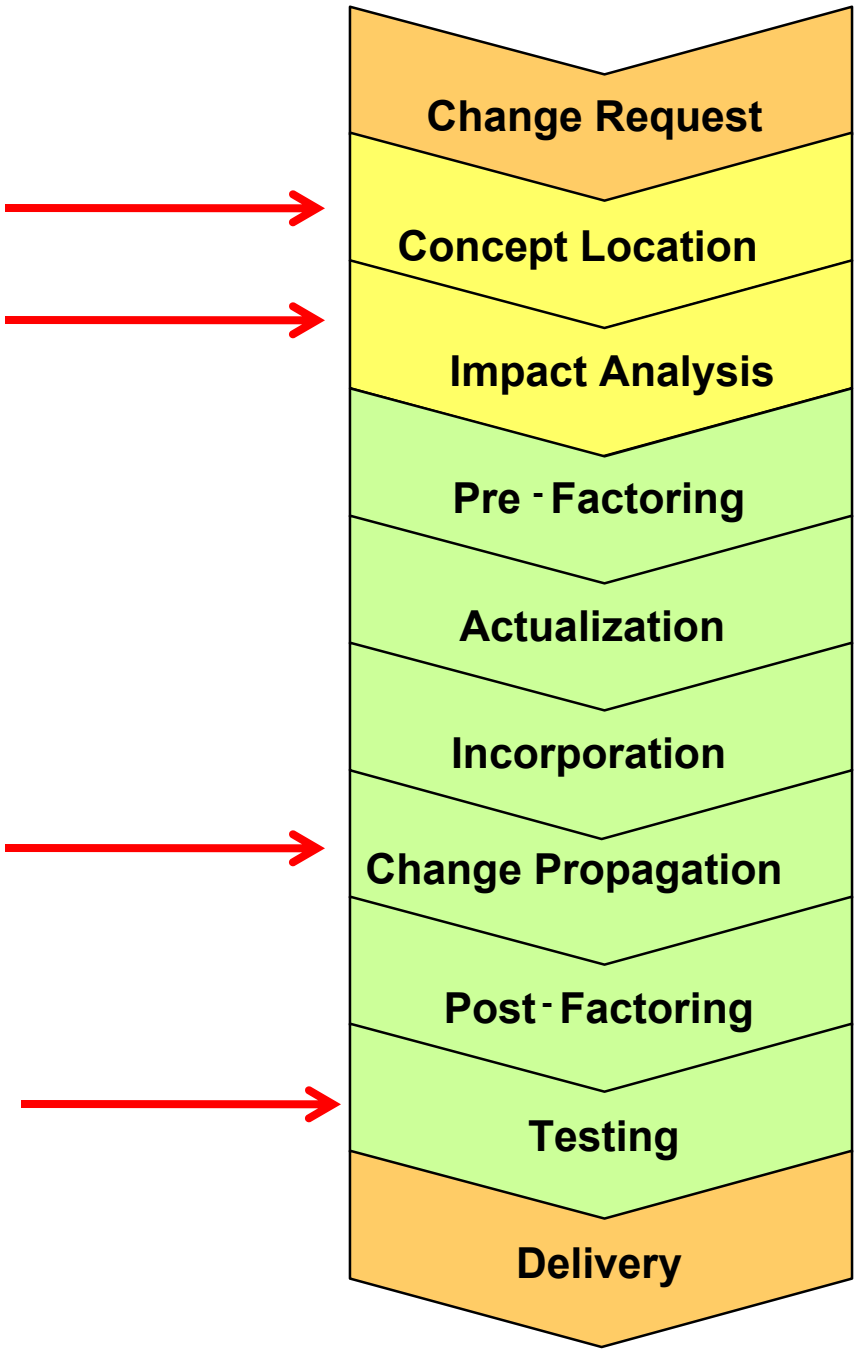
Using Traceability Links to Assess and Maintain the Quality of Software Documentation

