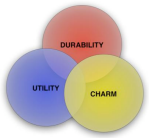
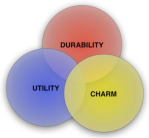


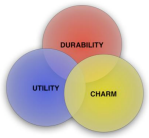
Class 1 SSW565

Gregg Vesonder
Stevens Institute of Technology



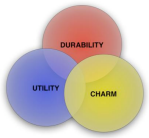
Roadmap

- Review syllabus
- Introductions
- logbook
  - me
  - volunteer
- Software Architecture, Design and Refactoring
- UML
- Exercise
- Readings this week:
  - http://www.sei.cmu.edu/architecture/definitions.html
- Readings next week: Dvorak paper, Fowler paper "Who needs an architect" - email to you, An Introduction to Software Architecture David Garlan and Mary Shaw – email to you, Vitruvius- On Architecture, Book 1 chapters 1-3, http://penelope.uchicago.edu/Thayer/E/Roman/Texts/Vitruvius/home.html
- My web site http://homepage.mac.com/vesonder
- My blog - check at web site



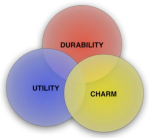
Goals of SSW565

- To understand the current state-of-the-practice and state-of-the-art in Software Engineering for Architecture, Design and Refactoring
- To further develop your sense of appreciation and "taste" for Architecture, Design and Refactoring
- To begin or accelerate a continuous learning approach to Software Engineering in theory and practice
- To impress on you the responsibility of your profession
- WHAT ARE YOUR GOALS?



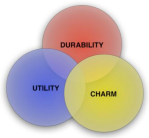
Syllabus

- Two books on design and refactoring - no book on architecture
- True graduate course in many ways
- Note grading policy:
  - 2 in class tests, each worth 35% of your grade
  - Final is **not** cumulative, but the knowledge is
  - extra credit on tests does not carry over, i.e., max grade is 100%
  - logbook worth 15%
  - papers, homework, class participation, blogging worth remaining 15%
  - Extra credit ONLY on tests
- **NO GRADE GRUBBING!!!! No plagiarism!**



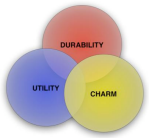
Log Book

- Preferably electronic (but as you use it, a bound book)
- Contains thoughts and insights about software architecture, design refactoring, …
- Should have at least a paragraph/biweekly (5 entries)
- Review at least one each week in class or on blog- with text copy
- NOT CLASS NOTES! NO PLAGIARISM!
- Submit on TBD, returned after exam
- Method to this - should be part of your professional life, time to start!



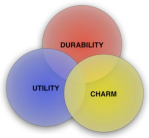
Key Dates

- ?



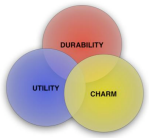
Software Engineering Ethics

- Numerous disasters due to software
- Software projects are pressure filled
- Software projects rely on relationships and trust
- IEEE Computer Society and ACM have developed a software engineering code of ethics with eight principles
- Think of the roles software plays in your life - health, transportation, finances, …



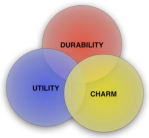
SE Code of Ethics

1. Public - shall act consistently with the public interest
2. Client and employer - shall act in a manner that is in the best interests of their client and employer and that is consistent with the public interest
3. Product - shall ensure that their products and related modifications meet the highest professional standards possible
4. Judgment - shall maintain integrity and independence in their professional judgment
5. Management - shall subscribe to promote an ethical approach to the management of software development and maintenance
6. Profession - shall advance the integrity and reputation of their profession consistent with the public interest
7. Colleagues - shall be fair to and supportive to their colleagues
8. Self - shall participate in lifelong learning regarding the practice of their profession and promote an ethical approach to the practice of the profession



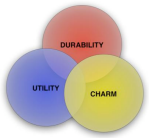
Vesonder's Relevant Bio

- Software for 30+ years
- PhD in Cognitive Psychology - Computer modeling of learning and memory
- [Bell|AT&T] labs for 30 years
- Dozens of projects
- Reviewer and served in Software Technology Center
- Stevens: SSW540 (web version too), SSW565(web version too), HCI and computer mediated entertainment
- University of Pennsylvania
- **Now your turn!**



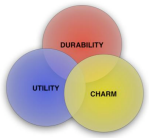
Logbook

- Honestly, measure twice cut once
- Any takers?

