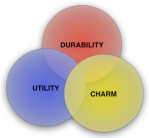
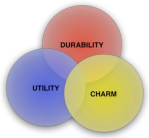


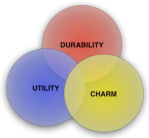
Class 6 SSW565

Gregg Vesonder
Stevens Institute of Technology



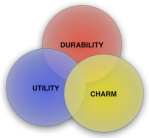
Roadmap

- ATAM
- Risk
- Agents
- Architecture Reviews
- Domain Driven Analysis
- Architecture Modeling
- Information Architecture
- Review
- Readings this week: Starr & Zimmerman and Hall
- Readings for next Monday: Part 1 in Evans, chapters 1-4



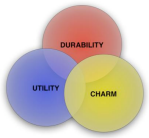
Key Dates

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



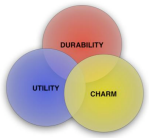
Log Book

From http://www.culture.gouv.fr/culture/arcnat/lascaux/en/



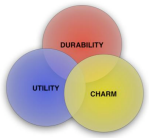
ATAM

- Another method from SEI, ATAM = Architecture Tradeoff Analysis Method
- Based on quality attributes and scenarios, it does not focus on functional correctness - brings stakeholders and technical team together
- scenario attributes similar to quality attributes shown on the next page
- some quality attributes: performance, security, availability, functionality, usability, modifiability, portability, reusability, integratability testability



Quality Attributes
(remember these?)

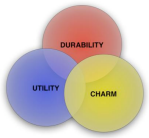
	main with sub	Abstract data	implicit	p&f
changes in data rep	-	+	+	-
changes in algorithm	-	o	o	+
changes in functionality	o	-	+	+
independent development	-	+	+	+
comprehensibility	-	+	o	+
performance	+	+	-	-
reuse	-	+	+	+



ATAM Steps

(standard steps for many software engineering methods)

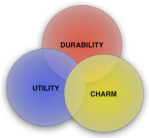
- Present ATAM methodology
- Present business drivers and context
- Present architecture (develop a common understanding)
- Identify architectural styles
- Generate quality attributes and assign priorities
- Examine styles in context of quality attributes identify risks and tradeoffs
- Generate scenarios and assign priorities to them
- Map scenarios onto styles and assess risks
- Provide report
- (WinWin like)



Software Risk Management

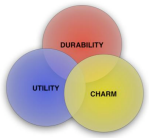
- (much of this adapted from http://www.stsc.hill.af.mil/crosstalk/2005/02/0502stsc.html and the SEI)
- Risk is defined as exposure to harm or loss, not only probability but affect as well.
- NOT RISK AVOIDANCE
- The SEI (and others) phases of risk analysis are:
  - Identify
  - Analyze
  - Plan
  - Track
  - Control

COMMUNICATION



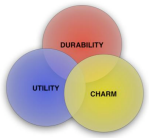
Identify risk

- Risks can be known, unknown and unknowable
- SEI method of risk identification (and management) based on following assumptions:
  - Risks are often known by tech staff but poorly communicated
  - A repeatable method is necessary for risk management
  - Must cover all areas
  - Attitude must be non-judgmental and supportive so that controversial views can be heard
  - Success or failure of the project can not be based solely on risk assessment

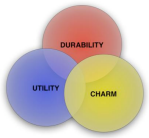


SEI Development Risk Taxonomy

- 3 major categories:
  - Product engineering - what is being developed
  - Development Environment - how it is being developed
  - Program constraints - contractual, organization and operational factors



Taxonomy Expanded
(adapted from CMU/SEI-93-TR-6)

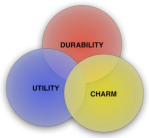


A. Product Engineering
   1. Requirements
      a. Stability
      b. Completeness
      c. Clarity
      d. Validity
      e. Feasibility
      f. Precedent
      g. Scale
   2. Design
      a. Functionality
      b. Difficulty
      c. Interfaces
      d. Performance
      e. Testability
      f. Hardware Constraints
      g. Non-Developmental Software
   3. Code and Unit Test
      a. Feasibility
      b. Testing
      c. Coding/Implement-ation
   4. Integration and Test
      a. Environment
      b. Product
      c. System
   5. Engineering Specialties
      a. Maintainability
      b. Reliability
      c. Safety
      d. Security
      e. Human Factors
      f. Specifications

B. Development Environment
   1. Development Process
      a. Formality
      b. Suitability
      c. Process Control
      d. Familiarity
      e. Product Control
   2. Development System
      a. Capacity
      b. Suitability
      c. Usability
      d. Familiarity
      e. Reliability
      f. System Support
      g. Deliverability
   3. Management Process
      a. Planning
      b. Project Organization
      c. Management Experience
      d. Program Interfaces
   4. Management Methods
      a. Monitoring
      b. Personnel Management
      c. Quality Assurance
      d. Configuration Management
   5. Work Environment
      a. Quality Attitude
      b. Cooperation
      c. Communication
      d. Morale

