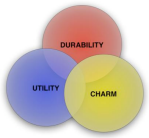
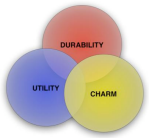


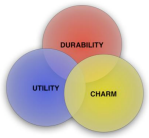
Class 2 SSW565

Gregg Vesonder
Stevens Institute of Technology



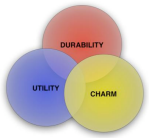
Roadmap

- logbook
  - me
  - Volunteer
- Vitruvius -> Fowler
- So what is an architecture?
  - NASA/JPL paper begins your apprenticeship
- Going through the process
- Next week styles and connectors
- Readings this week: Dvorak paper, Vitruvius, Fowler - emailed to you
- Reading next week: van Vliet chapter 10 (if you have/can borrow it), "connector" paper, Mehta, Medivdovic & Phadke, 2000



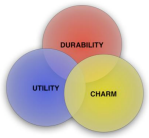
Key Dates

- Thursday, May 28th
- ?



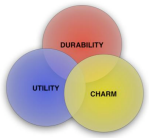
Clarifications

- Stakeholder - (IEEE 1471) is a person, group or entity with an interest in or concerns about the realization of the architecture.
  - (from Rozanski & Woods)
  - Architectures are created solely to meet stakeholders needs
  - A good architecture meets the objectives, goals and needs of the stakeholders

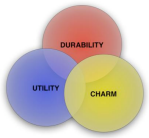


Logbook

- Think globally, code locally
- Your turn

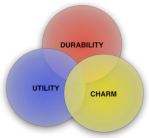


Your Favorite/Worst Arch Definitions



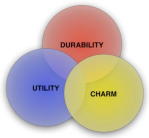
Vitruvius

- Insight into the formation of the concept of architect:
  - "definite directions for the conduct of works"
  - A science arising from many sciences
- Distinguishes theory and practice:
  - "Practice and theory are its parents. Practice is the frequent and continued contemplation of the mode of executing any given work, or of the mere operation of the hands, for the conversion of the material in the best and readiest way. Theory is the result of that reasoning which demonstrates and explains that the material wrought has been so converted as to answer the end proposed."
  - "In theory there is no difference between theory and practice, in practice there is" - Yogi Berra
- Must understand intention and the matter used to express that intention - you have to be a doer too



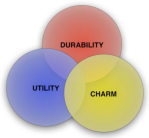
Vitruvius-2

- A good architect is a good writer, draftsman, versed in math and history
  - "commit to writing his observations and experiences to assist his memory" Logbooks!
  - "be above meanness in dealings"
  - "avoid arrogance"
  - "law should be an object of study" (just like physical laws)
  - "… from their early youth …" Is a journey
- An architect unlike an artist has to master many disciplines - "required of the architect to be generally well informed … he cannot hope to excel in each art"



