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

NOKIA

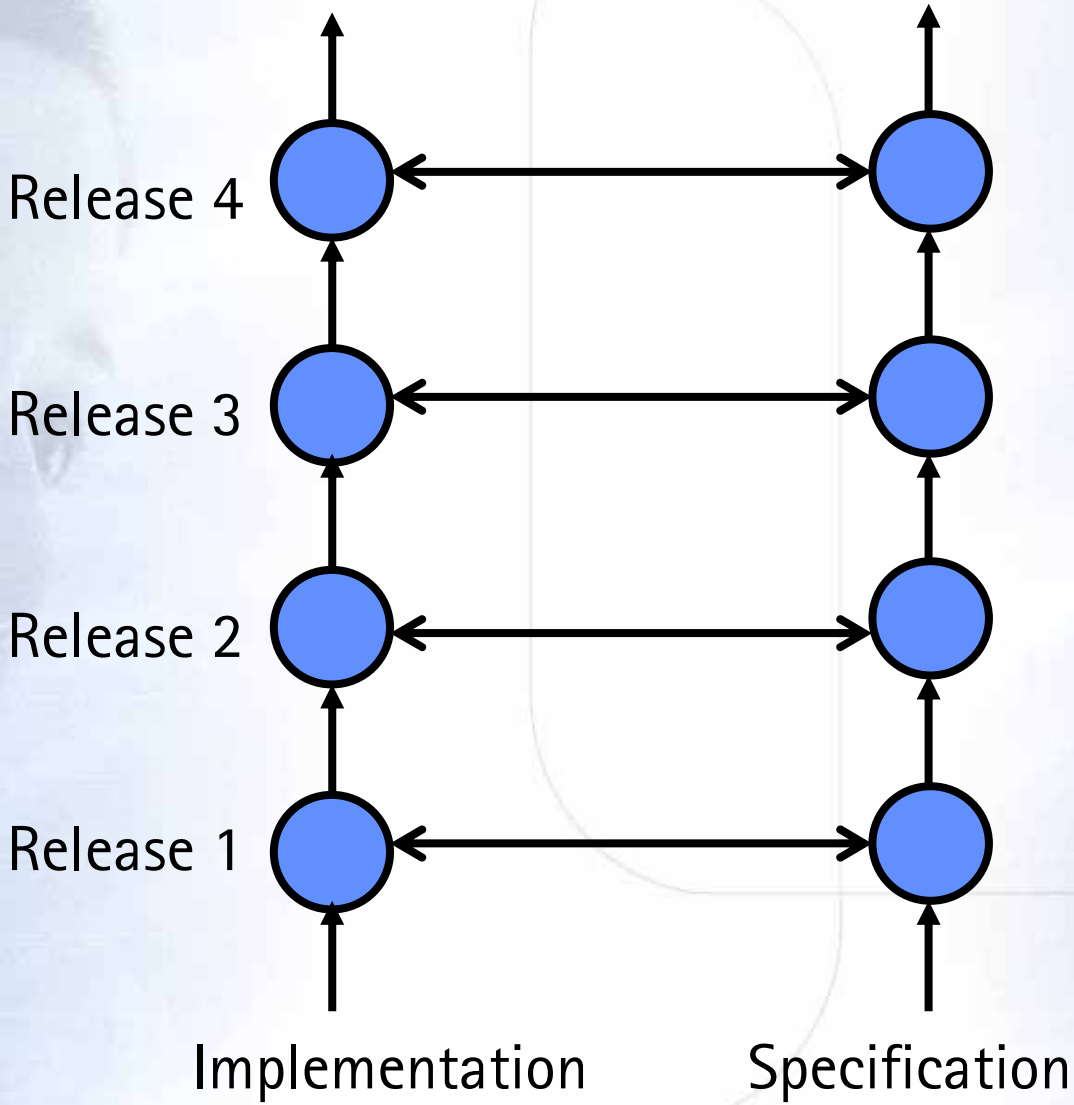
Existing Techniques

- 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

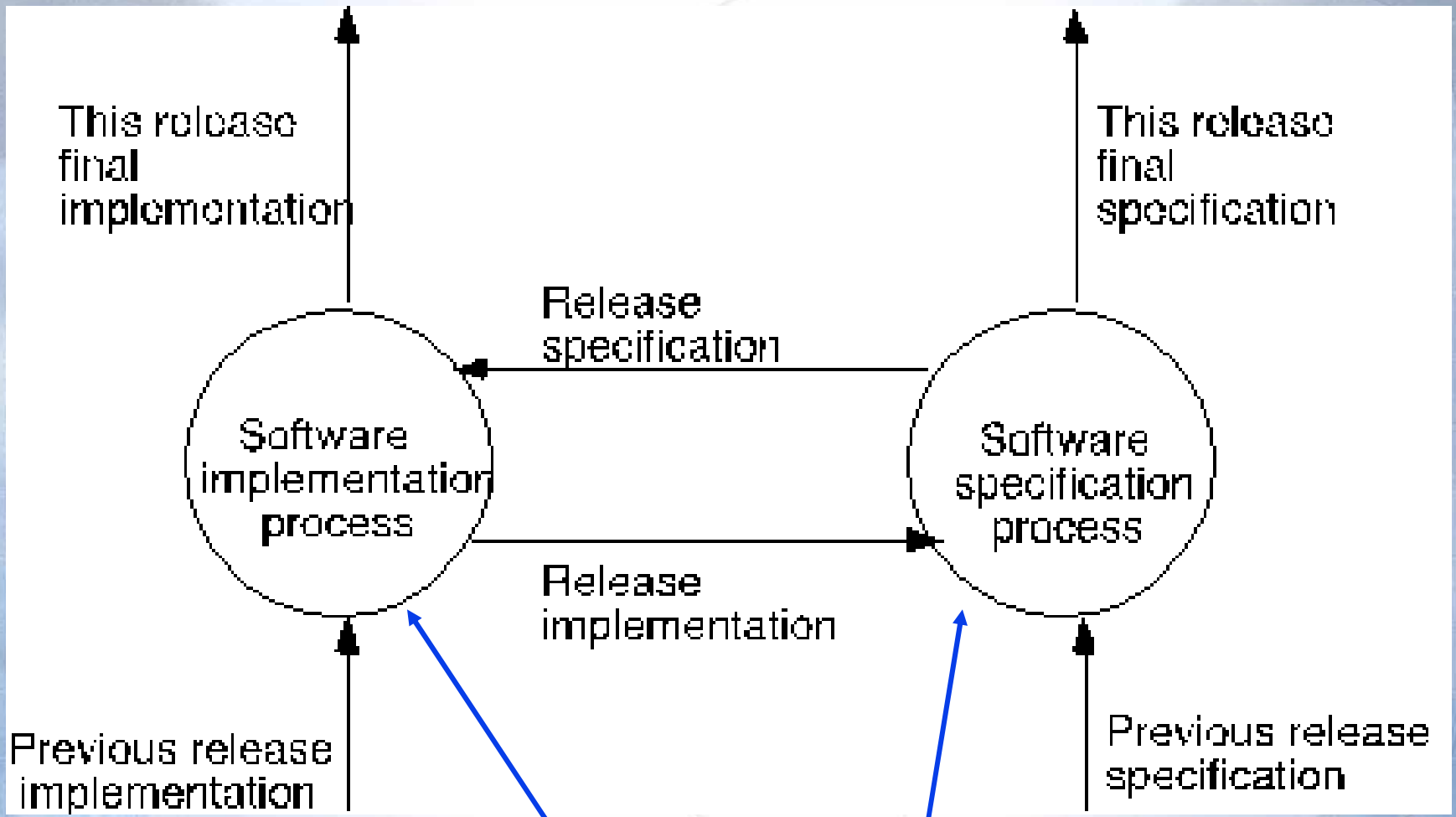
Goals

