Industrial Experience: building a Web Portal Product Line using a Lightweight, Reactive Approach

Ulf Pettersson
ST Electronics (Info-Software Systems) Pte. Ltd.
ulfp@stee.stengg.com

Overview

- Introduction
  - Challenges & Opportunities of Web Engineering
  - Product Lines
- Web Portal Product Journey
  - Conventional Starting Point - Portal Overview
  - Conventional Starting Point - Design
  - Reuse Goals and Benefits
- The XVCL Approach
  - Introduction to XVCL & Design of Web Portal Architecture
- Results and Experiences
- Conclusions
**Challenges of Web Engineering**

Web Applications play increasingly important role in businesses:

- Fuzzy, often changing requirements
- Tight development and maintenance schedules
- Growing complexity of Web Applications:
  - Web Sites → Web Portals → full-blown Business Enterprise Applications
- Web technologies change and multiply fast:
  - Constant evolution of web technologies (J2EE, .NET, etc)
  - Conventional methods and processes must be adjusted to meet the realities of the Web engineering

---

**Where are the opportunities?**

- Much Similarities across Web Applications
- Capturing and benefiting on similarities

- Web technologies supporting the above:
  - ASP, JSP, PHP
  - J2EE, .NET
  - Design patterns
  - General OO and component concepts
  - ...
  - XVCL
A Web Portal Product Line

A WP Product Line is a family of similar WPs

- Understand what’s similar and what’s different among WPs
- Manage WPs from a common base of reusable code (called a product line architecture)

Objectives of the PL approach

Apply reuse techniques in order to:

- Reduce the system development cost
- Reduce time to the market
- Simplify maintenance
- Expand range of products and address new market segments
Overview

- Introduction
  - Challenges & Opportunities of Web Engineering
  - Product Lines

- Web Portal Product Journey
  - Conventional Starting Point - Portal Overview
  - Conventional Starting Point - Design
  - Reuse Goals and Benefits

- The XVCL Approach
  - Introduction to XVCL & Design of Web Portal Architecture

- Results and Experiences
- Conclusions

Conventional Starting Point

- Conventional ASP Development
  - Personal Portal (home)
  - People Tracking (business product - SARS)

- Team Collaboration Portal (TCP) (office)
  - TCP Variants (office)

- XVCL-based development
  - Web Portal Product Line Architecture
  - New Portal Variants (generated)

Experiences and Results
Sj 5 indicate WPs
- used internally at ST Electronics
- business products (SARS)

Stan Jarzabek, 8/8/2005
Team Collaboration Portal - Overview

A Web Portal for supporting a Team & Project-based work environment.

Has elements of:

- Web portal
- Simple content management
- Simple workflow support
- Collaboration environment
Team Collaboration Portal – GUI

Technology Office Team Portal

TCP Entities
- Insert
- Update/Edit
- Delete
- Associate/Composite
- Query/List/Sort
- View
- Printer Friendly
- History/Versioning

Team Collaboration Portal – Modules

TCP

Profile Help
User (Admin) Access Control

Folder Feedback Forum Search Statistics
HTML Content Poll ToDo Action Log Session Log

“Standard Entities”
Managed by Admin
- Insert
- Update/Edit
- Delete
- Query/List/Sort
- View

“TCP Entities”
Managed by Portal Users
- Insert
- Update/Edit
- Delete
- Associate/Composite
- Query/List/Sort
- View

“Standard Features”
- Query/List/Sort
- View
Conventional Starting Point - Design

Top-level Portal Architecture:

Entity Module
Entity Module
Entity Module

Portal Foundation:
- Support of patterns
- Other conventional reuse...

Conventional Starting Point - Design

Entity Model Example:
Conventional Starting Point - Design

Meta-model related to Base Entity:

- **Base Entity**
  - **visualize**
  - **main**
  - **Entity View**
    - **Module Page**
- **Entity Controller**
  - **Edit Page**
  - **Display Page**
  - **List Page**
  - **Print Page**
  - **Post Page**

**Composition of views on a page:**

Conventional Starting Point - Design
**Conventional Starting Point - Design**

Composition of views on a page:

**Problems with conventional approach**

Pattern-rich approach

- Reuse of Entity Modules across portals
  - Difficult to manage selective changes and unify enhancements
    - Changes needed for a specific module but not for other modules
    - Same changes needed, but at different time

- Enhancement to patterns

- Changes in Portal Foundation result in Propagation Selection Problems

- Changes in Entity Module result in duplicated code in all Entity Modules
What we want to achieve

