A whiteboard for Saros

To develop a whiteboard functionality for distributed pair programming, iterative by prototypes

Michael Jurke
Institut für Informatik
Fu Berlin
25. Februar 2010

Overview

Freie Universität Berlin

I. Goals
II. Prototyping
III. Possible whiteboard requirements
IV. Drawing in Java and Eclipse
V. Requirements for a distributed whiteboard
VI. Test-driven development
VII. Concept and timetable (abstract)

Goals

Freie Universität Berlin

- Experiences with prototyping
- Develop a working whiteboard
  - Determine its requirements
    - By end user feedback
    - Compare and synchronize with scientific literature
- If possible complete unittests
  - How does test-driven development apply in this scope

Prototyping

Freie Universität Berlin

- Creating of incomplete software (parts)
  - to allow users to evaluate proposals
  - to show developers the needs
    - Maybe only by GUIs
      - Keyword UX-Design (User-experience Design)
      - Proprietary programmes like iRise, Axure, VisualPrototyper …
  - *Literature: The Mythical Man-Month: Essays on Software Engineering, Fred Brooks*
- Iteratively extending them by user feedback

Prototyping

Freie Universität Berlin

- Very up to date
  - some argue it should be used all the time
  - very effective for on-line systems, especially transaction processing
- Most beneficial for programmes having a lot of human-computer interactions
  - fits for developing a whiteboard

Prototyping steps

source: *English wikipedia*

Freie Universität Berlin

1. Identify basic requirement
   - desired in- and output
2. Develop Initial Prototype
   - if possible only user interface
3. Review
   - Customers examine prototype, feedback on additions and changes
4. Revise and Enhance the Prototype
   - improve specification and implementation
   - after changes, repeat step 3 and 4

Prototyping – types (selection)

Freie Universität Berlin

- Throwaway prototyping
  - developing of usually discarded snippets
    - quick feedback by users
    - user-testable interfaces
- Evolutionary prototyping
  - robust prototype in a structured manner
  - continuously refined
    - flexible, working basis until the final product
- Explorative prototyping
  - estimate of certain solutions
  - determine whether specifications and ideas are suitable

Prototyping – in the context of Saros

Freie Universität Berlin

- The whiteboard depends on Saros
  - No Throwaway prototyping
- It is quite probable that only students will test
  - we may not get a balanced feedback

- An evolutionary and explorative approach
  - to estimate some pre-defined possible requirements for a DPP whiteboard
  - to find new “real” ones by testing students

Prototyping – in the context of Saros

Freie Universität Berlin

- It is quite probable that only students will test

→ to boost the amount and diversity of testing participant we might decouple Saros from Eclipse (optional)
  - distribute it as RCP or standalone Jabber client

- Advantage: A lot of testers for Saros session, network, chat and whiteboard
- Disadvantage: Most prototype users won't be programmers

