **Implementation view**
 - Continuous structure in memory, a record of data and code elements
- **Formal view**
 - Object viewed as a state machine with a finite set of states and a finite set of state functions. State functions map old states and inputs to new states and inputs
- While modeling conceptual viewpoint is stressed
- **Tensions between a problem oriented (analysis) vs. solution oriented viewpoint (design)**

Objects

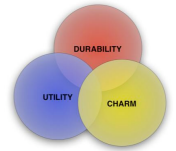
- Characterized by a set of attributes or properties
 - Attributes originate from Entity-Relationship Modeling
 - In ERM attributes represent intrinsic properties that do not depend on other entities
 - Shared properties among objects are denoted as relationships
- OO modeling uses attributes to denote any field in the underlying data structure
 - State includes intrinsic and shared properties
 - State includes set of structural attributes and operations = behavioral attributes.
 - Identity is an attribute

More on Objects

- Programming level;
 - Objects having same set of attributes belong to the same class
 - Individual objects of the class are called **instances**, when they are created they are **instantiated**
 - **Objects not only encapsulate state but also behavior - the way it is acted upon and acts upon other objects**
 - Behavior of an object is described as the services provided by that object
 - Services are invoked by sending messages from the requestor to the object acted upon
 - Client server model of objects, client object requests services from server object, services also are referred to as responsibilities

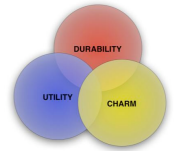
Relations Between Objects

Relationship	Example
Specialization/ generalization, isa	Table isa furniture
Whole-part, has	Table has tabletop
Member-of, has	Library has member



More on Relations

- Generalization-specialization can be expressed as a hierarchy
 - *Single inheritance*, tree
 - *Multiple inheritance*, directed acyclic graph
 - Define common attributes at a higher level and let descendants inherit the attributes (abstraction)
 - Object hierarchy can be viewed as a type hierarchy, chair and table are subtypes of furniture



More on Relations - 2

- Part of relationship aggregates components into a whole
 - It is a transitive relationship
- Member-of relationship represents the relation of a set and its neighbors
 - It is not transitive

OO Analysis

- Not considered with instances - concerned with object types, classes
- Major goal- identify set of objects (classes) with their attributes (states) and their services (behavior)

OO Analysis and Design Schemes

- Common notations of the schemes:
 - Class diagram - static depiction of objects as nodes and their relations as edges
 - State diagram - models dynamic behavior of single objects using a variant of a finite state machine representation. Nodes in state diagram represent state of object, edges possible transitions between states
 - Interaction diagrams - model sequence of messages in an interaction among objects
 - Sequence diagrams emphasize time orderings
 - Collaboration diagrams emphasize objects and their relationships relevant to a particular interaction

Will see this later with UML

OO Analysis and Design Assumptions

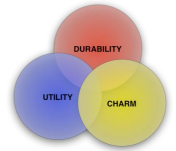
- **Assumes a stable problem statement**
 - Not strong on elicitation (but not weak either)
 - A collection of use cases are one view of the software architecture of a system

CRC Cards✓

- Class Responsibility Collaborator cards
- Documents collaborative decisions - usually done in a group, but useful individually
- Very helpful in early stages of software development
- Nice for small to medium size projects
- One of the techniques to use, very useful - a winner!
- Wilkinson, N. Using CRC Cards: An informal approach to Object-Oriented development, Cambridge University Press, 1995, ISBN: 0133746798

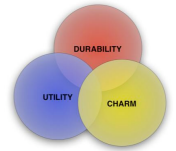
CRC Card

Class Name: Superclasses: Subclasses:	
Responsibilities	Collaborators



Analysis and Design Methods

- Approach:
 - Identify the objects
 - Determine attributes and services
 - Determine relationships between objects

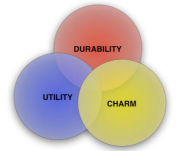


Identify the Objects

- Look for important concepts from the application domain
 - Domain specific entities are prime candidates for objects
 - Real world objects - books
 - Roles played - customer
 - Organizational units - department
 - Locations
 - Devices
- Look at existing classifications and assembly (whole, parts relationship)
 - Sometimes listing most of the **nouns** in the requirements specification or the problem statement
 - Eliminate from the noun list implementation constructs
 - Vague terms replaced by concrete terms or eliminated
 - Eliminate synonymous terms
 - Demote some terms to attributes
- Some information is from statement, some from tacit knowledge
- Diagram starts with relationships and evolves to more detailed descriptions (cardinality constraints and inheritance)

Identify Attributes and Services

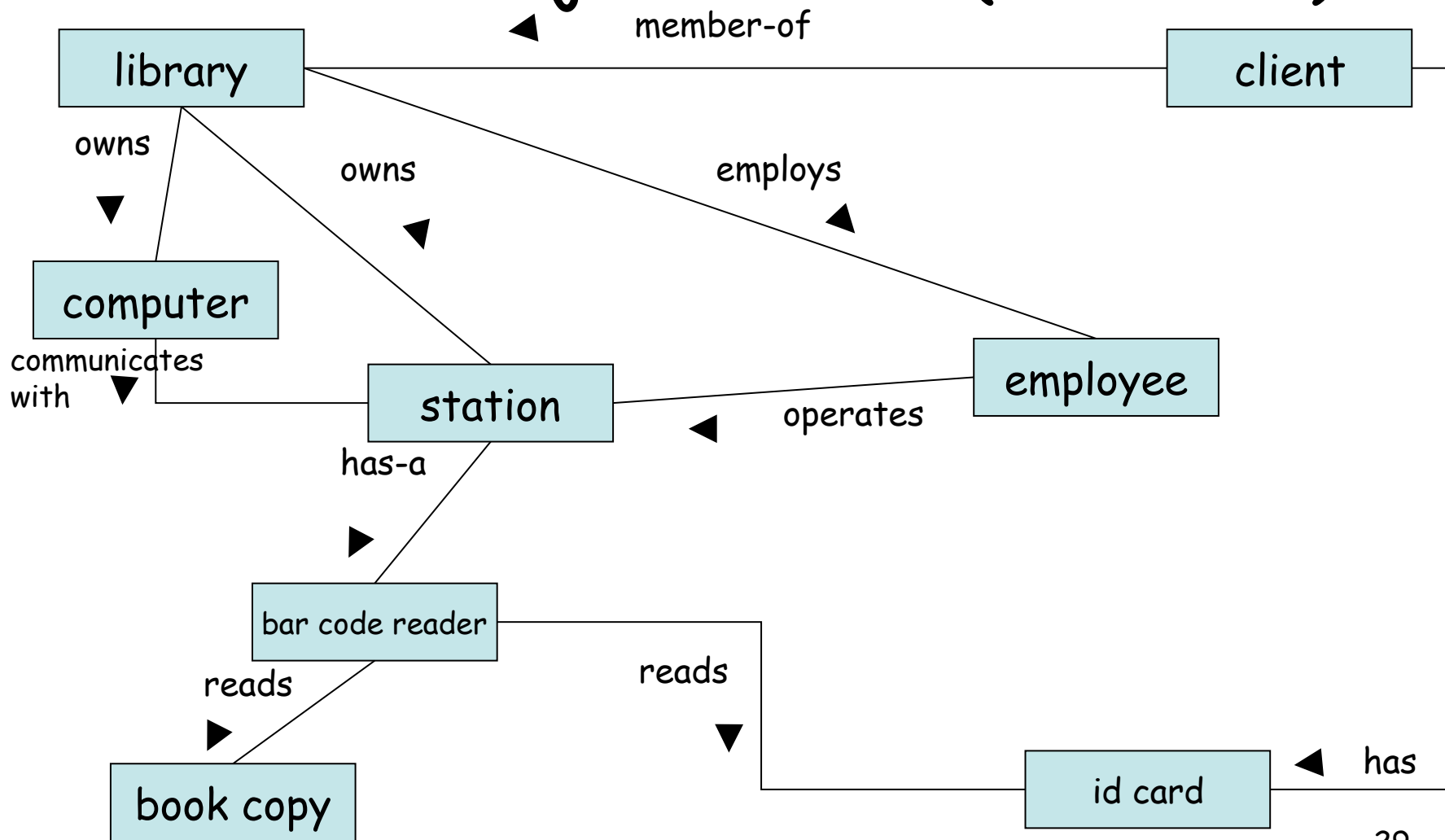
- Describe an instance of the object
 - The state of the object
 - Consider characteristics that distinguish instances but are common properties of the objects
 - Look for atomic rather than composite attributes - compute from atomic to composite if necessary
 - Services are related to life cycle and are usually **verbs** in the description, e.g., book is acquired, borrowed, returned, retired
 - Concern state of the object
 - Usage scenarios aid in discovery



