

CSSE 490 Model-Based Software Engineering: Domain Specific Language Introduction



Shawn Bohner

Office: Moench Room F212

Phone: (812) 877-8685

Email: bohner@rose-hulman.edu




ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

Plan for the Day

- Introduction to Domain Specific Language (DSL)
- Watch Martin Fowler clip on DSLs
- Plus/Delta for course
- DSL Reading Discussion (if time)





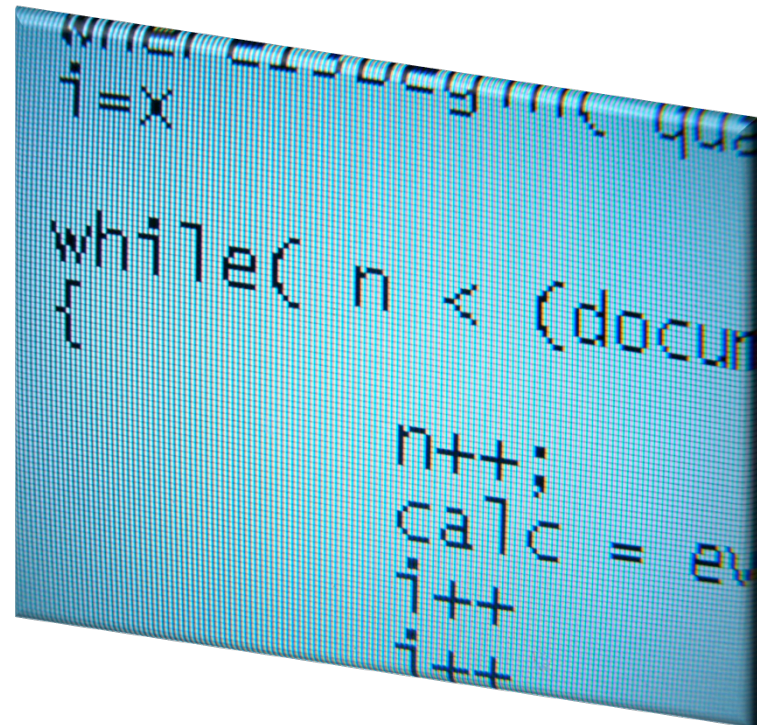
I bet you have already used some domain specific languages and you didn't even realize it. Can you think of some languages you have used that are domain specific?

- Think for 15 seconds...
- Let's talk... 😊



General Purpose Language (GPL)

- Turing complete
- Well understood and widely used
- Applicable to a wide range of problems
- Examples
 - C, C++
 - Java, C#
 - Fortran, Perl, Haskell
 - Smalltalk, Python...





GPL Strengths and Limitations

Strengths

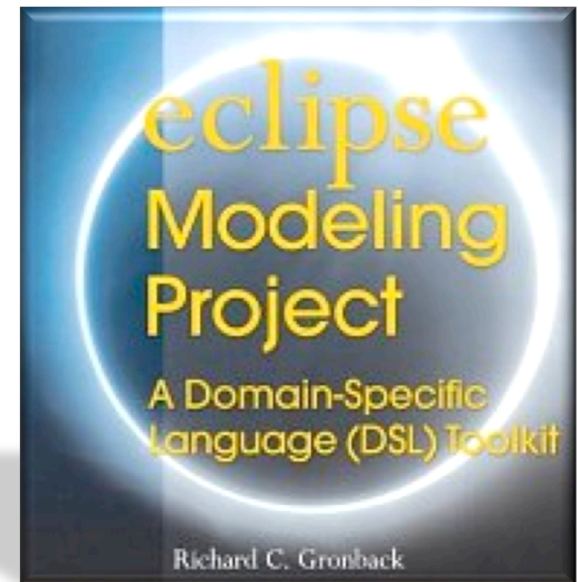
- Excellent IDE support
- Lots of experienced developers
- Growth via libraries

Limitations

- Lack of abstractions
- Language growth is often not possible
- Can be very verbose

Domain Specific Language (DSL)

- Small language targeted at application domain
- Expressive in its domain
- Often declarative
- Examples
 - Textual: LaTeX, Graphical: Pic
 - Declarative: HTML, Imperative: VHDL
 - Document: SVG, Executable: Ant/Maven
 - Compiled: yacc, Interpreted: SQL
 - Embedded: LINQ of C#
 - Diesel, Groovy, ...