Get possible DPP whiteboard requirements

Freie Universität Berlin

Overview

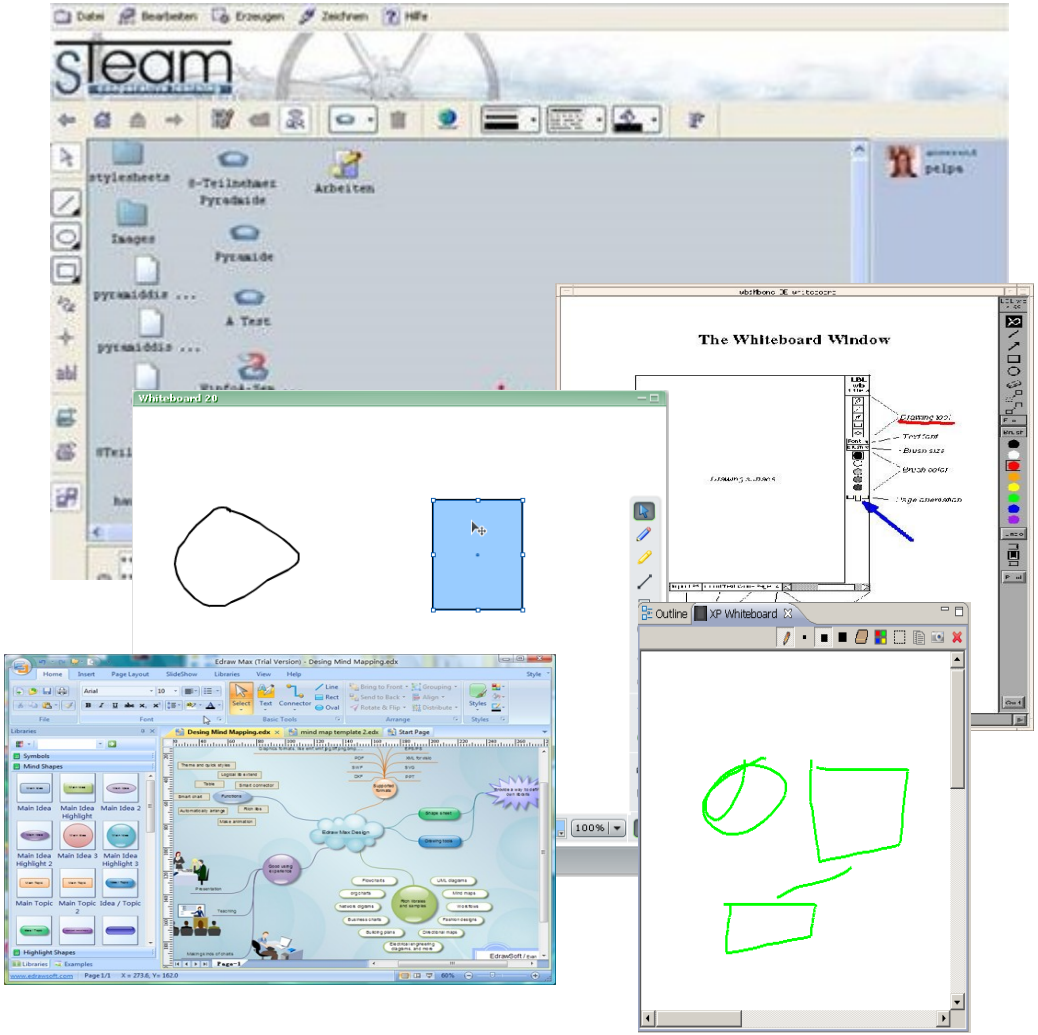
- existing whiteboards and their features
- why does DPP need a whiteboard
- and why might someone not be very likely to use one (like me)
  - which features would help
- a dream DPP whiteboard

State of the art
Existing distributed whiteboards

Freie Universität Berlin

- There are thousands

  sTeam, Adobe Connect Pro, MBONE, Inkscape, XPairtise, Drawboard, Groupboard, Edraw Mind Map …



What do they offer?

Freie Universität Berlin

- pencil: always
- usually standard shapes like line, rectangle and circle
- others
  - selection (Adobe Connect Pro, sTeam, Inkscape)
  - text
  - half transparent text marker (Adobe)
  - Mind Maps (Edraw Mind Map and others)
  - connectors (Inkscape)
  - desktop-like drawable home area (sTeam)
  - Screen shots, export (i.e. to JPEG or SVG)

Existing distributed whiteboards

Freie Universität Berlin

- Problem: why not use an existing on?
  - apart of Saros-integration
- Better question: what advantages has a whiteboard specialized on distributed programming?

Why does DPP need a whiteboard?

Freie Universität Berlin

- Giving overviews graphically
- Explaining abstract ideas
- Informality
  - *”What designers need are computerized tools that allow them to sketch rough design ideas quickly“*
    Landay, J.A., Myers, B.A. Interactive Sketching for the Early Stages of User Interface Design (1995)
    - Abstract and formal methods do not always fit to improve communication
      - may make it difficult to communicate with end users
    - ideas are faster to sketch than to specify formally

Why might a programmer not like to use a whiteboard ?

