

# Bridging Javadoc and design documentation using

## UML diagram image maps as spatial menus

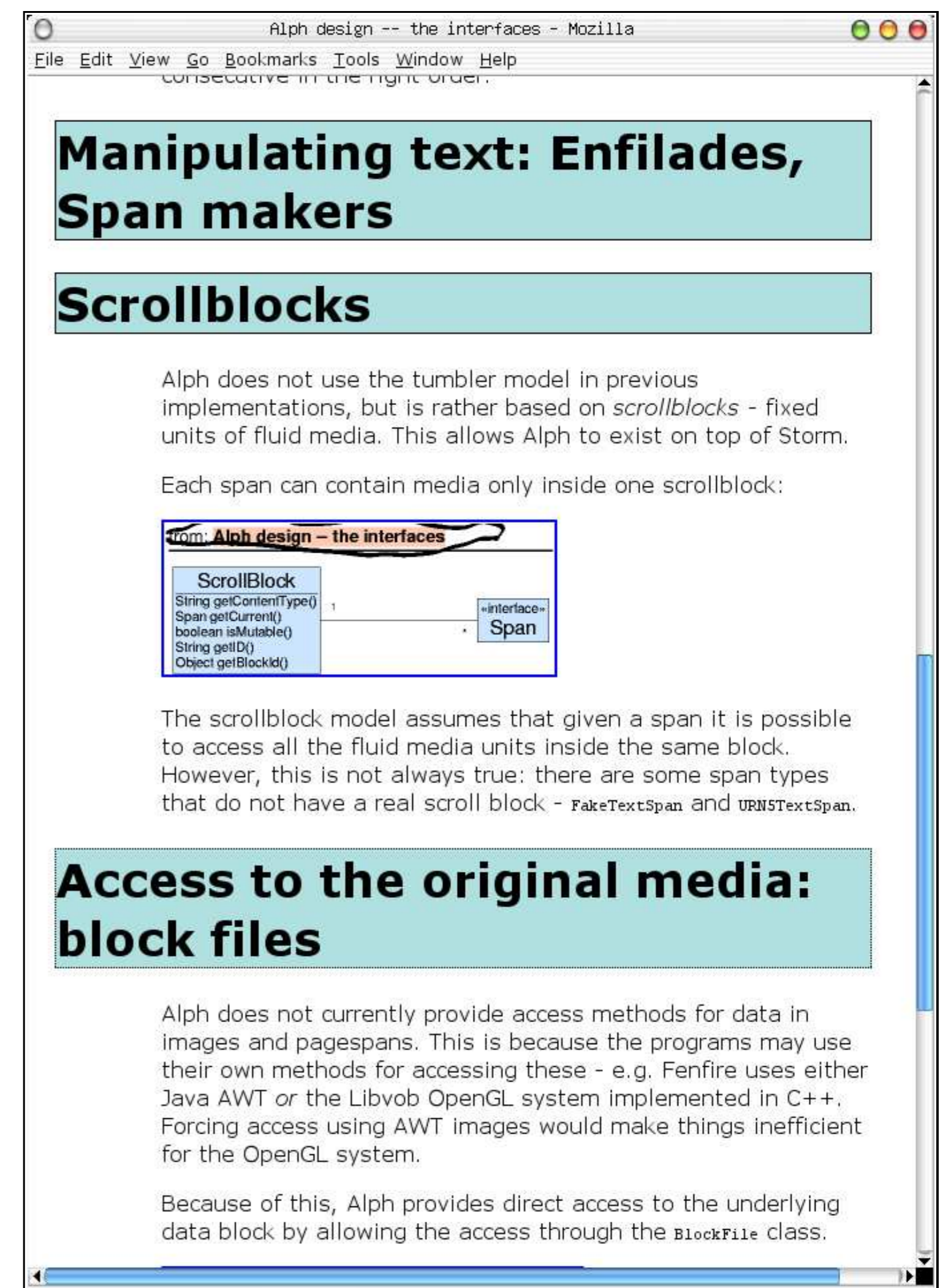
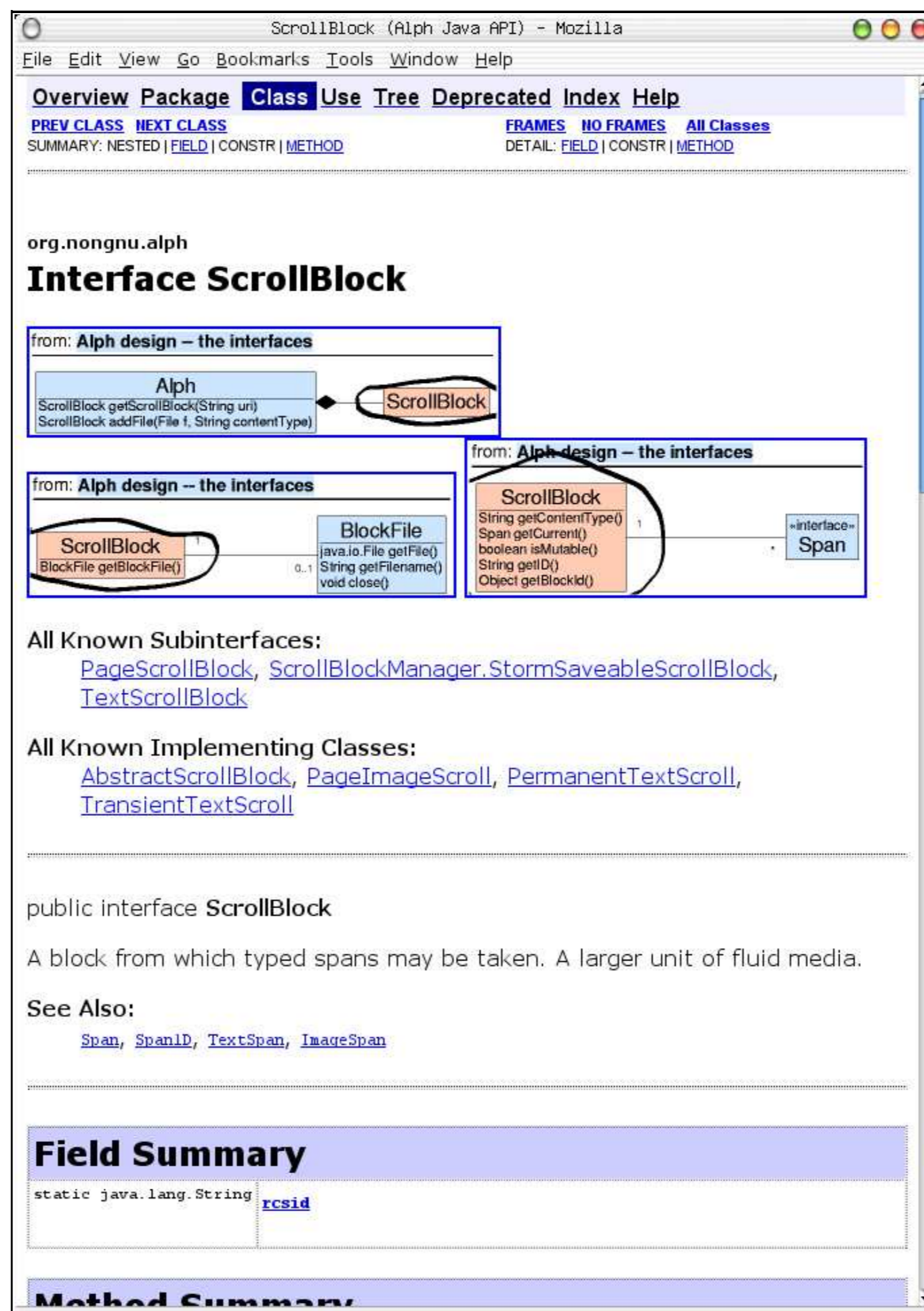