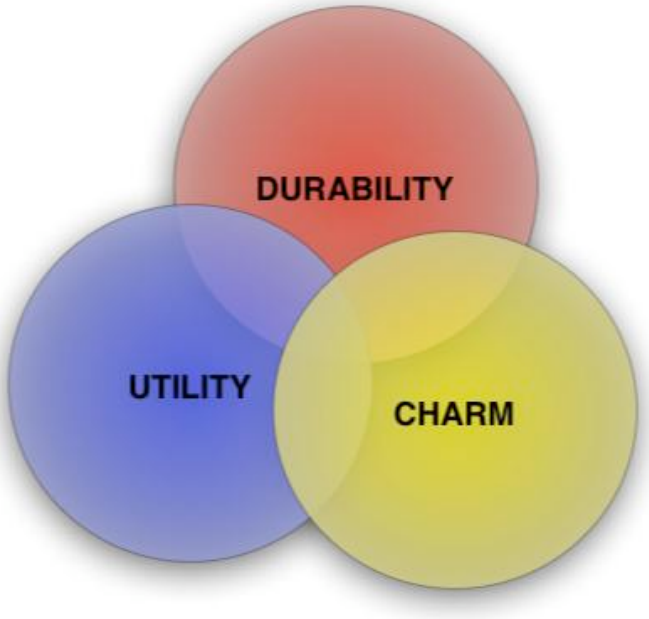
Vitruvius-3

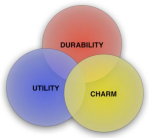
- Architecture depends on fitness and arrangement along with proportion, uniformity, consistency and economy
  - Consistency: templates, conforming to legacy
  - Economy: avoid materials not easily procured or prepared on the site (developer tools and skills)
- What we build should possess strength (durability), utility and beauty(charm)



Keys to Success

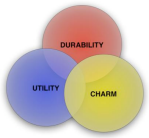
All these should possess strength, utility, and beauty. Strength arises from carrying down the foundations to a good solid bottom, and from making a proper choice of materials without parsimony. Utility arises from a judicious distribution of the parts, so that their purposes be duly answered, and that each have its proper situation. Beauty is produced by the pleasing appearance and good taste of the whole, and by the dimensions of all the parts being duly proportioned to each other. --Vitruvius, book 1, chapter 3, section 2





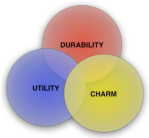
Fowler, On Architecture

- Contrarian opinion -- equal time
  - Agile methods - more later in the course
- "architecture is a word we use when we want to talk about design but want to puff it up to sound important."
- Ralph Johnson - "…there is no (one) highest level concept of a system" multiple views
  - Architecture as a shared understanding of the system design (more during design)
- Awareness of the entire project, intense collaboration, mentor develops -- be a guide
- Leads to



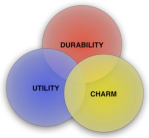
Get Rid of Architecture

- Relation to building architect failed metaphor, software is easy to change unlike actual buildings
  - Contain complexity by reducing irreversibility - Fowler thinks this is important, eliminating irreversibility
  - But be careful … "making something easy to change makes the overall system a little more complex, and making everything easy to change makes the entire system very complex" Johnson
- Johnson concludes: "(Software) is limited by imagination, by design, by organization. In short it is limited by properties of people, not by properties of the world. We have met the enemy and he is us."
- I think we can accommodate these views even in a more standard arch approach (agilists would disagree, but Yogi quote) - perfect for smallish systems, skilled folks



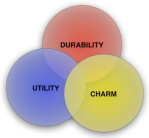
Every system has an architecture!

(however, it may not be described and it may not be very good)



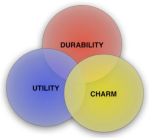
Architecture

- What is an architecture?
- What does an architect do?
- Your apprenticeship
- Snapshot in the JPL/NASA themes paper
  - "several architectural themes shaping the MDS design"
  - Object-oriented design, component architectures, domain specific frameworks.
- Problem to be solved: "No common architecture or frameworks for them (missions) to draw from."



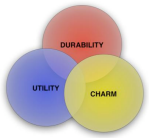
Objectives

- JPL/NASA Objectives:
  - Earlier collaboration of mission, system and software design
  - Simpler, lower cost design test and operation (general outcome of a good architecture)
  - Customer-controlled complexity
  - Evolvability to in place exploration and other autonomous applications (architecture considers future)
- So let's examine the themes that get them there