Freie Universität Berlin

- Difficult to draw a pencil with the mouse
  - Most users do not have a pen pad
- It may be clumsy to modify, extend or scale
- It may take to long to draw what you want to express
  - Classes, inheritance, concurrency drawn by pencil and rectangles
- The drawn sketches may not be used again
  - Hardly for documentation: good looking sketches are time intensive
  - Not for code generation (related work, see [traetteberg, 2002])
  - the area is quickly used off

Basic requirements – whiteboard for programming

Freie Universität Berlin

- Standard tools and shapes

  - It may be clumsy to modify and extend
  - → selecting, modifying and removing of element

  - The drawn sketches may not be used again
  - → Save and reload

- (requirements to be verified by the explorative prototyping)

Basic requirements – whiteboard for programming

Freie Universität Berlin

- Difficult to draw a pencil with the mouse and It may take to long to draw

→ connectors and templates for commonly used UML-like shapes (optional)

- the area is quickly used of

→ various pages, zooming (optional)

▶ (requirements to be verified by the explorative prototyping)

Dream whiteboard

Freie Universität Berlin

- A dream DPP whiteboard for me would be able to:
  - recognize patterns (like E-Chalk)
  - have a possibility to introduce semantic, i.e. Domain Specific Language
  - enable layouting
  - be able to generate elements by drag and drop (i.e. UML-like class shapes from package explorer)
  - be personalisable and extendible
  - …
- More an UML and Model-Drive-Architecture helper, however, it leads us to GEF and GMF

Drawing (and modelling) in Java and Eclipse

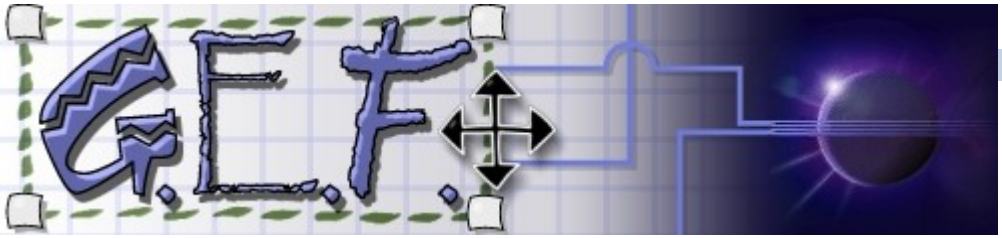
Freie Universität Berlin

Overview

- GEF and GMF
- Pixel based
- SVG (batik, SVG Salamander)

Graphical Editing Framework

Freie Universität Berlin

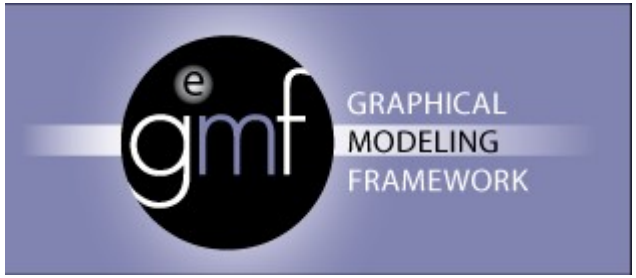


- helper Framework to edit models graphically
- based on Draw2d
  - SWT based Java2D compliant
- offers infrastructure to deal with shapes (figures)
  - strict MVC infrastructure
  - layouts and coordinate system
  - edit policies and commands

Graphical Modelling Framework

Freie Universität Berlin

- based on GEF and EMFs Ecore (Eclipse Modeling Framework)
- may automatically generate graphical editors and DSLs by XML-mappings



- has in my experience a too specialized scope

GEF and GMF

Freie Universität Berlin

- Both focus more on formal model editing
- To improve developing they require quite a big scope of usage (a real graphical editor)
  - and quite some knowledge

- No option for Saros' whiteboard
  - especially as it should improve communication by informal sketches

Pixel based

Freie Universität Berlin

- Like XPairtise's whiteboard or the SWT paint example
- Disadvantages:
  - No selection, moving, modifying of elements
  - difficult to synchronize
    - sending of whole repaint regions?
    - sync. execution on the server (might have a heavy delay)

Scalable Vector Graphics

Freie Universität Berlin

- open XML standard for 2D vector graphics
  - under w3c development since 1999
- supports
  - vector- and raster graphics as well as text
  - gouping, styling, modifying and composing of elements
  - animations, JavaScript and CSS
- renderable and dynamical modifyable in Java by
  - Batik
  - SVG Salamander
- extendible (i.e. introducing new tags, namespaces etc.)