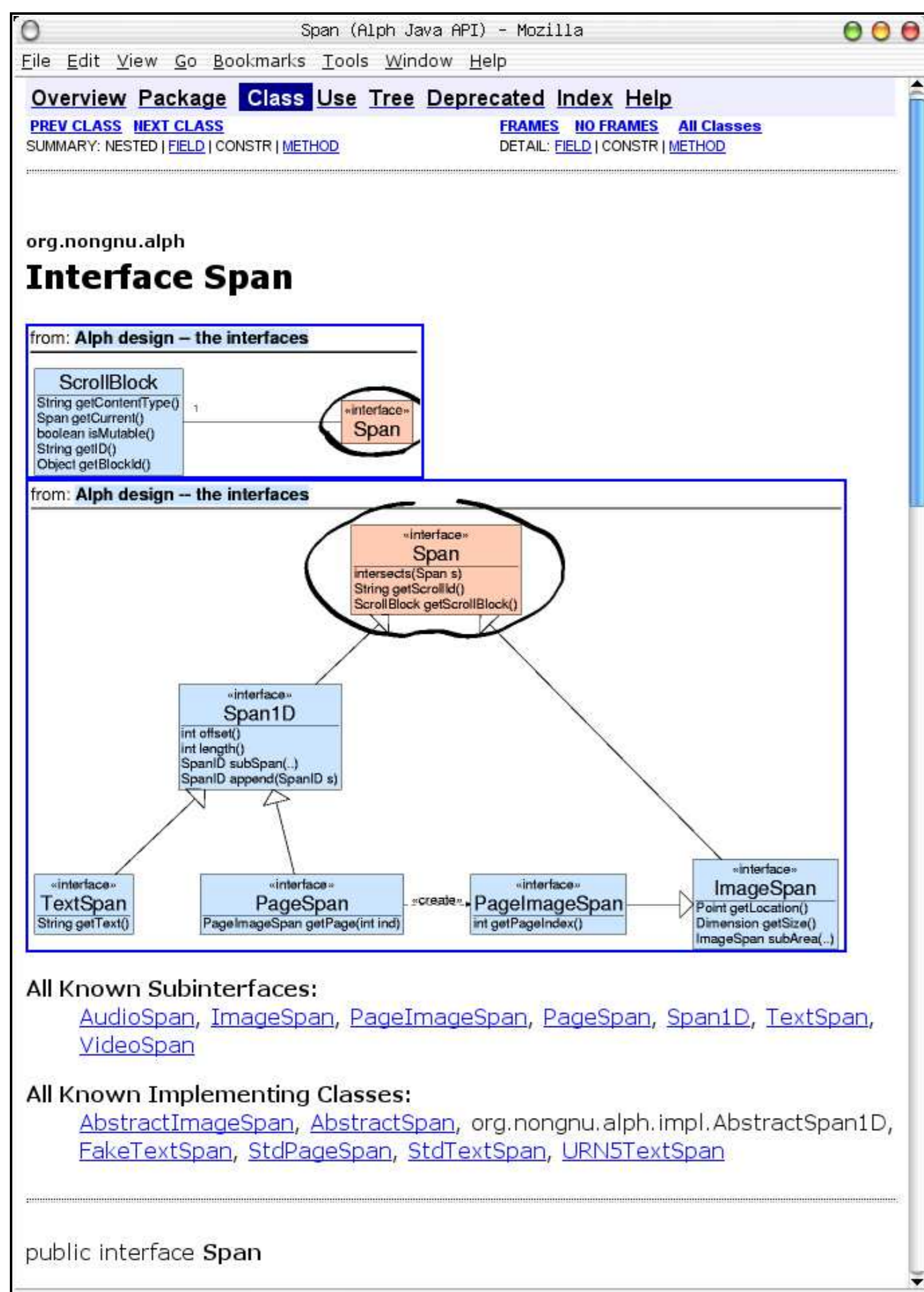
Asko Soukka, Tuukka Hastrup, Tuomas J. Lukka and Benja Fallenstein