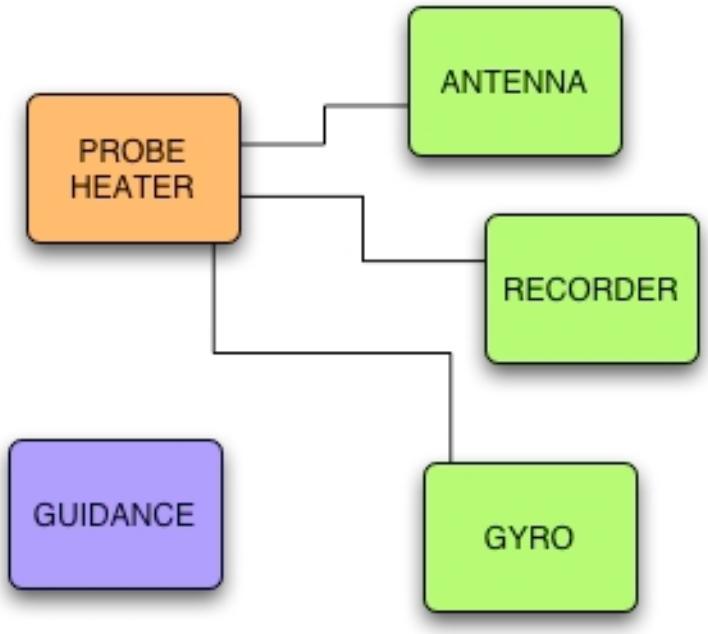


Use Architectural Elements

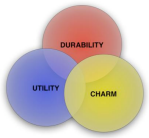
- "Construct subsystems from arch elements, not the other way around"
  - Work compartmentalized to function, e.g., navigation vs. power vs. propulsion vs. telecom
  - Managing interactions is foundation of good design
    - Example:power coordination service vs. individual module brokering -> decreases coupling between subsystems
    - Simplifies testing, not necessary to have two modules necessary, rather module and power broker
    - It is an arch element that is used in design - ubiquitous solution rather than lots of "one-ofs"



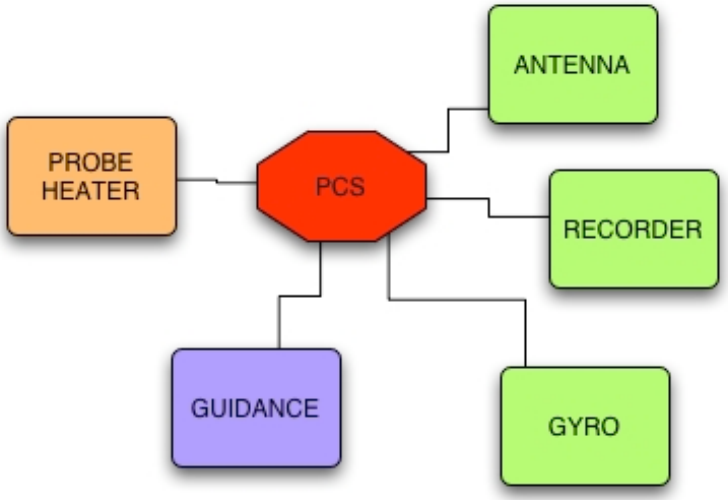
Power Distribution



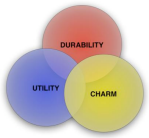
- Heater wants power
- Sends message to each of the systems
- Now engineers add guidance - need to accommodate but issue with priorities
- If antenna, recorder, gyro needs more power need to add more links and compute priorities w/in each module, …



Power Distribution Improved

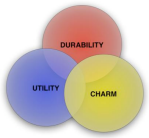


- PCS: Power Control System
- Central module - only one connection required from a module to central module regarding power
- Priorities are adjusted there
- Can modify power management in one place instead of in each module - Probe heater sends need 15 watts



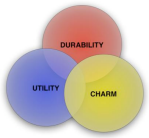
Theme 2: Simplify Operations

- "Migrate capability from ground to flight to simplify operations"
  - More onboard processing, less communication
  - Automate with inflight experience
  - Less human-in-the-loop
  - Quick reaction to events
  - Flight and ground must have shared arch (in order to migrate)



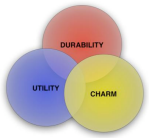
Theme 3: State and Models

- "State based arch - momentary condition of evolving system"
  - Resource levels, temperature, pressure
  - Some states are expressed as functions of other states
  - State variables, state timelines
  - Requires models that describe how state evolves
    - Operate
    - Predict future state
    - Control to desired state
    - Assess performance



