Individual Differences

► Personality
- Develops over a person’s lifetime
- Manifested into thoughts, feelings, attitudes and behavior
- Affected by nature (biological heritage) and nurture (life experiences, friends)
- Situational factors might cause personality to affect behavior

► Ability: what a person is capable of doing
- Cognitive ability
- Physical ability
- Emotional intelligence (the ability to understand and manage one's own feelings and emotions and the feelings and emotions of others)
- Affected by nature and nurture (education, practice, exercises)

Personality Traits

► Extraversion
- Predisposition to experience positive emotional states and feel good about themselves and the world around them

► Neuroticism
- Tendency to experience negative emotional states, feel distressed, and generally view themselves and the world around them negatively

► Agreeableness
- How well a person gets along with other people

► Conscientiousness
- The extent to which an individual is careful, scrupulous, and persevering

► Openness to experiences
- The extent to which an individual is original, has broad interests, and is willing to take risks

Personality Traits Relevant to Organizations

- Locus of control
- Self-monitoring
- Self-esteem
- Type A and Type B personality
- Need for achievement
- Need for affiliation
- Need for power

Expertise Differences

► Dreyfus’ model of skill acquisition
- Studied airplane pilots, chess players, car drivers, adult learners of a second language
- The road to mastery of an activity is essentially
  - a lifelong learning process.
  - can take place at times when one is not consciously aware that one is learning, but
  - generally requires a focused effort at understanding new concepts and practicing new skills

► Five stages: novice, advanced beginner, competence, proficiency, expertise

► Best way to observe stages:
  - Unstructured problems
  - Number of potentially relevant facts enormous
  - Variety of solutions extensive
Stages

► Novices
  • Follow specific rules for specific circumstances
  • No modifiers, "context free"
  • Don’t feel responsible for other than following the rules

► Advanced beginners
  • New "situational" elements are identified
  • Rules begin to be applied to related conditions
  • Decisions still are mostly made by rule application
  • Does not experience personal responsibility

► Competence
  • Recognize the complexity of task and a larger set of cues
  • Select and concentrate on the most important cues
  • No longer aware of absolute rules; they are assumed
  • Experimentation with new behaviors

Proficiency

► Proficiency
  • Unconscious, fluid, and effortless performance begins to emerge
  • Approach to problem molded by perspective arising from multiple real world experiences
  • “Holistic similarity recognition”
  • Learner uses intuition to realize "what" is happening
  • Conscious decision-making and rules used to formulate plan

► Expertise
  • No decomposition of situation into discrete elements
  • Pattern recognition extends to plan as well as diagnosis
  • Don’t make decisions
  • Don’t solve problems
  • Do what works

Application of Dreyfus’ model

► Expert vs. novice programmers

► Novice programmers
  • lack an adequate mental model of the area [Kessler and Anderson, 1989]
  • are limited to a surface knowledge of subject, have fragile knowledge (knows but fails to use when necessary) and neglect strategies [Perkins and Martin, 1986]
  • use general problem solving strategies (i.e., copy a similar solution or work backwards from the goal to determine the solution) rather than strategies dependent on the particular problem
  • tend to approach programming through control structures
  • use a line-by-line, bottom up approach to problem solution [Anderson, 1985]

► Expert programmers
  • have many mental models and choose and mix them in an opportunistic way [Visser and Hoc, 1990]
  • have a deep knowledge of their subject which is hierarchical and many layered with explicit maps between layers
  • when given a task in a familiar area, work forward from the givens and develop sub-goals in a hierarchical manner, but given an unfamiliar problem, fall back on general problem solving techniques
  • have a better way of recognizing problems that require a similar solution [Davies, 1990]
  • tend to approach a program through its data structures or objects [Petre and Winder, 1988]
  • use algorithms rather than a specific syntax
  • have better syntactical and semantical knowledge and better tactical and strategic skills [Bateson, Alexander & Murphy, 1987]

Regardless of expertise

► Given a new, unfamiliar language, the syntax is not the problem, learning how to use and combine the statements to achieve the desired effect is difficult.

► Learning the concepts and techniques of a new language requires writing programs in that language.
  • Studying the syntax and semantics is not sufficient to understand and properly apply the new language.

► Problem solution by analogy is common at all levels; choosing the proper analogy may be difficult.

► At all levels, people progress to the next level by solving problems.
  • "Practice makes perfect" has solid psychological basis.

► Discussion: should tools for novices:
  • Have less or more functionality than those for experts?
  • Have different functionality than those for experts?

Helping novices

► What happens when things go wrong?
  • Solve it yourself
  • Get help

► How do one gets help?
  • Look up (inquire) things in problem-solution database
  • Peer support (discussion boards, blogs)
  • Manual, documentation, helpdesk (human or automated)

► Good help
  • Elicitation: Helper asks user what is wrong. Get enough information from user to determine problem and choose solution.
  • Explanation: Helper tells user how to fix the problem. Explain why the solution worked, and how to avoid such problems in the future
Mismatch between expert and novice mental models

► The helper is an expert; the user is a novice
► Novice may lack technical vocabulary to understand the elicitation questions
► Novice may lack background knowledge to understand explanation of solution
► Expert may be unable to empathize with novice

User Support

► Help
  - Problem-oriented and specific (usually used in panic mode)
► Documentation
  - System-oriented and general (but users don’t often read it)
► Never a replacement for bad design, but essential
► Simple system
  - User walks up and uses it (but not everybody can)
► Most systems with rich features require help