Reuse Goals

Some goals of pragmatic reuse:

- Unify similarity patterns with generic structures at all levels:
  - Unify similar portal modules with "generic module"
  - Unify other meta-model design elements:
    - Entity, Controller, Views, Pages, etc.
- Provide a mechanism to derive concrete instances from generic ones.
- Establish understanding of what is generic and what is specific.
**Reuse Goals**

Some goals of pragmatic reuse:

- Unify similarity patterns with generic structures at all levels:
  - unify similar portal modules with “generic module”

- Unify similar views across modules with “generic views”:
  - Edit View [M], List View [M], …
  - Also applicable to Entity, Controller, Pages, etc.

- Provide a mechanism to derive concrete instances from generic ones:
  - Edit View [M] -> Edit View [User], Edit View [ToDo], ..

- Design a “generic TCP”

- Simplification and engineering benefit of unification

---

**Expected benefits of a “generic TCP”**

- Conceptually simpler, smaller size

- Easier to trace the impact of changes
  - better traceability
  - less number of modifications to implement an enhancement
  - controlled change propagation from generic to concrete instances
  - reduced risk of update anomalies

- Uniformity of design within TCP

- Reuse of generic solutions
  - within TCP
  - across portal family

- Improved maintainability
  - unique variations should not pollute generic code
  - clear separation of differences from similarities
  - avoiding explosion of similar components and files
Overview

- Introduction
  - Challenges & Opportunities of Web Engineering
  - Product Lines
- Web Portal Product Journey
  - Conventional Starting Point – Portal Overview
  - Conventional Starting Point - Design
  - Reuse Goals and Benefits
- The XVCL Approach
  - Introduction to XVCL & Design of Web Portal Architecture
- Results and Experiences
- Conclusions

Concepts behind XVCL

- Explicit models of commonalities and variations
- Generic solutions, non-redundancy
- Enhanced changeability (adaptability)
  - improved visibility of changes
  - automated propagation of changes across all program parts affected by changes
- Integration of architecture with code
  - all product line assets: documents, test cases, etc.
**XVCL concepts**

*an x-frame*: a generic adaptable meta-component

*x-framework*: a library of *x-frames*

- a product line architecture

---

**Generation of WPs**

from the WP Architecture, we generate many different WPs:

**TCP:**

- User Module
  - View [User] page
  - Edit [User] page
    - Edit [User] View
  - ...

- ToDo Module
  - View [ToDo] page
  - Edit [ToDo] page
    - Edit [ToDo] View
  - ...
  - ...

- Module n
**an x-framework: WP Architecture**

![Diagram](attachment:image.png)

**An x-frame in XVCL**

A kind of a template

A generic building block for programs

- class, architectural element, interface def, ...

Contains parameter slots:

- values, expressions, selection, iteration, ...
- other meta-components

Program generation:

- “composition with adaptation” of meta-components
Generating TCP the WP Architecture

XVCL adaptation mechanisms

- Adaptation of x-frames at the composition point
- Generic names: variables and expressions
- Selection among many given options
- Insertions at break points
- Iteration (code generation)
**XVCL adaptation mechanisms**

- Adaptation of x-frames at the composition point
- Generic names: variables and expressions
- Selection among many given options
- Insertions at break points
- Iteration (code generation)

---

**Entity**

```
Class ?@Name?
<select InheritBase>
  <case "true"> 'code for inheritance
  <break> "BrkName">
    'default code
  </break>
</select>
```

---

**ToDo**

```
<adapt Entity>
  <set Name="ToDo">
    <set InheritBase = "True">
      <insert-after "BrkName"> 'inserted code
    </insert-after>
  </set>
</adapt>
```

---

**XVCL adaptation mechanisms**

- Adaptation of x-frames at the composition point
- Generic names: variables and expressions
- Selection among many given options
- Insertions at break points
- Iteration (code generation)

---

**Entity**

```
Class ?@Name?
<select InheritBase>
  <case "true"> 'code for inheritance
  <break>
    'default code
  </break>
</select>
```

---

**ToDo**

```
<adapt Entity>
  <set Name="ToDo">
    <set InheritBase = "True">
      <insert-after "BrkName"> 'inserted code
    </insert-after>
  </set>
</adapt>
```

---
**XVCL adaptation mechanisms**

- Adaptation of x-frames at the composition point
- Generic names: variables and expressions
- Selection among many given options
- **Insertions** at break points
- Iteration (code generation)

