


Novice and Expert Programmers



Gild Project

University of Victoria
Jeff Michaud

Overview of Presentation

- General Definitions
- Psychological Studies
 - Novices
 - Expert
 - Common
- Novice-Expert Continuum
- Discussion

General Definitions

- Expert: someone who is more than just proficient in an area
- Novice: someone who is just entering an area

In the beginning

- 'Psychological' studies of programmers started in the 1970s
- Results are unfortunately not reliable
 - Weak methodologies
 - Complex programming practices
 - No psychologists ...

Enter the Psychologists

- Proper research methodologies used
- Studies start to show consistent results:
 - novices could understand the syntax and semantics of individual statements
 - but could not combine the statements into programs to solve the problem

Results of the Studies: Novices

- lack an adequate mental model of the area [Kessler and Anderson, 1989]
- are limited to a surface knowledge of subject, have fragile knowledge (something the student 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]

Results of the studies: Experts

- 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
- apply everything they know
- 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 [Chi, *et al.*, 1981; Davies, 1990]
- tend to approach a program through its data structures or objects [Petre and Winder, 1989]
- use algorithms rather than a specific syntax (they abstract from a particular language to the general concept)
- are faster and more accurate [Wieden-beck, 1986; Allwood, 1986]
- 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. The old saying that practice makes perfect has solid psychological basis.

The Continuum

- **Dreyfus and Dreyfus [Dreyfus, 1985] described the continuum from novice to expert with five stages**
 - **Novice**
 - Learns objective facts and features and rules for determining actions based upon these facts and features. (Everything they do is context free.)
 - **Advanced Beginner**
 - Starts to recognize and handle situations not covered by given facts, features and rules (context sensitive) without quite understanding what he/she is doing.
 - **Competence**
 - After considering the whole situation, consciously chooses an organized plan for achieving the goal.
 - **Proficiency**
 - No longer has to consciously reason through all the steps to determine a plan.
 - **Expert**
 - "An expert generally knows what to do based upon mature and practiced understanding."

Discussion

- Many tools for intro programmers are marketed as a 'scaled down' version of experts' tools.
- Should tools for novices:
 - Have less or more functionality than those for experts?
 - Have different functionality than those for experts.

Problem Solving and Pedagogy

- Problem Solving
 - Understand the problem
 - Determine how to solve the problem
 - Translate the solution into a computer program
 - Test and Debug
- Pedagogy
 - Learn the syntax and semantics of one feature at a time
 - Learn to combine this language feature with known design skills to develop programs to solve the problem
 - Develop general problem solving skills