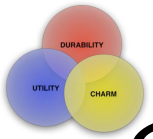
C. Program Constraints
   1. Resources
      a. Schedule
      b. Staff
      c. Budget
      d. Facilities
   2. Contract
      a. Type of Contract
      b. Restrictions
      c. Dependencies
   3. Program Interfaces
      a. Customer
      b. Associate Contractors
      c. Subcontractors
      d. Prime Contractor
      e. Corporate Management
      f. Vendors
      g. Politics

Figure A-1 Taxonomy of Software Development Risks



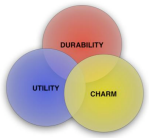
Analyze Risk

- Probability of risk, USAF Handbook categories are very low, low, medium, high and very high
- Impact of risk, USAF Handbook categories are negligible, marginal, critical and catastrophic
- Risks are rarely independent
- A matrix is used to determine overall risk for different categories (e.g., effort, performance, schedule, cost, support)



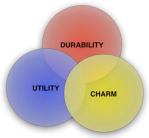
Sample Impact/Probability Matrix
(used to calculate overall risk)

Impact/Probability	Very High	High	Medium	Low	Very Low
Catastrophic	H	H	M	M	L
Critical	H	H	M	L	0
Marginal	M	M	L	0	0
Negligible	M	L	L	0	0



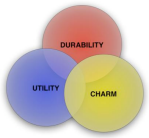
Plan for the Risks

- What can you do:
  - Mitigate impact by developing a contingency plan should risk occur and identify the trigger to initiate the contingency plan
  - Avoid the risk by changing something
  - Accept the risks and the consequences if it occurs
  - Study the risk further so that you can decide on one of the above
- In addition:
  - Specify why risk is important
  - What info is need to track status of risk
  - Who is responsible for Risk Management and what is the cost



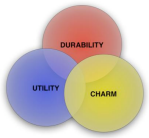
Risk Tracking and Control

- Track like everything else in the project monitoring status of risks and actions taken to address them. Appropriate risk metrics should be in place.
- Control, the risk process should be in place in the beginning, deviations to the plan should be corrected, triggering events should be handled and the process should be assessed for effectiveness.

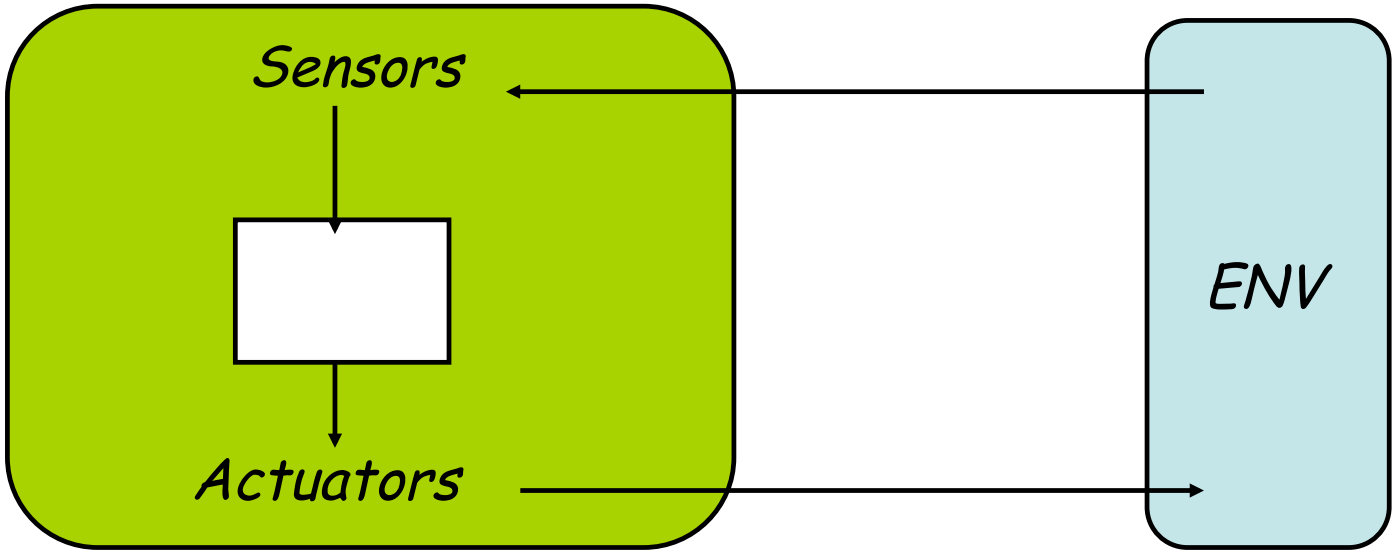


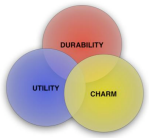
AI & Agents

- Agent: something that acts, operates under autonomous control, perceiving the environment, persisting over long time periods, adapting, being able to take on another's goals
- A rational agent achieves the best outcome or, given uncertainty, the best expected outcome
- Perfect rationality - always doing the right thing, is not feasible in complicated environments
- Limited rationality - acting appropriately when there is not enough time (or ability or feasibility) to do all the calculations one might like



