The Cornell Program Synthesizer: A Syntax-Directed Programming Environment

Tim Teitelbaum and Thomas Reps
CACM 1981

Summary by Prof. Thomas LaToza
SWE 795, Spring 2017
Software Engineering Environments
Syntax-Directed Programming Environments

- Offer a unified programming environment that enables developers to work with code at higher level of abstraction

- Enable step-wise refinement, where developers start at high-level and work downwards

- Spare developers from mundane and frustrating details of programming syntax
Key Idea: Program Templates

• Rather than work with characters, developers use commands to insert program templates
• Inserts all necessary keywords
• Leaves *placeholders* that can be filled by text of the correct type
  • e.g., statement, condition
Working with program templates

IF ( k > 0 )
THEN PUT SKIP LIST (\texttt{List-of-expressions} );
ELSE PUT SKIP LIST ( ‘not positive’ );

- All edits occur through templates
  - Cursor only moves through template, phrase, placeholder
- Errors can occur only in phrases, not templates
  - Error detection can give more precise immediate feedback
- Structural modification commands can edit, delete, move, copy program units
- Can fold (hide) program elements to summarize less relevant sections
Questions for discussion

• Overall reaction to the paper

• Would you use such a system for your everyday programming?
  • Why or why not?

• What are the pros and cons of structured editors compared to modern IDEs?