Identify Relationships

- Services are one way objects can be related
 - OO flavor comes from whole-part, gen/spec relationships
 - Consider similarities of objects as basis for specification of a more general object
 - Publication is an abstract object, one that has no instances
 - Attributes and services defined at publication level constitute a common interface for its descendants
 - Generalization/Specialization relationship can lift services to higher levels in the hierarchy and they are represented by virtual functions - services for which a default implementation is provided and can be redefined by specializations

Initial Object Model (van Vliet)



Comments on OO Analysis and Design

- Instances of an object should have common attributes if not repartition or reconsider
- Over evolution/time, object hierarchy should remain stable but attributes and services may change
- OO can be considered a middle-out design method!
 - Set of objects constitute the middle design level
- OO could prosper if there were a future where collections of domain specific classes become available - domain libraries - DIFFICULT

Design Patterns✓

- Portland Design Repository - <http://c2.com/ppr/>
- Motivation came from a "real" architect, Christopher Alexander-
 - "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem in such a way that you can use this solution over a million times without ever doing it the same way twice."
 - Referring to buildings and towns, e.g. couple's realm, children's realm, sleeping to the east, ...

Couple's Realm

- Consists of following centers:
 - Bed Alcove
 - Couple's Realm Main Area
 - Fireplace
 - Connects to Bed alcove, dressing room, Porch/balcony, still pond, bathing room and its child centers
 - Dressing Room
 - Porch/Balcony
- Connects to Bathing Room and Child Centers
- <http://www.simplybuilding.net/center/view/39>

Categories of Patterns

- **Creational** - abstract instantiation, strive for independence - problem is creating a complex object.
 - Used to control creation of objects, including families of objects. Often replace constructors.
- **Structural** Patterns - how classes and objects are composed to form larger structures--uses inheritance - problem is representing a complex structure
 - Lists, collections and trees with convenient interfaces
- **Behavioral**- algorithms and assignment of responsibility, describe objects and communication - problem is representing behavior
 - -application behavior options at runtime
- **Pros, more flexibility**
- **Cons, increased complexity and potential performance issues**

OO Metrics ✓

Source	Metric	OO Construct
Traditional	Cyclomatic Complexity	Method
Traditional	Lines of Code	Method
Traditional	Comment Percentage	Method
OO	Weighted Methods per Class (WMC)	Class/Method
OO	Response for a Class (RFC)	Class/Message
OO	Lack of Cohesion of Methods (LCOM)	Class/Cohesion
OO	Coupling between Objects (CBO)	Coupling
OO	Depth of Inheritance Tree(DIT)	Inheritance
OO	Number of Children(NOC)	Inheritance

OO Metrics

- WMC (Weighted Methods per Class) is a measure of size of class, assumes larger classes are less desirable - usually a count of the number of methods
- RFC (Response For a Class) is number of methods in class + number of methods called by each of these class methods where each method is counted once
- LCOM (Lack of Cohesion of Methods) is number of disjoint sets of methods of a class, any 2 methods in the same set share at least one local state variable, preferred value is 0, cohesion metric
- CBO (Coupling Between Objects) is coupling metric, 2 classes are coupled if a method of one class uses a method or state variable of the other class, high values = tight bindings, undesirable
 - Gradations
 - Law of Demeter - methods of a class should only depend on top level structure of own class
- DIT (Depth of Inheritance Tree) is the distance of the class from the root of the tree. Language dependent
 - Strive for inheritance trees of medium height, not narrow and deep, not shallow and broad
- NOC (Number of Children) is number of immediate descendants of a class, large number suggests improper abstraction

Non OO
metrics soon!

OO: Hype or Hammer?

- OOA and OOD are similar with OOD adding implementation specific classes, OOA should be problem oriented, OOD should be solution oriented
- Transition to OO takes time
- Issues: handling of real time requirements, less mature, measuring progress is hard, no good cost models, scalability and interoperability with non OO systems pose problems
- Traditional functional models are easier to understand by the customer
 - Users do not think in objects they think in tasks - use cases as a way to bridge this
- BUT...

The UML

- the UML = Unified Modeling Language
- Actually the books we will be working with use either the UML or code
- In your efforts you can use any of these techniques, another such as CRC cards or something you with which you are comfortable, just so long as you describe it and are systematic in its use.
- I provide a very basic introduction to the UML as a default
- There is an object oriented bias in modern notations
- **Opinion: UML has run amok, specification is 400+ pages**

What is the UML?