- 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



The Ladder Process



Feature implementation
Feature corrections
Automatic unit testing
Refactoring

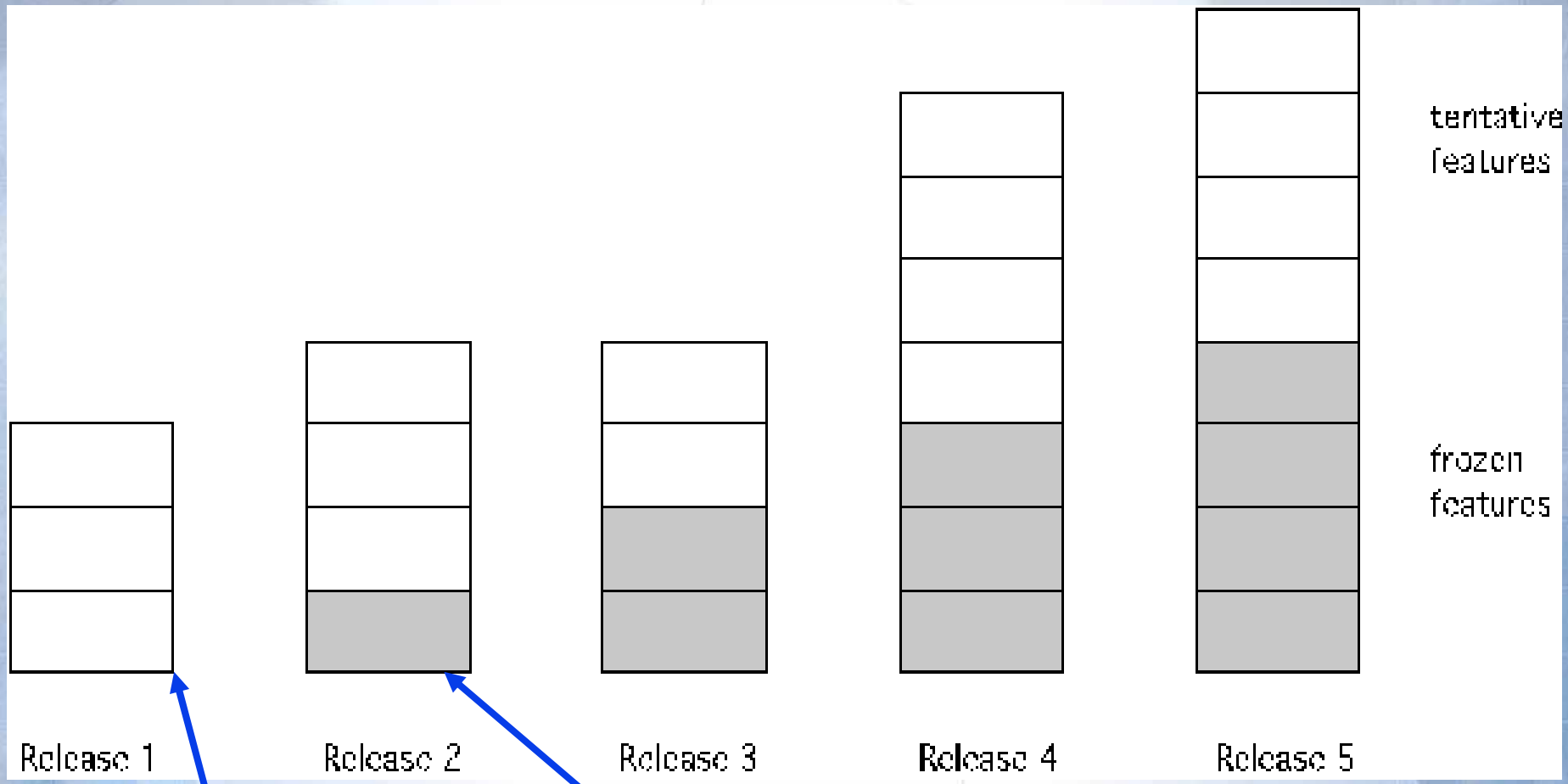
Feature specification
Specification corrections
Conformance testing
Usability testing

