Scalable Vector Graphics to the eXtreme
Documenting software in Extreme Programming

By
Abraham Rodriguez Mota. abraham@dcs.shef.ac.uk
Anthony J. Cowling. a.cowling@dcs.shef.ac.uk

The Verification and Testing Lab
Department of Computer Science
The University of Sheffield, UK
Outline:

- Objectives
- The software development process
  - The waterfall model
  - The XP methodology
- The role of pictures in design
  - The Unified Modeling Language
- The research approach
- The technologies
  - The Eclipse Framework
  - SVG
  - Apache- Batik
- Combining the technologies: the eXtreme Programming Designing (XPD) tool
- Conclusions
Objectives:

- Analyze the nature and possible products of the design activity for an Extreme Programming approach to software development.
  - Emphasis on artifacts for documenting, communicating, maintenance and evolution of software products.
  - Proposals for how development tools might integrate these different artifacts.
- Produce light-weight software documentation.
  - For developers following an Extreme Programming (XP) approach.
  - To keep in synchrony with code easily.
Software development process:

Software development is possible through the execution of a set of development activities or phases (the software development process or the software development life cycle).

- The waterfall model
  - Requirements analysis
  - System analysis
  - Design
  - Implementation
  - Testing
  - Deployment
  - Maintenance
Software development process: Extreme Programming

Is a light-weight methodology for small-to-medium-sized teams developing software where requirements are vague or change rapidly

- The XP approach
  - Communication
  - Simplicity
  - Feedback
  - Courage
Software development process: Extreme Programming

Major areas of practice in XP

- The planning game
- Releases
- Metaphor
- Simple design
- Testing
- Refactoring
- Pair programming
- Collective ownership
- Continuous integration
- 40-hour week
- On-site customers
- Coding standards
Software development process: Extreme Programming

XP is distinguished from other methodologies by

- Early, concrete and continuing feedback
- Incremental planning approach
- Flexible scheduling of the implementation
- Reliance on automated tests
- Reliance on oral communication
- Reliance on an evolutionary design
- Reliance on the close collaboration
- Reliance on practices
The role of pictures in design

“My objection is not to pictures, but to trying to keep multiple forms of the same information synchronized” (Kent Beck).

(XP) The pictures are not saved

If there is a kind of source code that is best expressed as a picture, then the picture will be created, edited and maintained.

(General) Trouble drawing the pictures can give developers subtle clues about the health of a design.

Speed. In the time that a developer would require to code one design, he/she can compare and contrast various designs using pictures.
The Unified Modeling Language (UML)

A family of graphical notations backed by a single meta-model

- Covers a large and diverse set of application domains
- Is helpful for describing and designing software systems, particularly object-oriented (OO) systems
- Has almost become the de-facto standard language for OO modeling
- Is the main tool used by the participants in our control group

Class Diagrams: Sketch and Blue Print mode
The research approach

A tool that produces light-weight documentation

- How can light-weight documents be created?
- How can they be kept in synchrony with the code?

Test the tool

- Abstraction levels
- Which features are more valued by developers following the Extreme Programming approach?
- How could developers benefit from this approach?

Proposals for how development tools might integrate different artifacts.
The selected technologies

SVG

The Eclipse Framework
- Easy integration of software development tools
- Platform for creating applications and tools
- Java language development capabilities
- Java language analysis capabilities
- Out-of-the-box functionality

The Scalable Vector Graphics representation
- Text based / easy editable
- Light-weight, searchable, zoomable, structured
- Interactivity, linking, user-trigger events and animation events
- Portable

Apache-Java technology for processing and displaying SVG
- Developed in java
- Compliance with the SVG standard
The Eclipse Framework

Java Development Tooling (JDT)

Is a set of plug-ins that allow users to write, compile, test, debug, and edit programs written in the Java programming language.