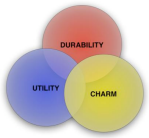
Very Simplified Agent





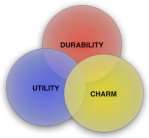
Classifying the Agent Task Environments

- Fully vs. Partially observable -can agent detect all relevant aspects?
- Deterministic vs. Stochastic - is next state of environment solely determined by current state and agent's actions
- Episodic vs. Sequential - does current decision affect future decisions?
- Static vs. Dynamic - can the environment change while the agent is deliberating?
- Discrete vs. continuous applies to state, time and to percepts & actions of the agents
- Single agent vs. Multi-agent - competitive or cooperative? Communication?
- **You're in trouble with partially observable, stochastic, sequential , dynamic, continuous and multi-agent!**



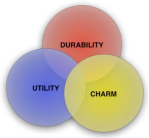
Types of Agents

- Simple Reflex Agents
  - Simple rule bases
- Model based reflex agent
  - Model state reflects part of the "percept history"
  - Information needed:
    - How world evolves independent of agent
    - How agent's actions affect world



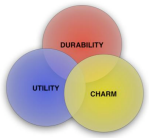
Types of Agents - 2

- Goal based agent
  - Agents use goals to select actions
  - Search and Planning devoted to action sequences achieving agent's goals
- Utility based
  - If one world state preferred to another then it has higher utility for agent
  - Utility functions are the measure
    - Enables tradeoffs on conflicting goals
    - Provides a way to measure likelihood of success against importance of goals



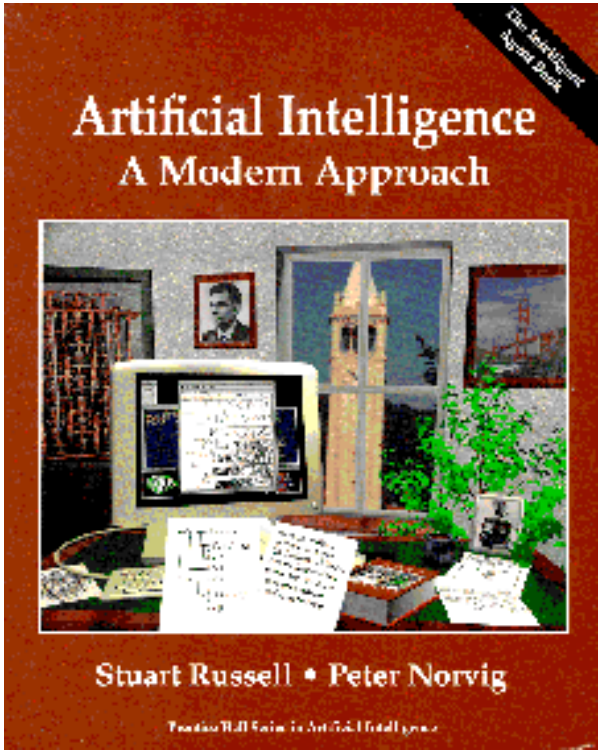
Learning and Agents

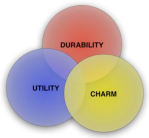
- Four components
  - Learning element
  - Performance element - selects external actions (The agent)
  - Critic - provides feedback
  - Problem generation - suggests actions leading to new informative experiences (exploration)



Reference

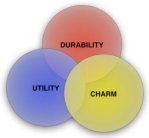
Russell & Norvig Artificial Intelligence: A Modern Approach 2nd Edition, Prentice Hall, ISBN: 0137903952 (now green cover)





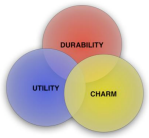
Architecture Reviews

- Adapted from Starr & Zimmerman(2002) and personal experience
- These reviews have been around for more than a decade - 500+ reviews at AT&T and Lucent (SARB, Systems Architecture review Board)
- Stresses architecture and evaluating it early in the project
- Architecture, in this context, is viewed as a solution to a problem for a client and should cover a broad range from cost to client needs
- After you establish an architecture:
  - Decide what and when to test
  - Establish success criteria
  - Make decisions based on your findings



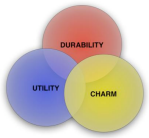
How and what do you address?