NOKIA

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

Case: Teenage Girl Diary

Always with you

For teenage girls

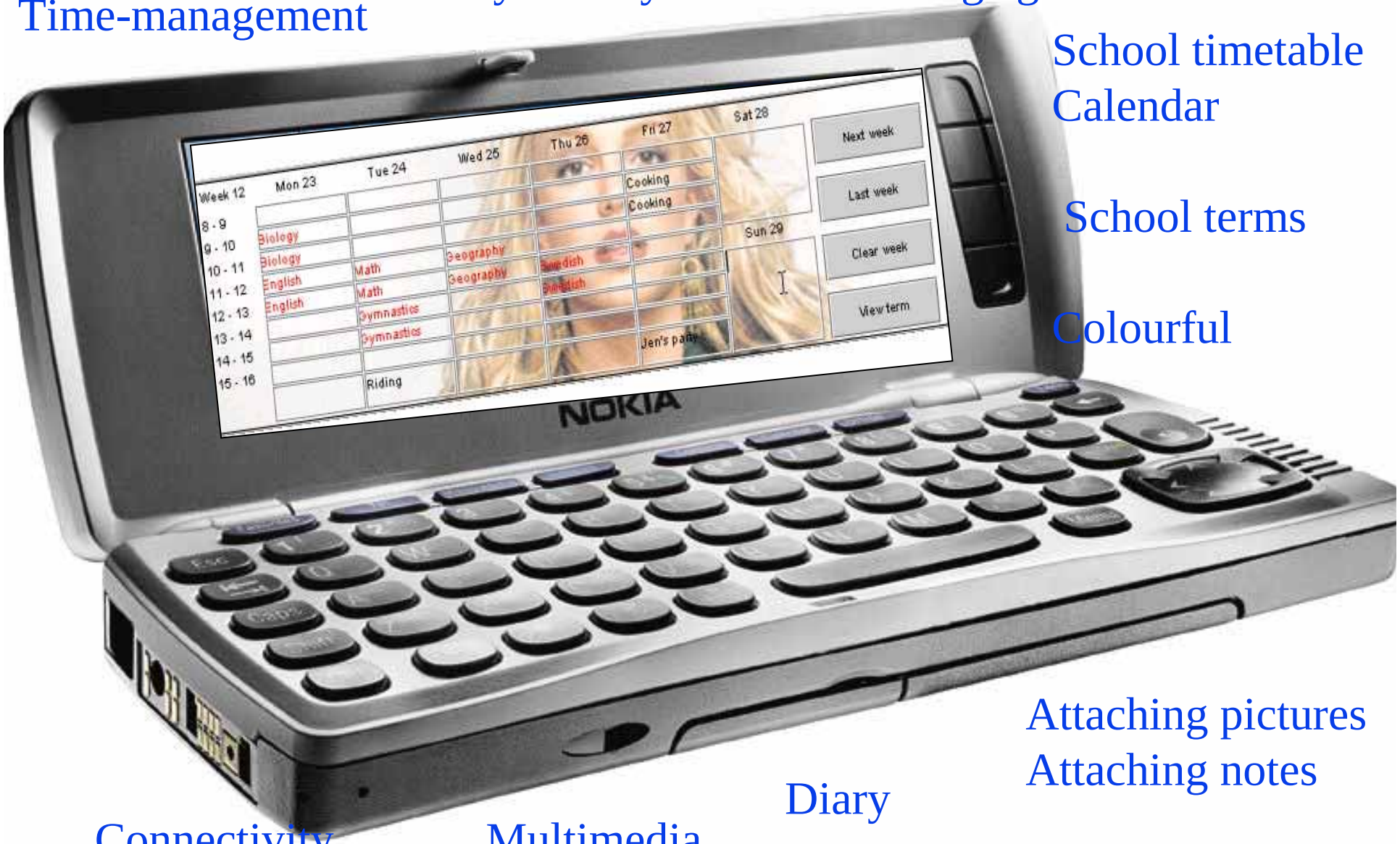
Time-management

School timetable

Calendar

School terms

Colourful



Attaching pictures

Attaching notes

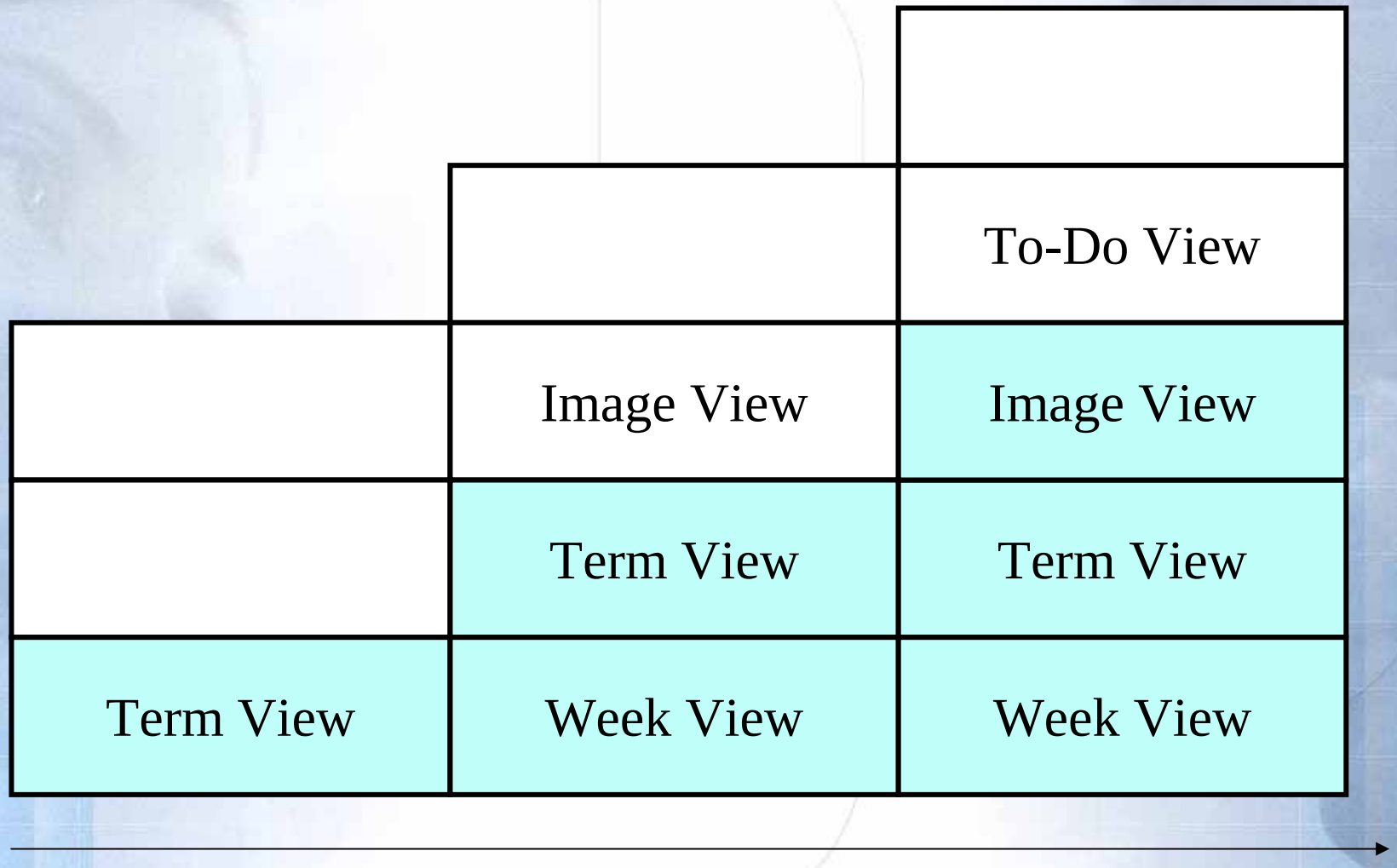
Diary

Connectivity

Multimedia

Running the Specification Process

Long-term planning



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 ke
- 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

USE CASE 4	Adding images to diary week view or term view (R2/5).
Goal in Context	User can add background image or small image/s to term view or diary week view.
Scope & Level	Secondary task ?
Preconditions	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).
Success End Condition	A user can add background image or small image/s to the selected vie
Failed End Condition	A user can't add background image or small image/s to the selected vi
Primary, Secondary Actors	A user.
Trigger	User wants to personalize/decorate the device.

