Class 12 CS540

Gregg Vesonder
Stevens Institute of Technology

Roadmap- Class 12

- Clarifications from last class
- Log Book
- HCI
- Open Source
- Microsoft
- Games
- Reading this week:
- Reading next week - finish Brooks, chapter 15 - Epilogue, Parnas paper: Parnas, D.L. and Wurges, H. "Response to undesired events in software systems." In D. M. Hoffman and D.M. Weiss, Software fundamentals: Collected papers by D.L. Parnas, 2001.

Key Dates

- December 12th, the final

Clarifications

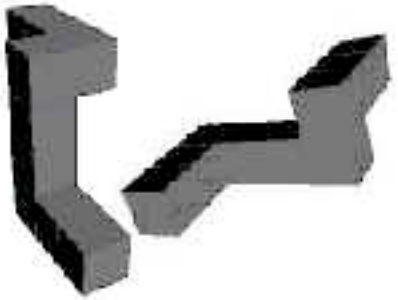
- Attention-hang in there

Logbook

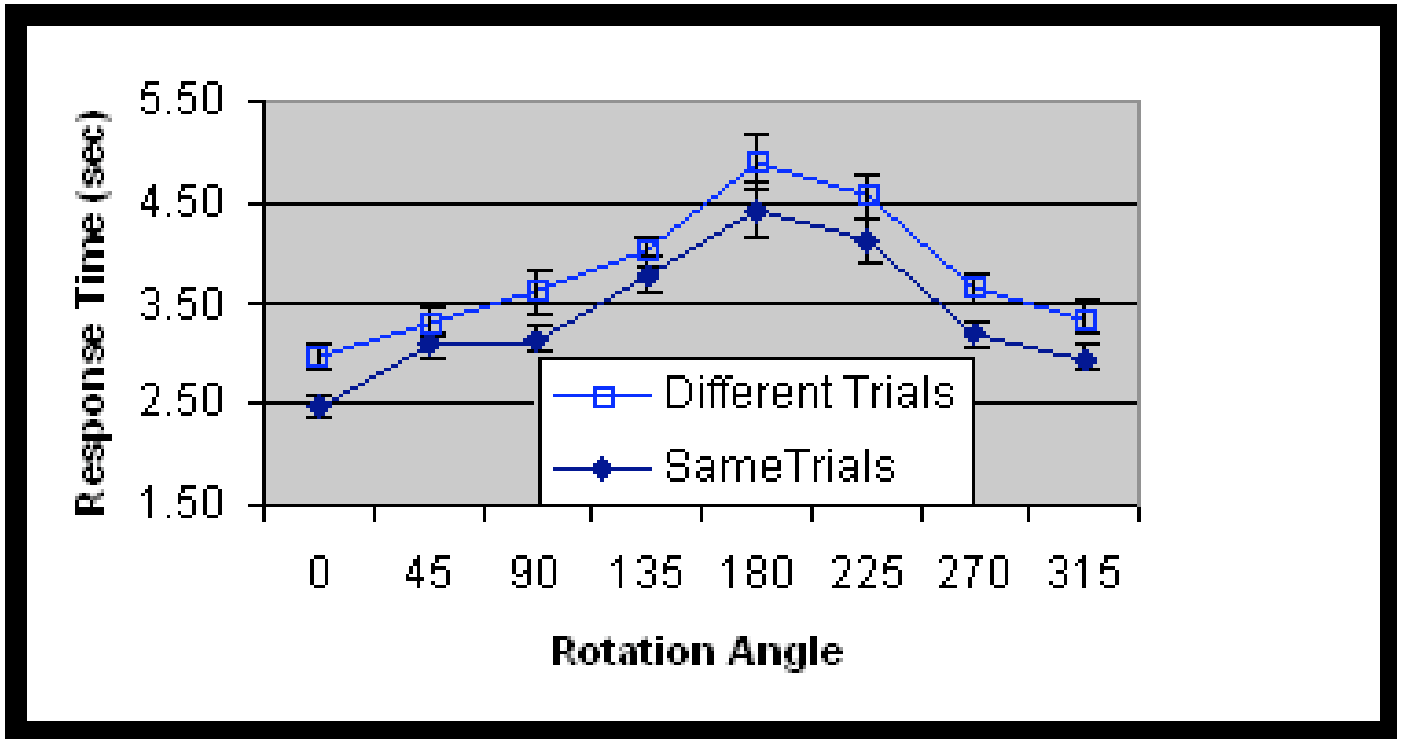
- The Dog and the Bone - on closure

Mental Rotation

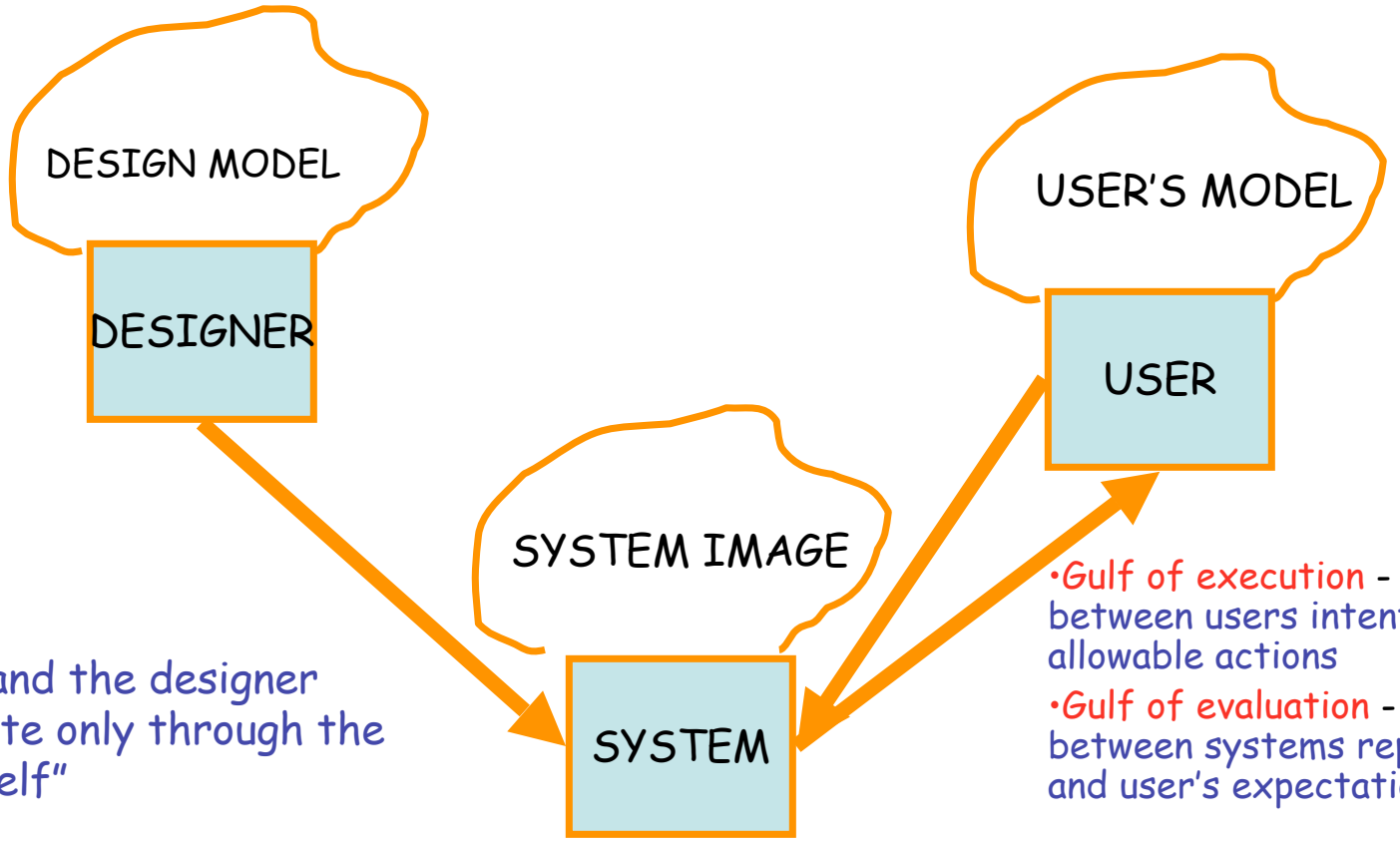
- Task is to decide whether figures are same or different, reaction time is measured, given a bunch of problems, from Shepard and Metzler(1971).