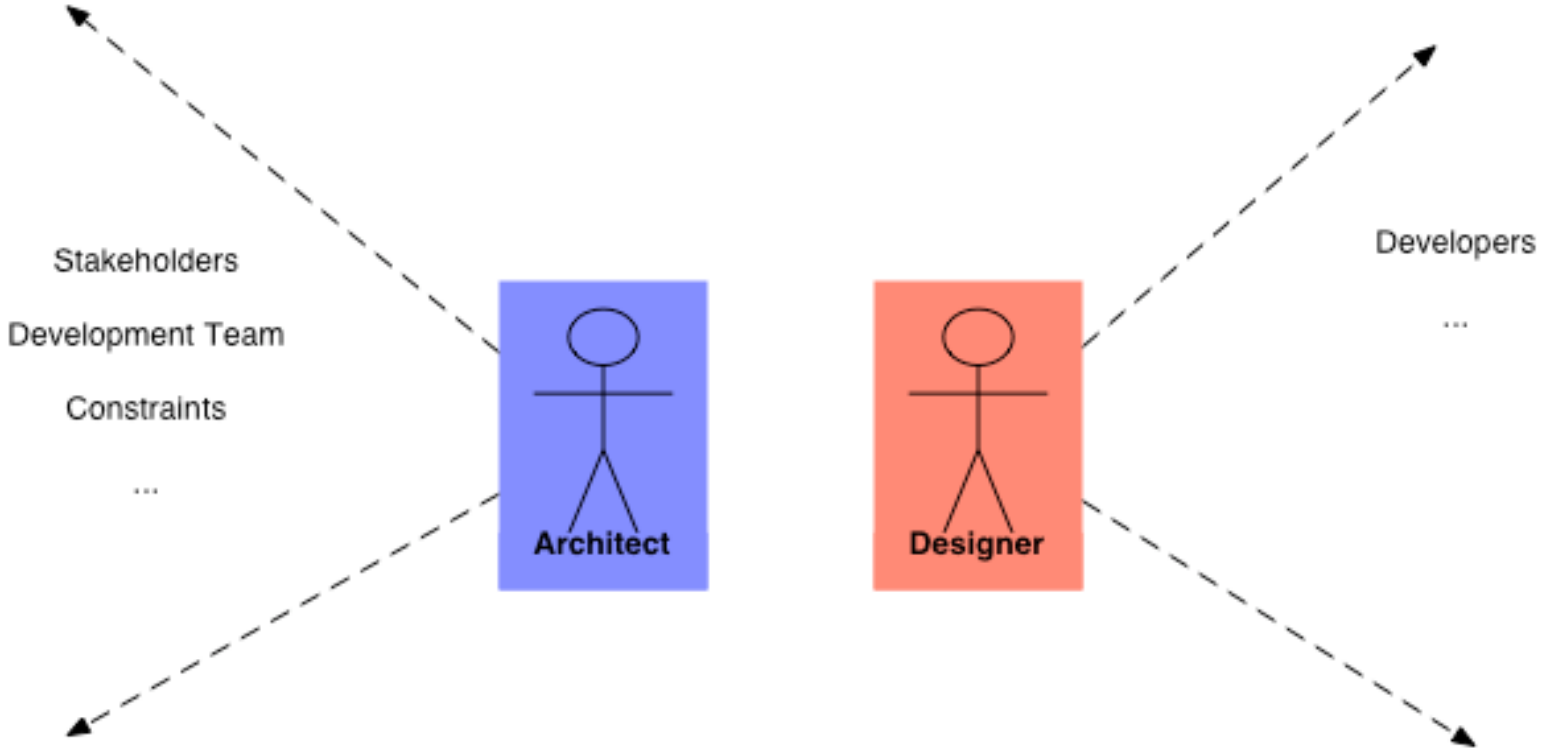


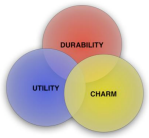
Software Refactoring

- Improving the design of existing software code.
- Refactoring does not change the observable behavior of the software: it improves its internal structure. (source webopedia)
- Improvement includes:
  - easier to understand
  - cheaper to modify
  - Less code(hopefully)
- Of the 3, the easiest to define
- http://www.refactoring.com



