Language-Oriented Programming: The Next Programming Paradigm

By Sergey Dmitriev, JetBrains

Presented by Danielle Watson
Introduction

• The current approach to software programming has built-in assumptions that hold it back
• The problem with object-oriented programming is that it requires the programmer to think like a computer, rather than vice versa
Overview

- Problems with mainstream programming
- Introduction to language-oriented programming
To be able to program freely, we must first reduce how dependent we are on languages.

There are two methods:

- General-purpose languages
- Domain-specific languages
Process for Mainstream Programming

- Think
- Choose
- Program
Process for LOP

- Think
- Choose
- Create
- Program

Figure 2: Language-oriented programming with domain-specific languages.
What exactly is wrong with mainstream programming?

- Time delay to implement ideas
- Understanding and maintaining existing code
- Domain learning curve
Details of LOP

• To solve a problem, we first think of a model, and then translate it into code
• This qualifies as a solution because the mental model can be explained to and understood by others
Text-Based Programming

- Everyone thinks of programs and programming languages in terms of text
- Compilers work by processing the text as an abstract syntax tree
- If LOP is to become easy, then the representation of the code has to be separated from the code itself
Ceci n’est pas une pipe.
Introduction to Meta Programming System

• So, how can we make languages easy to create?
• We create domain-specific languages for the domain of ‘creating new languages’
• When practising LOP, there would be two levels of programming:
  • Meta level: define the language
  • Program level: write the program
Structure Language

- We need to be able to define the structure of the language to be created using a language
- The structure of a language is described with the types (features and content) of the language
- Each type is partially defined by its relationships to other types
Editor Language

- The editor is part of the language
- It would use cells to edit different parts

Figure 4: Definition of an editor for the “Method” concept
Transformation Language

- For code created with this to be executable, it has to go through compilation
- This has two phases: translation from the target language to the final result, and transformation from the initial language to the target language
- To make this transformation as easy as possible, we need a transformation language
Templates and Patterns

• Templates allow you to add macros!
• Patterns work as regular expressions for models
Platforms, Frameworks and Libraries

- Domain specific languages do exist as class libraries
- But since they are libraries, they aren’t supported by the environment
Platform Languages

- Base Language
- Collection Language
- User Interface Language
Conclusion
Some additional comments

- Designing a DSL is not easy
- Complex libraries are not necessarily complex because they are part of OOP
- DSL don’t have the same level of support that standard languages do (debugging, etc.)