---

**X-frame illustration**

**x-frame Portal X**

```xml
<x-frame Portal X>
  <set Modules="ToDo, User, Folder, ChangeRequest"/>
  <while Modules>
    <adapt @Modules.xvcl/>
  </while>
</x-frame>
```

**x-frame ToDo**

```xml
<x-frame ToDo>
  ...<adapt DisplayView.xvcl>
  <insert Advertisement>
    - special advertisement code...
  </insert>
</adapt>
...<adapt ListView.xvcl>
  <insert FilterSettings>
    <set Filter="true"/>
    <set FilterAttr="Project, AssignedTo,..."/>
  </insert>
</adapt>
```

**x-frame DisplayView**

```xml
<x-frame DisplayView>
  ...<break Advertisement>
    - default advertisement code...
  </break>
  ...<ifdef Filter>
    - insert filter attributes code...
  </ifdef>
</x-frame>
```

**x-frame ListView**

```xml
<x-frame ListView>
  ...<break FilterSettings/>
  ...<ifdef Filter>
    - insert filter attributes code...
  </ifdef>
</x-frame>
```
Our Focus Area

Focus Area:
- Unify Patterns of similar design across Entity Modules

Examples of Entity Module Differences:
- Entity class properties and types
  - Different properties & property types
  - Data dictionary based properties
  - Inheritance through include (no inheritance in ASP)
- Associations
  - Same entity has different association relationships in different portals
- Business logic
  - Change Request workflow (different logic in different projects)
- User Interface characteristics
  - Display characteristics (coloring based on property values, etc)
  - Input type (text, dropdown, list popup, date calendar, etc)
  - Input validation
**Structure of WP Architecture**

- **Web Portals (meta level 1)**
  - User
  - Folder
  - ToDo
  - Change Request

- **Entity Modules (meta level 2)**
  - Entity
  - List Page
  - List View
  - Statistics
  - Edit Page
  - Edit View
  - Controller

- **Basic Module Features (meta level 3)**
  - Data Dictionary
  - Enumeration Pop-up
  - List Filter

- **Any Building Block (meta level 4)**

**Elements of conventional design in WP Architecture**

- **Web Portals (meta level 1)**
- **Entity Modules (meta level 2)**
- **Basic Module Features (meta level 3)**
- **Any Building Block (meta level 4)**

- **Adapts**
Elements of conventional design in WP Architecture

Conventional refinement of WP Architecture

Specialization of pages duplicated in all entity modules

Specialization of pages duplicated once in the generic module
Special refinement of WP Architecture

Change Request
(enhancements for large teams)
Specific attributes
Specific business logic

Change Request
(small teams)

Adapt

Entity Module Organization

Specific Portal n
Specific Portal A
Adapted
Specific
Entity Module
Entity Module
Entity Module
Entity Module

Common
Entity Module
Entity Module

Standard
Entity Module
Entity Module

Generic
Entity Module

Specific

Common

Standard & Generic
Runtime Perspective

Portal Construction from WP Architecture

Overview

- Introduction
  - Challenges & Opportunities of Web Engineering
  - Product Lines
- Web Portal Product Journey
  - Conventional Starting Point – Portal Overview
  - Conventional Starting Point - Design
  - Reuse Goals and Benefits
- The XVCL Approach
  - Introduction to XVCL & Design of Web Portal Architecture
- Results and Experiences
- Conclusions
**Results and Experiences (1)**

Code Lines before and after application of XVCL:

<table>
<thead>
<tr>
<th>Entity Module</th>
<th>Original TCP (ASP)</th>
<th>WP Architecture (XVCL)</th>
<th>Generated TCP (ASP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portal Foundation</td>
<td>15180</td>
<td>14021</td>
<td>16401</td>
</tr>
<tr>
<td>Specific Entity Modules</td>
<td>16322</td>
<td>1577</td>
<td>30474</td>
</tr>
<tr>
<td>Generic Entity Module</td>
<td>N/A</td>
<td>4119</td>
<td>N/A</td>
</tr>
<tr>
<td>Σ</td>
<td>31502</td>
<td>-37%</td>
<td>19717</td>
</tr>
</tbody>
</table>

Table 1

**Results and Experiences (2)**

Code Lines before and after application of XVCL:
- Entity Modules – *Our focus area for unification of similarities*