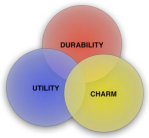
Architecture and Design





Software Design

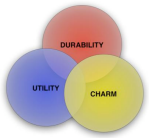
- The blueprint and parts list that is followed during construction
- The parts list emphasizes this is the main time to consider reuse (after architecture)
- Concentrate on domain models, patterns and refactoring



Software Architecture

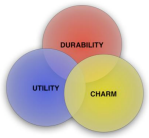
- sei web site, http://www.sei.cmu.edu:
  - sei = Software Engineering Institute - key web site
- **Boehm, et al., 1995**
  Barry Boehm and his students at the USC Center for Software Engineering write that:
  A software system architecture comprises:
  -A collection of software and system components, connections, and constraints.
  -A collection of system stakeholders' need statements.
  -A rationale which demonstrates that the components, connections, and constraints define a system that, if implemented, would satisfy the collection of system stakeholders' need statements.
- "The architecture of a software intensive system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them." (Rozanski & Woods, 2005, derived from SEI)
- Ralph Johnson via Martin Fowler "… Architecture is about the important stuff, whatever that is"

The Matrix Reloaded



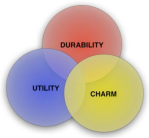
Sommerville on Architecture

- "Architecture represents a critical link between requirements and design" p242
  - "There is a significant overlap between the processes of requirements engineering and architectural design."p243
- Stakeholder communication, system analysis and large scale reuse



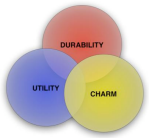
Relevant Current Issues

- Why use an architect?(adapted from Rozanski & Woods 2005)
  - Users have become more demanding
  - Long system development times result in churn in focus and architecture and design
  - Systems usually are constructed from a mix of off the shelf and custom components increasing functional and structural complexity
  - Modern systems rarely exist in isolation they are usually embedded in or are themselves systems of systems
  - How the system behaves (non functional requirements) is just as important as what the system does (functional requirements)
  - Technology is advancing rapidly and more off the shelf platforms are being introduced seemingly daily



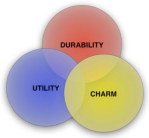
Where do Architectures come from?

- (from SAP)
- System Stakeholders
  - customer
  - marketing
  - end users
  - developers
  - development organization
  - maintenance organization
  - ...
- Technical and Organizational Factors
  - business needs may affect schedule, less time,different arch
  - technical milieu
  - background and experience of the architects
  - current investments in hardware and software
  - ...



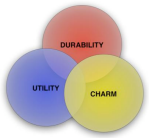
Architecture Based Process Steps (SAP)

- Create business case - constraints
- Understand requirements (elicit)
- Creating/selecting architecture - conceptual integrity (Brooks) ONE ARCHITECT!
- Representing and communicating architecture
- Analyzing/evaluating architecture
- Implementing the system based on architecture
- Ensuring implementation conforms to architecture



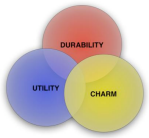
What makes a good Architecture?

- SAP - Process Recs
  - product of a single or small group of architects
  - technical requirements + clear, recorded list of qualitative properties (e.g., portability) with priorities
  - architecture documented in a notation/scheme all stakeholders can understand **with minimum effort**
  - stakeholders are actively involved in review of architecture
  - architecture should be assessed on quantitative (throughput) and reviewed on qualitative (modifiability) properties early on - before too late to change
  - amenable to implementation of skeletal system with minimal functionality in order to test communication paths
  - should provide a small, specific set of resource contention areas which are budgeted and well publicized (time, cpu, memory, network use ...)



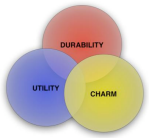
What makes a good Architecture?

- (SAP) structural
  - should feature well-defined modules with functional responsibilities allocated on the principles of information hiding and separation of concerns with a well defined interface (cohesiveness and coupling)
  - should have modules that can be worked on by independent teams
  - information hiding should also consider encapsulation idiosyncracies of the computing infrastructure (COM, Corba, ...)
  - should never depend on a particular version of a commercial product or tool and, if this is unfortunately necessary, should minimize the effects
  - modules that produce data should be separate from modules that consume data
  - the architecture should feature a small number of interaction patterns, i.e., the system should do the same thing in the same way throughout



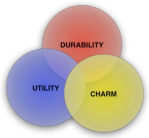
Sommerville's Architectural Questions

- (p. 245) Is there a generic application architecture that can act as a template for the system that is being designed?
- How will the system be distributed across a number of processors?- multiple cores too!
- What architectural style or styles are appropriate for the system? – more later, Garlan and Shaw
- What will be the fundamental approach used to structure the system? Client-server, layered
- How will the structural units in the system be decomposed into modules? OO, function oriented
- What strategy will be used to control the operation of the units in the system? Centralized, event-driven
- How will architectural design be evaluated?
- How should the architecture of the system be documented?