Hyperstructure Group

Agora Center, University of Jyväskylä

P.O. Box 35, FIN 40014

humpake@iki.fi, Tuukka@iki.fi, lukka@iki.fi, b.fallenstein@gmx.de



Three HTML pages modified by Navidoc. Clicking on the Scrollblock element in the diagram on the leftmost page takes the user to the Scrollblock javadoc (the center page). Clicking the heading at the top of the diagram traverses to the design document (the rightmost page), where the diagram is excerpted to be described. The currently active node is highlighted in each diagram.

### Background

- Usually in software projects, two main classes of documentation:
  - Design documentation
  - Embedded documentation, e.g. Javadoc
    - Big picture
    - Detailed use of classes
    - Human-made, semantically meaningful UML with selected context relevant classes
    - Comprehensive; includes all classes of a package
    - Sometimes (e.g., with Doxygen) autogenerated UML diagrams, showing the inheritance tree
- Both internally linked
- Usually no hyperlinks between the two types
- Example: Java2 SDK 1.4.1 class `java.security.PublicKey`
  - Can be found through package and class lists by name when looking how to represent a cryptographic public key
  - Javadoc page less than useful:
    - no information on how to create these
    - On the "Use" page, references to this interface listed
    - No information about contextual importance!
    - Accessor methods that return a `PublicKey` contained somewhere listed alongside factory methods
  - Should have a link to the design documentation where the reader could then understand the **intention** of the designer
  - Problem: laborious to link manually

### UML (Unified Modeling Language)

- A standard way to diagram software architectures and constructs
- Usually a natural part of design documentation
- Each diagram exposes a particular **aspect** of the design
  - A single element (e.g., class) appears on several diagrams highlighting different aspects of its context
  - For a given class, the diagrams it appears on and their locations in the design documentation give a good overview of its use

### The central idea: human-made UML diagrams in design docjo as spatial menus

- Human-made**
  - Important: autogenerated diagrams are often too full of irrelevant details
- Spatial menus**
  - The diagram (as an imagemap) is inserted to all pages which describe elements on the diagram
  - The current page is highlighted
  - The automatically inserted versions are reduced in size by default to avoid clutter, but can be zoomed while browsing
  - The originating pages are added to the diagram to provide a backlink to the page

### Implementation: Navidoc

- A light-weight post-processor built on top of
  - Docutils
  - Javadoc / Doc++ / other embedded documentation tools
  - Our own UML diagram language
  - Metapost, image manipulation tools
- GPL-licensed, works in a fully free software environment

- | Pros  | Cons   |
|---|--|
| <ul style="list-style-type: none"><li>Easy to get started, simple syntax</li><li>Support for different embedded documentation formats, more can be added modularly</li><li>Extensible in python</li></ul> | <ul style="list-style-type: none"><li>Syntax errors may cause obscure error messages from MetaPost</li><li>Yet, no support for interactive placement of elements inside diagram</li><li>Currently requires design documentation to be in reStructuredText format</li></ul> |

### Experiences

- According to the initial statistics from the WWW server of our project, the UML diagrams have been used to traverse between javadoc and design documentation (approx. 10% of all page loads, unfortunately cannot tell difference between navidoc testing and real use)
- Anecdotally the UML diagrams are used to traverse between Javadoc pages even more often than to design documentation
- Currently more detailed statistics are being recorded -without skewing by extra traffic from testing and debugging Navidoc itself
- More advanced and experimental use of Navidoc beyond this poster:
  - Cross-linking between documentations of distinct projects, which share common packages and classes
  - Cross-linking between documentations and a reStructuredText and Docutils based Wiki -environment
  - Enlarging diagram syntax to cover RDF -symbols

### Conclusions

- The human-made UML diagrams from the design documentation are used as spatial menus on the relevant Javadoc or other embedded documentation pages
  - Allows traversal between the design documentation and the Javadoc pages
  - Gives the user an idea of the context of the current class
  - Is likely to include only meaningful parts of the system's semantics

### Further information

- Navidoc project at Savannah <http://savannah.nongnu.org/projects/navidoc/>
- Navidoc processed documentation examples <http://himalia.it.jyu.fi/gzzdoc/> <http://himalia.it.jyu.fi/ffdoc/>
- Fenfire project <http://fenfire.org/>
- David Goodger's Docutils: Python Documentation Utilities <http://docutils.sourceforge.net/>
- An Introduction to reStructuredText <http://docutils.sourceforge.net/spec/rst/introduction.html>