Combining Stepwise Feature Introduction with User-Centric Design

Heikki Anttila, Ralph-Johan Back, Pekka Ketola, Katja Konkka, Jyrki Leskelä, Erkki Rysä

Nokia Mobile Phones
Abo Akademi University and TUCS
Contents

• Introduction
• The Ladder Process
• Case: Teenage Girl Diary
• Evaluating the Ladder Process
• Related Work
• Conclusions
Introduction

- Context of the Study
- Existing Techniques
- Goals
Context of the Study

• The environment of software construction getting turbulent
  – User needs and technology changing unpredictably
  – Software is often an evolving artefact that needs continuous adaptation

• It is necessary to provide an architecture and process to make the software evolution possible

• Two recent techniques address the problem domain
  – Stepwise Feature Introduction (Back, 2002)
  – User-Centric Design (ISO13407, 1999)
Existing Techniques

- **Stepwise Feature Introduction**
  - Architecture for constructing software in very thin layers
    - Each layer introduces one new feature in the system
    - Each layer forms a complete application that can be tested against requirements conformance
    - The structure of layers is maintained during updates

- **User-Centric Design**
  - Iterative approach for concept and design creation
    - Understand users requirements and environment
    - Identify users' tasks
    - Define the success criteria for the product, per task
    - Incorporate HCI knowledge (visual/interaction/usability)
    - Produce design specification
    - Evaluate the design specification against success criteria
    - Repeat when the criteria are not met
Goals

• Combine Stepwise Feature Introduction and User-Centric Design into a new incremental software development process called the **Ladder Process**.

• Aim of the process:
  - The process is steered by continuous feedback from users
  - Faster and more flexible response to end-user needs
  - Thin increments feature by feature
  - Improves flexibility of SW architecture
  - Ease of SW maintenance
  - Increased reliability through incremental testing
  - Easy to make product variants
  - Better co-operation between teams

• Evaluate and refine the Ladder Process by studying a concrete case.
  - Teenage Girl Diary
  - For Nokia Communicator -like platform
The Ladder Process

Release 4
Release 3
Release 2
Release 1

Implementation  Specification
The Ladder Process

This release final implementation

Release specification

Software implementation process

Release implementation

Previous release implementation

Software specification process

This release final specification

Feature implementation  Feature specification
Feature corrections  Specification corrections
Automatic unit testing  Conformance testing
Refactoring  Usability testing
Stepwise Feature Introduction

Release 2
- edit text
- cut & paste
- styles

Release 1
- edit text

Refactoring (Fowler, 1999)
- Modifying the class structure
- Not adding functionality
- To improve design of existing code
User-Centric Design

Main activities of UCD as defined in ISO 12407

- identify need of human-centred design
- evaluate designs against requirements
- system meets specified functional, user & organisational requirements
- produce design solutions
- understand & specify the context of use
- specify the user & organisational requirements

User, Environment, Users tasks

Success criteria (for example how quickly user gets task done)

Incorporate HCI knowledge (visual/interaction design, usability)
Application of User Centric Design

- UCD works best in the very early development, like concept definition
- Continuous challenges in applying UCD in traditional sequenced SW development
  - Timetable pressures (separate design step not always possible)
  - Confidentiality limitations (involving real users)
  - Increasing complexity of systems (a lot of SW designed and implemented one-shot)
  - Complex and evolving user requirements
- Opportunity for Stepwise Feature Introduction
  - Design iteration easier with thin layers
  - Easier to coordinate future release plan based on user feedback
  - Iteration between releases is natural activity
  - System complexity grows in small steps
  - User requirements can be checked between each release
Long-Term Planning of the Releases

- Simple editor specified
- Super editor drafted
- Simple editor implemented
- Better editor specified
- Super editor drafted
The Release Specification Sub-Process

Release n implementation process

Release specification

Release n specification process

Implementation usability tests and analysis

Specification compliance check

Requirements

Use cases

Task analysis

Use scenarios

New feature proposals

New feature proposals from sources

Start: user needs studies, feature proposals

Document the activity

Compose and analyse requirements

Turn requirements to concrete UI specs

Compose stories, user scenarios

Analyse the steps

Understand the user needs for the release

Turn requirements to concrete UI specs

4.4.2002 / Jyrki Leskelä
The Release Implementation Sub-Process

Release n implementation

- Releasing
- Unit testing
- Integration
- Implementation
- Refactoring
- Layer design
- Release specification

