Win Friends and Influence People
...with DSLs

Dr. Vladimir Bacvanski  vladimir.bacvanski@scispike.com
Lloyd Mangnall    lmangnal@vha.com
Joel Denton       jdenton@aptitude.com
The Beginning: Common Challenges

Business & Technology Rapidly Changing

- Aggressive new competition
- Broad regulatory changes

Extremely Brittle & Aging Legacy Environment

- Silo Architecture Resembling “3rd World Hillside of Huts”
- Significant degree of functional & technical redundancy
- >1500 point-to-point ETL integrations
- Regularly takes a year or more to “integrate” new systems
- **Executive Mandate: No longer “business as usual”**
- Forget changing the wheels…**we’re changing the bus into an airplane, while it’s moving!!!**
Your mission, should you accept to…

…like we really have a choice…

Rapidly deploy a scalable, flexible architecture that can respond to new business opportunities and leverage emerging technologies.

Key Business Drivers:

• Increase the “Hit Rate” of Business Innovations
• Respond Rapidly to Customer Feedback
• Reduce Time to Market
• Reduce Cost
• Quality! Quality! Quality!

Key Technical Drivers:

• Scalable, Supportable, Flexible Architecture
• Interoperable (SOA) vs. Interfaced (ETL)
Why Domain Specific Languages

Consistently Capture Our Problem Domains

• Keep the level of abstraction closer to business
• Transition from tacit knowledge hidden in legacy code to explicit representation in DSL

Automate Transition from High-level Models to Code

• Apply standard transformations & patterns
• Facilitate ongoing adoption of emerging technologies

SciSpike has been applying agile DSL development in similar scenarios → this was a good match
Win Friends and Influence People?

Win friends and influence business stakeholders:

• Solve business problems faster through fast software delivery
  - Enable new opportunities to be rapidly realized

Win friends and influence developers:

• Use the cutting edge software technology
• Create a productive environment
• Experience making a significant contribution to the project
Using Domain Specific Languages

DSLs provide for expressing solution in a vocabulary of the problem domain

Graphical DSL

- UML with stereotypes or custom notation
- Business users are often challenged by complex graphical notation
- Complex tool development

Textual DSL

- Better acceptance by our users
  - Specially trained analysts
- Much easier tool development
Our DSL Tools: EMF and Xtext

Eclipse Modeling Framework

• Our technology of choice for implementing meta-models

Xtext

• A framework for developing DSLs and supporting tooling
• Created tooling integrates with Eclipse
The CTO’s Business Case

Focus on Systematic Risk Mitigation
Rework resulting from unmitigated risk is EXPONENTIALLY MORE COSTLY the later it is discovered along the delivery lifecycle.

Cost of Error

1/10x  1/2x  1x  2x  5x  10x  100x

Reqs  Design  Develop  Unit  System  Accept  Prod

Agile DSL  Agile  Waterfall  Prod
Traditional Agile Product Delivery Profile
Focused on Systematic Risk Mitigation

Business:
- ~5%
- 10%
- $100K

Technology:
- 20%
- 30%
- $400K

Implementation:
- 65%
- 50%
- $1.3M

Acceptance:
- 10%
- 10%
- $200K

＜-Risk:
- ＜-Effort
- ＜-Schedule
- ＜-Cost

Project X:
- UC = 275
- AUCP = 1368
- Effort = 20523 eh
- TTM = 10 months
- Total Cost = $2M

VHA Inc. Confidential Information
Introduction of DSLs into the Process…
Still Focused on Systematic Risk Mitigation

Goal: Generate 100% of x% of the overall application

Project X:
UC = 275
AUCP = 1368
Effort = ??? eh
TTM = ?? months
Total Cost = $???
What We’ve Seen So Far…

Recast our projected effort to ~14000 eh (30% productivity increase)

- Experienced an initial increase in technical debt
  - Building initial languages
  - Selecting initial architectural components
- 3 developers generating 100% of ~75% of the application

Project X:
- UC = 275
- AUCP = 1368
- Effort = ~14000 eh
- TTM = 10 months
- Cost = $2M
What We Hope to See in the Future…
Sustained 50-60% overall productivity increase

Project N:
UC = 275
AUCP = 1368
Effort = ~8000 eh
TTM = 6 months
Cost = $800K
Iterative DSL Lifecycle

- Define/Update DSL
- Validate DSL UX
- Generate
- Develop/Use Framework
- Add Handcrafted Code
- Refactor/Refine
- Model
- Metamodel
- Continuous
  - Build
  - Integration
  - Refactoring
  - Collaboration
- increased iterations
- Domain/Business/Requirements
Obstacles and Misconceptions

Modeling

• "Just drawings"
• Perceived disconnect between modeling and programming

Process

• Reverting to waterfall
• Expecting to write application code early on

Dealing with UI

• Specialized UI teams may prefer to go into very detailed waterfall-like design early on
The problem: What is the product?

• Just the generated solution?
• Or the DSL language, the tooling, and the underlying framework?

Is management interested in the difference?

• They just want a solution to business problem!
• Need to sell them on the tools!

Evangelizing DSL approach is a must from early on

• But: it must be expressed in the business terms (they did not hear before)
• Move from project to product line / rapid change thinking helps
People and Roles

**Language builders**
- Exposure to multiple languages helpful

**Tool builders**
- Able to think in meta-models (test early!)

**Domain experts**
- Available and motivated to help

**Language users**
- Business users paired with developers familiar with the DSL
1. **Focus on getting the domain model right**

2. **Rapidly prototype and try out the language with users**

3. **Cultivate a champion DSL designer to work closely with domain expert**

4. **Keep the language minimal**

5. **Keep underlying framework clean and refactor early**

6. **Documentation is essential**

7. **Staff that is not afraid of learning new things**
1. Pressure to fall back to waterfall

2. Pressure to "finally start coding the application"

3. Pressure to "add more bodies"

How to avoid these?

- Must manage business and management stakeholders early on
- Must resist consensus based decision making under pressure and preserve the purity of the DSL solution
- Requires a technical architect with a strong vision
Seven Habits of Successful DSL Projects…

1. **Look for opportunities where DSLs are a good match**
   - Be Proactive

2. **Focus on an easy to use DSLs that users like**
   - Begin with the end in mind

3. **Focus on area with the most impact**
   - Put first things first

4. **Work together with business and solve real problems**
   - Think win/win
…Seven Habits of Successful DSL Projects

5. Understand the domain, then choose the matching approach
   • Seek first to understand, then to be understood

6. Explore synergy with similar projects where DSL may work
   • Synergize

7. Be on the lookout for new DSL tools and practices
   • Sharpen the saw