Domain Driven Design and Domain Specific Modeling

Peter Bell
SystemsForge
Overview

- What is DSM?
- DSM and DDD
- Common Misconceptions
- Key Distinctions
- When to Use
- Discussion/Q&A
Who Am I?

- **Researcher** - Published in DSM forum/ooPSLA
- **Programmer** - 30-40 projects/yr.
- **SystemsForge** - Software Product Line
- **Write/Present** - DSM, Code Gen, Agile, etc.

Calibration

- Have you:
  - Used DDD on a project?
  - Generated code from UML?
  - Created parser for custom XML config files?
  - Written parser for textual DSL?
What is DSM?

- Using Domain Specific Languages to implement some/all of the application functionality
Domain Specific Language

“An executable language targeted to a specific problem domain”

- Specific - can’t do everything
- Executable - not just conceptual
- Raising level of abstraction
DSM and DDD

- Ubiquitous Language -> Domain Specific Language
  - Guaranteed sync between model and code
  - Generate different projections for different stakeholders
  - Quicker development of complex apps
  - Less testing required of user readable statements
Common DSM Misconceptions

✓ It needs to be:
  - UML/MDA
  - Domain expert writeable
  - BDUF
  - 100%

✓ It can’t be:
  - 100%
DSM: Key Distinctions

- Horizontal vs. Vertical DSLs
- Abstract Grammar vs Concrete Syntax
- Internal vs. External DSLs
- Code Gen vs. Interpretation
- Extension Mechanisms
- Tooling Options
Horizontal vs. Vertical

- **Horizontal** (tech focused)
  - SQL
  - RegEx
  - Import description language
  - General workflow language

- **Vertical** (business focused)
  - Insurance policy language
  - Product configuration language
  - Shipping rules language
DSL Key Concept

Abstract Grammar vs. Concrete Syntax

*What* you say vs. *How* you say it
DSL Key Concept

Abstract Grammar vs. Concrete Syntax

**What** you say vs. **How** you say it

```xml
<Object title="User">
  <Name>FirstName</Name>
  <Name>LastName</Name>
</Object>
```
DSL Key Concept

Abstract Grammar vs. Concrete Syntax

What you say vs. How you say it

<Object title="User"> 
  <Name>FirstName</Name> @FirstName 
  <Name>LastName</Name> @LastName
</Object>
Types of DSLs - Internal

- **API**
  - `UserService.import("test.csv", ",", 
  "FirstName,LastName,Email", "tbl_User", "update");`

- **Method Chaining (Fluent)**
  - `User.IsValid.FirstName()`
Types of DSL - External

- Comma delimited
  - "test.csv", ",", "FirstName,LastName,Email", "tbl_User", "update"
- XML
  - `<import filename="test.csv" record-delimiter=""," field-name-list="FirstName,LastName,Email" source="tbl_User" method="update" />
- Databased
- “Little Language”
  - Import test.csv using commas update tbl_User with FirstName,LastName,Email
- Visual
Code Gen vs. Interpretation

```xml
<businessObject>
  <extends>BaseObject</extends>
  <title>Event</title>
  <name>Event</name>
  <IDPropertyName>EventID</IDPropertyName>
  <titlePropertyName>Title</titlePropertyName>
  <defaultOrderBy>PublicationDate</defaultOrderBy>
  <uniqueInstancePropertyName>EventID</uniqueInstancePropertyName>

  <properties>
    <calculatedField title="Event ID" name="EventID" columnName="EventID" dataType="int" sqlDataType="int" calculateOn="_insertobject" calculationType="ID"/>
    <field title="Publication Date" name="publicationdate" columnName="publicationdate" dataType="date" sqlDataType="DateTime"/>
    ...
  </properties>
</businessObject>
```
Extension Mechanisms

✓ Inheritance
✓ Includes
✓ Mixins
✓ AOP
✓ Protected Regions
Tooling Options

- MetaEdit+
- openArchitectureWare
- Visual Studio
- JetBrains MPS (beta)
- Intentional (one day . . .)
- NOT Antlr, lex/yacc, etc
When to Use DSM

- Smart UI vs Layered Design (pp 76)
- Adding DSM
  - Complexity of problem
  - Skillset of team
  - Stability of domain model
  - Quantity of domain rules
Discussion/Q&A