- **JDT Core** - the infrastructure for compiling and manipulating Java code.
- **JDT UI** - the user interface extension that provides the IDE that developers see.
- **JDT Debug** - program launching and debug support specific to the Java programming language.

**Abstract Syntax Tree (AST)**

- **org.eclipse.jdt.core** - defines the classes that describe the Java model.
- **org.eclipse.jdt.core.compiler** - defines an API for the compiler infrastructure.
- **org.eclipse.jdt.core.dom** - supports Abstract Syntax Trees (AST) that can be used for modeling the structure of a compilation unit down to the statement level.
- **org.eclipse.jdt.core.eval** - supports the evaluation of code snippets in a scrapbook or inside the debugger.
- **org.eclipse.jdt.core.jdom** - supports a Java Document Object Model (DOM) that can be used for walking the AST structure of a Java compilation unit.
- **org.eclipse.jdt.core.search** - supports searching the workspace's Java model for Java elements that match a particular description.
- **org.eclipse.jdt.core.util** - provides utility classes for manipulating the files of compiled code and associated Java model elements.

Source: Eclipse JDT's documentation
The Eclipse Framework

Java Development Tooling (JDT)

Source:Eclipse JDT’s documentation

org.eclipse.jdt.core
The Apache-Batik API

**Core Modules**

- **SVG DOM** an implementation of the SVG DOM API defined in the SVG recommendation. It lets the programmer manipulate SVG documents in a Java program.

- The **Swing components** module includes, primarily, the JSVGCanvas class, a UI component that can display SVG content and let the user interact with that content (zoom, pan, rotate, select text, etc.).
Combining the technologies: the eXtreme Programming Design (XPD) tool
Combining the technologies: the eXtreme Programming Design (XPD) tool
Combining the technologies: the eXtreme Programming Design (XPD) tool

A user creates a class

Eclipse \rightarrow Wizard

Compilation Unit

XPD

SVG

<svg>
  <background>
    <relations>
      <content>
        <g id="class_name">
          <text display="none">[class path of the class]
          <g id="class_name.classBox">
            <text> [classHead]
              <text> [stereotype]
              <text> [class_name]
              <g id="class_name.fields">
                <text> [field's signature]
              </g>
              <g id="class_name.methods">
                <text> [method's signature]
              </g>
              <g id="class_name.relations">
                <text> [relation's id]
              </g>
          </g>
        </g>
      </content>
    </relations>
  </background>
</svg>
Combining the technologies: the eXtreme Programming Design (XPD) tool

A user modifies a class

```
<svg>
  <background>
    <relations>
      <content>
        <g id=[class_name]>
          <text display="none">[class path of the class]
          <g id=[class_name.classBox]>
            <rect>
            <rect>
          </g>
          <g id=[class_name.classHead]>
            <text> [stereotype]
            <text> [class_name]
          </g>
          <g id=[class_name.fields]>
            <text> [field's signature]
          </g>
          <g id=[class_name.methods]>
            <text> [method's signature]
          </g>
          <g id=[class_name.relations]>
            <text> [relation's id]
          </g>
        </g>
      </content>
    </relations>
  </background>
</svg>
```
Combining the technologies: the eXtreme Programming Design (XPD) tool

A user modifies a class
Conclusions

- Improves actors experience
  - Actors involved in the process learn and communicate in different ways, and therefore may require in a different extend more detailed representations or multiple representations of the project.
  - It should be easier for users to think about features of the system being developed at the most appropriate level of abstraction, rather than always having to work at the lowest level of detail.
  - The developed SVG representation can resemble the file system structure as well as the OO code structure with no further changes.
  - Code comprehension by filtering the information being displayed on the diagram and facilitating its storage and display in different formats and media definitions (abstract representation of the code from which general metrics can be obtained).
  - Facilitate software design but helps to apply good design practices and techniques during the whole software development process.

- In the agile area we expect that the proposed data representation can be a gateway for the introduction, or a more extensive use, of other technologies such as audio and animation as documentation artifacts.
THANKS