Some more Specific DSL Definitions

A domain-specific language is a:

- language designed to provide a **notation tailored** toward an application domain, and is based only on the relevant concepts and features of that domain
- programming language or executable specification language that offers, through appropriate **notations** and **abstractions**, **expressive power** focused on, and usually restricted to, a particular problem domain
- small, usually declarative, language **expressive** over the distinguishing characteristics of a set of programs in a particular problem domain



DSL Strengths and Limitations

Strengths

- Tailored to domain
- Expressive and easy to use
- Little languages, little maintenance, increased productivity
- Communication with domain experts

Limitations

- Extra investment
 - Language engineering
 - Learning new language
- Weak IDE support
 - Editor, Debugger, Refactoring
- Migration can be hard
- Evolving into generality



More DSL Definitions (continued)

- Domain-specific description (DSD)
 - A **program** (specification, description, query, process, task, . . .) written in a DSL
- Domain-specific processor (DSP)
 - A software tool for **compiling, interpreting** or **analyzing** domain-specific descriptions

As an **analogy** to general purpose languages
DSL ~ Java
DSD ~ Java source files
DSP ~ Java Compiler + Virtual Machine

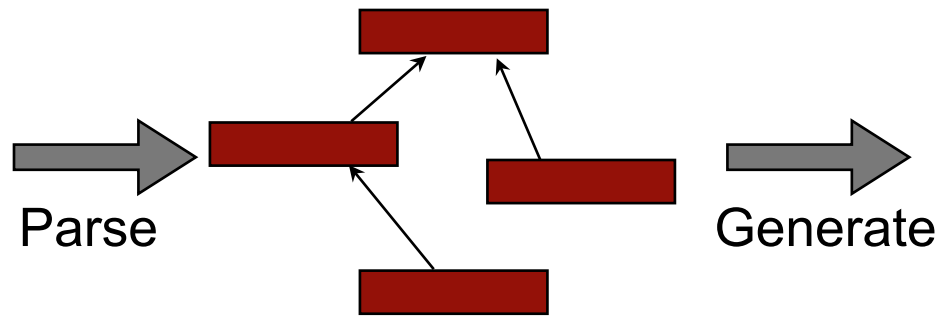


DSL ARCHITECTURE

Basic DSL Architecture

```
computer:  
  processor:  
    cores 2  
    i386  
  disk:  
    size 150  
  disk:  
    size 75  
    speed 7200  
    sata
```

DSL Script



Semantic Model

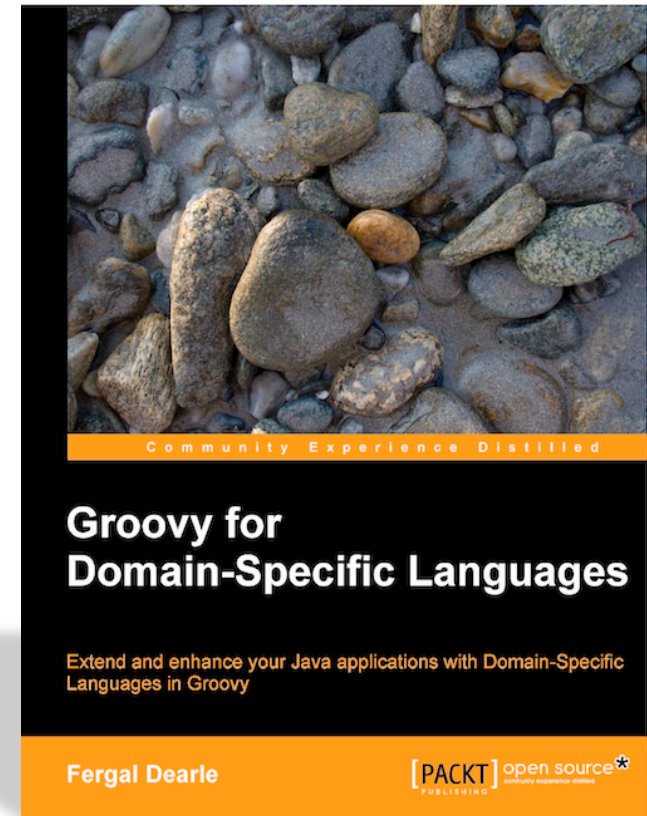
```
Processor p = new Processor  
  (2, Processor.Type.i386);  
Disk d1 = new Disk(150,  
  Disk.UNKNOWN_SIZE,  
  null);  
Disk d2 = new Disk(75,  
  7200,  
  Disk.Interface.SATA);  
return new Computer(p, d1,  
  d2);
```