Spikes

Release n+1 implementation process

Release n implementation process

- Release n specification process

Release n-1 implementation process

Add new layer with subclassing
Re-factor e.g. when duplicated code
When support needed to lower layers
- Inject ancestor classes as helper layer
- Minimum changes as last resort
The Integrated Process

Release n+1 implementation process
- Releasing
- Unit testing
- Integration
- Implementation
- Layer design

Release n specification process
- Implementation
- Usability tests and analysis
- Specification compliance check
- UI specification
- Requirements
- Use cases
- Task analysis
- Use scenarios

New feature proposals
- Release n specification process
- Release n+1 specification process

Release n-1 implementation process
- Refactoring

Release n implementation process
- New feature proposals
- Release n+1 implementation process
- Release n specification process
- Release n-1 implementation process
Case: Teenage Girl Diary

Always with you

For teenage girls

School timetable

Calendar

School terms

Colourful

Attaching pictures

Attaching notes

Connectivity

Multimedia

Diary
Running the Specification Process
## Long-term planning

<table>
<thead>
<tr>
<th>To-Do View</th>
<th>Image View</th>
<th>Image View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term View</td>
<td>Term View</td>
<td>Term View</td>
</tr>
<tr>
<td>Term View</td>
<td>Week View</td>
<td>Week View</td>
</tr>
</tbody>
</table>
Stories / User Scenarios

Scenario: “Helmi, 12 years, wants to add a photo from a scout camp she went last summer as a background image to school term 2, which is the current school term. In addition, she likes to have a small photo of her friend Pekka on the current week, because during that week is Pekka’s birthday.”
Tasks Analysis

**Steps**

· Helmi has her smart diary open on the current week.
· She activates the term part of the week view (specification 2d).
· She selects add background image from menu/Add background image.
· She selects the camp photo from the files and, clicks ok (hypothetically the devices has file structure, and the image is already modified to be suitable as a background image).
· The camp image appears as a background image to the current term.
· Helmi activates the diary part of the week view of her smart diary from soft key.
· She selects add image from menu/Add image
· She selects Pekka’s photo from the files and clicks ok.
· Pekka’s photo appears to the week view as a small movable object.
**Use cases**

<table>
<thead>
<tr>
<th><strong>USE CASE 4</strong></th>
<th>Adding images to diary week view or term view (R2/5).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goal in Context</strong></td>
<td>User can add background image or small image/s to term view or diary week view.</td>
</tr>
<tr>
<td><strong>Scope &amp; Level</strong></td>
<td>Secondary task ?</td>
</tr>
<tr>
<td><strong>Preconditions</strong></td>
<td>User has the smart diary open on the diary week view or on the term week view depending on to which view s/he is adding an image/images. User has images saved to the device (hypothetically the device has file structure, and images are already modified to be suitable).</td>
</tr>
<tr>
<td><strong>Success End Condition</strong></td>
<td>A user can add background image or small image/s to the selected view.</td>
</tr>
<tr>
<td><strong>Failed End Condition</strong></td>
<td>A user can’t add background image or small image/s to the selected view.</td>
</tr>
<tr>
<td><strong>Primary, Secondary Actors</strong></td>
<td>A user.</td>
</tr>
<tr>
<td><strong>Trigger</strong></td>
<td>User wants to personalize/decorate the device.</td>
</tr>
</tbody>
</table>
## Use cases (continued)

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A user has the smart diary open on the current week.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>User activates the term view (R2/5).</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>User selects add background image from menu.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>User selects a photo from the files and, clicks ok</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>The photo appears as a background image to the current term.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>User selects add image from menu.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>User selects a photo from the files and, clicks ok.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Photo appears to the week view as small movable object.</td>
<td></td>
</tr>
</tbody>
</table>

### EXTENSIONS

<table>
<thead>
<tr>
<th>Step</th>
<th>Branching Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td></td>
</tr>
</tbody>
</table>

### SUB-VARIATIONS

<table>
<thead>
<tr>
<th>Branching Action</th>
</tr>
</thead>
</table>
R2.Req.4. Add and remove images

Operations on images:

- Removing images from term view.
- Removing images from diary week view.
- Select image file.
- Select background/photo (file).
- Activate image (scroll-on).
R2.UIspec.4: Adding and removing images

Add background image

User can set a background image to selected term or week view. Each view (term, week) can have max. 1 image.