Typical Results



Knowledge in the World and in the Head



"the user and the designer communicate only through the system itself"

- Gulf of execution - mismatch between users intention and allowable actions
- Gulf of evaluation - mismatch between systems representation and user's expectations

The Design Challenge

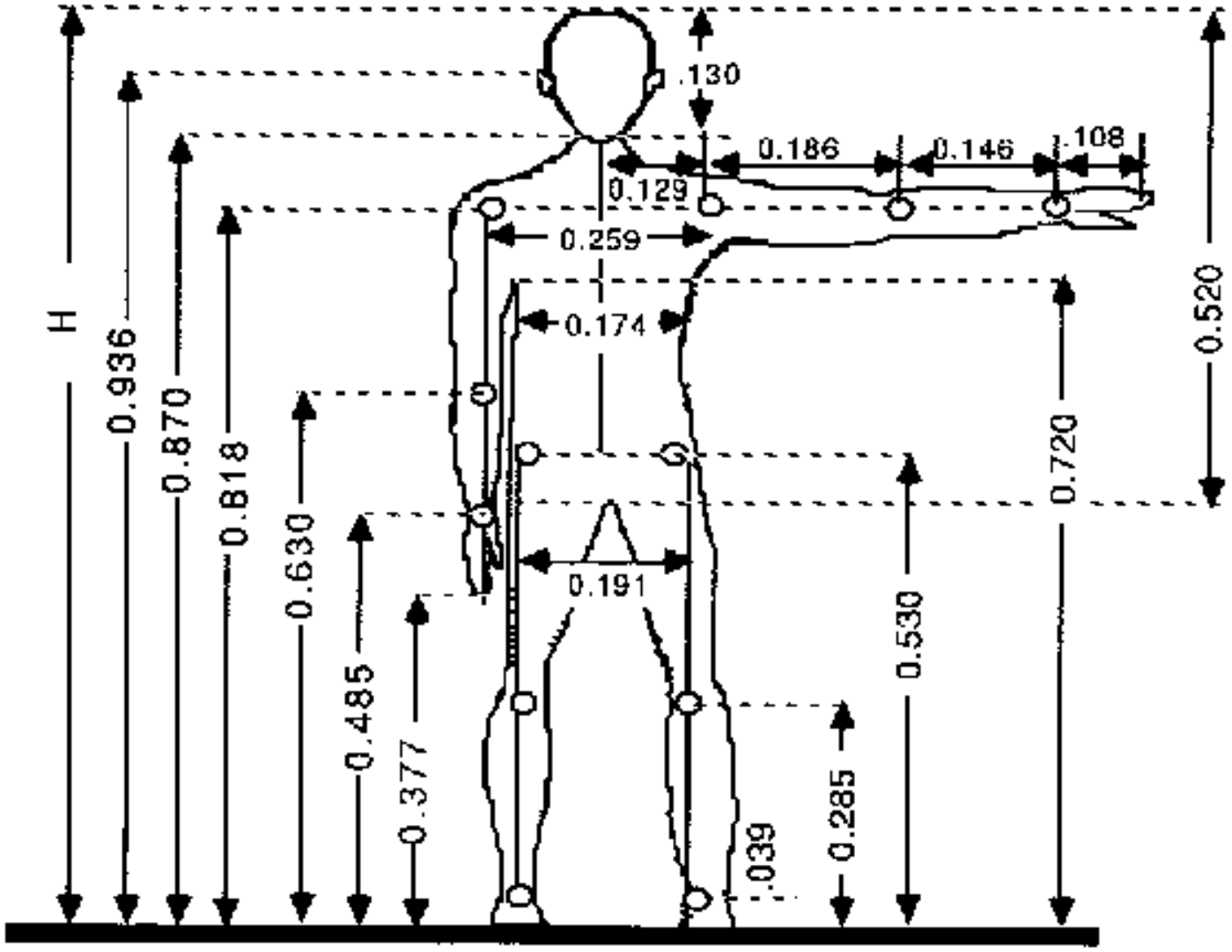
Interaction Styles

Style	Advantages	Disadvantages
Direct Manipulation	Visually presents task concepts, easy learning, easy retention, avoids errors, encourages exploration, high subjective satisfaction	Hard to develop, requires graphics display & pointing device
Menu Selection	Shortens learning, reduces keystrokes, structures decision making, can use dialog management tools, easy support of error handling	Danger of many menus, slows frequent users, consumes screen space, requires rapid display rate
Form Completion	Simple data entry, modest training, convenient assistance, use of form management tools	Consumes screen space
