

Class 27: The Evolution of Some Key Language Features

Held: Friday, 8 April 2011

Summary: Today we consider a variety of issues pertaining to Ryder et al.'s paper. Some issues are more general (e.g., how do you read a paper), while others bear on the specific relationships the group identifies between software engineering and programming languages.

Related Pages:

- EBoard.

Notes:

- Are there questions on the mid-semester exam? It is now due Monday evening. Your hash tables should copy the strings.
- Reading 1 for Monday: Dijkstra's "Goto Considered Harmful".
- Reading 2 for Monday: Bratley, P. (1967). Algorithm 306: Permutation with Repetitions [G6]. *Communications of the ACM* 10(7), July 1967, pp. 450-451.
- Both are in the ACM DL.
- EC for CS table today [academic].
- EC for Sita Sings the Blues tonight [academic].

Overview:

- The Story of Mel, a Real Programmer.
- Reading Ryder et al.
- Major Characteristics.
- More History.

The Story of Mel, a Real Programmer

- What were the key points?
- Why might this suggest that programming is an art?
- What does this say about language design?

Reading Ryder et al.

- What is the thesis?
- What evidence is used to support that thesis?
- Is the evidence sufficient?
- How was that evidence gathered?
- What biases are there in the way the thesis is stated or argued?

- Are there any parts of the paper that undermine the thesis?

Major Characteristics

- What are some major characteristics of programming languages that Ryder et al. describe?
- How long did the evolution of those characteristics from theory into regular practice take?
- Were there particular characteristics you were surprised to see?
- Suppose we accept Ryder et al.'s implicit assertion that the primary driving principle in early language design was better software engineering. Do we see a same driving force in modern language design, or are there other important driving forces?

More History

- If we have time, we'll return to issues from the previous class's outline.