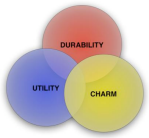
Why is Software Architecture important?

- Communication among stakeholders
- Early design decisions and analysis
- (Blueprint expressing the constraints)
- Transferable abstraction/Reuse
- Has the architect maintaining the conceptual integrity



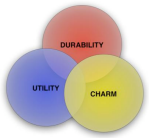
Software Architecture and Notation

- Boehm again (Foreword of Shaw and Garlan's Book)
  "...the shortage of intermediate abstractions that connect the characteristics of system users' need to the characteristics that software engineers can build"



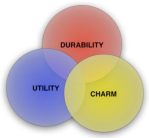
The UML

- 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

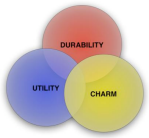


Class Diagram

Employee
Name Address Age Salary
ChangeField() ReadField()

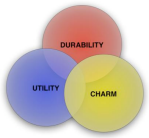
Robot
SerialNumber Color MemoryCapacity Fuel
Move() FireWeapon() SeekEnergy()

Class Name
Attributes Attributes
Operations Operations

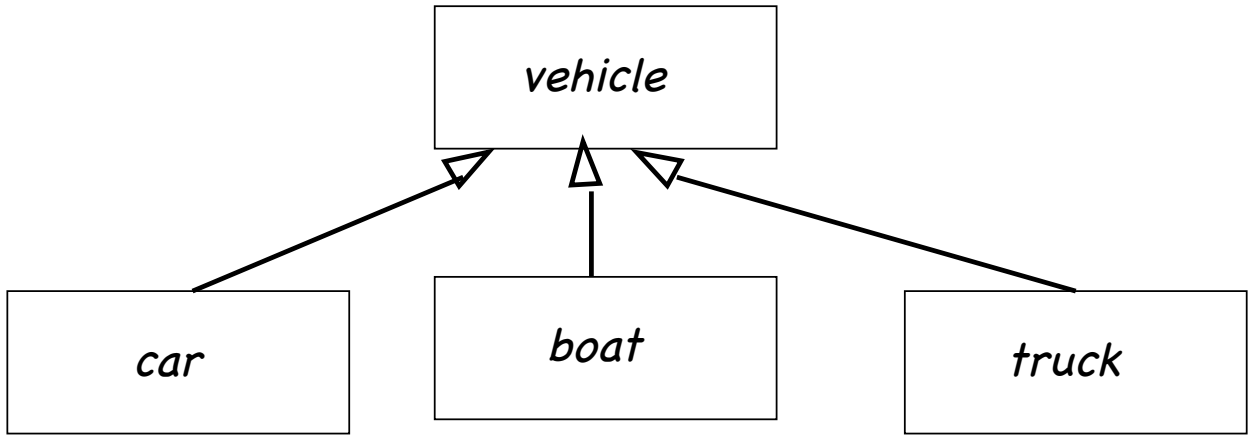


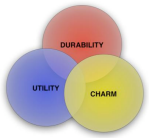
Object Diagram

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

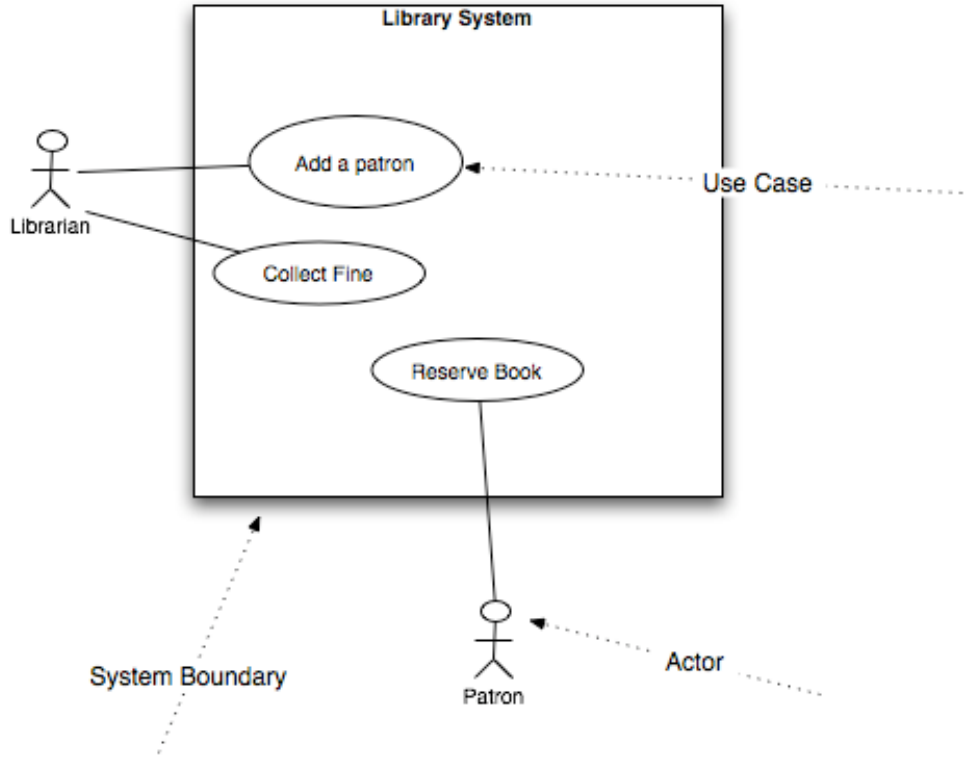


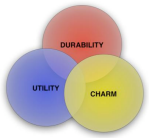
Inheritance



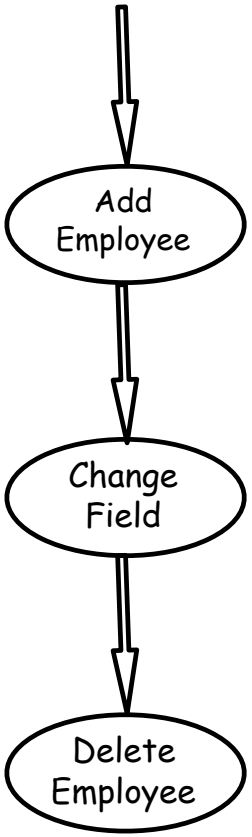


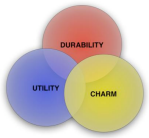
Use Case Diagram