Generated Code

DSL Script

A language to build, configure or do in your domain

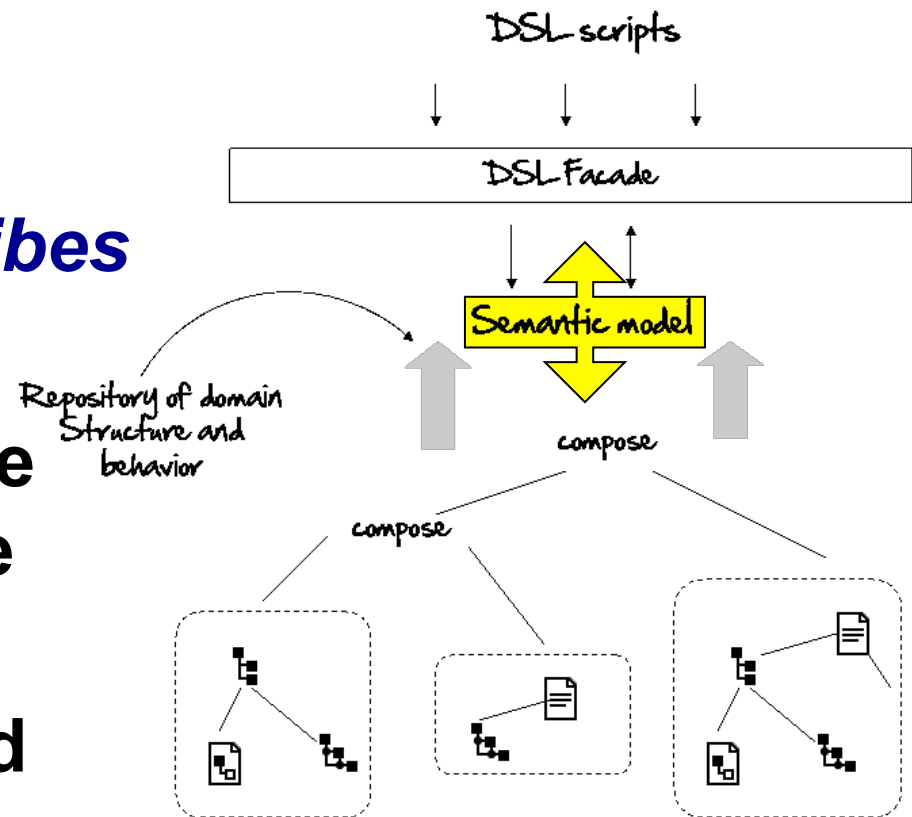
- Not necessary textual, could be graphical
- Might reuse syntax of host language