Command Language	Flexible, power users, user initiative, creation of macros (customizing)	Poor error handling, long training, memorization
Natural Language	Relieves burden of learning syntax	Clarification dialog, more keystrokes, contest is hard, unpredictable

Physical abilities and surroundings

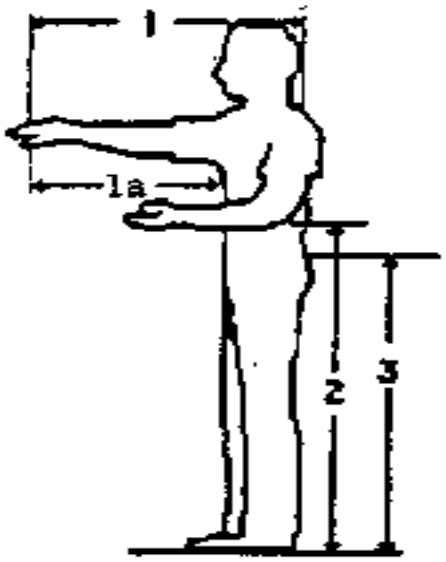
- Anthropometry - basic data about human dimensions (range of dimensions)
  - Not only static (size of hand) but also dynamic, reach distance while seated, speed of finger presses, strength of lifting, …
- Human Factors engineering of computer work stations
  - Work surface and display support height
  - Clearance under work surface for legs
  - Work surface width and depth
  - Adjustability of heights and angles for chairs and work surfaces
  - Posture adjustments, arm rests, foot rests, chair coasters
  - Lumination levels, glare, flicker, noise, air temperature, movement and humidity

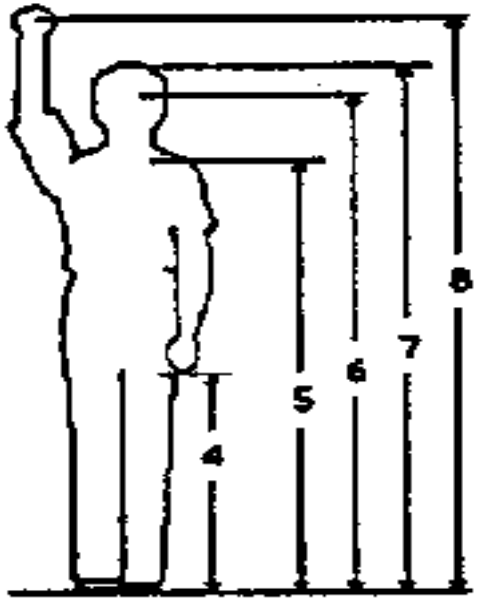
Relative Dimensions of Average Human Body



