Democratizing Software Creation

Markus Völter
Independent Consultant for ISC

Henk Kolk
CTO Financial Services, Capgemini
The Challenge

• Business needs to accelerate innovation

• Software is a main vehicle of innovation

• Innovation through software is hampered by communication between domain experts and programmers
The Key Players

Domain Expert  Programmer
...to fully integrate Domain Experts into software creation.
A brief history of software

A struggle to distinguish and treat separately the problem and the program
No such faith comforts the software engineer. Much of the complexity he must master is arbitrary complexity, forced without rhyme or reason by the many human institutions and sys-
A brief history continued

- When we fail to treat separately the problem and the program
  - The problem and the program get mixed up creating the complexity we hear about
  - We get stuck with improving the resulting mess
Complexity of scrambled eggs
After refactoring ;-}
Domain Orientation trends:

- Domain Specific Languages (DSL)
- Domain Specific Modeling (DSM)
- Domain Driven Design (DDD)
- Model Driven Architecture (MDA)
- Model Driven (Software) Development (MD(S)D)
- Generative Programming (GP)
- Language Oriented Programming (LOP)
- Code Generation
- Meta Programming (Ruby)
- ...

© 2007 Intentional Software Corporation
DSL + Code Generation

Problem Description → Generator → Program Source

Built by Domain Expert

Built by Generator Programmer

© 2007 Intentional Software Corporation
Key benefits of domain orientation

• Domain Expert participation feasible - business knowledge isolated from technology

• Complexity is reduced – separation of concerns

• Programmers create a more valuable artifact – capture process instead of “messy” result
What prevents mainstream use?

1. Integrate Domain Experts fully
   - Matching existing notations
   - Mixing graphical/textual notations

2. Multi-domain
   - Compose independent domains
   - References between domains

3. Domain evolution

4. Groupware
Through **Intentional Domain Workbench™**

bring the current trend of domain orientation to a new level to *fully integrate Domain Experts into software creation.*
• Intentional Domain Workbench™

  • Intentional Tree domain representation
  • Projectional multi-view editing
Intentional Tree

- Extendible, uniform representation
- Strong identities throughout
- No fixed meta-levels
- Versioned storage
- Separated concerns

```
return a = b / (c + 1);
```

© 2007 Intentional Software Corporation
Projectional editing

\[ \text{return } a = b / (c + 1); \]
\[ \text{return } a = \frac{b}{c + 1}; \]
\[ \text{return } a = \text{Display / Input} \]

Domain Code  Projector  Display / Input

© 2007 Intentional Software Corporation
Integrate Domain Experts

- Matching existing notations
- Mixing notation graphics / text

- Projectional editor supplies appropriate notation
- Projectional editor decouples domain code from notation
  - Multi-view, embedding, extension…
- Graphics / text are treated uniformly
- Notation can change on domain or other selected boundaries

© 2007 Intentional Software Corporation
Multi domain

- Compose independent domains
- References between domains
- Tree structure accommodates composition
- Inter-domain references connecting domains
Domain evolution

- Tree storage is independent of schema – will not “break” if schema changes
- Notation can keep up with evolution
- Further parametrization is always possible
- Independent concerns can be added without interfering with others
Groupware

- Tree storage requires rethinking groupware
- Change logs for fully general solution
- Edit “conflicts” are a “mini domain” – integrated with notations
- Versioning and audit trails
- Familiar metaphors – versions, branches, merges, checkins
Intentional Domain Workbench™ Status

- Technology is fully capable of handling the Domain Workbench requirements.
- We are nearing operational use in selected domains.
- We are working with selected OEM customers only, for example with Capgemini.
Henk Kolk
CTO Financial Services Capgemini
Problems for Pension Companies

Need for pension product innovation

• Governmental interest
  – New Pension Laws
• Mergers
• Transparency

Problems

• Time to market
• Abstract product models
• Ensuring quality
Old way: disconnected domains