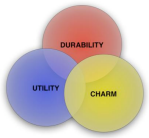
Theme 4: Explicit Use of Models

- "Express domain knowledge explicitly in knowledge rather than implicitly in program logic"
  - Programs logic expressing domain knowledge is hidden and hard to validate and reuse (architect for test)
  - Expressed in spreadsheets, rules, state machines
  - Separate domain knowledge form the logic to apply that knowledge -- mini expert systems



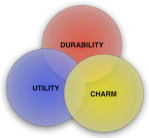
Theme 5: Goal Directed Operation

- "Operate missions via specifications of desired state rather than sequences of actions"
  - Past used linear command sequences designed on ground.
    - Must know current state of the spacecraft (difficult due to distance, …)
    - If actual state differs from expected state, system should fail rather than do harm (Asimov's laws of robotics)
  - Rather control states by goals
    - Goal = prioritized constraint on value of a state variable
    - Differs from command in that it specifies intent in the form of the desired state rather than a specific action
    - Goal is easier to specify than a command
    - Provides option of alternate command paths



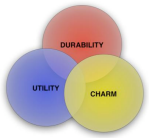
Theme 6:Closed Loop Control

- "Design for real-time reaction to changes in state rather than for open loop or earth-in-the-loop commands"
  - Accomplished by goal achieving module (GAM) compares present state to desired state (remember hobbits and orcs) and deciding how to change state if necessary
  - Always keep goal issuer informed of status
  - GAMs often issue subgoals to achieve goals
  - Self-checking (always comparing current state with goal or subgoal state) so goal failures are visible, as is the trace



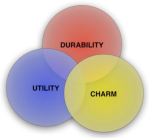
Theme 7: Integrated Fault Protection

- "Fault protection must be an integral part of the design, not an add-on"
  - Fault detection, localization and recovery usually designed after control system and late in the project
  - Minimum level of fault protection is that they must report when an active goal is not being achieved
  - Also may specify recovery strategy



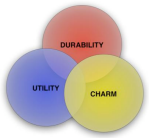
Theme 8: Real Time Resource Management

- "Resource usage must be authorized and monitored by a resource manager"
  - (see Theme 1)
  - Overuse of resources is a bad thing and ground controllers have been real conservative which may unnecessarily bound what can be accomplished
  - Reserve resources and assign priorities, can oversubscribe, manager doles out until exhausted
    - Because of close loop on craft knows actual values and the effect of realizing each request (which may be different than what was asked for)
  - Arch for resource management a key to the stakeholders



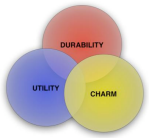
Theme 9: Separation of State Determination and Control

- "For consistency, simplicity and clarity separate state determination logic from control logic"
  - Controllers sometimes share state variables (feet and meters, ☺) their estimates or units of measurement may differ
  - Two distinct tasks in same module, cohesion issues, no separation of concerns
  - Simplifies design, programming and testing



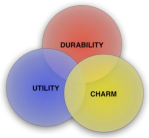
Theme 10: Acknowledge State Uncertainty

- "State determination must be honest about the evidence; state estimates are not facts"
  - Many roads to uncertainty: conflicting evidence, sensor degradation, rapid dynamic change
  - Provide degree of (un)certainty - potential reason for abandoning goal



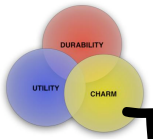
Theme 11:Separate Data Management From Data Transport

- "Separate data management duties and structures from those of data transport"
  - Craft as nodes in an inter-planetary network
  - Data management as separate function with objects and files can be updated, abstracted and aged, some never hit ground
  - Data products treated consistently in both places (file management on rover)
  - Data transport can access any data product and prepare if for transport between flight and ground -- formats and protocols are hidden from data management
  - They are decoupled



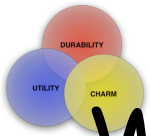
Theme 12: Join Navigation with Attitude Control

