

CSSE 490 Model-Based Software Engineering: More on Domain Specific Languages



Shawn Bohner

Office: Moench Room F212

Phone: (812) 877-8685

Email: bohner@rose-hulman.edu



ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

Plan for the Day

- **Plus/Delta Summary**
- **Watch Martin Fowler clip on DSLs**
- **DSL Reading Discussion**
- **Milestone 2 Discussion**
- **More on Domain Specific Language (DSL)**



+/ ∂ Feedback: Lectures



Pace

- 0 – much too fast
- 4 – somewhat too fast
- 1 – Somewhat too slow
- 0 – much too slow

Working well

- Class slides and material
- Project will be very helpful
- Examples
- Group discussions
- Daily Quizzes
- Move to Conference/classroom

Improvements

- Make lectures less abstract/
more specific (2)
- On Target (1)
- More examples like video (1)
- More detailed explanation of
terms (1)
- Slow down (1)
- More class time on project (1)



+/*o* Feedback: Quizzes

Quizzes

- 2** – Very helpful
- 3** – somewhat helpful
- 0** – somewhat unhelpful
- 0** – Very unhelpful

Improvements

- Quizzes are fine (4)
- Don't know/NA

Working well

- Focuses lecture for me
- Questions work well
- Good study guide
- Indicates high points
- Question #'s on slides



+/*o* Feedback: Reading and Homework

Reading

- 0** – all of it
- 5** – most of it
- 0** – little of it
- 0** – none of it

Homework Difficulty

- 0** – much too difficult
- 4** – a bit too difficult
- 1** – a bit too easy
- 0** – much too easy



+/ ∂ Feedback: Homework Helpfulness

Homework Helpfulness

- 0** – very helpful
- 3** – somewhat helpful
- 2** – somewhat unhelpful
- 0** – very unhelpful

Working well

- In-class discussions from readings
- Summaries relate information from class
- Summaries reinforce reading
- Interesting topics

Improvements

- With fairly difficult topics, more time for readings (1)
- Shorter papers (2)
- Target more clear or at least less abstract readings
- More concrete papers and assignments



+/*o* Feedback: Workload

■ Workload

- 1** – much higher than average
- 2** – somewhat higher than average
- 3** – somewhat lower than average
- 0** – much lower than average

■ General Comments

- Moving a bit fast
- Not feeling ready for exam or implementing the project
- Could use a middle option for questions 1, 9, & 12
- While load is lower, the material is more difficult
- Encouragement (4), Neutral (1) Discouragement (0) 😊

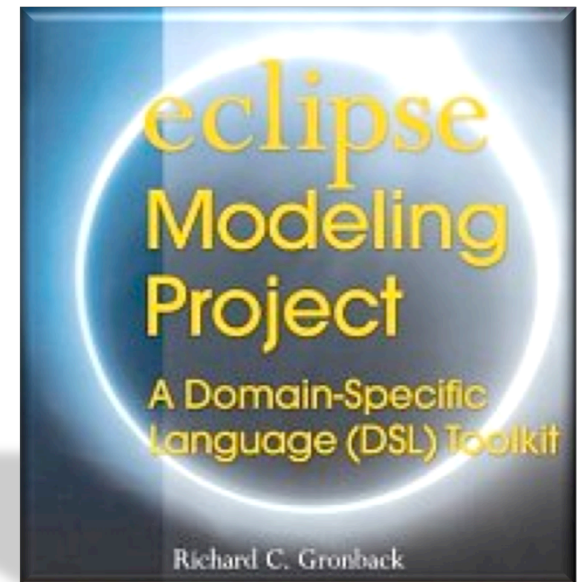


Summary of $+/\partial$ Actions

- More concrete papers (where available) for case study assignments
- More examples in class
 - More time in class on project
- Slow the pace in class for more discussion
- Homework and projects due at 11:55pm

Recall: Domain Specific Languages

- 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, ...





Recall: 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 pervasiveness of DSLs?
- What difference between an internal and external DSL?
- What does he say about Language Workbenches?