- Code inspections examine source code, Design reviews examine models, modules and interfaces
- However a single document or artifact rarely captures architecture
- Architecture is reviewed as an interactive oral presentation which varies from a chalk talk to a few overheads
- Review should be done before contracts are signed or announcements made



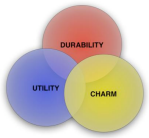
The Architecture Problem Statement

- 1-4 pages in length
- It is criteria to test the architecture and includes:
  - What project must do (functional aspects)
  - What it must be (constraints)
  - Economic constraints (time, schedule, money)
  - How system relates to past (backward compatibility), present and future (evolvability)
  - Unique aspects of the problem (e.g. risks)
- NOT requirements document



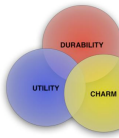
Preparing for the review

- Problem statement should be approved by architect, client, development manager and other stakeholders (customer representatives, system engineers/analysts). Also should iterate with review coordinator.
- Review team selected (review team << project team):
  - Review leader - technical person trained in SARB and facilitation and responsible for writing review report
  - Angel - manager not associated with project, interface between review team and project management
  - Reviewers - several internal tech experts or outside consultants with relevant expertise, including: design, technology, management, domain knowledge, wild card reviewer

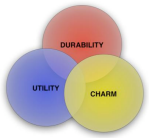


Review process

- Pre-review meeting a week before review
- Review meeting (2-3 days):
  - Begins with problem statement - what success means
  - Overview of proposed solution
  - Presentation on "ilities" reliability, maintainability, …
  - Reviewers ask questions and record issues and strengths on snow cards "out in the open" "reconfigurable"
  - Review team has caucus for several hours
    - Snow cards (50-200) arranged by functional categories (requirements, management, interfaces, …) and severity
    - Process provides overall assessment of chances for success and key messages

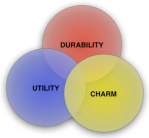


Snowcard example



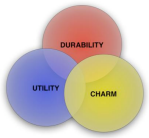
Review process (cont'd)

- Readout:
  - Encourage project team to invite clients and stakeholders
  - Provides overall assessment, strengths and issues
  - Elicits feedback for Project team
- Followup:
  - Within 2 days snow cards are numbered, recorded and sent to project team
  - If there are immediate issues angel and leader compose letter to client
  - Within 2 weeks detailed writeup of key issues and strengths
  - Within 2 months review leader provides report to SARB, assessment, key issues and review critique
- SARB creates annual report of key issues and trends



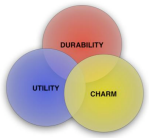
SARB Heuristics

- Requires cultural change:
  - Cost to project, reviewers
  - Project team safety
  - Ubiquity
- Key steps:
  - Combine executive and grass roots support
  - Establish and maintain self reinforcing support for experts
  - Recognize contributors
  - Maintain safety for team and reviewers
  - Start small



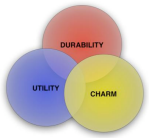
On team safety - key element

- Protect individuals from blame
- Keep no secrets from project team
- Treat all findings as confidential:
  - Specific review findings only reported to team and board
  - Written report is property of project team
  - Only general/anonymous findings are part of annual report
- Review team are partners and colleagues do not have role as policeman



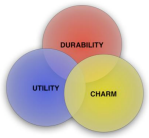
Arch Review Payoff

- Average review pays back 12 times its cost
- Reduced development effort and interval - find defects early
- Higher product quality
- Lower product cost
- Faster less costly product evolution (planned)
- Company wide learning - annual report
- Yes, projects were canceled after reviews and the attitude of the project team was often surprising



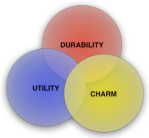
More General Review Heuristics

- When do you review -- early and often, note arch review is very early
- Roles in the review:
  - Author - responsible for updating entity afterwards (e.g., actual architect, developer)
  - Moderator - enforces roles and responsibilities -- manages meeting and is "neutral"
  - Scribe - accurately documents review points (active role, sometimes shared role with monitor), distributes drafts and with moderator and author publishes final review report
  - Reviewers - find issues, try not to solve at that point (can volunteer to help) constructive
- ALL IN THE TONE



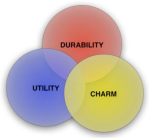
Active Reviews

- Parnas suggestion to get folks engaged:
  - Ordinary assumption, smaller number of comments during design review indicates higher quality work - NOT!
  - Parnas suggests it indicates a superficial review, besides effective review is not reading a document end to end but using it to
    - Evaluate correctness of document
    - Usefulness of it for future tasks
  - Give reviewers questions on the Design Document so they have to use information - Active Review
  - And a rigorous active design review process should include questionnaires written by all groups needing information, not just authors



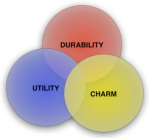
Domain Specific Software Architecture

- "A DSSA not only provides a framework for reuseable software components to fit into but captures the design rationale and provides for a degree of adaptability."
- Map user needs into requirements that given implementation constraints define a DSSA
- Stresses separation of user needs from system requirements and implementation constraints
- Spiral process model to develop it (although not much mention of risk analysis).