- "Navigation and attitude control must build from a common mathematical base"
  - "common architectural mechanisms for common problems"
  - Share a common model - both are solving geometry problems

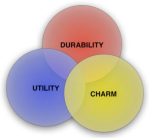


Theme 13: An Architecture with a Future

- "Design interfaces to accommodate foreseeable advances in technology"
  - Balance between exploiting new technologies and maintaining architectural stability -- for cost reasons
  - Accomplished by careful design of architectural interfaces
    - Create a subset of the interfaces to accommodate current needs.
  - Design of a family
  - Architect and design for the future



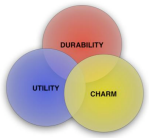
What are the Main Concerns of the Stakeholders?



An Exercise

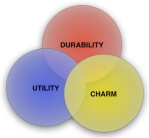
What themes would you use to build an interplanetary internet architecture for robots on Mars and the Moon?

Planet	Roundtrip Time
Moon	2.6 seconds
Mars	6.5 - 44 minutes



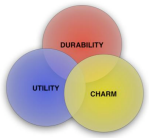
Achitecture - Key Concerns

- System Decomposition:
  - How do we break the system up into pieces?
  - Do we have all the necessary pieces?
  - Do the pieces fit together?
- Cross-Cutting Concerns:
  - Broad-scoped qualities or properties of the system
  - Tradeoffs among the qualities
  - Aspect oriented development
- System Integrity
  - System integrity cannot be achieved bottom-up.



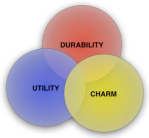
Architecture - KC2

- Alignment with Business
  - With business strategy
  - With business environment
    - Legacy and existing investments
    - Organizational capabilities and culture
  - With customers and channel
- System Evolution
  - Architectures are long-lived,
  - They must provide the blueprint for implementing today's strategy, and
  - They must to be able to evolve, to match the changes in the business strategy.



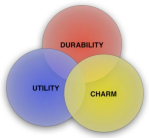
Bernstein on Architecture

- Framework of modules
- Design of interfaces
- Constraints on modules such as size and execution
- Enforces properties of capacity, throughput, consistency and module compatibility
- (note that NASA/JPL was designing a family so they dictated analysis characteristics too)
- Sees three components: processing, communication and data architecture