- Three Amigos: Grady Booch, James Rumbaugh & Ivar Jacobson
- Merged their notations into the UML
 - Class Diagram - represents details of the class
 - Object Diagram - represents instance of a class
 - Use Case Diagram - description of system behavior from a user standpoint
 - State Diagram - represents the current state of the object
 - Sequence Diagram - represents system over time
 - Activity Diagram - represents the sequence of an activity
 - Collaboration Diagram - represents interaction of elements
 - Component Diagram - represents a software component
 - Deployment Diagram - represents physical architecture of the system

Class Diagram

Employee
Name
Address
Age
Salary
ChangeField()
ReadField()

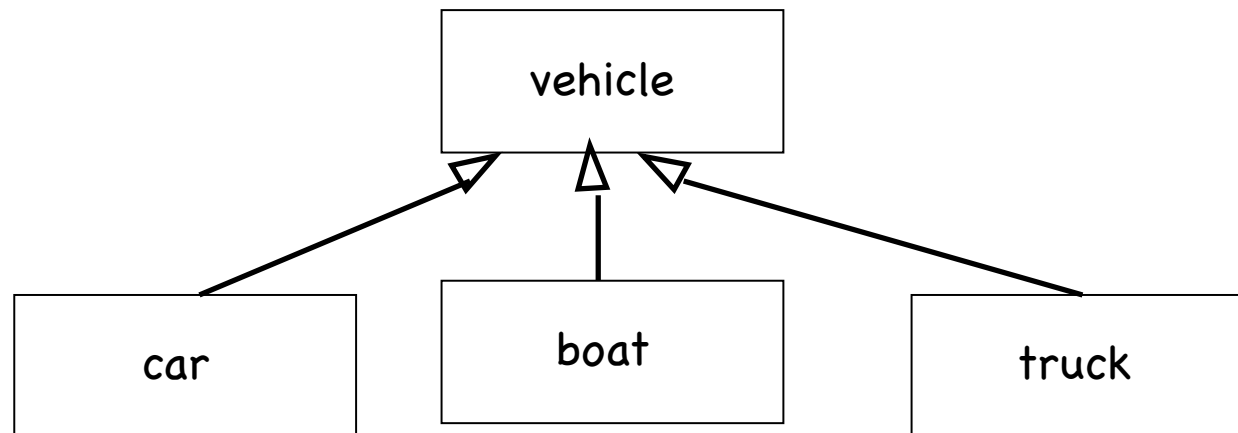
Robot
SerialNumber
Color
MemoryCapacity
Fuel
Move()
FireWeapon()
SeekEnergy()

Class Name
Attributes
Attributes
Operations
Operations

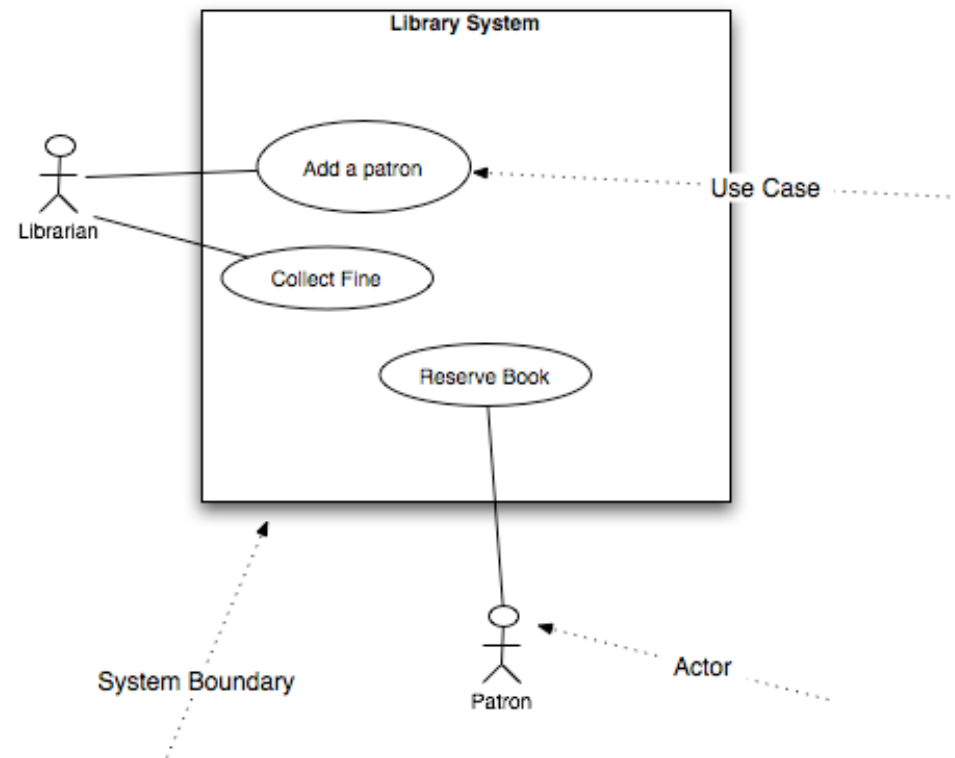
Object Diagram

Robot C3P0
SerialNumber: C3P0 Color: Gold MemoryCapacity:100 Fuel: nuclear
Move() FireWeapon() SeekEnergy()