Current issues:
• Expensive handovers
• Traceability
New way: connected domains

Pension Plan Analysis

No apparent handover
Traceability comes for free

Functional design

Technical design

Test

Program

Validation & Handover

Execute
**Old way: Excel & Word**

### 20.6 T-PREMIEM-VI-V1-2006

**Definition:**
- \( T-PREMIEM-VI-V1-2006 \) (IDAT, EDAT)

**Formula:**
- Berekening van de totale pensioenboete vanaf de overdrachtdatum tot het vermogen.

**Waargemaakt:**
- 1

**Gebruikt door:**
- Bij het jaarwerk.

**Ingetekend:**
- 01-01-1999

**Strak:**
- 2

**RKT:**
- 1

**ALS:**
- 1

**DAN:**
- T-PREMIEM-VI-V1-EDAT

**ADS:**

### 20.7 T-PME-OP-TM-VI-VA-06

**Definition:**
- \( T-PME-OP-TM-VI-VA-06 \) (IDAT, EDAT, overdracht, contract, @, @-

**Werkopname:**
- Berekening van de pensioenkosten vanaf het vermogen tot de uitgifte. Dit bedrag moet worden doorgegeven bij de PVD:
  - 0 = normale pensioenkosten
  - 1 = pensioenboete
  - 2 = pensioenboete verlengd
  - 3 = pensioenboete verlengd gedeeltelijk

**Waargemaakt:**
- 1

**Gebruikt door:**
- Bij het jaarwerk.

**Ingetekend:**
- 01-01-1999

**Strak:**
- 2

**RKT:**
- 3

**ALS:**
- 3

**DAN:**
- 0

**ADS:**
- T-PME-OP-TM-VI-VA-06(IDAT,EDAT,PAR,3) + T-PME-OP-TM-VI-VA-06(FR,1,PAR,3)

---

<table>
<thead>
<tr>
<th>Id</th>
<th>Sal</th>
<th>Ud</th>
<th>Es</th>
<th>Ve</th>
<th>WI</th>
<th>Inv</th>
<th>IV</th>
<th>IO</th>
<th>Or</th>
<th>Ix</th>
<th>Groep</th>
<th>Toez</th>
<th>Stat</th>
<th>VN</th>
<th>Ac</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Invogegevens</th>
<th>Uitvoergegevens</th>
<th>Berekening</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code pensioensysteem</td>
<td>Code pensioensysteem</td>
<td>als Code pensioensysteem = 1 (Eindlinc)</td>
</tr>
<tr>
<td>Opgebouwde diensttijd vorige mutatie</td>
<td>Opgebouwde aanspraak vorige mutatie</td>
<td>begin</td>
</tr>
<tr>
<td>Toekomst diensttijd parttime</td>
<td>Toekomst diensttijd parttime</td>
<td>dan</td>
</tr>
<tr>
<td>Uitzicht diensttijd parttime</td>
<td>Uitzicht diensttijd parttime</td>
<td></td>
</tr>
<tr>
<td>Opgebouwde aanspraak vorige mutatie</td>
<td>Opgebouwde aanspraak</td>
<td></td>
</tr>
<tr>
<td>Aanspraak per dienstjaar fulltime vorige mutatie</td>
<td>Toekomst aanspraak</td>
<td></td>
</tr>
<tr>
<td>Aanspraak per dienstjaar fulltime deze mutatie</td>
<td>Uitzicht aanspraak</td>
<td></td>
</tr>
<tr>
<td>Code afronding aanspraak</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Uitvoergegevens:**
- Opgebouwde aanspraak
- Toekomst aanspraak
- Uitzicht aanspraak

---

<table>
<thead>
<tr>
<th>Id</th>
<th>Sal</th>
<th>Ud</th>
<th>Es</th>
<th>Ve</th>
<th>WI</th>
<th>Inv</th>
<th>IV</th>
<th>IO</th>
<th>Or</th>
<th>Ix</th>
<th>Groep</th>
<th>Toez</th>
<th>Stat</th>
<th>VN</th>
<th>Ac</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

| Berekende percentage | Opbouwpercentage OP | Overgangspercentage OP-NP | OP-NP |
New way: Pension Workbench