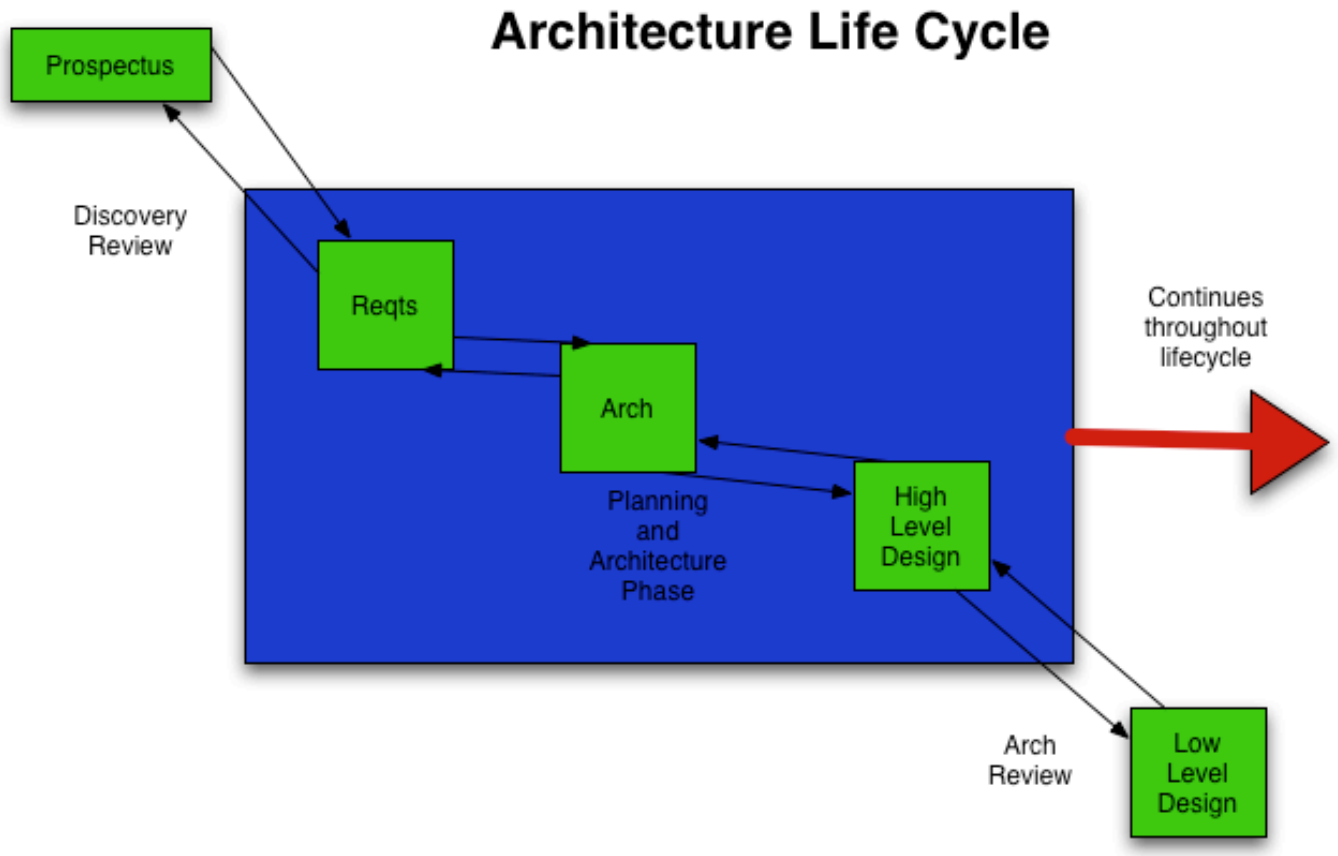


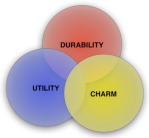
More Bernstein

- Advocates architect as role
- "the framework for all technical decisions"
- Architecture evolution in a project
- 4+1