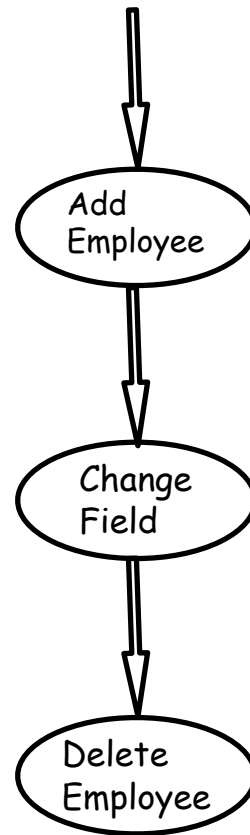
Inheritance



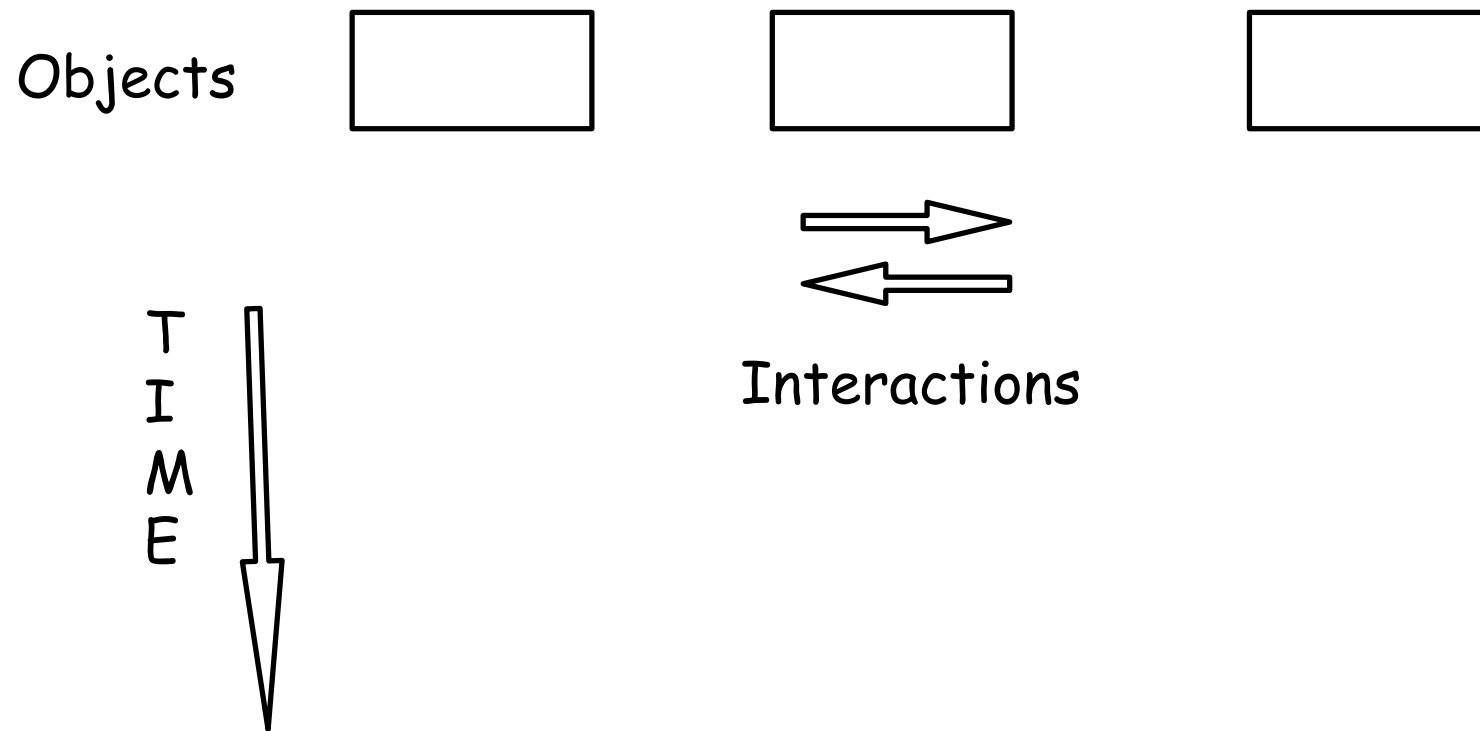
Use Case Diagram



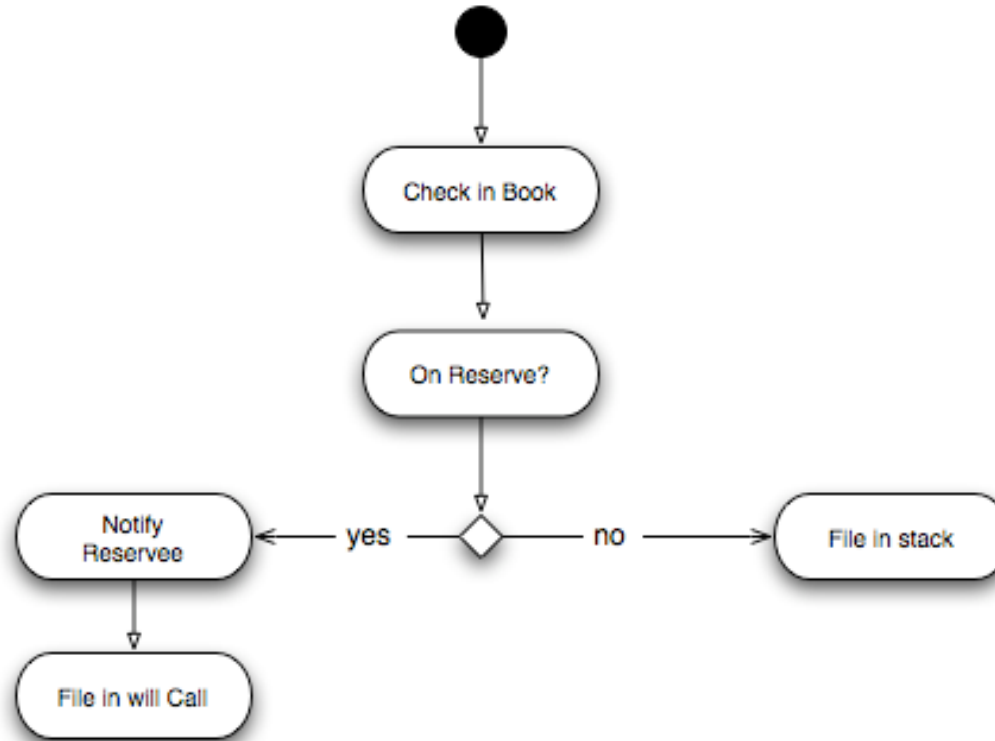
State Diagram



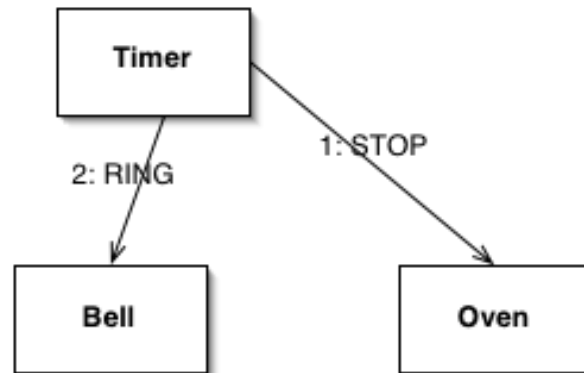
Sequence Diagram



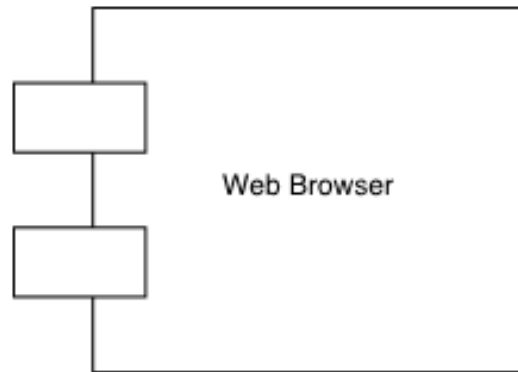
Activity Diagram



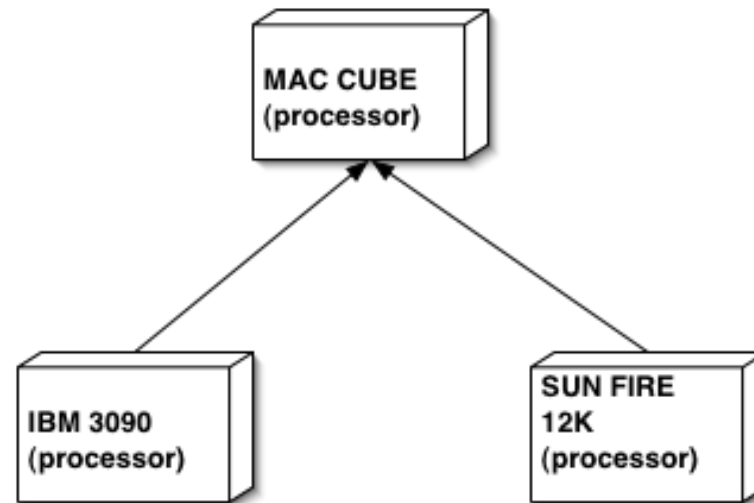
Collaboration Diagram

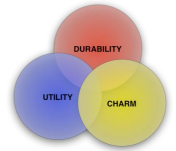


Component Diagram



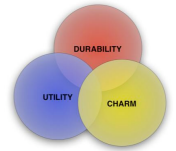
Deployment Diagram





Thought Problems

- You are beginning an OO project, what analysis style would you use, e.g., the UML, CRC cards, (your favorite method)? Do you think it would depend on the size of the project?
- What so you think are relevant criteria for deciding if you should use OO methodology?