Standing Posture Data

Measurement (Inches)	Males			Females		
	95th	50th	5th	95th	50th	5th
1- Forward Grip Reach	33.3	30.9	28.5	30.1	28.0	25.8
1a- Bust to Grip Reach	21.9	20.9	19.8	18.3	18	17.5
2- Elbow Height	46.9	43.5	40.2	43.1	40.2	37.2
3- Hip Height	39.2	36.0	32.9	35.8	32.9	29.9
4- Fingertip Height	28.5	26.0	23.4	27.4	24.8	22.2
5- Shoulder Height	61.0	56.7	52.4	56.1	52.2	48.2
6- Eye Height	71.9	67.3	62.8	64.2	60.0	55.9
7- Stature	73.6	69.1	64.6	68.1	64.0	59.8
8- Vertical Grip Reach	87.0	81.9	76.8	80.5	75.8	71.1





HCI Systems Approach

- Proper functionality
  - Task analysis is central
  - Avoid excessive functionality (palm pilot vs Microsoft)
- Reliability, availability, data integrity
  - Building/destroying users trust in the system (data integrity is #1) crash on me, but don't lose my powerpoint
- Standardization integration, consistency and portability
- Schedules and budget

In Other Words

- Population
- Tasks
- Methods
- Techniques
- Evaluation
- Heuristics

Relativity of Design

- Each user and each task should have precise objectives:
  - Average time to learn
  - Speed of performance
  - Error rate by users
  - Retention over time (frequency of use is a factor)
  - Subjective satisfaction - surveys satisfaction scale
- Tradeoffs:
  - lengthy learning -> better performance
  - Rate of errors vs. speed of performance

Why spend effort on the UI? (redux)

- Increased efficiency
- Improved productivity
- Reduced errors
- Reduced training - strive for game like training
- Improved acceptance
- So evaluation determines how well we did

Usability Characteristics Evaluation

ISO 9241	Schneiderman	Nielsen
Efficiency Effectiveness Satisfaction	Speed of performance Time to learn Retention over time Rate of errors by users Subjective satisfaction	Efficiency Learnability Memorability Errors/Safety Satisfaction

Evaluation

- Not only in course of design process but as part of the system - throughout process, continually evaluate
- BEFORE: scenario based, manual based, story board based - evaluation as prototyping, experimentation
- AFTER: (have a prepared baseline of all tasks in previous environment) study and MEASURE how users are doing - in the beginning and at regular intervals
  - Casual interfaces - kiosks should go quickly: seconds to minutes
  - Week on task interfaces - telemarketing: minutes to hours
  - Month on task interfaces - help desk: days
- Observe the entire environment before and after for days
  - Include what is on their desk, tacked to wall and interactions
- **SATISFACTION AND JOY -** what follows are some heuristics to get there

In Other Words

- Population
- Tasks
- Methods
- Techniques
- Evaluation
- Heuristics

Heuristics on the User Interface

- If there's a substantial UI component have full time UI person involved from beginning plus artist/designer
  - UI person is not converted developer
- Avoid Natural Language interfaces
- Understand the environment and the users and the types of users
  - Auditory interface in high noise or long dialog text is not recommended
- Test it and observe - prototypes, user manuals, storyboards
- Do not stray too far from current interfaces, unless …revolution
- Do not be tempted by direct manipulation/"Matrix mode" unless ample time and software/hardware - but be inventive

More on UI

- **Do automate!**
- Do not ignore the users needs
- Do talk to the users
- Do understand your user population
- Do be predictable
- Do use common examples in documentation - Unix Man pages
- Do use designers/artists
- Do use paper, stickers, job aids, …
- Do consider Ergonomics
- Do consider special needs
- Joy is an important aspect
- One unsatisfied customer can hurt more than two satisfied customers can help!