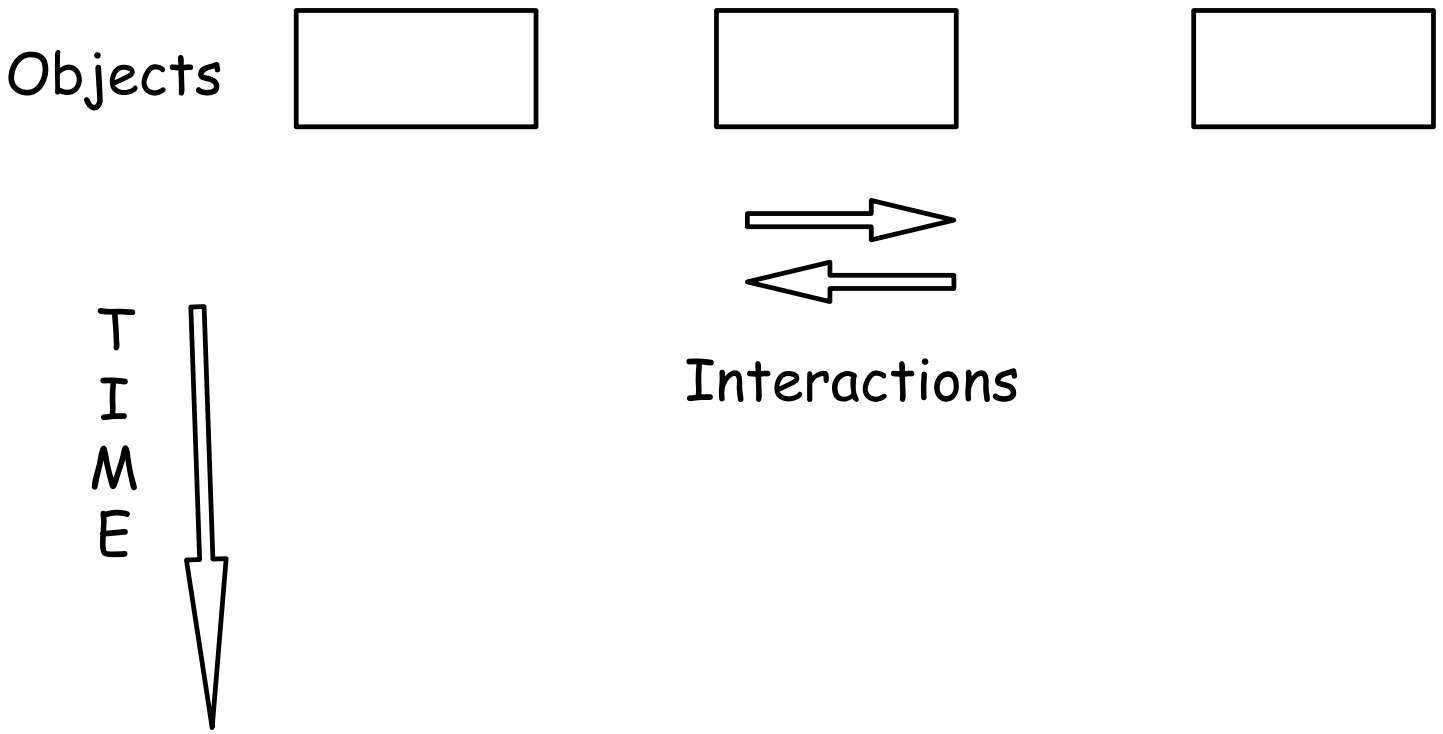


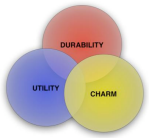
State Diagram



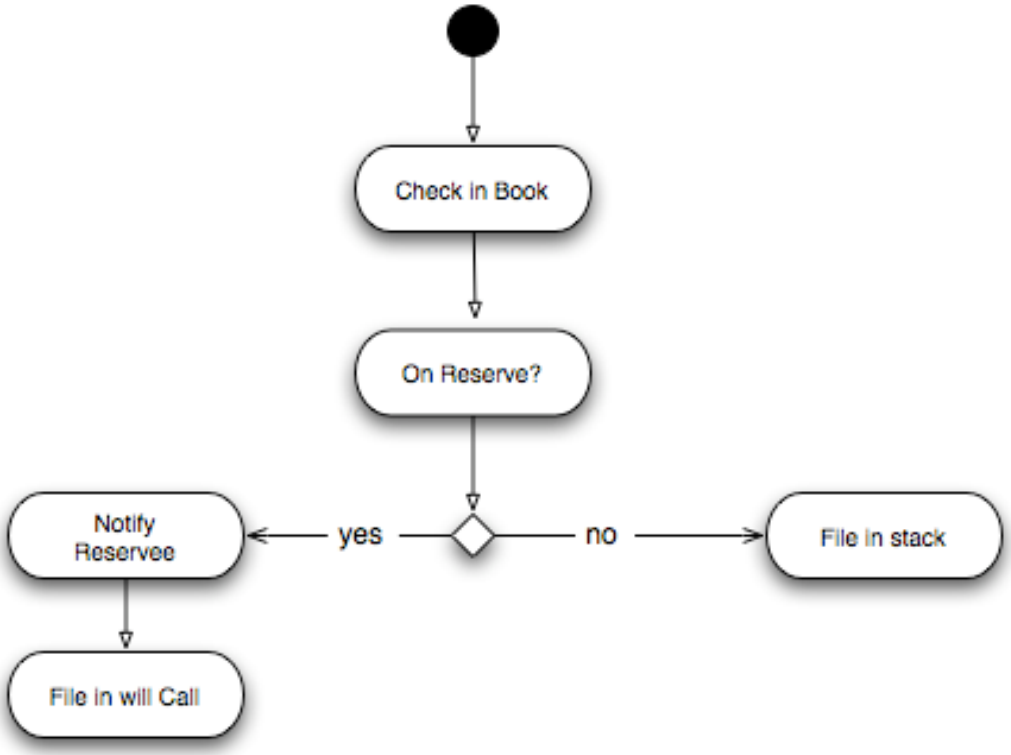


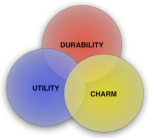
Sequence Diagram



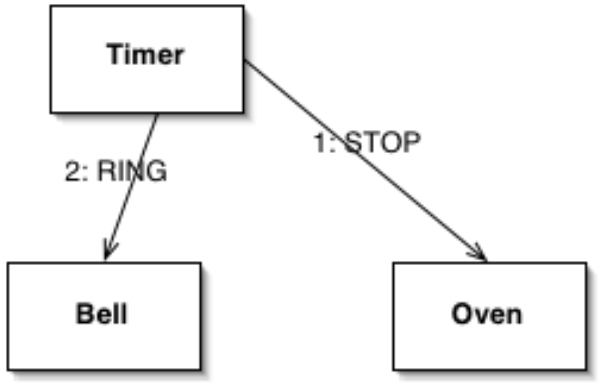


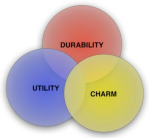
Activity Diagram



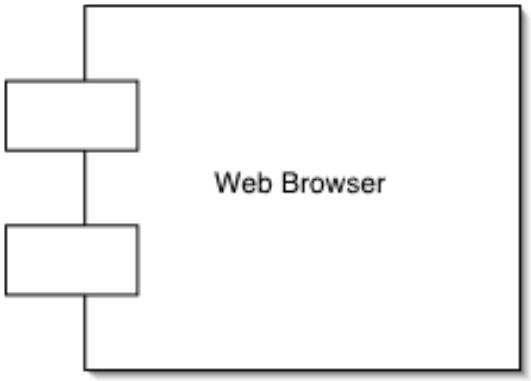


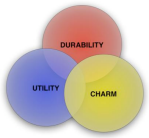
Collaboration Diagram



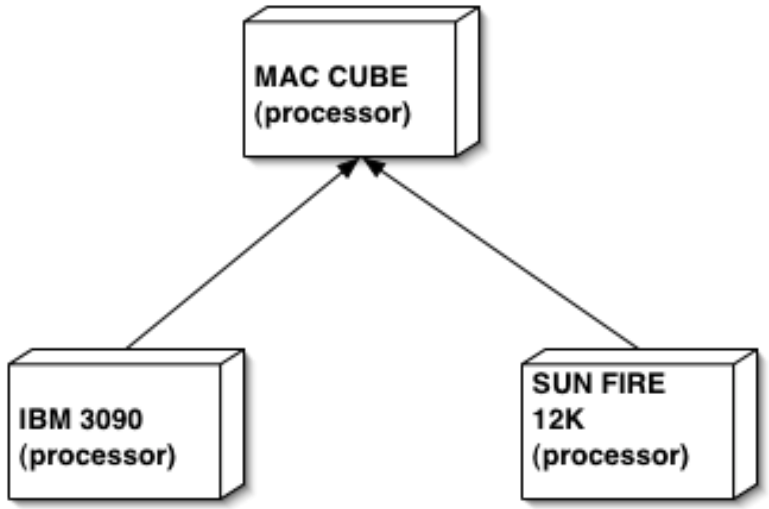


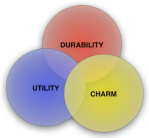
Component Diagram





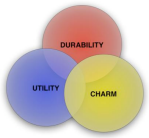
Deployment Diagram





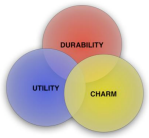
In Summary

- The course
- What refactoring, architecture and design are
- Architecture Overview
- The UML



Assignment 1

- Go to the SEI architecture definition site, select the definition you most agree with and present it along with a paragraph explaining why
- Send me your email address



Other References

- Martin Fowler, UML Distilled, Addison-Wesley, 2004.
- http://www.sei.cmu.edu/architecture/definitions.html
- Bass, L., Clements, P. & Kazman, R. Software Architecture in Practice, 1998, Addison-Wesley.
- Rozanski, N. & Woods, E. Software Systems Architecture, Addison-Wesley, 2005, ISBN 0-321-11229-6.
- Sommerville, I. Software Engineering, 8th Edition, Addison-Wesley, 2007, ISBN: 0-321-31379-8