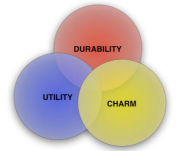
Thought Problem

- You want to nurture architects in your organization, what is a plan to do that and what styles will you encourage?
- You are asked to decide on a design strategy for your company - will you go OO?

On managing software quality

Q

Peters & Waterman, In Search of Excellence, a key factor of successful companies

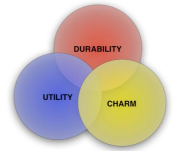


On Quality

- The cost of low quality ranges from dissatisfied customers to costing lives.
- Premise is that the quality of a product is largely based on the Quality of the process that leads to the product.
- We should strive to numerically describe quality - the other Q - Quantitative, "from response time is very fast to an average response time of < 1 second with no response taking over 3 seconds."
- What Quality is differs by audience - each (tester, end user, administrator) has their own perspective.

Approaches to Quality

	Conform	Improve
Product	ISO 9126	Best Practices
Process	ISO 9001 SQA	CMM SPICE Bootstrap

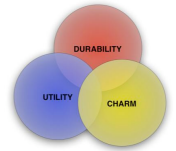


Approaches to Quality -2

- TQM, Total Quality Management, emphasizes the eclectic view, the pursuit of excellence in everything
- SQA - sees to it that the work is done the way it should be done
- CMM (and SPICE and Bootstrap) improves the development process or the process to improve the development process

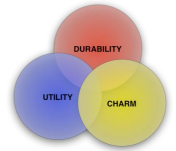
Quality Activities

- **Quality assurance** - organizational processes and standards
- **Quality planning** - processes and standards selected for a particular product
- **Quality control** - processes enacted to ensure development team has followed quality procedures and standards



On Quality

- Quality is tough to measure but easy to recognize (and there is agreement, inter rater reliability)
- IEEE defines it as "the degree to which a system, computation or process meets customer or user needs and expectations"
- Perspectives:
 - User: degree to which requirements are met: correctness, reliability, usability, ...
 - Customer: factors relating to the structure of the system: maintainability, testability, portability



McCall's Taxonomies

- (taxonomies are common in the early stages of a science)
- High level Q Factors, external attributes that are measured indirectly, and Q Criteria measure subjectively or hopefully objectively. By combining Q Criteria one measures the Q Factor. Subjective measures can be decomposed even further to find an objective base
- 3 classes of Q Factors: operation, revision and transition
- Q factors are not independent

McCall Examples

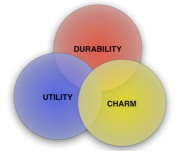
- Operation: Correctness (completeness & **consistency**), Usability (communicativeness, operability & training)
- Revision: Maintainability (conciseness, **consistency**, modularity, self-documentation, simplicity)
- Transition: Portability (hardware independence, modularity, self-documentation, software system independence)

ISO 9126

- Another attempt to define Quality characteristics and subcharacteristics
- Addresses product not process
- Only subcharacteristics visible to user
- Provides metrics for each subcharacteristic
- The current answer - develop your own based on these and relevant to your needs but remember that quality requirements that cannot be measured cannot be controlled

ISO 9126

- Characteristics and sample subcharacteristics:
 - Functionality (suitability, security)
 - Reliability (fault tolerance)
 - Usability (understandability)
 - Efficiency (resource use)
 - Maintainability (changeability)
 - Portability (replaceability)

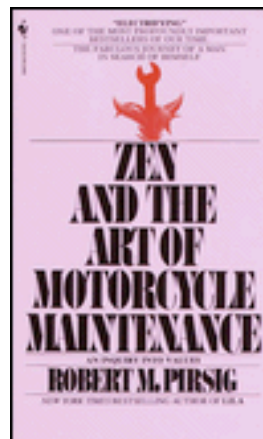


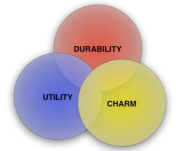
Garvin on Quality

- 5 perspectives/definitions of quality:
 - Transcendent, innate excellence, obvious
 - User based, fitness for use, addresses needs (acceptance test)
 - Product based quality as in ISO 9126
 - Manufacturing based - conformance to specs
 - Value based - show me the money

Pirsig

- Zen and the Art of Motorcycle Maintenance: An Inquiry Into Human Values, Bantam Books, 1974. ISBN: 0-535-27747-2



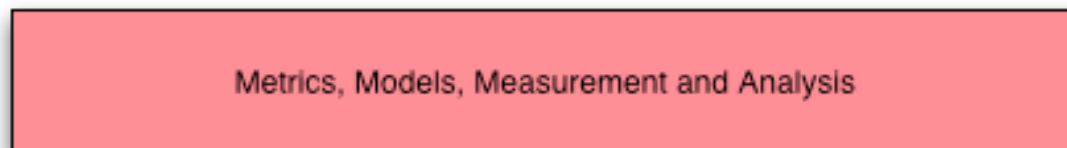
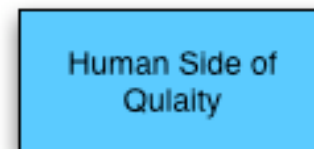
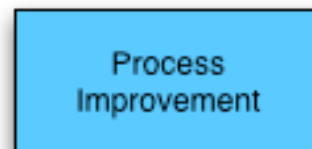
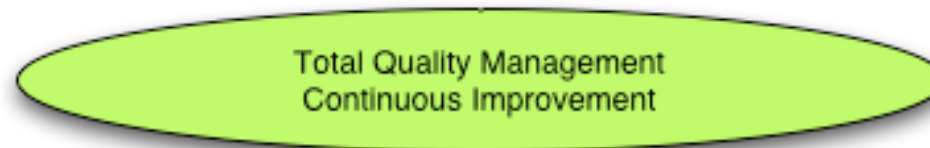


TQM- Total Quality Management

- Stresses improvement rather than conformance, CMM builds on TQM
- 3 principles:
 - **Customer value strategy** - benefits vs sacrifices caused by the product
 - **Organizational system** - eliminate complexity not people, people are a critical resource and focuses on how software melds with organizational practices
 - **Continuous improvement** - proactively based rather than reactively based.