User Support Requirements

► Availability
  - Available any time the user is operating the system
► Accuracy & Completeness
  - Accurate (tricky with changing versions) and covers all aspects of application
► Consistency
  - Across different sections, between on-line and paper documentation, in terminology, content and style
► Robustness
  - Should be predictable and free of errors
► Flexibility
  - Appropriate for novices through experts, maybe by having expandable sections of details
► Unobtrusiveness
  - Shouldn’t distract from or interfere with normal work flow

Types of Doc/Help

1. Tutorial
   - For start-up
   - Gets user going
   - Convey conceptual model
   - Communicate essential items
   - Sometimes see on-line tour or demo
2. Quick reference/review
   - Reminder or short reference
   - Often for syntax
   - Can be recall aid for expert
   - Allows novice to see what’s available
   - Detailed command descriptions
   - Usually for experts
   - E.g., Unix on-line manual pages
4. Context-sensitive (task-specific) help
   - System provides help on current situation
   - E.g., ToolTips
5. Adaptive help
   - Tailor help level and style to the particular user
   - Usually requires a system to maintain a user model

Doc Organization

► Present concepts in logical sequence, increasing order of difficulty – for each concept:
  - Explain reason for concept
  - Describe concept in task-domain semantic terms
  - Show computer-related semantic concepts
  - Offer syntax
► Avoid forward references
► Make sections have roughly equal amounts of material
► Have plenty of examples, complete sample sessions
► Always have table of contents and searchable
► Keep reading level simple (5th grade best)
► Try to predict common states and problems

Helping when Users Make Errors

► What can we help on?
  - Avoiding and preventing
  - Identifying and understanding
  - Handling and recovering
► Three phases
  - Read-scan phase -- Perceptual errors
  - Think phase -- Cognitive errors
  - Respond phase -- Motor errors
► Perceptual errors: result from insufficient or poor perceptual cues
  - Display of objects that are visually similar
  - Invisible or poorly expressed states
  - Failure to capture user’s attention
  - Lack of perceivable feedback
### Helping when Users Make Errors

**Cognitive errors:** caused by taxing the memory and problem solving capabilities
- Tax recall memory
- Lack of or poor mnemonic aids
- Inconsistency
- Lack of context or status info
- Mental calculations and translations

**Motor errors:** Taxing the eye-hand coordination and motor skills
- Awkward motor movements
- Highly similar motor sequences
- Pressure for speed
- Require a high degree of hand-eye coordination
- Requiring special types of motor skills (type)

### Example Studies

1. **170 experienced UNIX users over 9 days**
   - Individual commands had error rates of 3-50%
   - Kraut et al, CHI '83

2. **300 security system users over 20 months**
   - 12,117 error messages
   - Most common 11 errors -> 65%
   - 2517 involved repeated errors (with no non-errors in between) within 10 minutes
   - → Bad error recovery/help
   - Mosteller & Ballas, Human Factors '89

### Slips

**Automatic (subconscious)** error that occurs without deliberation

1. **Capture error** - Continue frequently done activity instead of intended one (similar starts)
   - Confirm deletion of file instead of cancel
2. **Description error** - Intended action has much in common with others possible (usually when distracted, close proximity)
   - ctrl key instead of Mac key
3. **Data driven error** - Triggered by arrival of sensory info which intrudes into normal action
   - Call to give someone a number, dial that number instead
4. **Associative activation** - Internal thoughts and associations trigger action
   - Skype phone rings, pick up normal phone
5. **Loss of activation** - Forgetting goal in middle of sequence of actions
   - Launching a web browser to search for something on google, then forgetting what to search
6. **Mode errors** - Do action in one mode thinking you’re in another
   - Zip all files, but you’re in wrong directory

### Error Prevention Guidelines

- Provide visible cues for modes
- Use good coding techniques (color, style)
- Maximize recognition, minimize recall
- Design non-similar motor sequences or commands
- Minimize need for typing free text

### Slips

- **5. Loss of activation** - Forgetting goal in middle of sequence of actions
  - Launching a web browser to search for something on google, then forgetting what to search

- **6. Mode errors** - Do action in one mode thinking you’re in another
  - Zip all files, but you’re in wrong directory

### Error Recovery Guidelines

- Provide appropriate type of response
  - **Gag** - Prevent user from continuing
    - Erroneous login
  - **Warn** - Warn user an unusual situation is occurring
    - Bell or alert box
  - **Nothing** - Just don’t do anything (Careful, user must determine problem)
    - Mac: move file to bad place
  - **Self-correct** - Guess correct action & do it
    - Spell-check correction
  - **Dialog** - System opens dialog with user
    - Go into debugger on run-time crash
Error Recovery Guidelines
► Ask user what should've been done, then allow error action as legal one
  ❖ Command language naming error
► Provide undo function
► Provide cancel function from operations in progress
► Require confirmation for drastic, destructive commands
► Provide reasonableness checks on input data
  ❖ Did you really mean to order 5000?
► Return cursor to error field, allow fix
► Provide some intelligence
  ❖ Guess what they wanted to do
► Provide quick access to context-sensitive help

Error Message Guidelines
► Explicit
  ❖ "You seem to want to attach a file to this message. Would you like to attach one now?"
► Human-readable language
► Polite phrasing that doesn't blame users
  ❖ Not "illegal command"
► Precise descriptions of exact problems
  ❖ Not "Syntax error"
► Constructive advice on how to fix the problem
  ❖ Instead of "out of stock", say "available 4/27/11".
► Visible and highly noticeable
► Preserve as much as the user's work as possible
► Reduce the work of correcting the error