A Collaborative Writing Mode for Avoiding Blind Modifications

Claudia-Lavinia Ignat, Gérald Oster, Pascal Molli and Hala Skaf-Molli
{ignatcla,oster,molli,skaf}@loria.fr
LORIA-INRIA Nancy Grand Est
Nancy, France
Collaborative Editing

- Communication modes: synchronous/asynchronous
- Awareness: "understanding of the activities of others which provides a context for your own activity" [DourishCSCW92]
Synchronous / Asynchronous Work Modes

- **Synchronous editing**
  - Reduces conflicts and task completion times [CookAPSEC05]
  - Disadvantage: no support for work in isolation

- **Asynchronous editing**
  - Support for work in isolation
  - Workspace copies kept in consistent states
  - Risk of blind modifications
    - Useless work
    - Redundant work
Main Issue

- How to allow work in isolation and avoid blind modifications?

- Our proposition
  - Provide information in real-time about group activities
  - Deal with trade-off awareness/privacy
  - Annotate the document with uncommitted changes
Outline

- Motivating Example
- Our Proposition
- Revisiting Motivating Example
- Comparison with Existing Approaches
- Conclusions and Future Work
Motivating Example

**Actions User\textsubscript{1}**
removes method \textit{isReal}

**Actions User\textsubscript{2}**
updates method \textit{isReal}

**Actions User\textsubscript{3}**
creates test class \textit{IntegerTest}
## Scenario with standard VCS

<table>
<thead>
<tr>
<th>Step</th>
<th>Actions User₁</th>
<th>Actions User₂</th>
<th>Actions User₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>op₁ = removes method <em>isReal</em></td>
<td>op₂ = updates method <em>isReal</em></td>
<td>op₃ = creates test class <em>IntegerTest</em></td>
</tr>
<tr>
<td>2</td>
<td>COMMIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>UPDATE (conflict op₁ ↔ op₂)</td>
<td>UPDATE (<em>IntegerTest</em> does not compile)</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Reinsert new method <em>isReal</em></td>
<td>Remove test for <em>isReal</em></td>
</tr>
<tr>
<td>5</td>
<td>COMMIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>UPDATE &amp; COMMIT (no test for <em>isReal</em>)</td>
</tr>
<tr>
<td>7</td>
<td>UPDATE</td>
<td>UPDATE</td>
<td></td>
</tr>
</tbody>
</table>
Our approach

- Assumption: users continuously connected
- Send non-committed operations in real-time
- Preserve user privacy by filtering the non-committed operations (ghost operations)
  - Example:
    - $\text{op}_1 = \text{insert(}\text{User}_2,\text{"file.txt",4,"Preventing blind modifications"})$
    - $g(\text{op}_1) = \text{insert(}\text{User}_2,\text{"file.txt",4,30})$
    - $g(\text{op}_1) = \text{edit(}\text{"file.txt",4})$
  - Trust metric may be used for automatic filtering
- Integrate ghost operations as annotations within the document
Revisiting Motivating Examples

<table>
<thead>
<tr>
<th>Generated Operations</th>
<th>Privacy filter</th>
<th>Ghost operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>op₁ = delete(User₁, Integer.java, 15-18)</td>
<td>No filter</td>
<td>g(op₁) = delete(User₁, Integer.java, 15-18)</td>
</tr>
<tr>
<td>op₂ = update(User₂, Integer.java, 16, &quot;return false&quot;)</td>
<td>Filter content</td>
<td>g(op₂) = update(User₂, Integer.java, 16, -)</td>
</tr>
<tr>
<td>op₃: create IntegerTest.java</td>
<td>No ghost</td>
<td>-</td>
</tr>
</tbody>
</table>
Interface User₁

$op₁ = \text{delete} (\text{User}_1, \text{Integer.java}, 15-18)$

$g(op₂) = \text{update} (\text{User}_2, \text{Integer.java}, 16,-)$

no ghost operation from User₃

User₁ will not validate the removal of method $\text{isReal()}$
Interface User₂

\[ \text{op}_2 = \text{update}(\text{User}_2, \text{Integer.java}, 16, \text{"return false"}) \]

\[ g(\text{op}_1) = \text{delete}(\text{User}_1, \text{Integer.java}, 15-18) \]

no ghost operation from User₃

User₂ is aware of removal of \text{isReal()} – can initiate communication
Interface User₃

op₃ : create IntegerTest.java

g(op₁)=delete(User₁,Integer.java,15-18)
g(op₂)=update(User₂,Integer.java,16,-)

User₃ can examine changes in Integer and postpone testing
Related Work – VC^2

- Awareness tool integrated with existing VCSs
- Notifications sent to users when a document is modified concurrently
- Concurrent changes are not localised

Related Work – Edit Profile


No awareness about concurrent uncommitted changes
Related Work – Tukan

- No filtering of information

T. Schummer, J.M. Haake, **Supporting distributed software development by modes of collaboration**, *In Proceedings of European Conference on Computer Supported Cooperative Work, Bonn, Germany, 2001*
Related Work – State Treemap

- Inform users about states of shared documents
- LocallyModified, PotentiallyConflict, WillConflict
- Granularity of awareness is the document
- Impossible to locate concurrent modifications within the document

Related Work – Palantir


severity metric not sufficient for inferring potential conflicts
Related Work – Divergence Metrics

- Metrics computed using operations and not events
- Example metrics
  - Compute the amount of changes on each document
  - Compute an amount of conflicting/overlapping changes

- No localisation of changes


• No privacy issues, no secrets between members of a team

Conclusions and Future Work

- Awareness approach for avoiding blind modifications (in text, wiki, source code documents)
- Ghost operations: trade-off awareness/privacy
- Future work
  - Develop a model for our proposed interaction mode (Operational Transformation Mechanism)
  - Provide users suitable interfaces for filtering
  - Implement a prototype for awareness in software engineering
  - Perform user studies