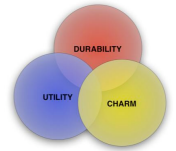
TQM

From S.H. Kan, Metrics and models
in software quality engineering,
Addison-Wesley, 2003.



ISO Quality System

- ISO set up a series of standards for quality management
- ISO 9001 most suited for software - model for quality assurance in design, development, production, installation and servicing
- ISO 9004-1 contains guidelines for individual elements of various standards
- ISO 9000 process includes third party auditor, with audits every 6 months and reregistration every 3 years - expensive
- Necessary for some customers



Software Quality Assurance

- Basic idea: improve quality by monitoring software and its development process
 - Ensure compliance with established standards
 - Ensure that inadequacies are brought to the managements attention and fixed
 - The quality organization performs reviews and audits and must be separate from production
 - Support of management, the QA organization must have go/no go authority over the product
 - Must be technically competent
- IEEE 730 provides a framework for Quality Assurance plan
- IEEE 983 is a complement to 730 and offers further guidelines including implementation, evaluation and modification.

NSA and Assurance

- Brian Snow -
<http://www.acsa-admin.org/2005/papers/Snow.pdf>
- Some highlights
 - Very strong use of formal methods, SEI level 5, TSP and PSP
 - Mutual suspicion - modules auditing and alarming each other's behavior - same with developers!

IEEE 730

- Purpose
- Reference documents
- Management
- Documentation
- Standards, practices, conventions and metrics
- Reviews and audits
- Test
- Problem reporting and corrective action
- Tools, techniques and methodologies
- Code control
- Media control
- Supplier control
- Records collection, maintenance and retention
- Training
- Risk management

CMM and others

- CMM - been there done that
- BOOTSTRAP - separate maturity rating for each of its practices
- SPICE (ISO/IEC 15504) - international initiative Software Process Improvement and Capability dEtermination

QAW

- Quality Attributes Workshop - facilitated method engaging stakeholders early to discover driving Quality attributes of a software intensive system
 - Results in creation of prioritized and refined scenarios
 - Provides description of Quality requirements before architecture is developed
 - It is system centric and stakeholder focused, done before software architecture is created
- Critical Quality attribute must be articulated and well understood early so it influences architecture - move to the left!
- Quality attribute examples: **security, reliability, modifiability, performance, interoperability, portability**

View of Traditional System Development

- Operational Descriptions
- High Lev Functional Requirements
 - Legacy Systems
 - New Systems
- A Miracle Occurs - Quality attributes are often missing from requirements document or, at best, vaguely understood and described
- Specific System Architecture
- Software Architecture
- Detailed design
- Implementation

QAW-2

- Motivation is to not rely on the miracle but clearly articulate what is needed by scenarios describing the stimulus, describes agent or factor that initiates system to react and a response, the systems reaction to the stimulus. Including the environment, the context (e.g. peak load, normal operation, maintenance mode).

Steps of QAW

- Step 1 - QAW presentation - description and participant introduction
 - 5 to 30 stakeholders
 - Stakeholders = usual suspects = end users, installers, administrators, trainers, architects, system and software engineers.
- Step 2 - Stakeholders present systems business/mission context, including high level requirements, constraints and identified Quality attributes
- Step 3 - Architecture Plan presentation: notional, preliminary architecture describing how requirements will be satisfied, key technical requirements and constraints and description of the system environment

Steps of QAW-2

- Step 4 - Identification of Architectural Drivers - facilitators summarize these from presentations in Steps 2&3, ask stakeholders for clarification, additions & deletions. Results in final list of Quality attributes to drive scenario stage.
- Step 5 - **Scenario Brainstorming**, stakeholders generate scenarios, each stakeholder contributes 2, facilitators assure at least one scenario for each Quality attribute driver of Step 4. The Quality attributes are operationally defined by these scenarios, hopefully, avoiding ambiguity of vocabulary.

Steps of QAW-3

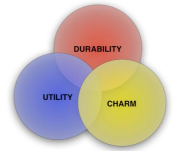
- Step 6 - Scenario Consolidation, similar scenarios are consolidated
- Step 7 - **Scenario Prioritization**, each stakeholder has N votes ($N = 30\%$ of # of scenarios), 2 passes, 1/2 of votes on each pass
- Step 8 - Scenario Refinement, work hard on clarifying descriptions of highly rated scenarios, including describing business/mission goals affected by scenario and describing relevant Quality attributes

Example Scenario

- Scenario - when a garage door opener senses object in door's path, stops door in less than 1 msec.
- Business Goals - safest system; feature rich product
- Quality Attributes: safety, performance
- Stimulus - object in path of garage door
- Stimulus Source - object external to system, bicycle
- Environment - garage door is closing
- Artifact - system motion sensor & motion control software
- Response measure - 1 msec
- Questions - How large must an object be before detected
- Issues - train installers to prevent malfunctions

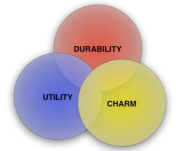
QAW Benefits

- **Increased stakeholder communication**
- Informed basis for architectural decisions
- Improved architectural documentation
- Support for analysis and testing throughout life of the system



Metrics

- Project: metrics used specifically but not solely by management to control current projects and provide feedback for future projects
- Technical (Individual?): used by an engineers to improve their performance



Technical/Individual Metrics

- Halstead
- McCabe
- Fan-in/Fan-out

Halstead - "software science"

- Stresses syntactic units rather than LOC
- Model components:
 - Operators - actions: +, -, *, /, if-then-else,...
 - Operands - data: variables and constants
 - 4 basic entities (used in a bunch of equations)
 - n_1 - # of different operators
 - n_2 - # of different operands
 - N_1 - total occurrences of operators
 - N_2 - total occurrences of operands
 - Length of Program for Halstead: $N = N_1 + N_2$

Halstead uses

- Simple to calculate, no in-depth examination of structure
- Measure of overall quality of programs - simplicity/bloat criteria
- In conjunction with others, helpful in maintenance and initial programming
- Substantial literature
- At surface level requires completed code so not good in estimation but with certain assumptions N can be calculated early on
- Does not account for complexity of interfaces

McCabe's Cyclomatic Complexity

- Based on a directed graph showing control flow of program thereby showing the number of independent paths in the program
- Cyclomatic Complexity, $CV = e - n + p + 1$ where:
 - e = # of edges
 - n = # of nodes
 - p = # of connected components (1 for main program 1 for each procedure)
- 10 should be upper limit of complexity for a component according to McCabe

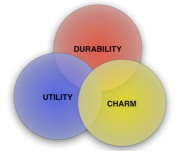
Eclipse metrics plugin

McCabe Uses

- Great for testing because it uncovers all linearly independent paths
- Does not add more complexity to nested loops and in general does not consider context
- Unlike Halstead does take into account control flow complexity, but not data therefore, often used together
- Useful for individual developer feedback and during maintenance