<table>
<thead>
<tr>
<th>Entity Module</th>
<th>Original TCP (ASP)</th>
<th>WP Architecture (XVCL)</th>
<th>Generated TCP (ASP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help*</td>
<td>722</td>
<td>133</td>
<td>1490</td>
</tr>
<tr>
<td>Web Page</td>
<td>514</td>
<td>50</td>
<td>1428</td>
</tr>
<tr>
<td>User</td>
<td>755</td>
<td>54</td>
<td>1717</td>
</tr>
<tr>
<td>. . .</td>
<td>. . .</td>
<td>. . .</td>
<td>. . .</td>
</tr>
<tr>
<td>ToDo (Task)</td>
<td>1295</td>
<td>147</td>
<td>2429</td>
</tr>
<tr>
<td>Σ</td>
<td>16322</td>
<td>-90%</td>
<td>1577</td>
</tr>
<tr>
<td>Generic Entity Module</td>
<td>NA</td>
<td>4119</td>
<td>NA</td>
</tr>
<tr>
<td>Σ</td>
<td>16322</td>
<td>-65%</td>
<td>5696</td>
</tr>
</tbody>
</table>

Table 2
Results and Experiences (3)

Code Lines – Portal Specific:

<table>
<thead>
<tr>
<th>Portal Abbrev.</th>
<th>Entity Modules</th>
<th>Portal specific XVCL**</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reused*</td>
<td>Adapted</td>
</tr>
<tr>
<td>Apol</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>Cq</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>Csap</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>Demo</td>
<td>18</td>
<td>4</td>
</tr>
<tr>
<td>Ecap</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>Feptp</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>Gered</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>Eses</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>Ework</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>Σ</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

* Common & Standard Modules
** Adapted & Specific Modules + other specific code

Table 3

Results and Experiences (4)

Code Lines – Portal Product Line:

<table>
<thead>
<tr>
<th>Item</th>
<th>XVCL</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portal Foundation</td>
<td>14021</td>
<td>(from Table 1 )</td>
</tr>
<tr>
<td>Generic Entity Module</td>
<td>4119</td>
<td>(from Table 1 )</td>
</tr>
<tr>
<td>Common Entities</td>
<td>2481</td>
<td>General Entities shared across WPs</td>
</tr>
<tr>
<td>Portal Specific Code</td>
<td>3953</td>
<td>(from Table 3)</td>
</tr>
<tr>
<td>Σ</td>
<td>24574</td>
<td>Total for 9 portals</td>
</tr>
</tbody>
</table>

Original Portal: 31502  WP Architecture (9 Portals): 24574

-22%
Results and Experiences (5)

Improvement areas:

- XVCL vs Integrated Development Environment (IDE)
  - No use of WYSIWYG generators
  - Developers first perception is primitive “feel”

- XVCL and debugging
  - Debugging of XVCL not possible (NUS working on tools)
  - Debugging of generated code is troublesome

- XVCL (XML) code mixed with native code
  - IDE setup can improve readability (keyword coloring, etc)

- No standards on how to approach XVCL development
  - Design and coding guidelines would be useful
  - Project Team organization and roles

Overview

- Introduction
  - Challenges & Opportunities of Web Engineering
  - Product Lines

- Web Portal Product Journey
  - Conventional Starting Point – Portal Overview
  - Conventional Starting Point - Design
  - Reuse Goals and Benefits

- The XVCL Approach
  - Introduction to XVCL & Design of Web Portal Architecture

- Results and Experiences

- Conclusions
Conclusions (1)

- Reactive Approach ➔ Low investment & Fast results
  - No systematic analysis before construction of WP Architecture
  - Incremental refinement and enhancement applied
  - Shorter than expected learning cycle for XVCL
    - 2 persons for 2 weeks to build first version of the WP Architecture

- Development Environment below “today's expectations”
  - No IDE integration, command line approach
  - Debugging difficult
  - Lack of design standards and representation

Conclusions (2)

- Significant Source Code Reduction ➔ Effort Savings
  - 22% reduction in code from original ASP portal to 9 XVCL portals
  - 65% reduction in code for Entity Modules
  - 90% reduction in code for NEW Entity Modules
  - 4-8 times reduction in effort for NEW Entity Modules
  - Less effort to maintain 9 portals in WP architecture than original 3

- Few redundancies and XVCL ➔ Ease of change
  - Easier to change
  - XVCL-approach effective in pattern-rich development
  - Few issues and little rework related to interface and other changes
  - Surprising ability to accommodate changes to foundation services, interfaces and patterns
    - Ease of evolving the ASP-XVCL WP in new directions