The image is added with Menu/Add Image/, with selection from submenu To Background. When this menu command is activated, a list of available images is given.
Running the Implementation Process
User Interface Implementations

Image View

Term View

Week View
Main Refactorings

• Change from Python/TKinter to Java/Swing
  – Need to work in familiar language and GUI
  – Java more realistic example for Nokia Mobile Phones

• Splitting the original Week Layer into a simplified Week Layer and Term Layer
  – Improving structure of the software

• Introducing an auxiliary Image Helper Layer
  – Simplify the introduction of images to the diary model

• Refactoring parallel development into strictly sequential layers
  – Java supports only single inheritance
  – With multiple inheritance, parallel feature introduction would have been possible
Conclusions
Pros and Cons

• Pros
  – Process is simple to follow
  – Parallel user testing
  – Reduced risk
  – Layered structure makes the overall architecture clear

• Cons
  – Resulting code may be difficult to understand (due to inheritance)
  – Refactorings can be potentially quite large
  – Not optimised for embedded SW

• Improvements
  – Extreme programming in the implementation process
  – Support for distributed development
## Goals Achieved?

<table>
<thead>
<tr>
<th>Goal</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>The process is steered by continuous feedback from users</td>
<td>Excellent</td>
</tr>
<tr>
<td>Faster and more flexible response to end-user needs</td>
<td>Excellent</td>
</tr>
<tr>
<td>Thin increments feature by feature</td>
<td>Excellent</td>
</tr>
<tr>
<td>Improves flexibility of SW architecture</td>
<td>Excellent</td>
</tr>
<tr>
<td>Ease of SW maintenance</td>
<td>Not Tested</td>
</tr>
<tr>
<td>Increased reliability through incremental testing</td>
<td>Good</td>
</tr>
<tr>
<td>Easy to make product variants</td>
<td>Not Tested</td>
</tr>
<tr>
<td>Better co-operation between teams</td>
<td>Excellent</td>
</tr>
</tbody>
</table>
Future Research

• Integrating Ladder Process with Extreme Programming.
• More thorough case study how Ladder Process works for constructing software products for mobile terminals.
• Study the construction of product variants in Ladders. Possibilities for open source development.
• Deeper study of testing, verification and validation of the specification and implementation.
Related work

- Extreme programming (Beck, 2000)
  - short iteration cycles, planning game
  - code as the main asset \textit{De-emphasises careful documentation and design}
  - striving for simplicity by avoiding planning far in future
  - frequent automatic testing and integration, tests first
  - Refactor duplicate code: \textit{Not much more guidance how to structure the code.}
  - on-site customer: \textit{has much of the specification responsibility... that activity is not defined clearly}

- Aspect-oriented programming (Miller, 2001)
  - Method for combining features but weaving them to the SW structure rather than clear layering.

- Layers in general common in SW systems
  - Stepwise feature introduction uses very thin layers, each layer introduces small increase of functionality
References


References


Appendix 1

- Experiences from the Diary Specification
  - Data is mainly platform independent
  - Platform elements not defined in the process (for example how common menu functionality works)
    - Guessing things in the implementation
    - Corrected by selecting 9210 as reference UI
  - There was a trade off when cursor keys were very costly to implement as pointing device for Swing UI
Appendix 2

• Experiences from the Diary Implementation
  – Week Layer implementation
    • Basic application structure
    • Separate tracer and tester classes for application model
  – Term Layer implementation
    • First experiences on layering
    • Model and view were extended significantly, dialogs added
    • Run-time flexibility of GUI (Swing) was needed
    • Data structures of layers kept separate when possible
  – Menu layer implementation
    • Menu was added as extension layer for the view
    • The UI components installed in lower layers slightly moved
  – Image layer implementation
    • New layer derived from view to add and remove images
    • Injecting image storage support into existing data elements with helper layer at the bottom (model not derived)
    • Entirely new classes such as image file selection