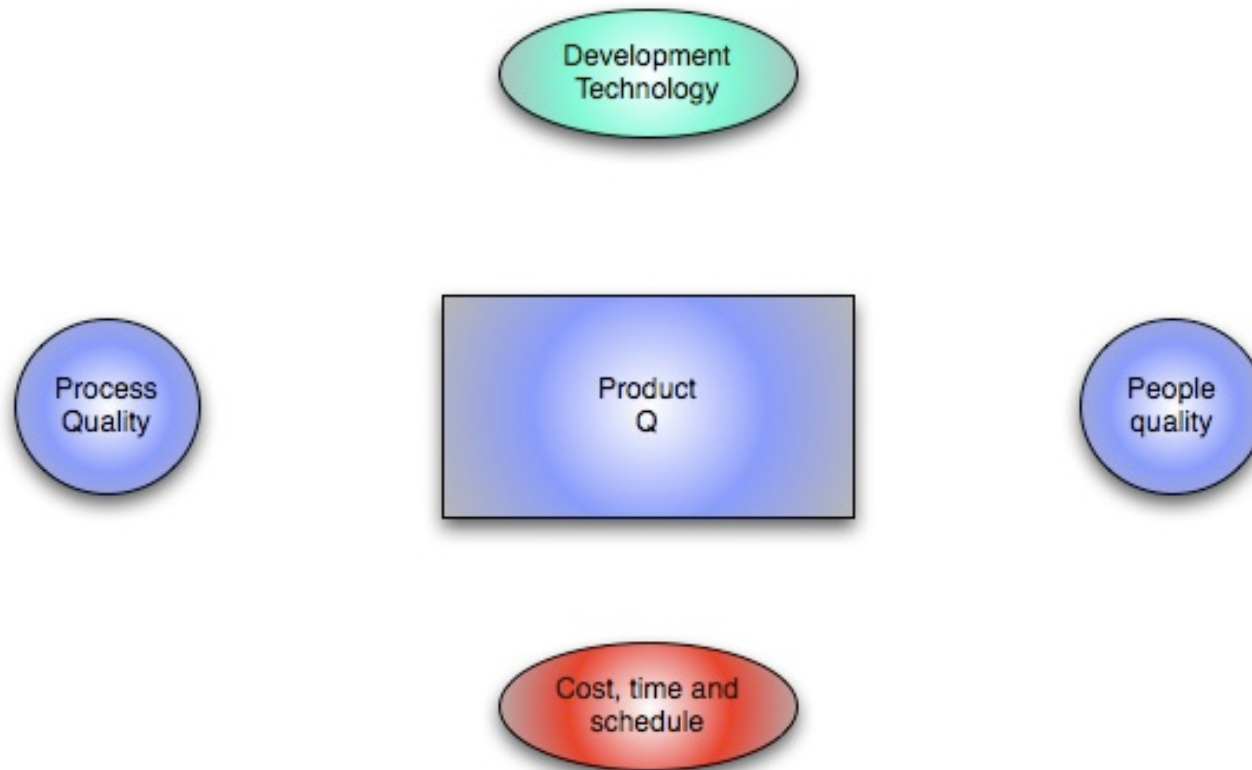
Fan In/Fan Out

- Measures interaction - basically, # of modules that call a given module and number of modules called by a given module.
- High degrees of fan-in/fan-out are undesirable
- Typical equations: $[LOC | Cyclomatic\ complexity]^* (\#fan-in * \#fan-out)^2$
- Takes into account data driven programs but underestimates (of course) complexity for programs/modules with little interaction



Quality Universe

Sommerville p.667



Reality Check

- The business is software, danger of a shift from developing software to developing processes, but ...



- Quality is recognizable

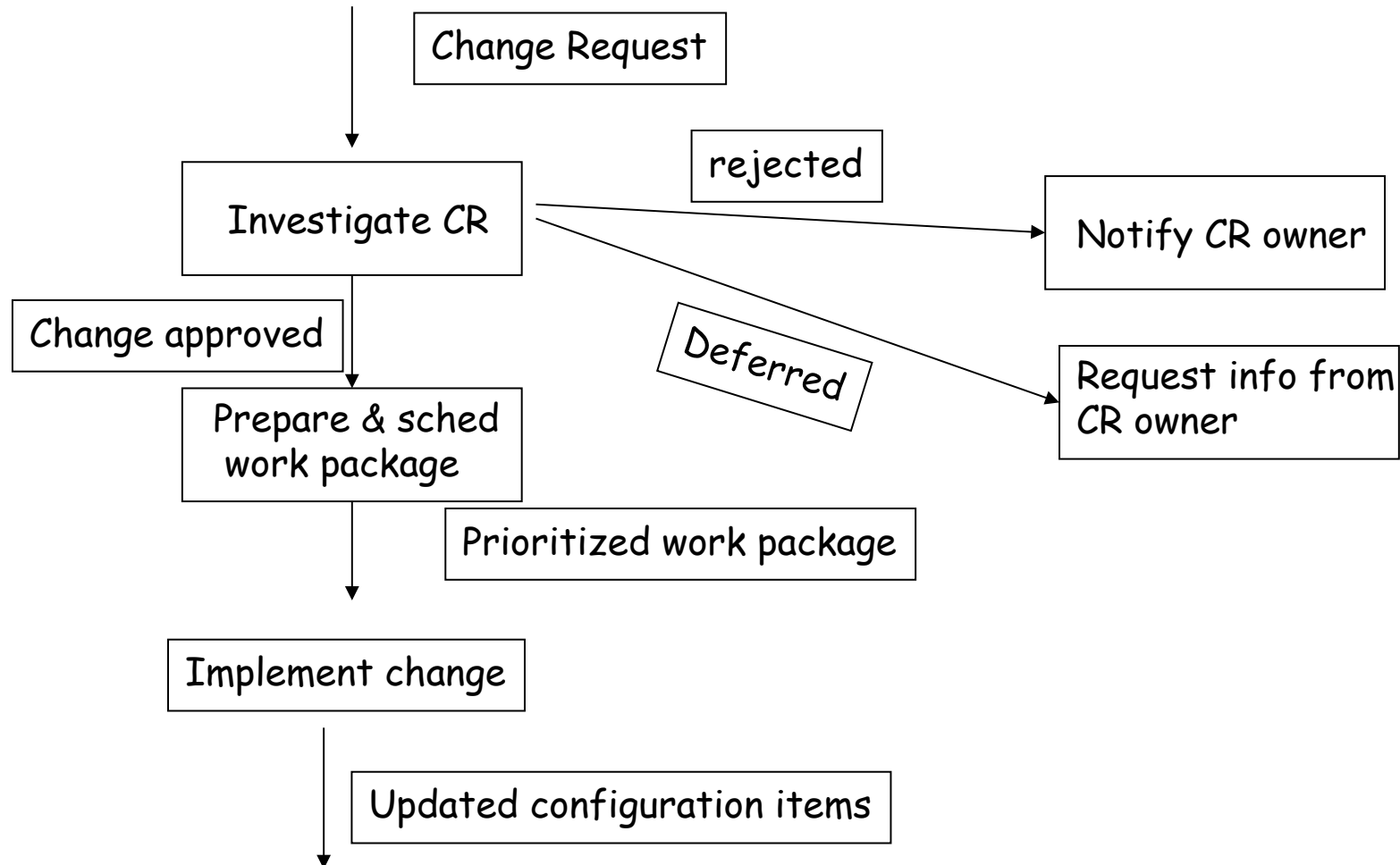
Configuration Management the problem

- Not a simple task!
 - Different versions of software usually is in the field during the life cycle
 - Different parts of the team are on different versions of the software and documents
 - The same release of a software product may have multiple versions consisting of different combinations of software components
- Configuration management is both a development and production issue