Denys Poshyvanyk and Andrian Marcus

International Symposium on Grand Challenges in Traceability
(GCT'07 / TEFSE'07)

Lexington, Kentucky

Incremental Change

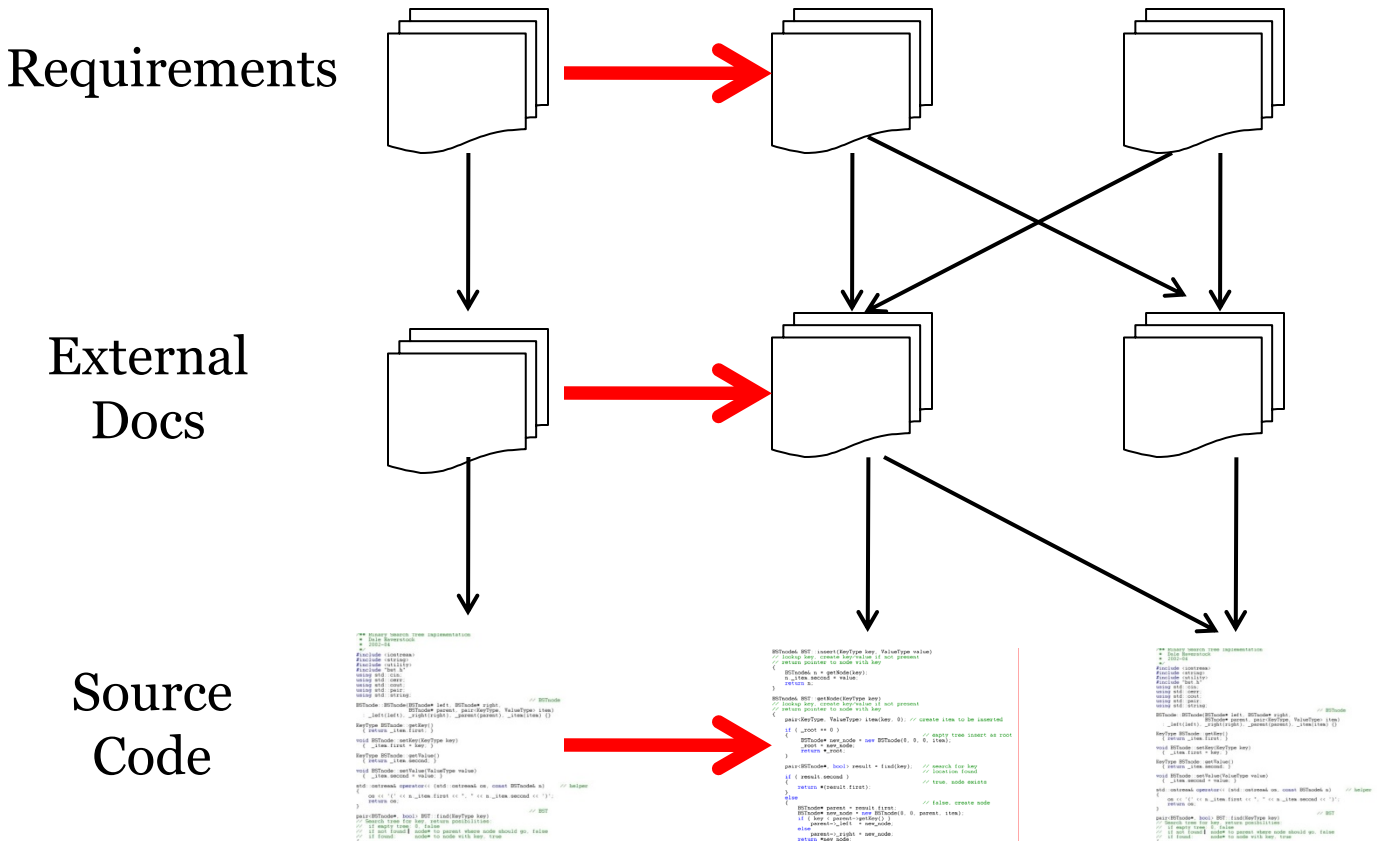