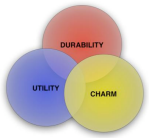
Five Steps to DSSA

- Define scope of the domain
- Define/Refine Domain specific concepts and requirements
- Define/Refine Domain-Specific Design and Implementation constraints
- ----------first discover, then do
- Develop domain models and architecture
- Produce gather reuseable work products



What's a Model in DSSA

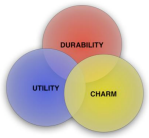
- Distinct, identifiable interface and design element, module, object that could be part of a large model (they are composable)
- Scalable unit of engineering technology with known structure and performance
- Specification (solution) for a class of problems



Ilities of the DSSA

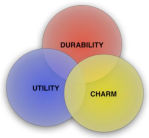
- Understandability
- Usability
- Complexity
- Adaptability
- Configurability
- Extensability
- scalability

- Composability - ability of components to be combined with other components -- think pipes and filters
- Compatibility with standards, technologies, …
- Predictability
- Quality, of course
- Salability



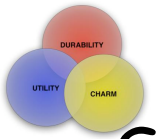
Step 1 - Scope of Domain Analysis

- What is the domain of interest and what is/are the problem(s) being solved including reuse goals
- Outputs:
  - Box and line diagram on inputs and outputs to DSSA and relationships between major functional units
  - List of contacts to serve as future reference and for validation
  - List of related projects with doc and code
  - List of needs to be met (problems to be solved) by applications in this domain



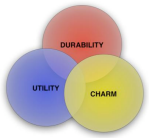
Step 2 - Define Domain Specific Concepts & Requirements

- Goals are to compile a domain specific dictionary and thesaurus of domain specific terminology (remember Universe of Discourse) and identifying commonalities and isolating difference between applications in the domain
- Outputs include:
  - Data dictionary with thesaurus
  - Type/inheritance hierarchy
  - Generic high level block diagram/architecture
  - Data flow and control flow diagrams
  - Rationale and relationships between elements in the domain



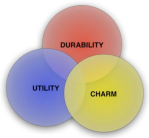
Step 3- Define and Refine Domain Specific Design and Implementation Constraints

- List and describe both stable and variable requirements. Stable are the what and variable are the how.
  - What it does, versus how (often fast, big, accurate, implemented, delivered, presented, operated)
- Outputs of this stage:
  - List of hardware constraints
  - List of software constraints
  - List of performance constraints
  - List of implementation constraints
  - Map of these constraints to the functional elements of the domain



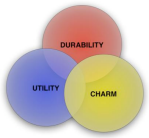
Step 4 - Develop Domain Architectures/Models

- Goal of this stage is to generate generic architectures and specify the syntax and semantics of the modules and components that form them
- Outputs:
  - DSSA
  - Domain specific models and analysis results
  - Mappings between the entities described in step 2



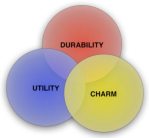
Step 5 - Produce or Gather Reusable Components

- Identify COTS and in-house components, specify what needs to be built
- Outputs:
  - Reusable components, test cases and documentation
  - Mappings of components to requirements, constraints and architecture.



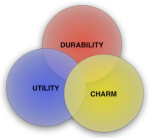
OneTESS

- OneTESS, Tactical Engagement Simulation Systems (laser tag like)
- Key components of Real Time Engagement Adjudication:
  - Shooter
  - Target
  - Adjudication (hit? <things between shooter and target> and, if so, how bad)
- Several architectures proposed -> early analysis through moderate fidelity abstract models