The Baseline

- IEEE - "reviewed and agreed upon basis for further development which can be changed only through formal control procedures"
- Contained in the baseline are configuration items: source, objects, requirements (p.75)
- Configuration management maintains integrity of these artifacts
- Major error- retrace steps through code, design documents and requirements specification -TRACEABILITY

Workflow of CR (MR)



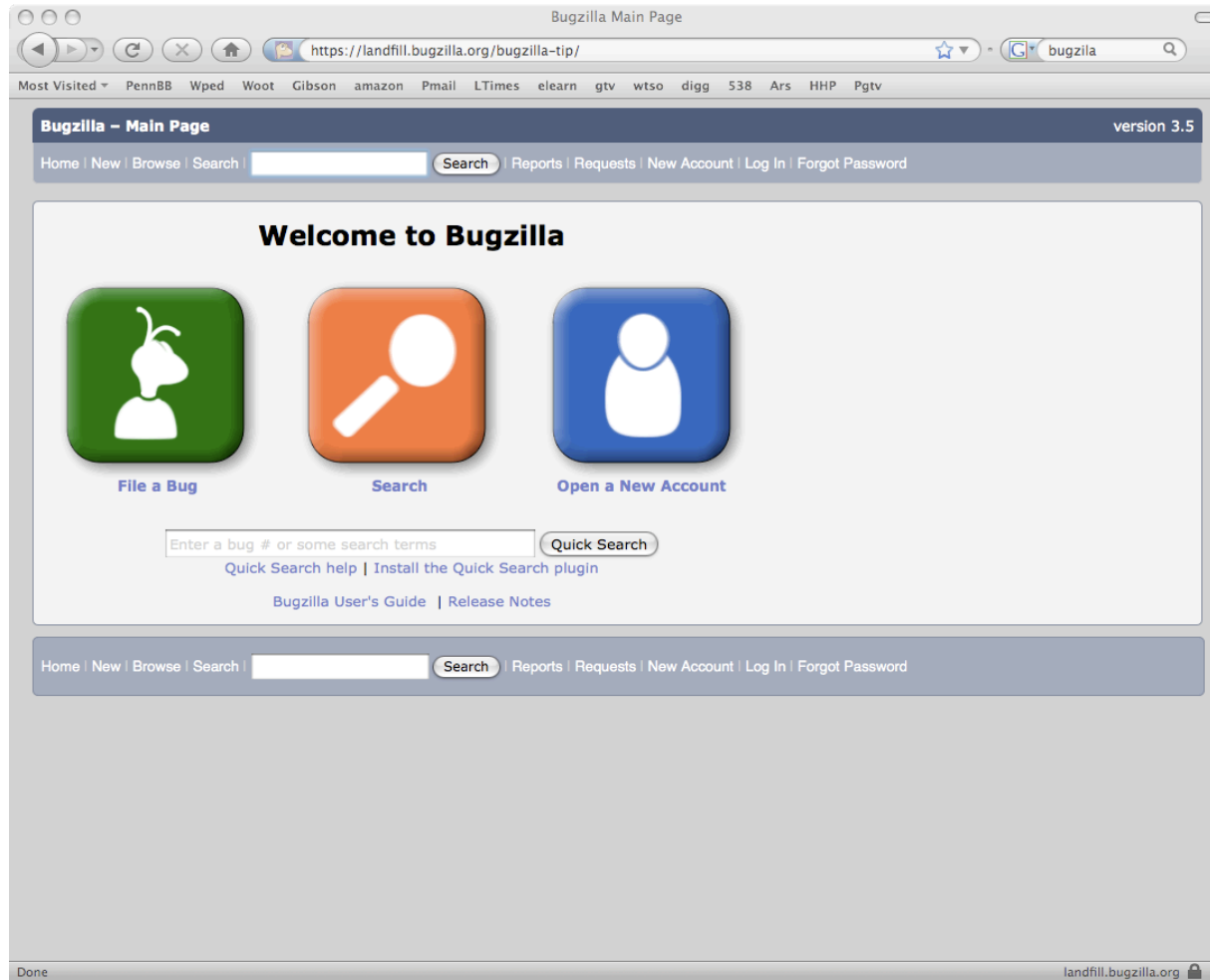
Configuration Management Tools

- Manage the workflow of CRs
- If item is to be changed, developer checks it out and item is locked to other users
- When item check back in revision history is stored
- All versions are recoverable
- Should be able to accommodate branching - necessary more times than you think!
- Configuration management tools are very sophisticated, keeps only the changes, the deltas and the remarks, timestamps and who did what - essential for Buildmeister and testers
- New tools are change oriented release configuration is identified by a baseline plus a set of changes.

Configuration Management Plan

- Main parts:
 - Management: how project is organized and who has responsibilities related to configuration management. How are change requests handled?
 - Activities:
 - Who is on CCB, what are their responsibilities
 - What reports are required
 - What data is collected and archived - IMPORTANT

Bugzilla



The screenshot shows the Bugzilla Main Page in a browser window. The browser's address bar displays the URL <https://landfill.bugzilla.org/bugzilla-tip/>. The page title is "Bugzilla Main Page" and the version is "version 3.5". The navigation bar includes links for Home, New, Browse, Search, Reports, Requests, New Account, Log In, and Forgot Password. The main content area features a "Welcome to Bugzilla" heading and three large buttons: "File a Bug" (green), "Search" (orange), and "Open a New Account" (blue). Below these buttons is a search input field with the placeholder text "Enter a bug # or some search terms" and a "Quick Search" button. At the bottom of the page, there are links for "Quick Search help", "Install the Quick Search plugin", "Bugzilla User's Guide", and "Release Notes".

References

- Futrell, Shafer & Shafer, Quality software project management, Prentice Hall, 2002, ISBN 0-13-091297-2
- Robertson, S. and Robertson, J., Mastering the requirements process, 1999, Addison-Wesley.
- Endres, A. and Rombach, D. A handbook of software and systems engineering. 2003, Addison-Wesley.
- Wirfs-Brock and Schwartz -
http://www.wirfsbrock.com/pages/resources/pdf/the_art_of_writing_use_cases_slides_and_notes.pdf
- Others embedded in text