Semantic Model

An in-memory representation of the subject the DSL describes

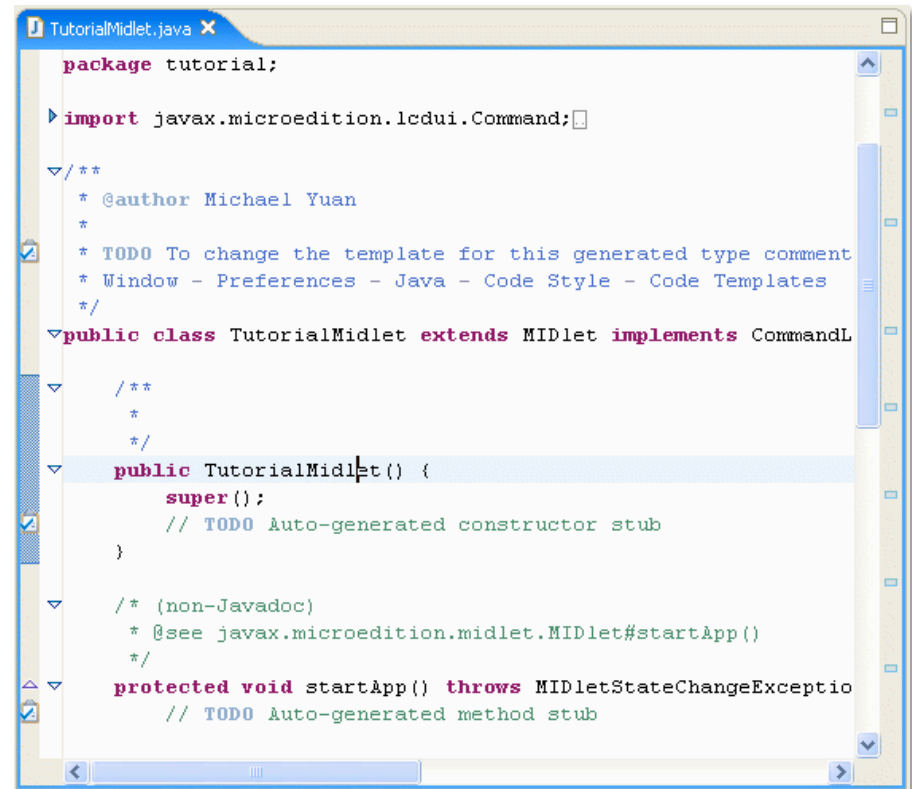
- Sometimes this is the Abstract Syntax Tree (AST)
- Separates parser and generation



Generated Code

Executable representation of the DSL

- Evaluation during parsing
- Interpretation of semantic model



```
package tutorial;

import javax.microedition.lcdui.Command;

/**
 * @author Michael Yuan
 *
 * TODO To change the template for this generated type comment
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class TutorialMidlet extends MIDlet implements CommandL

    /**
     *
     */
    public TutorialMidlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /* (non-Javadoc)
     * @see javax.microedition.midlet.MIDlet#startApp()
     */
    protected void startApp() throws MIDletStateChangeExceptio
        // TODO Auto-generated method stub
    }
}
```



General DSL Development Activities

1. Identify problem **domain** of interest and gather the relevant **knowledge** in this domain
2. Capture domain knowledge in **semantic** notions and operations
3. Construct a **library** of **components** that implement the semantic notions and operations
4. Design a **DSL** that concisely describes applications in the domain
5. Develop a **compiler** (DSP) that translates DSL programs to a sequence of library calls
6. Write DSL **programs** (DSD) for all desired applications and compile them

Let's watch Martin Fowler clip...

- What did Martin Fowler have to say about the usefulness of DSLs?
- What were some of the key elements of a DSL?
- How do DSLs get used today?



Help Me Help You

- Plus/Delta course evaluation on ANGEL
- Been doing some things that are new and want your impressions
- Please take 10 minutes to help me improve the course



Paper Discussion: DSL Paper

“When and How to Develop Domain-Specific Languages”

- What are the main thrusts of the paper?
- What are the controversial points and your positions?
- What did you get out of reading about Domain Specific Languages?





Homework and Milestone Reminders

- **Continue Reading DSL Survey Paper...**
 - Be prepared to discuss and even lead the discussion
 - Write a brief summary of observations on the paper
- **Milestone 2: Establish a repository and structure for assembling components for your FacePamphlet application**
 - Due by 11:55pm Friday, April 1st, 2011 (no foolin'!)