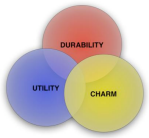
Project Arch Life Cycle



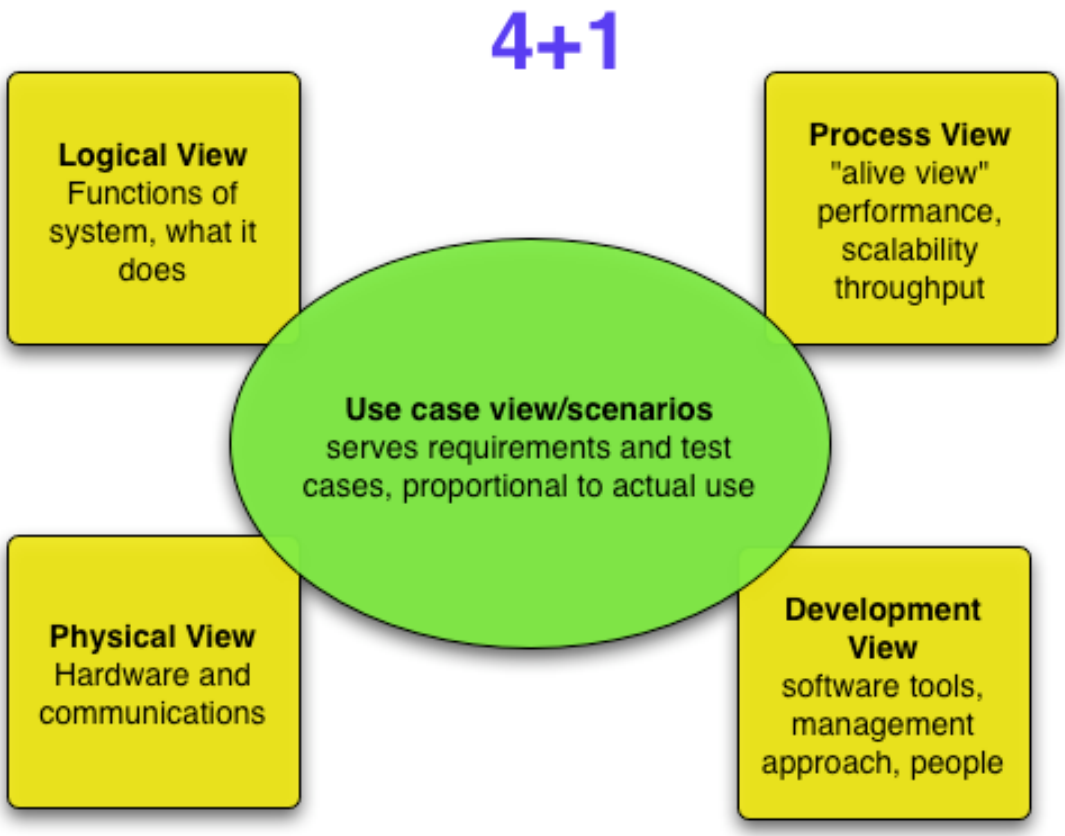


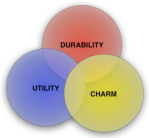
The "4+1" Architecture Model

- Proposed by P. Kruchten in 1995.
- It is an architecture model for describing the architecture of software-intensive systems based on the use of multiple, concurrent views.
- Allows the addressing of concerns separately of the various stakeholders including end-user, developers, system engineers, project managers, etc.
- Allows handling of functional as well as non-functional requirements.



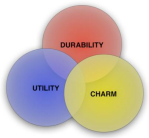
Architecture Approach





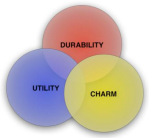
The "4+1" Architecture Model

- The Logical Architecture
  - The logical architecture presents an object-oriented decomposition of the system.
  - It supports primary the functional requirements.
  - The decomposition:
    - Produces objects or object classes.
    - Identifies common mechanisms and design elements across the various parts of the system.
  - To represent the logical architecture:
    - O-O approach (Rational Rose)
      - Class diagram
      - Class template
    - Non-O-O approach can use the E-R diagrams



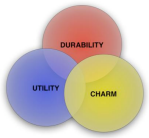
The "4+1" Architecture Model

- The Process Architecture
  - The process architecture presents the process decomposition.
  - It takes into account some non-functional requirements.
  - It uses multiple levels of abstraction to address different concerns.
  - Processes represent the level at which the process architecture can be tactically controlled, and can be replicated to handle increased processing load as well as improved availability.



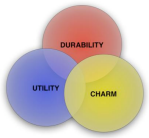
Background on Process Arch

- A process is a grouping of tasks that form an executable unit.
- The software is partitioned into a set of independent tasks.
- A task is a separate thread of control.
- Major tasks are:
  - Architecture elements that can be uniquely addressed.
  - Communicating via a set of well-defined inter-task communication mechanisms (RPC, MQ, event broadcast, etc).
  - Not always collocated in the same process or processing node.
- Minor tasks are:
  - Light-weight threads providing local helps.
  - Communicating by rendezvous or shared memory



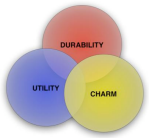
The "4+1" Architecture Model

- The Development Architecture
  - It focuses on the actual software module organization on the software development environment.
  - It is represented by model and subsystem diagrams, showing the export and import relationships.
  - The complete development architecture can only be described when all the elements of the software have been identified.
  - The rules governs the development architecture:
    - Partitioning
    - Grouping
    - Visibility