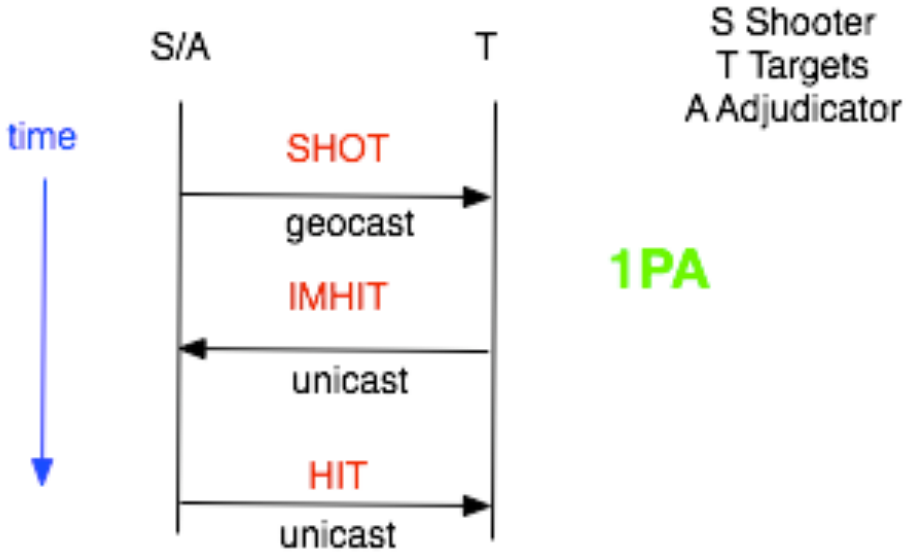


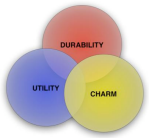
OneTESS modeling

- Abstract essential elements, "just enough detail"
- Component are Shooter, Target, Adjudicator and Casuality Assessment
- Architectures differ in where A resides and what info is transmitted to whom:
  - First Person Adjudication
  - Second Person Adjudication
  - Third Person Adjudication
  - Use of Geocast versus point-to-point
- Verify with high fidelity, though more expensive Qualnet model

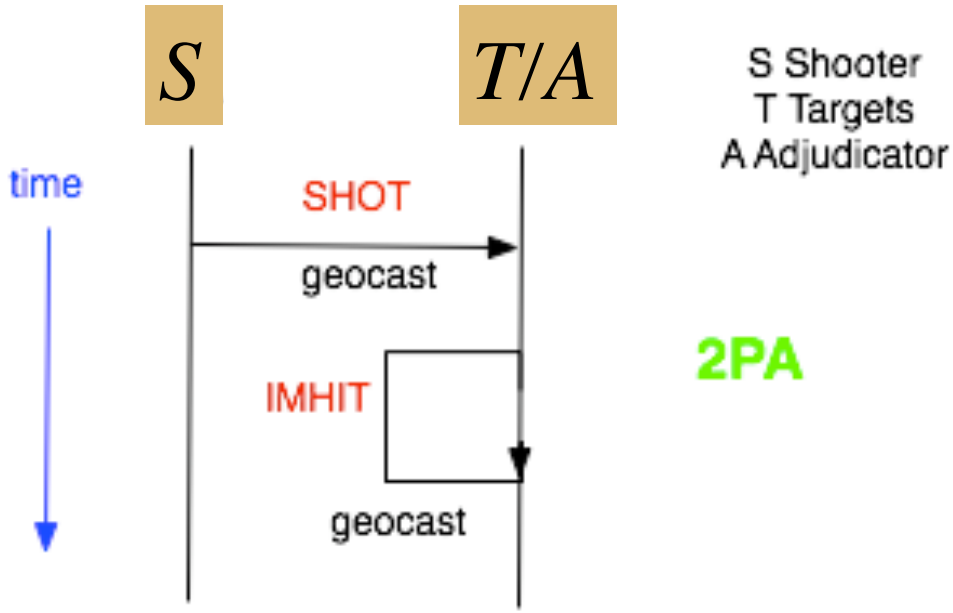


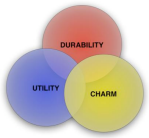
Abstract Model 1



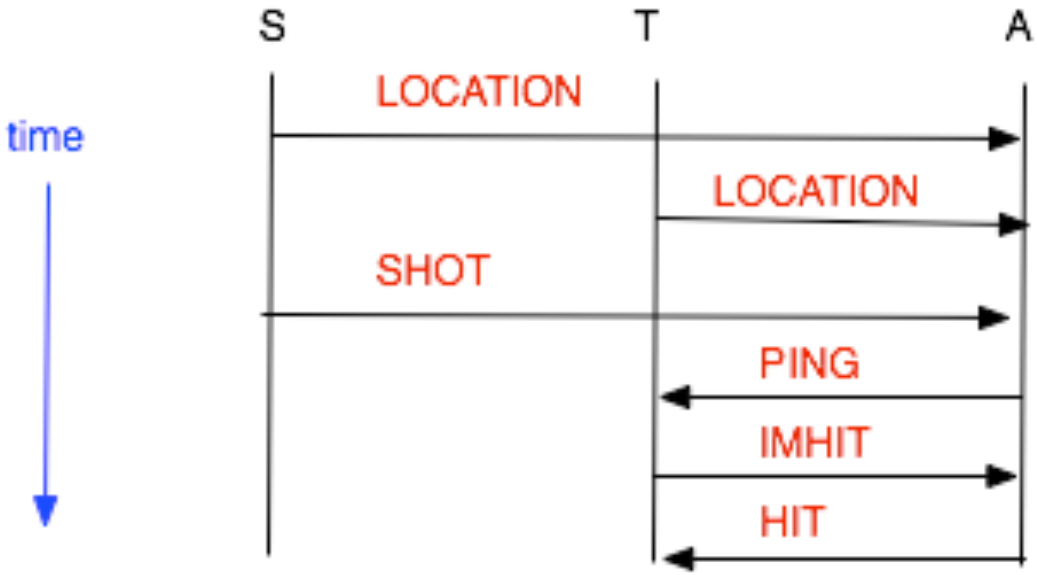


Abstract Model 2





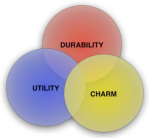
Abstract Model 3



S Shooter
T Targets
A Adjudicator

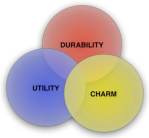
3PA

(all unicast)