B&Y Heuristics

- Simplicity
- Speak the user's language
- Be consistent
- Minimize what users must remember
- Design for flexibility and efficiency
- Design aesthetics and minimalist graphics
- Satisfaction
- Recognize the power of chunking
- Predictability
- Screen layout
- Naturalness
- Structure progressive levels of detail
- Navigation
- Safety

Information Visualization

- Shneiderman and Plaisant
  - Overview
  - Zoom
  - Filter
  - Details on demand
  - Relate - among items
  - History
  - Extract

Visualization Example

And So Much More

- Psychology of computer programming
- More on ethnography
- Psychology of online communities - Mail vs IM generations
- Computer supported cooperative work
- Psychology of embedded device interfaces
- Challenges of every new leap in technology
- …

Open Source Software Cathedral and the Bazaar

- Cathedral - commercial software world, bazaar - linux and open source
- Key names:
  - Richard Stallman - emacs, gnu, Free Software Foundation
  - Linus Torvalds, linux, open source process, open source license (General Public License (GPL), BSD, Perl's Artistic) License
  - Larry Wall - PERL

Flavor of Open Source

- Torvalds style - release early and often, delegate be very open -VERY developer centric!
- Axiom 1 - Every (?) good work of software begins by scratching a developer's itch (Axioms do not conform to numbers in the paper)
- 2- Good programmers know what to write, great ones know what to rewrite and reuse
  - Constructive laziness
- 3-Plan to throw one away, you will anyhow
  - You do not understand problem until after first time you implement
- 4- If you have the right attitude interesting problems will find you
- 5- When you lose interest in a program, your last duty is to hand it off to a competent successor

Flavor of OS - 2

- 6- treating your users as codevelopers is your least hassle way to rapid code improvement and effective debugging
  - If you have that luxury
  - Torvalds "too lazy to fail"
- 7- release early, often and listen to customers
  - Released a new linux kernel in the early days more than once a day!
- 8- given a large enough beta tester and code developer base, almost every problem will be characterized quickly and the fix obvious to someone
  - Linus' Law "Given enough eyeballs, all bugs are shallow"
  - Delphi effect
  - Debugging is parallelizable
  - Brooks: more users find more bugs
  - Non source aware users do not provide great bug reports

Flavor of OS 3

- Team: Usually 1-3 core developers, beta testers and contributors in the 100s
- 9- If you treat your beta testers as if they're your most valuable resource, they become it
- 10- the next best thing to having good ideas is recognizing good ideas from users. Sometimes the latter is better
- 11-often the most striking and innovative solutions come from realizing your concept of the problem was wrong
- 12 - Perfection (in design) Is achieved not when there is nothing to add, but rather when there is nothing to take away, Antoine de Saint-Exupery
  - Debugging is not only parallelizable, so is development and exploration of the design space!

Flavor of OS 4

- 13 - any tool should be useful in the expected way but a truly great tool lends itself to uses you never expected
- 14 -International flavor of participants is a plus in globalization (extract)
- 15-To solve an interesting problem start by finding a problem that is interesting to you

Preconditions for the Bazaar Style

- Hard to originate code in this style, test, debug, improve yes
  - You need to have something running - attractor
  - It has to run and convince others that it can evolve into something neat in reasonable time
- Leader/coordinator of Open Source project does not need to be a great designer but needs to recognize good ideas from other folks:
  - Robust and simple rather than cute and complicated
  - Community's internal market in reputation exerts subtle pressure in self-selecting competent leaders
  - A bazaar leader must have good people and communication skills

Social Context for Open Source

- Evolution of software in the presence of a large, active community of users
- Programmer time is not fungible - small number of codevelopers obeys Brooks communications links
- Programmers cannot be territorial about code, encourage others to look for bugs and improvement XXP!
- "while coding remains an essentially solitary activity, the really great hacks come from harnessing the material and brain power of entire communities"
- Internet helped
- Development of leadership style and set of cooperative customs
- Utility function of linux hackers is maximizing in their own eyes satisfaction and reputation among peers and users
- Boring is essential - software and documentation
- Open source is fun - joy as an asset
- Not so easy as to be boring, not too hard to achieve!

