xText

Ivan Gibbs

University of Nevada, Reno

April 11, 2013
Modeling?

- build a smaller, simpler, cheaper model first
Modeling?

- build a smaller, simpler, cheaper model first
- learn from the model
Modeling?

- build a smaller, simpler, cheaper model first
- learn from the model
- the model guides the implementation
Model Driven Engineering

- the top down perspective
Model Driven Engineering

- the top down perspective
- solve top level problems first
Model Driven Engineering

- the top down perspective
- solve top level problems first
- use models to generate code
Domain Specific Language

- specific language to model software systems
Domain Specific Language

- specific language to model software systems
- textual or graphical language
Domain Specific Language

- specific language to model software systems
- textual or graphical language
- can be combined with each other to make larger systems
Textual DSL

- write a script in the DSL
Textual DSL

- write a script in the DSL
- program splits script into tokens
Textual DSL

- write a script in the DSL
- program splits script into tokens
- tokens are organized via a grammar
Textual DSL

- write a script in the DSL
- program splits script into tokens
- tokens are organized via a grammar
- an abstract syntax tree
Textual DSL

- write a script in the DSL
- program splits script into tokens
- tokens are organized via a grammar
- an abstract syntax tree
- a semantic model is made
Textual DSL

- write a script in the DSL
- program splits script into tokens
- tokens are organized via a grammar
- an abstract syntax tree
- a semantic model is made
- code is generated from the semantic model
Textual DSL–Graphviz Example

- dsl.dot is defined as:

```plaintext
digraph dsl {
    rankdir=LR;
    "DSL script" -> "Lexer";
    "Lexer" -> "Parser";
    "Parser" -> "AST";
    "AST" -> "Code Generation";
}
```

- then we run the command

```
dot dsl.dot -Tpng -odsl.png
```

- then we get the following image:
Entry defined in XML:

```xml
<Person active="true" role="work">
    <Name>
        <First>Sergiu</First>
        <Last>Dascalu</Last>
    </Name>
    <Address>
        <Email>dascalus@cse.unr.edu</Email>
    </Address>
    <Information>Phd advisor.</Information>
</Person>
```
Then we use XSL to map XML to LDIF

```xml
<xsl:text>dn: </xsl:text>
<xsl:value-of select="Name/First"/>
<xsl:text> </xsl:text>
<xsl:value-of select="Name/Last"/>
<xsl:if test="Address/Email">^M
  <xsl:text>, </xsl:text>
  <xsl:text>mail=</xsl:text>
  <xsl:for-each select="Address/Email">
    <xsl:value-of select="text()"/>
  </xsl:for-each>
</xsl:if>
<xsl:text>
objectclass: person

<xsl:text>cn: </xsl:text>
<xsl:value-of select="Name/First"/>
<xsl:text> </xsl:text>
<xsl:value-of select="Name/Last"/>
<xsl:text>
mail: </xsl:text>
<xsl:value-of select="Address/Email"/>
<xsl:text>
modifytimestamp: 0
</xsl:text>
```
Then we run command

dxsltproc -o People.ldif ldif.xsl People.xml

to get the following LDIF file:

dn: Sergiu Dascalu, mail=dascalus@cse.unr.edu
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
objectclass: mozillaAbPersonAlpha
cn: Sergiu Dascalu
mail: dascalus@cse.unr.edu
modifytimestamp: 0
We import into Mozilla Thunderbird email program
Or we transform it into HTML instead of LDIF
Benefits

▶ one location for names and addresses
Textual DSL

Benefits

▶ one location for names and addresses
▶ easy to create multiple output formats
Textual DSL

Benefits

- one location for names and addresses
- easy to create multiple output formats
- low-tech tools needed (no C++ or Database)
Textual DSL

Benefits

▶ one location for names and addresses
▶ easy to create multiple output formats
▶ low-tech tools needed (no C++ or Database)
▶ provides a human readable interface
Textual DSL

Benefits

- one location for names and addresses
- easy to create multiple output formats
- low-tech tools needed (no C++ or Database)
- provides a human readable interface
- helps interoperation through mapping
Textual DSL

Drawbacks
  ▶ reliance on tools
Textual DSL

Drawbacks
- reliance on tools
- requires time to create
Textual DSL

Drawbacks

- reliance on tools
- requires time to create
- require Language designers
Textual DSL–xText

Generate Text Artifacts video
Textual DSL–xText

Make Java Project Video
Textual DSL–xText

Define Language Instance Video
NetCDF DSL
Question and Answer?
You are assigned to manage a small team of 8 people to develop a missile guidance system. You have a specification and must use a proprietary military language and compiler. What type of development process would you choose?

- a: Agile
- b: Waterfall
- c: Incremental Development
- d: Other

Describe your reasoning.
Question 1 for Test

The answer here can be any method along with explanations.
What are the arguments for and against Model Driven Engineering (MDE)
What are the arguments for and against Model Driven Engineering (MDE)

For
- reduce errors
- speed up design and implementation
- reusable, platform-independent models
- code implementations can be generated from the model

Against
- model abstractions may conflict with implementation abstractions
- implementation may be easier with COTS
- platform-independence is only valid for long-lived systems