Use cases (continued)

DESCRIPTION	Step	Action
	1	A user has the smart diary open on the current week.
	2	User activates the term view (R2/5).
	3	User selects add background image from menu.
	4	User selects a photo from the files and, clicks ok
	5	The photo appears as a background image to the current term User activates the diary week part of the smart diary (R2/5).
	6	User selects add image from menu. User selects a photo from the files and, clicks ok.
	7	Photo appears to the week view as small movable object.
EXTENSIONS	Step	Branching Action
	1a	
SUB-VARIATIONS		Branching Action

Requirements

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).

UI Specifications

R2.UIspec.4: Adding and removing images

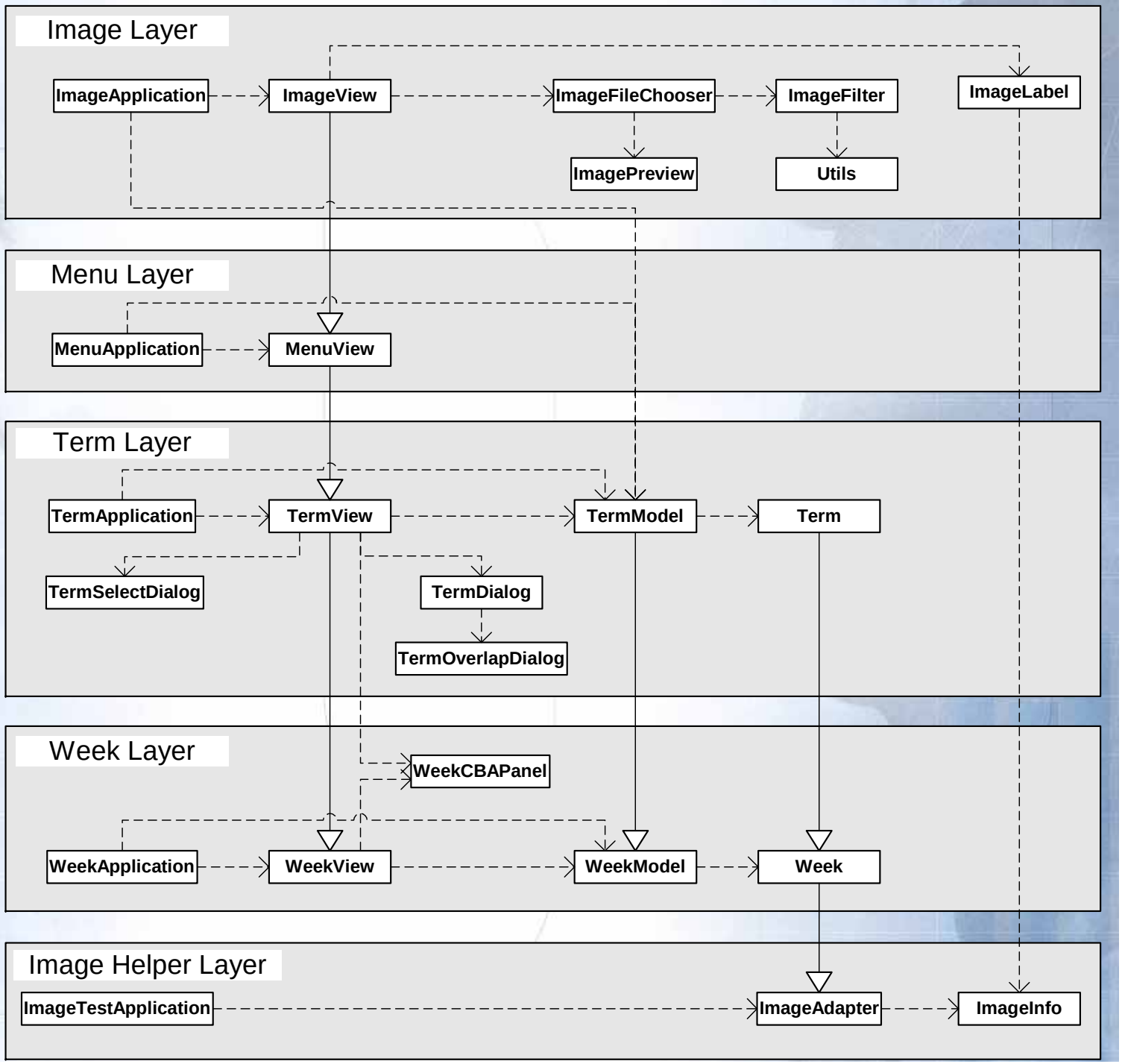
Add background image

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

Implementation Layers

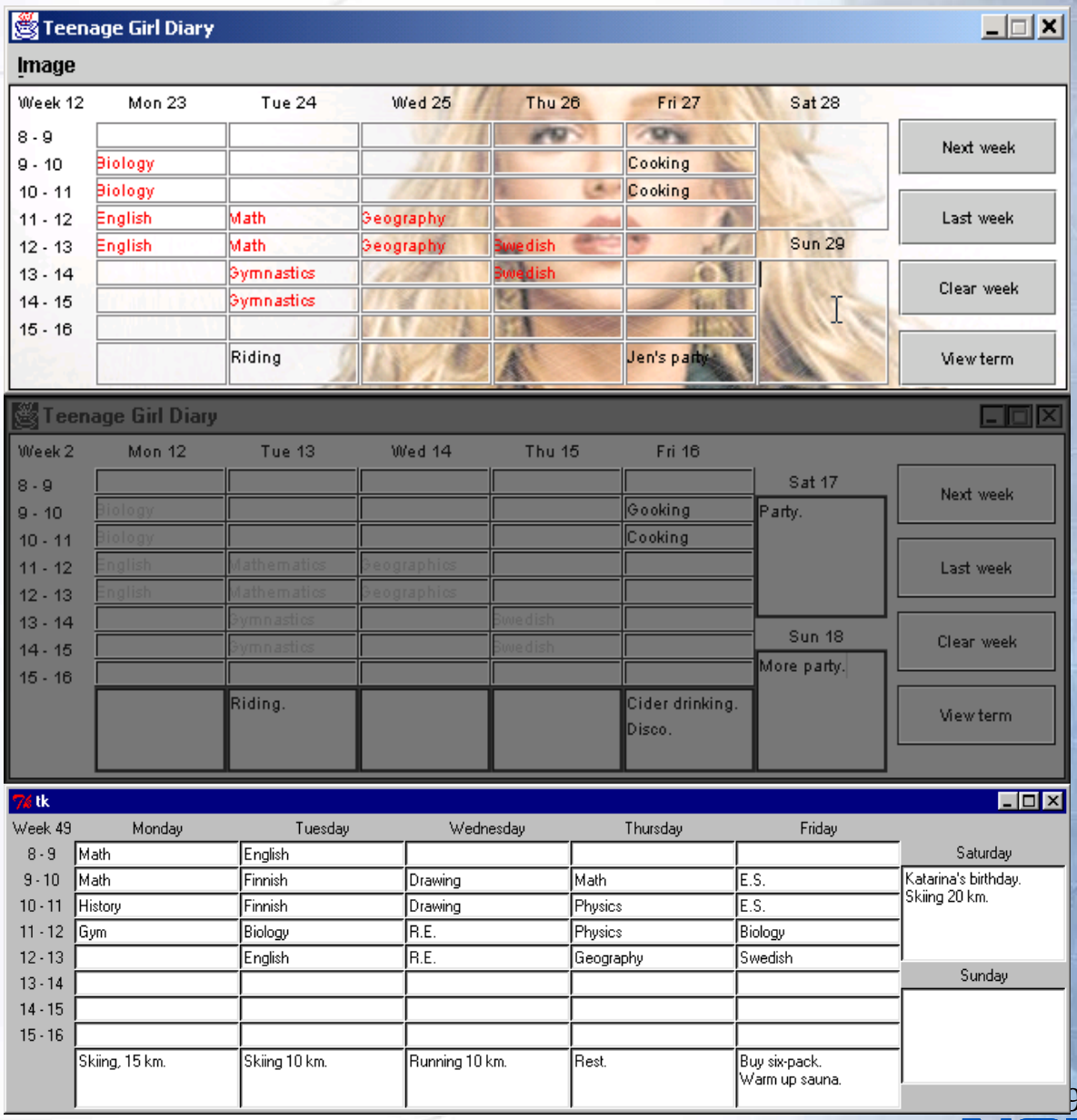


User Interface Implementations

Image View

Term View

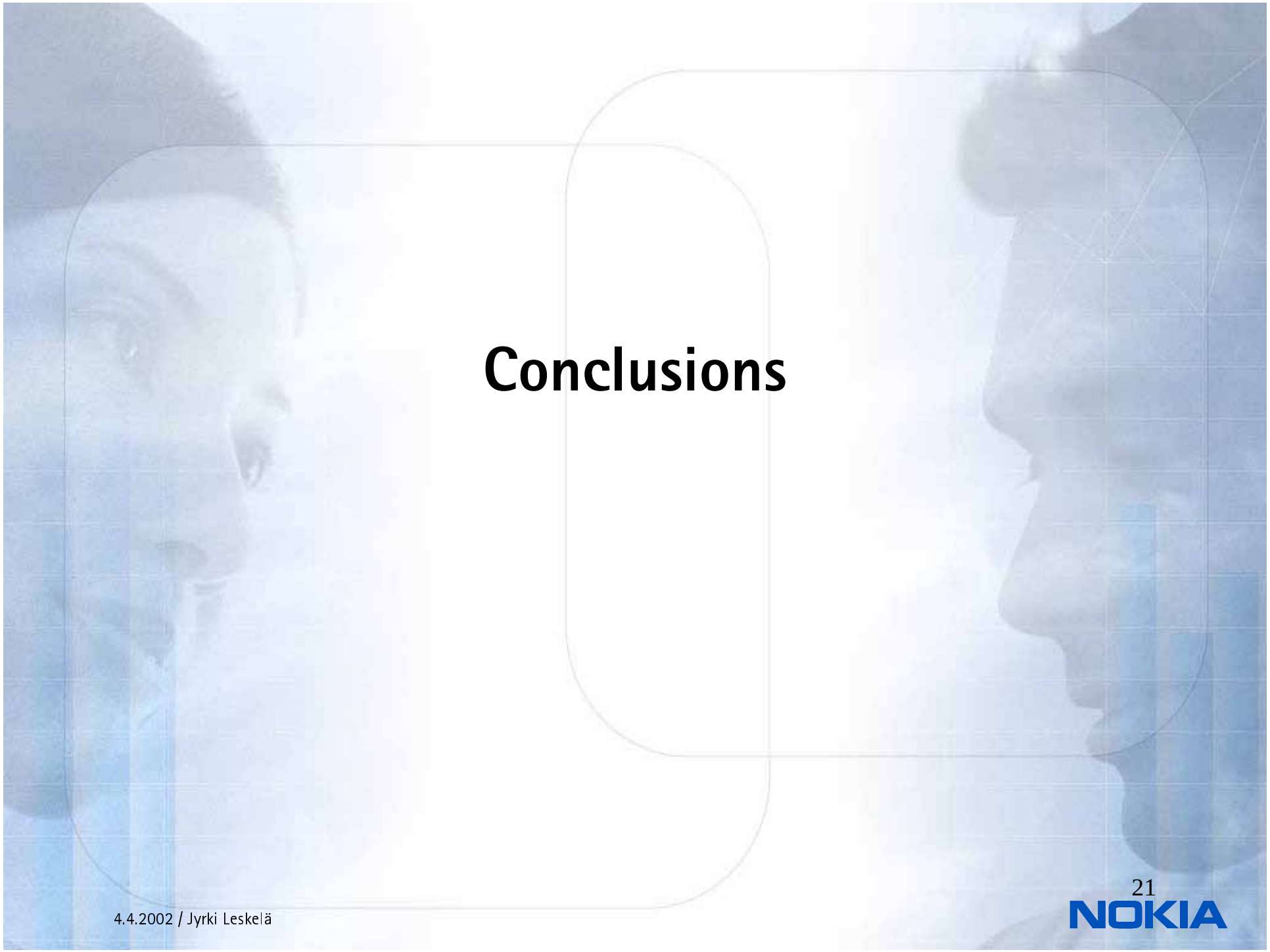
Week View



NOKIA

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 ?

The process is steered by continuous feedback from users	EXCELLENT
Faster and more flexible response to end-user needs	EXCELLENT
Thin increments feature by feature	EXCELLENT
Improves flexibility of SW architecture	EXCELLENT
Ease of SW maintenance	NOT TESTED
Increased reliability through incremental testing	GOOD
Easy to make product variants	NOT TESTED
Better co-operation between teams	EXCELLENT

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 *De-emphasises careful documentation and design*
  - striving for simplicity by avoidin planning far in future
  - frequent autimatic testing and integration, tests first
  - Refactor duplicate code: *Not much more quidance how to structure the code.*
  - on-site customer: *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

- [1] Back, R.J.R.: Software Costruction by Stepwise Feature Introduction. In *ZB2002: Formal Specification and Development.* In Z and B (eds. D. Bert, J. Bowen, M. Henson and K. Robinsons), pp. 162-183. Springer Lecture Notes in Computer Science 2272, January 2002.

- [2] Back, R. J. R, L. Milovanov, I. Porres and V. Preoteasa: XP as a Framework for Practical Software Engineering Experiments. To be presented at the Third International Conference on eXtreme Programming and Agile Processes in Software Engineering, May 2002, in Alghero, Sardinia, Italy

- [3] Beck, K.: *Extreme Programming explained*. Addison Wesley 1999.