- Matching existing notations
  - Pension experts record pension world in their notations

Old spreadsheet

Pension Workbench
Multiple Views with Graphics

Pension Plan versions

Rule dependencies
## Compose Business Domain

- Domain Schema
- Projectional Editors

### Rules

<table>
<thead>
<tr>
<th>Result</th>
<th>Name</th>
<th>Documentation</th>
<th>Tags</th>
<th>body</th>
</tr>
</thead>
</table>
| Premium old age pension| Rule for \( \text{Premium old age pension} \) | final pay 1970-1988, final pay 1988-1999, final pay 1999-2004, avg pay 2004-2005, avg pay 2006 | (Premium percentage OP * (Pension base - 1)) * Part time % | Always, Pension.RL~, At, Pension.RL~, Average, Pension.RL~, Begin, Pension.RL~, Branch, Pension.RL~, Break, Pension.RL~, Coalesce, Pension.RL~, CurrentOrNext, Pension.RL~, CurrentOrPrevious, Pension.RL~, Date, Kernel~, DaysOf, Pension.RL~, Div, NPL~, Each calendar year, Pension.RL~, Eq, NPL~, False, Kernel~, For, Pension.RL~, Ge, NPL~, If, NPL~, In, NPL~, Lt, NPL~, Ne, NPL~, Not, NPL~, Or, NPL~, Pred, NPL~, Rel, NPL~, Sex, NPL~, Symbol, NPL~, Time, NPL~, True, NPL~, Variable, NPL~, Year


| Salary gap             | Rule for \( \text{Salary gap} \) | final pay 1970-1988, final pay 1988-1999, final pay 1999 | when(((previous(Gross salary) = 0 and current(Gross salary) = 0) or previous(Gross salary) = 0) or (Gross salary - previous(salary) / previous(salary) > 10 %)) ) |
Integrate Rule Test Domain

- Unit Tests for pension rules
- Real time evaluation

<table>
<thead>
<tr>
<th>Name</th>
<th>Documentation</th>
<th>Tags</th>
<th>Valid time</th>
<th>Fixture</th>
<th>Expected value</th>
<th>Actual value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1990-1-31</td>
<td>Piet Van Dijk</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1990-1-31</td>
<td>Jan De Jong</td>
<td>true</td>
<td>Nil</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1991-1-31</td>
<td>Piet Van Dijk</td>
<td>Nil</td>
<td>Nil</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1992-1-31</td>
<td>Piet Van Dijk</td>
<td>Nil</td>
<td>Nil</td>
</tr>
</tbody>
</table>
Integrate System Test Domain

- Test cascading rules and their interrelations
- Real time evaluation
Build Code Generators

- Multiple implementation target languages
Capgemini Pension Language

Unified Pension Language

Capgemini Pension expert discussion

Client Pension Language 1

Client Pension Language 2

Client 1: “Please raise abstraction level of my language”
Testing – Lack of “groupware”

- Rules domain
- Rules
- Test cases (VBA)
- XML Export domain
- Multiple users

Issues
- Version control
- Consistency of 800+ separate files
- Debugging VBA code
Integrating Pension Experts

Herman Gerbscheid, Pension Architect:
• “This is the stuff I had to do mentally and keep consistent in my head all the time. It’s great to finally have tools for it.”

Suzanne Pront, Pension Expert:
• “Normally I know what I want, but don’t know how to tell engineers. Now I can do this myself. This is a revolution!”

Sybren den Hartog, Java Architect:
• “Now we can generate business rules and domain structure, which we could not do in UML based MDA.”
Summary

• Intentional Software is helping us to accelerate Pension Product innovation for our clients

• We were able to demonstrate a radical change in time to market and quality

• We used Pensions as a pilot, but we see many opportunities in other domains