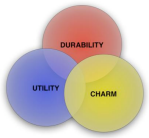
The "4+1" Architecture Model

- The Development Architecture
  - It takes into account internal requirements:
    - Constraints imposed by:
      - The programming language
      - The toolset available
    - Ease of development:
      - Software management
      - Reuse
      - Commonality
  - It serves as the basis for:
    - Requirement allocation
    - Cost evaluation and planning
    - Distribution of work to teams



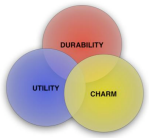
The "4+1" Architecture Model

- The Physical Architecture
  - It takes into account primarily the non-functional requirements of the system.
  - It maps the various elements (networks, processes, tasks, and objects) onto the various nodes (a network of computers, or processing nodes).
  - The mapping needs to be highly flexible and have a minimal impact on the source code.



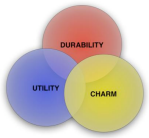
The "4+1" Architecture Model

- Scenarios
  - This architecture is redundant with the other 4 (hence the "+1").
  - It serves the following roles:
    - As a driver role – to discover the architectural elements during the architecture design.
    - As an illustration role – when the architecture design is completed.
    - As a validation role – the starting point for the tests of an architecture prototype.
  - The scenarios are an abstraction of the most important requirements, and illustrate how the elements in the 4 views work together seamlessly.



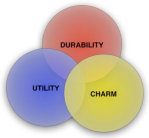
The "4+1" Architecture Model

- Correspondence Between the Architectures
  - These various architectures are not orthogonal or independent.
  - Elements of one architecture are connected to elements in other architectures, following certain design rules and heuristics.
  - Look at:
    - Logical Architecture ➔ Process Architecture
    - Logical Architecture ➔ Development Architecture
    - Process Architecture ➔ Physical Architecture



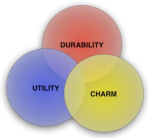
The "4+1" Architecture Model

- Don't Forget to Document It !
  - Key outlines of a software architecture document:
    - Scope
    - References
    - Software Architecture
    - Architecture Goals & Constraints
    - Logical Architecture
    - Process Architecture
    - Development Architecture
    - Physical Architecture
    - Scenarios
    - Size and Performance
    - Quality



Non Glamorous Features

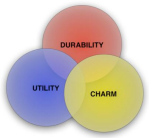
- Training users
- Configuring computer to run software
- Defining network requirements
- Populating data, tables, data bases
- Monitoring and maintaining adequate response time
- Trouble shooting -- common logging architecture -- essential
- Your favorite



Rozanski and Woods

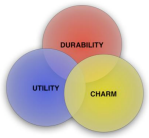
- Functional Viewpoint
- Information Viewpoint
- Concurrency Viewpoint
- Development Viewpoint
- Deployment Viewpoint
- Operational Viewpoint

An alternative to 4+1



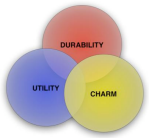
R & W Perspectives

- Security
- Performance & Scalability
- Availability & Resilience
- Evolution
- Accessibility
- Development Resource
- Internationalization
- Location
- Regulation
- Usability



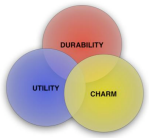
In Summary

- Discussed alternate views of the value of an architect
- Looked at both ancient and modern views of architecture
- Looked at two architecture schemes 4+1 and Rozanski and Woods



An example: Video Game

- A very simple Doom like video game - players walk through a maze, shooting software generated bad creatures with some intelligence. Sometimes these critters are called AI or bots.
- There can be more than one player playing and in this instance, the games are hosted on a centralized site.
- Let's call the game Gloom



Other References

- Dvorak, D., Rasmussen, R., Reeves, G. & Sacks, A. "Software architecture themes in JPL's mission data system." AIAA, 1999.
- http://www.sei.cmu.edu/architecture/definitions.html
- Vitruvius, On Architecture
- Fowler, M. "Who needs an architect?" IEEE Software, 2003
- N. Rozanski & E. Woods Software Systems Architecture (2005)