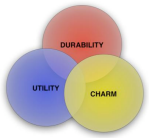
The Simulator

- Merely counts transmissions and receptions, no modeling of collisions, queuing, …
- Assumed smallish message
  - Size of packets would affect transmission
- Measures
  - Percentage of shots adjudicated correctly
  - Total packet transmissions/sec average and maximum (bursty)
  - Total packet transmission/sec/player unit
  - Total packet reception/sec/player unit
  - Power is a key factor: packet transmissions will correlate with battery depletion



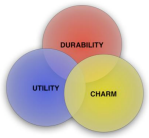
Scenarios

- Marketplace:
  - PARAMETERS: # of shooters, # of noncombatants, motion pattern
  - CHALLENGES: high player density, chaotic movement
- Convoy:
  - PARAMETERS: # of convoy troops, # of attackers, # of noncombatants
  - CHALLENGES: player density, indirect weapons in dense scenarion, player motion around buildings (shot & radio)
- RadioHeck: small units complex, radio limiting environment
- Chase: through large wilderness



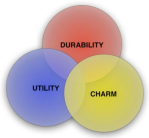
Results - Marketplace

	Reliability	Tmax	Tave	N1	N2
1PA	100%	131	72.7	2.40	2.38
2PA	100%	304	126.6	2.13	2.10
3PA	100%	594	227.0	192.5	1.52



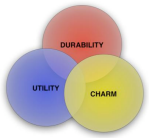
Chase

	Reliability	Tmax	Tave	N1	N2
1PA	100%	831.7	550.6	16.6	16.3
2PA	99.97%	3205.0	1516.2	67.2	65.4
3PA	43.1%	4060.6	760.5	263.3	81.2



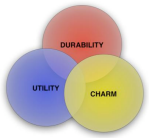
Conclusions

- As player density increase 1PA is better in marketplace
- Convoy 3PA had slightly higher reliability, until density increases and 1PA and 2PA get better
- In general 3PA will exhibit hotspot effect
- 1PA uses less bandwidth than 2PA
- 1PA results validated with Qualnet (more comparison necessary
- Situation dependent suggests adopting hybrid arch



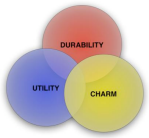
Information Architecture
Morville and Rosenfeld(2007)

- What it is:
  - Information - duh!
  - Structuring, organizing and labeling
  - Finding and managing
  - Art and science
- But not quite:
  - Graphic design
  - Software development
  - Usability engineering



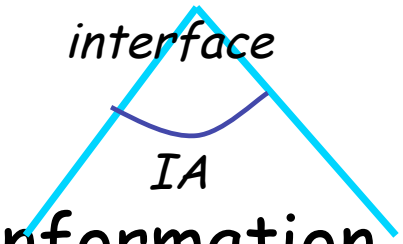
Why IA Matters

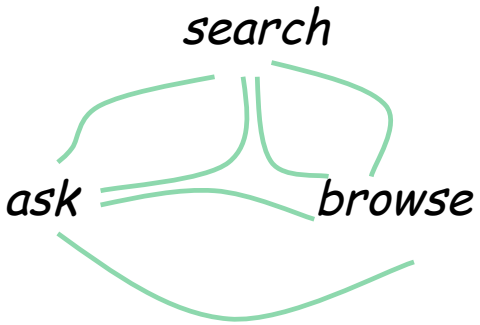
- Cost of finding information
- Loss of not finding information
- Value of education
- Cost of construction, maintenance and training
- Value of brand - important at all levels

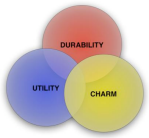


IA Concepts

- Complex systems
  - Context, content and users
- Invisible work
- Knowledge networks - aka information ecologies
- Information seeking