Problem Description

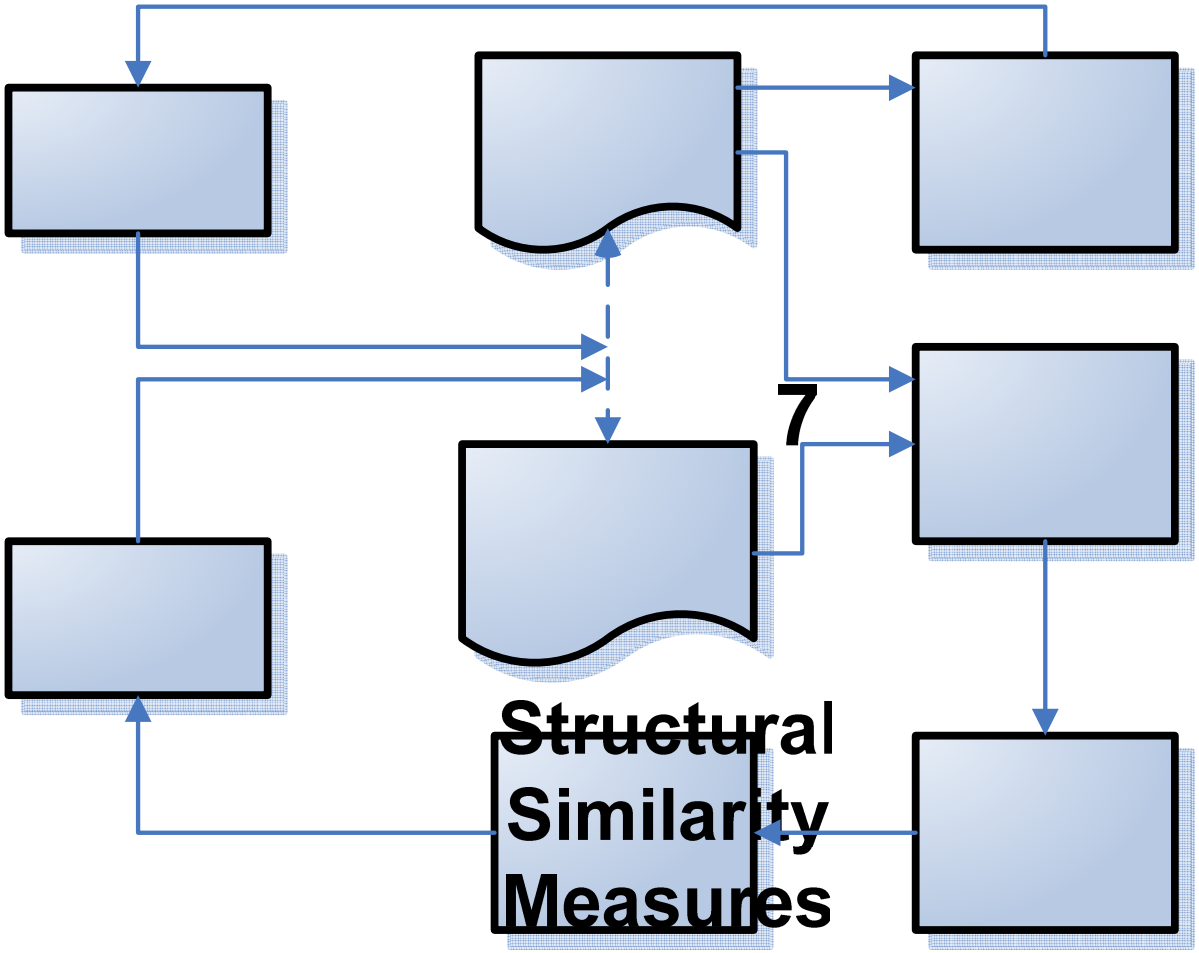
- Documentation structure does not reflect in many cases the structure of the source code (missing links among related sections in the docs)