- [4] Cockburn, A.: *Writing effective use cases*. Addison-Wesley 2000.

References

- [5] Fowler, M.: *Refactoring: Improving the Design of Existing Code*. Addison Wesley, 1999.
- [6] ISO13407:. *Human-centred Design Processes for Interactive Systems.* International Organisation for Standardization, Genève, Switzerland. 1999.
- [7] Kasesniemi E-L. and Rautiainen P.: *Kännyssä piilevät sanomat (Embedded messages in mobile phones, In Finnish)*. Tampere University Press. Tampere, Finland, 2001
- [8] Miller, S.K.: Aspect-oriented Programming Takes Aim at Software Complexity, *Computer*, April 2001.

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

Contents

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

Introduction

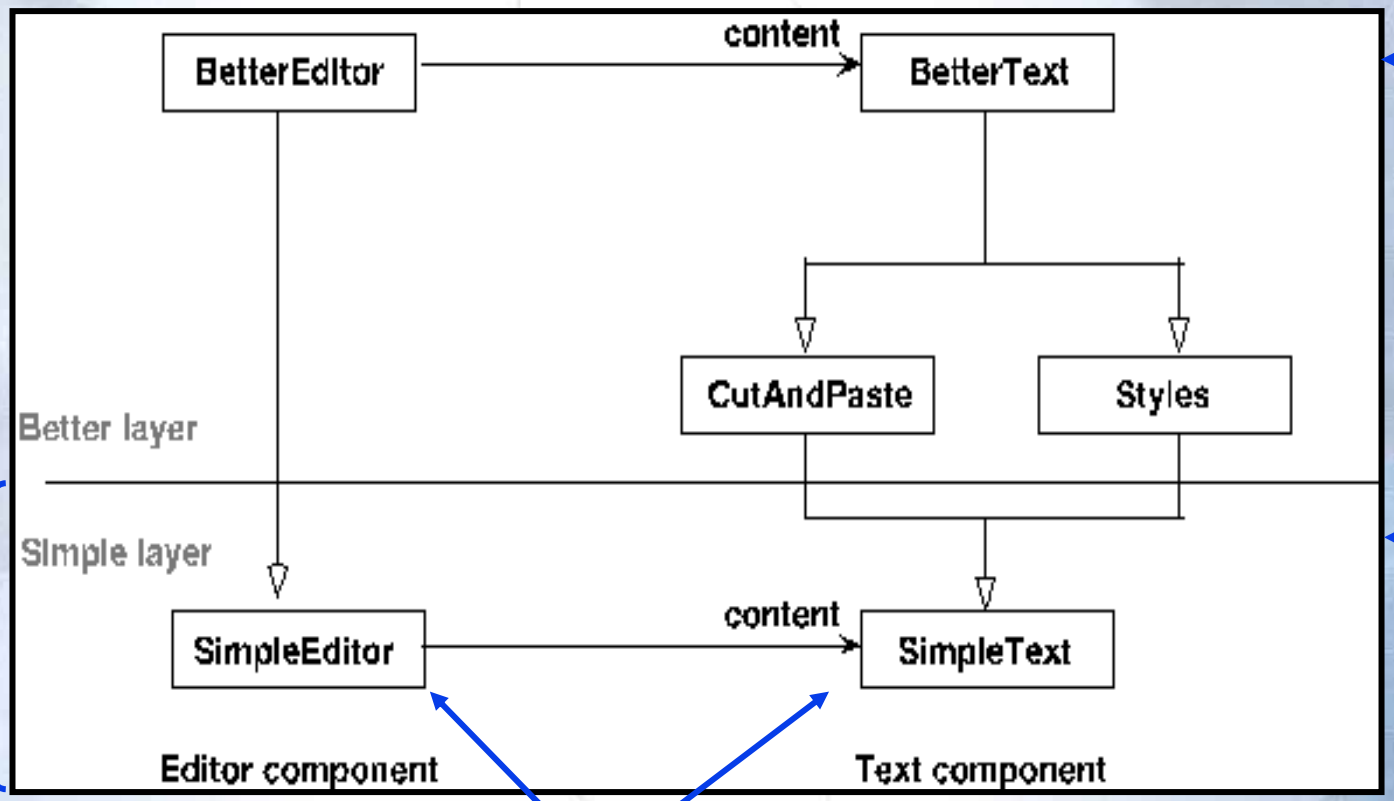
- Context of the Study
- Existing Techniques
- Goals

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)