- Best SVG 1.1 implementation after Opera
- generate XML by drawing
  - maybe not used in our case
- display an XML
- handle selection by SVG listeners
- Swing based
  - need to use an embedded Frame for eclipse
- after writing some test snippets it works fine for me within eclipse

Requirements 2

Freie Universität Berlin

- What requirements may be given by distributed drawing?
  - same picture for every peer
  - fast drawing
  - informing if any other user is drawing
  - blocking of elements that are in modification (i.e. text edit)

    (not complete yet)

Requirements 2 – distributed whiteboard

Freie Universität Berlin

- usage of SVG would be an advantage
  - text-based
    - smaller amount of transferred data
    - easier to synchronize
  - standard conform ...
    - there are several SVG + XMPP approaches to study about in this thesis, however, not yet standardised
      - http://xmpp.org/extensions/inbox/whiteboard2.html
      - could we make our whiteboard compatible with TransVerse? (XMPP+SVG whiteboard, initiators of proposal above)

Test-driven development

Freie Universität Berlin

1) A test case is written
   - to define desired improvement
2) Run all tests and the new one will fail
   - if not, the feature is already implemented or the test case is wrong!

- Note: I do not distinguish to test-first approach as only recently they may be seen as different approaches. Test-first is said to be more likely part of extreme programming

Test-driven Development

Freie Universität Berlin

3) Write the code to make the test work

- don't write more than necessary
  - would result in lack of test cases

4) Rerun the tests

- should pass now

5) Refactor the code

… and repeat all steps

Test-diven development

Freie Universität Berlin

- Not (yet) interesting for this work in the beginning
  - Not very suitable for user interfaces
    - because full success or failure is required
    - a lot of non testable functionalities (i.e. flicker)
      - Note: Writing snippets, I spend almost all my time on this
  - or when depending on external libraries
    - a lot of mock objects are needed
  - or when working distributed
- In this cases it might be an interesting study for experienced test-driven developers

Test-diven development

Freie Universität Berlin

- … continued
  - Not very suitable as Saros is not implemented test-driven
    - test coverage gaps lead to false confidence
    - the test cases won't help to localise problems there
  - fits better for programming in groups or pairs (best agile programming)
- However, in the latter state of this work when receiving iteratively new requirements it might be interesting to apply TDD on a new (separated) part of the whiteboard
    - how test-driven development does apply in the conditions mentioned above (optional)

Concept and timetable (abstract)

Freie Universität Berlin

- March
  - 1st three weeks
    - extending snippet to first prototype
      - Tools (pencil and common shapes) and Toolbar
      - Saros integration
    - reading about SVG+XMPP
  - 4th and 5th week
    - consolidate class structure
    - writing test cases and bug search
    - How to find test users
      - find appropriate lectures for Saros exercises; ask professors/tutors
      - ask for volunteers using studi-replies@math.fu-berlin.de

Concept and timetable

Freie Universität Berlin

- April
  - 1st week
    - if necesary, finishing left over work
    - creating questionnaires and exercises
  - rest of April
    - with semester start some students should be using Saros
      - helping installing and using Saros
      - getting feedback and evaluate questionnaires
    - searching and comparing literature
    - further planning, maybe TDD

Concept and timetable

Freie Universität Berlin

- Mai
  - continue iterative prototyping and refining
  - if there is time to implement and evaluate optional requirements
  - extract a standalone Saros version (optional)
    - or at least finding out necessary steps
  - reading, reading, reading
- June
  - start writing thesis
- Note: tight calculation, but there is a month left at the end

Questions and comments

Freie Universität Berlin

Viele Dank!

Sources

Freie Universität Berlin

english and german wikipedia
google search (especially for whiteboards and distributed whiteboards)
www.w3c.org
www.w3.org/Graphics/SVG/
www.eclipse.org/gef/
www.eclipse.org/gmf/
Thesis: Model-based User Interface Design, Traetteberg, Norwegian University of Science and Technology, 2002
Work on TDD by Bettina Selig
http://www.inf.fu-berlin.de/inst/ag-se/teaching/S-AGILE-2004/ausarbeitungen/Bettina_Selig_TDD-Ausarbeitung.pdf
http://xmpp.org/extensions/inbox/whiteboard2.html
http://xmlgraphics.apache.org/batik/
https://svgsalamander.dev.java.net/
various whiteboard sites and whiteboard examples