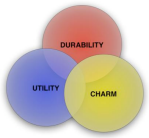






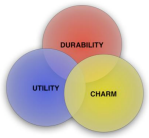
IA Systems

- Search systems
- Navigation systems
- Semantic networks

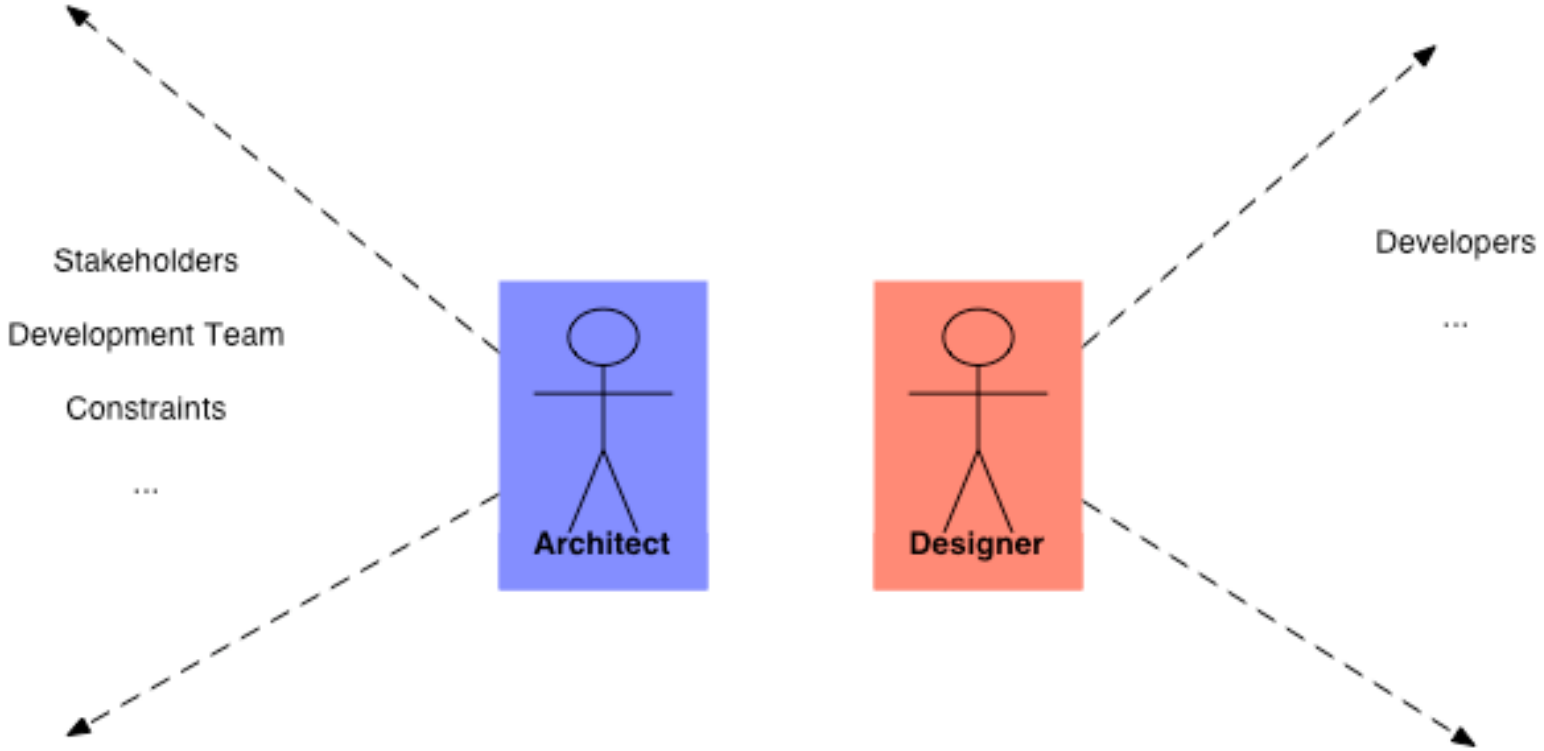


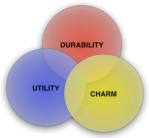
IA Deliverables

- Wireframes
- Controlled vocabularies
- Blueprints (site map)
- Metadata scheme



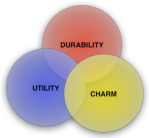
Architecture and Design





Review

- First 6 classes are architecture, next 4 Design and Domain Driven Design, last 2 refactoring + special topics
- Architecture:
  - Definition, brief review of UML
  - NASA arch paper
  - 4+1 architecture description
  - Case studies
  - Architectural styles and examples
  - Architectural connectors and types
  - ADL
  - Sha simplicity and complexity
  - UI architecture
  - Ilities: performance, concurrency, state, transactions (ACID) …
  - Arch reviews and discoveries
  - Agents
  - Domain Specific Architectures
  - Service Oriented Architectures



Other References

- Hall, R.J. "Combinatorial communications modeling or real-time tactical engagement adjudication architectures," MILCOM, 2005.
- Morville, P. & Rosenfeld, L. Information architecture for the world wide web, 3rd Edition, O'Reilly, 2006, ISBN: 978-0-596-52734-1.
- ATAM: http://www.sei.cmu.edu/pub/documents/00.reports/pdf/00tr004.pdf