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?



Let's talk about the project...

- How is the repository coming?
 - Basic component organization?
 - Metamodel for components?
- How is Albert's backup strategy coming?



Mainstream DSL Approaches

- External DSL
- Internal DSL
- DSL Workbench

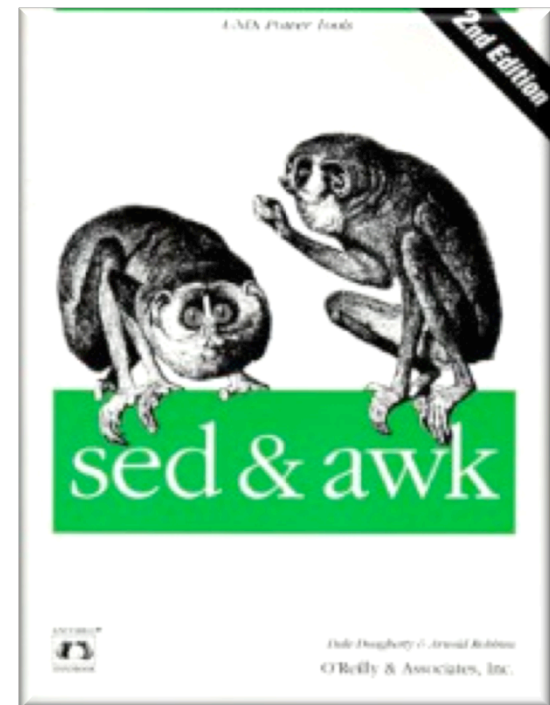


External DSLs

DSLs that use a different syntax to the main language that uses them

■ Examples

- make, flex, yacc, bison
- XPath, SQL, regexp
- sed, awk





External DSL Strengths / Limitations

Strengths

- Language Designer
 - Tools for compiler construction can be used
- Language User
 - Simpler to use than a GPL

Limitations

- Language Designer
 - Expensive to implement
- Language User
 - Weak tool support
 - Yet another language to learn
 - Often targeted towards a particular GPL
 - Often difficult to closely integrate with GPL



Internal DSLs

DSLs that *share the same syntax* to the main language that uses them

- A subset of the host language is used
- Examples
 - PetitParser (Smalltalk)
 - rake, rspec (Ruby)
 - jQuery (JavaScript)
 - RPython (Python)



Internal DSL Strengths / Limitations

Strengths

- Language Designer
 - No special tools
 - No new grammar
- Language User
 - Intermixable with GPL
 - Tools continue to work
 - No new language to learn

Limitations

- Limited expressivity
- Unnecessary syntactic noise
- Constrained by host language