- As source code evolves (structure changes), the documentation should also reflect those changes within its internal structure (e.g. major refactoring of the source code)

Traceability and Structural Links



Proposed Approach



Source co

Applications

- On systems with existing traceability links between source code and documentation
- To recover initial traceability links to enhance the structure of documentation

Improving Documentation in LEDA

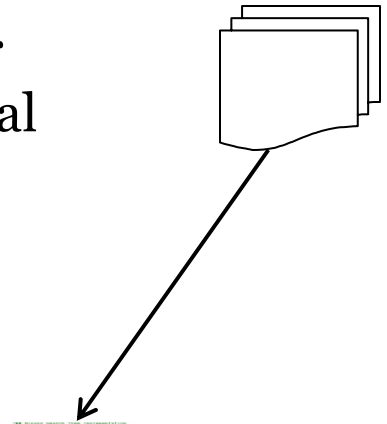
- LEDA implemented in C++
- 115 manual sections in English
- 219 classes
- Traced links from documentation to the source code

Coupling Measures

- Coupling measures computed with Columbus [Ferenc'04] :
  - CBO, RFC, MPC, DAC, ICP, ACAIC, OCAIC, ACMIC, OCMIC
- Conceptual coupling measures (CoCC) computed with our IRC^{3}M tool

Supporting Link Recovery

User manual



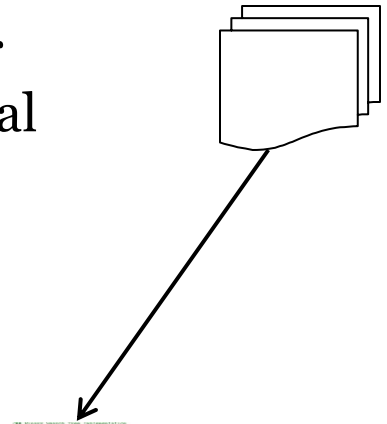
Class A

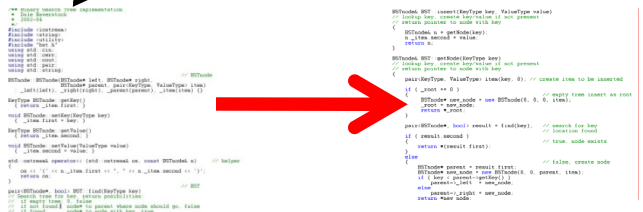
Class B

Class A
Class D
…
…
Class B

Supporting Link Recovery

User manual





Class A

Class B

Class A
Class B
…
…
Class D

Other Applications

java.util

Class ArrayList<E>

```
java.lang.Object
  └ java.util.AbstractCollection<E>
      └ java.util.AbstractList<E>
          └ java.util.ArrayList<E>
```

```
public class ArrayList<E>
extends AbstractList<E>
implements List<E>, RandomAccess, Cloneable, Serializable
```

Resizable-array implementation of the `List` interface. Implements all optional list operations, and permits all elements, including `null`. In addition to implementing the `List` interface, this class provides methods to manipulate the size of the array that is used internally to store the list. (This class is roughly equivalent to `Vector`, except that it is unsynchronized.)

See Also:

`Collection`, `List`, `LinkedList`, `Vector`, Serialized Form

Future Work

- Other types of artifacts, e.g. design docs, requirements
- Case studies on single version of software
  - to evaluate the impact of using structural/conceptual similarities to refine the structure of documentation
  - which combination of coupling measure can adequately reflect the structure?
- Case studies on multiple versions of software
  - to evaluate the impact of structural similarities in source code due to incremental changes