Stepwise Feature Introduction

Planning



Release 2
- edit text
- cut & paste
- styles

Release 1
- edit text

Test

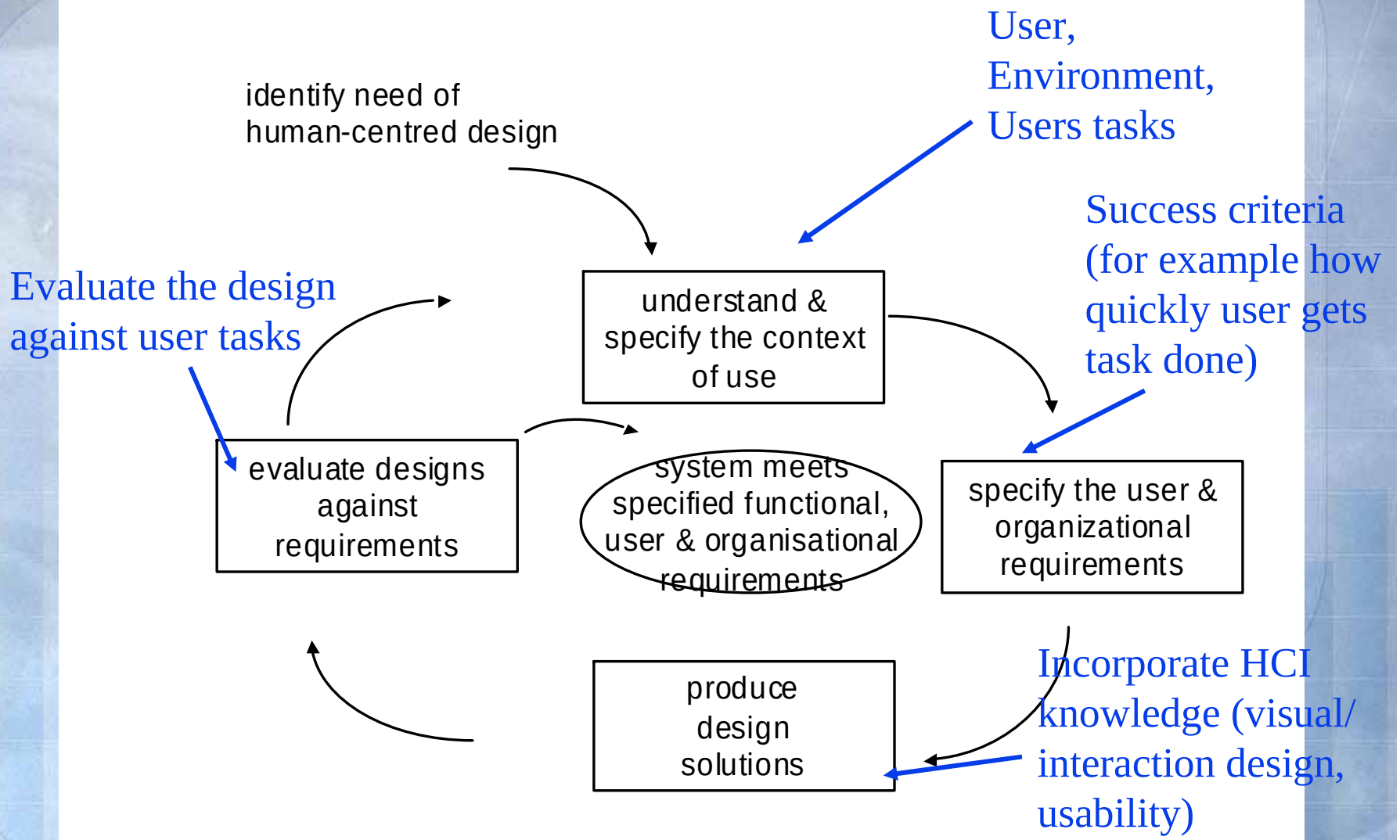
Test

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

NOKIA

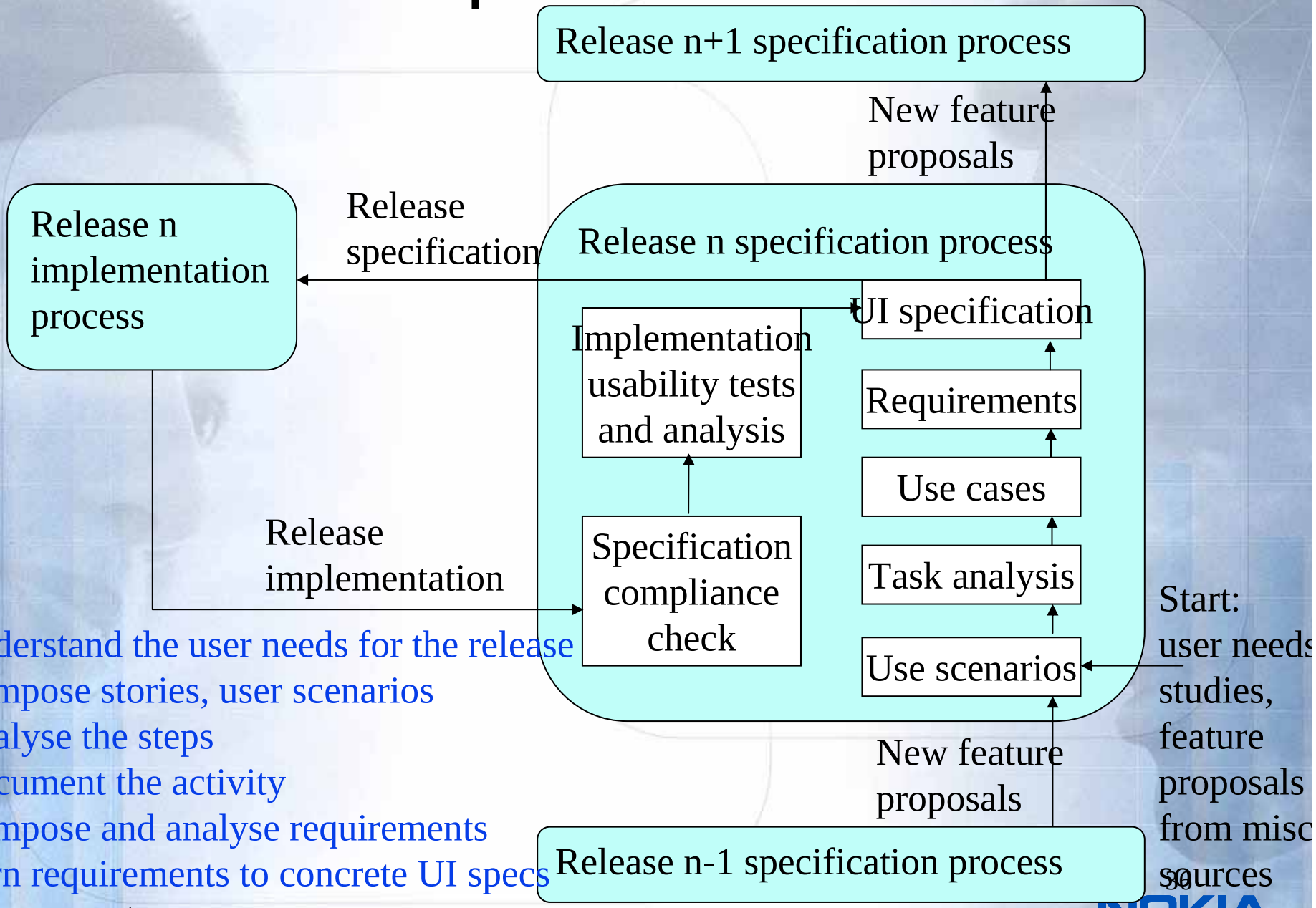
User-Centric Design

Main activities of UCD as defined in ISO 12407



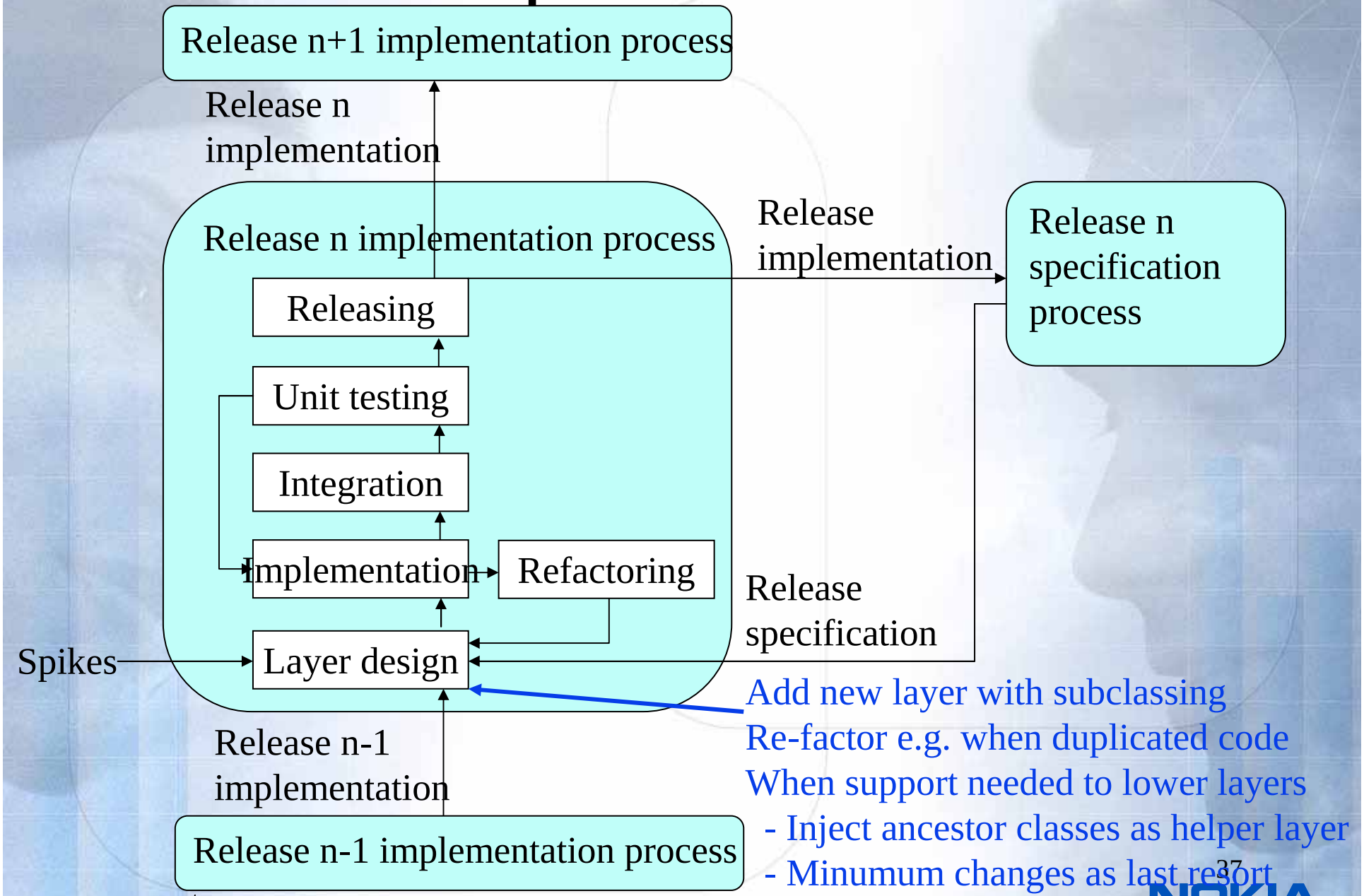