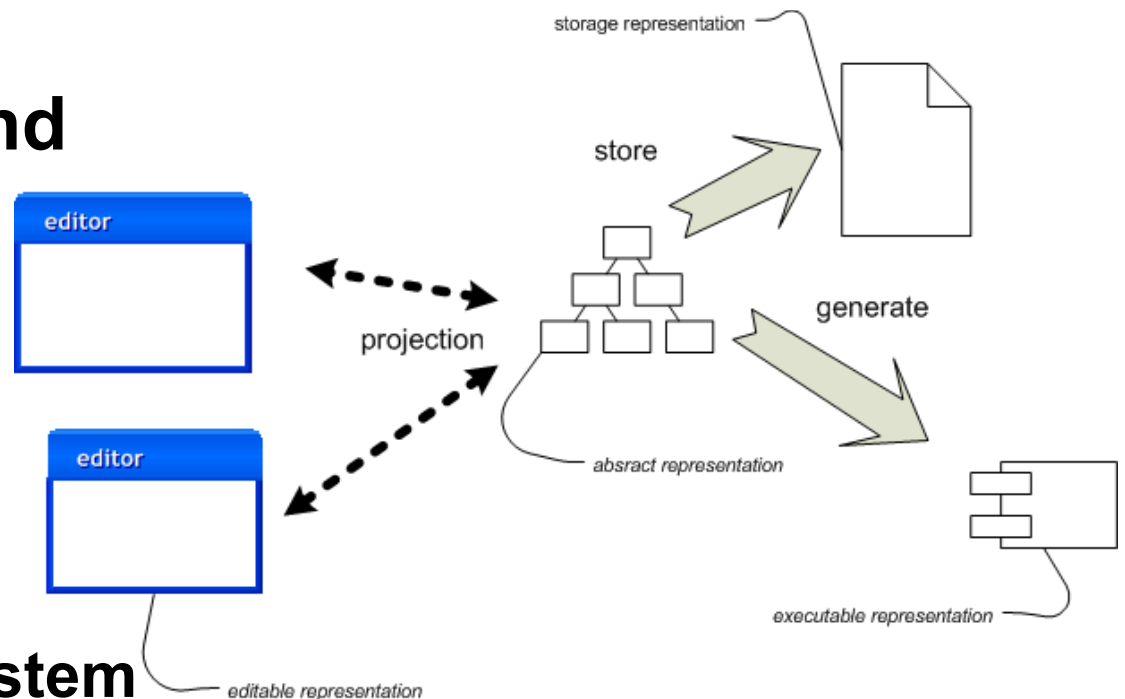
Language Workbenches

IDEs designed for building DSLs

- Common host and domain specific language representation

- Examples

- JetBrains Meta Programming System (MPS)
- openArchitectureWare



theSimpleLanguage - jetbrains.mps.samples.theSimpleLanguage.structure/TheSimplestConcept - JetBrains MPS (Yellow Fox) #1161

File Edit View Go To Generate Build Tools Version Control Window Help

Project /Applications/M... View as: Logical View

Project

- jetbrains.mps.samples.theSimpleLanguage
 - jetbrains.mps.samples.theSimpleLanguage
 - sandbox (generation required)
 - Hello
 - jetbrains.mps.samples.theSimpleLanguage
 - structure
 - TheSimplestConcept
 - implements : INamedConcept
 - propertyDeclaration : text
 - editor
 - TheSimplestConcept_Editor
 - cellModel : collection
 - childCellModel : collection
 - childCellModel : constant
 - childCellModel : collection
 - constraints
 - typesystem
 - generator/baseLanguageGenerator
 - jetbrains.mps.samples.theSimpleLa
 - main@generator
 - Hello
 - visibility : public
 - constructor : Hello()
 - rootTemplateAnnotationSa
 - staticMethod : main(String)
 - propertyMacro\$property_a
 - main
 - rootMappingRule : map Th
 - runtime
 - all models
 - modules pool

```

the simplest concept instance Hello
text = hello text

```

```

concept TheSimplestConcept extends BaseConcept
implements INamedConcept

instance can be root: true

properties:
text : string

children:
<< ... >>

references:

```

Structure Editor Constraints Behavior Typesystem Intentions Find Usages Data Flow Generator

```

editor for concept TheSimplestConcept
node cell layout:
[
  [> the simplest concept instance {name}<]
  <no text>
  [> text = {text}<]
]

```

Structure Editor Constraints Behavior Typesystem Intentions Find Usages Data Flow Generator

Hello

```

public class $[Hello] extends <none> implements <none> {
  <<static fields>>

  <<static initializer>>
  <<fields>>
  <<properties>>
  <<initializer>>
  public Hello() {
    <no statements>
  }

  <<methods>>
  public static void main(String[] args) {
    System.out.println("[Hello]");
  }

  <<static inner classifiers>>
}

```

Structure Editor Constraints Behavior Typesystem Intentions Find Usages Data Flow Generator



Homework and Milestone Reminders

- **Take home examination tomorrow...**
 - Review book and paper readings
 - Review class slides

- **Milestone 2: Establish a repository and structure for assembling components for your FacePamphlet application**
 - Due by 11:55pm Tuesday, April 5th, 2011