Homesteading the Noosphere

- Open Source culture: zealotry varies, hostility to commercial software varies
- Open source must protect an unconditional right of any party to modify (and redistribute modified versions of) open source software
- Taboos of Open Source:
  - Strong social pressure against forking projects
  - Distributing changes to a project without cooperation of moderators is frowned upon
  - Removing a person's name from the project history is not done without the person's explicit consent

Ownership in Open Source

- Owner of the software project is the person who has the exclusive right to distribute modified versions
- How to own:
  - Start the project - homesteading
  - Have ownership handed to you - deed transfer
  - Observe that a project needs work and owner has lost interest- try to find owner to get to have ownership handed to you - "adverse possession, moves on and improves"

Hacker Culture = Gift Culture

- Excludes crackers and warez d00dz - different culture
- Not what you control but what you give away - reputation among peers
- Along with sheer joy of making something work
- Craftsmanship model as a corollary - still linked to reputation
- In hacker culture status is based on critical judgment of peers

On Reputation

- Ego is despised, yet system runs on it
- One's work is one's statement, no one attacks one's technical competence, emacs bugs not Stallman's bugs
- More prestige in founding a project than working on an existing one
- More prestige for innovative rather than me too
- Being carried in a major distribution (Red Hat, SuSE) is prestigious
- Continued devotion to hard, boring work (debug, write doc) is more praiseworthy than fun and easy hacks.

Governance

- Project leader - codevelopers - contributors
- Apache has a voting committee
- PERL has rotating dictatorship among codevelopers

Open Source Resources

- www.opensource.org - jump off point, Halloween papers
- SourceForge.net - project pages
- Freshmeat.net - products and product announcements
- Slashdot.org - fun

Microsoft's Response

- The Halloween documents - response to Open Source, quotes Raymond
- Considers Open Source a threat especially in server market (H II - also in desktop)
- Commercial quality can be achieved!
- To target Open Source, you target a process not a product
- Open source is long term credible - you cannot FUD it!
- Implementation provides a high visibility showcase for OS
- Linux has done well in mission critical commercial environments
- Linux can win as long as services and protocols are commodities

More Microsoft

- Current count 9 Halloween papers - not all from Microsoft!
- From Halloween VIII - "First they ignore you, then they laugh at you, then they fight you, then you win" - Gandhi
- Halloween IX SCO + Microsoft??

Revenge of the Hackers

- Again by Eric Raymond
- Linux was a watershed from parts to a working car
- How did the Linux community beat Brooks' Law?
- Cathedral and Bazaar - provided language so they could improve their "process"
  - Hackers loved it around the world
- Netscape open sourcing their browser was "the shot heard round the world"
  - Failure would discredit open source
  - Mozilla

RH - 2

- Origin of "Open Source" and Open Source Initiative
- Needed marketing and the "Free Software" movement was a problem
  - Free = no price & liberty
  - Association of hostility to IP rights, communism and radicalism was not mainstream IT
- Needed Positive Stereo types: 1) pragmatic tales, 2) high reliability, 3) lower cost (not free), and 4) better features

RH-3

- Top down not bottom up marketing to CEOs/CTOs/CIOs
- Raymond as spokesperson - "sound challengingly weird but convey an aura of honesty and simplicity"
- More than Linux:
  - Apache 50% of server market
  - Perl
  - Sendmail - dominant mail router

RH-4

- O'Reilly and his company helped
- Key win -- Oracle and Informix offer linux ports, started bandwagon
- Halloween Docs gave it credibility - Microsoft was concerned
- Titanic - rendered by a roomful of Linux boxes
- Beowulf - supercomputer on the cheap, Open Source on cutting edge
- Red Hat, SuSE and the rest, proprietary Unix losing share
- Watch out for BSD
- Desktop versus server

Open Source Landscape

- From Code Reading by D. Spinellis

Language	# of Projects	% of Projects
C	8,393	21.2
C++	7,632	19.2
Java	5,970	15.1
PHP	4,433	11.2
Perl	3,618	9.1
Python	1,765	4.5
Visual Basic	916	2.3
Unix Shell	835	2.1
Assembly	745	1.9
JavaScript	738	1.9