NOKIA

The Release Specification Sub-Process

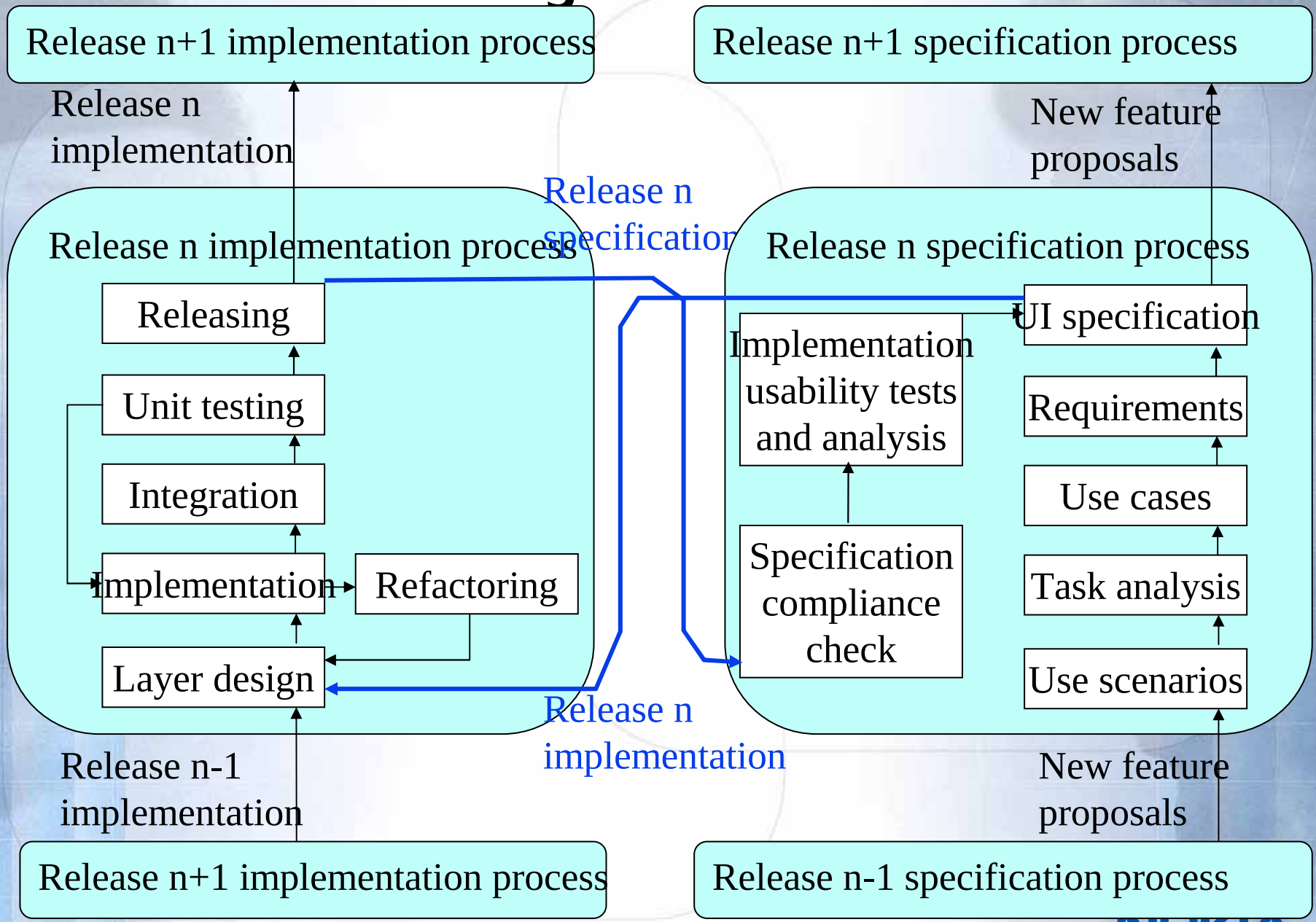


Understand the user needs for the release
Compose stories, user scenarios
Analyse the steps
Document the activity
Compose and analyse requirements
Turn requirements to concrete UI specs

The Release Implementation Sub-Process



The Integrated Process



NOKIA