Microsoft Development - Lucovsky

- NT: 6 guys from DEC, 1 guy from Microsoft
- Design goals for NT family:
  - Portability - abstract away machine dependencies
  - Reliability - nothing should be able to crash the OS
  - Extensibility
  - Compatibility
  - Performance but all of the above are more important
- Design Workbook - written by engineers for engineers
  - Every functional interface was defined and reviewed - small teams essential
  - Spread review duties and everyone shares culture

Time to Big

- To scale a team you need to establish a culture
  - Common way of evaluating designs, tradeoffs
  - Common way of developing code
  - Common way to establish ownership of problems
- Goal setting as foundation for the culture - hard as it grows
- Every decision made in the context of design goals
- Everyone owns all the code, so whenever something is busted anyone has a right and duty to fix
  - Works in small groups (< 150!)
- Sloppiness is not tolerated
- Accept that mistakes will happen

NT Source Code Control System

- Internally developed, by a non-NT tools team - no branch capability
- Small hard drive could hold whole tree (6M LOC), 10-12 source projects
- Easy to stay in synch
- Onto Win 2000 needed branching, (29M LOC), 180 source projects - full source, 50 gig, up to date machine, 2 hours to sync

NT Build

- 4 hours sync period, could check in code the other 20 hours
- Build lab syncs during 4 hour period (in morning) and begins a complete build - 5 hours on 486/50
- Preliminary test is done then off to 4pm stress test on ~100 machines

Win 2000 Build

- No source tree changes w/o explicit permission
- Build lab approves ~100 changes each day and manually syncs and builds
  - A developer mistyping a build instruction can stop build which stops 5000 people
- Build 8 hours on 4 way PIII Xeon with 50 gig disk and 512K
- Build boot and baseline tested then 4pm stress testing on 1000 machines

Team Sizes

PRODUCT	Dev Team	Test Team
NT 3.1	200	140
NT 3.5	300	230
NT 3.51	450	325
NT 4.0	800	700
Win2000	1400	1700

Defect Rates Data

Product team size	Defects/yr/ dev	Time to fix/defect	Defect/day	Total fix time
NT 3.1, 200	2	20 min	1	20 min
NT 3.5, 300	2	25 min	1.6	41 min
NT 3.51, 450	2	30 min	2.5	1.2 hours
NT 4.0, 800	3	35 min	6.6	3.8 hours
Win2000, 1400	4	40 min	15.3	10.2 hours

Game Development

- Roles of Game Development:
  - Producer - person responsible for managing people and processes responsible for the game
  - Game Designer - overall vision of the game and maintaining it
  - Level Designer - implements game using content creation tools created by programmers and assets generated by artists
  - Programmer - tool builder
  - Game Graphic Artist - know current context but be very broad
  - Much more "creative" based, developer as tool builder, amenable to software process factoring in this large difference
  - May (should) become more common

Thought Problems

- You want to become part of the Open Source Movement - How should you begin?
- What steps would be needed to change your current software process to a process more similar to the Software Gaming Industry?

So Far

- Software Process Models, Software Project Planning (woosh!), Requirements, Estimation, Risk Analysis, Multics case study, Architecture Reviews, Questionnaire Design
- Software Quality Assurance, Configuration Management and Testing, Architecture and Design, Software Engineering skills: Problem Solving, meeting, stat, … (and finished Arch and Design) and OO
- Lightweight Methodologies, XP, CHI and Human Factors, Part 1
- This Time: CHI and Human Factors Part 2, Open Source, Microsoft, Gaming
- Next Time: Brooks, Reliability, Parnas, Formal Methods, Out sourcing

References

- Schneiderman, B. Designing the User Interface, (wait for 4th edition! - Spring 2004) Addison-Wesley
- Mental rotation: http://psychexps.olemiss.edu
- E.S. Raymond:
  - The cathedral and the bazaar
  - Homesteading the Noosphere
  - Halloween papers
- Mark Lucovsky, "Windows a software engineering odyssey" - build centric view
- Game Developers Association
- Body Chart: http://www.mech.utah.edu/ergo/educate/safety_modules/ctd-anthropometry/