Coordination in Distributed Software Development

Anita Sarma
University of Nebraska, Lincoln
May 16, 2014
My Research

Understanding online communities

Supporting coordination in distributed teams

End User Software Engineering
My Research

- Understanding online communities
- Supporting coordination in distributed teams
- End User Software Engineering
Software Development
Project Dependencies

Dependencies among packages in PERL language

Relationships among developers in CPAN

© Visualcomplexity.com
Project Evolution

Evolution over different versions

© Visualcomplexity.com
Conflicts in Distributed Software Development

- **Direct Conflicts**: Two developers edit the same file concurrently (Merge conflicts)

- **Indirect Conflicts**: Conflicts arising because of changes in one file affecting changes in another (Build and Test conflicts)
## Conflicts in Distributed Software Development

<table>
<thead>
<tr>
<th>Project</th>
<th>#Merges</th>
<th>#conflicts</th>
<th># conflicts</th>
<th># Res. Days Avg (Med)</th>
<th># conflicts</th>
<th># Res. Days Avg (Med)</th>
<th># conflicts</th>
<th># Res. Days Avg (Med)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perl</td>
<td>185</td>
<td>74 (40%)</td>
<td>14 (8%)</td>
<td>23 (10)</td>
<td>4 (2%)</td>
<td>0.7 (1)</td>
<td>56 (30%)</td>
<td>31 (14)</td>
</tr>
<tr>
<td>Storm</td>
<td>88</td>
<td>39 (44%)</td>
<td>17 (19%)</td>
<td>6 (2)</td>
<td>9 (10%)</td>
<td>5 (8)</td>
<td>13 (15%)</td>
<td>8 (3)</td>
</tr>
<tr>
<td>Jenkins</td>
<td>505</td>
<td>204 (54%)</td>
<td>68 (14%)</td>
<td>23 (4)</td>
<td>74 (15%)</td>
<td>5 (2)</td>
<td>28 (6%)</td>
<td>7 (2)</td>
</tr>
<tr>
<td>Voldemort</td>
<td>380</td>
<td>170 (34%)</td>
<td>55 (15%)</td>
<td>20 (4)</td>
<td>16 (4%)</td>
<td>2 (0.75)</td>
<td>133 (35%)</td>
<td>6 (4)</td>
</tr>
</tbody>
</table>

- Merge conflicts: 8% to 19%
- Build conflicts: 2% to 15%
- Test conflicts: 6% to 35%
Coordination in Distributed Software Development

How can we

– identify emerging conflicts?
– predict the severity of conflicts?
– be proactive and avoid conflicting situations?
Coordination in Distributed Software Development

How can we

- identify emerging conflicts?
- predict the severity of conflicts?
- be proactive and avoid conflicting situations?
Workspace Awareness

- Monitor ongoing changes in remote workspace
- Identify potential conflicts
  - merge conflicts (direct conflicts)
  - conflicts arising from dependency violation (indirect conflicts)
- Notify developers of emerging conflicts
Results

- Conflicts are detected as they emerge
- Developers undertake action upon noticing a potential conflict
- Fewer conflicts grow “out of hand”
- The resulting code is of higher quality
- The penalty may be a small increase in time *now*
  - but the experiments do not account for the time *later* that developers must otherwise spend on resolving conflicts that are committed to the CM repository
Other Workspace Awareness tools

- Current tools (Conflict mitigation):
  - CollabVS [Dewan et al., ECSCW’07]
  - FastDash [Biehl et al., CHI’07 ]
  - Crystal [Brun et al. FSE’11]
  - ...
Limitations

- Conflicts identified after they occur
- Developers have to understand the significance and self-coordinate
- Coarse grained impact analysis
- Potential for information overload and Interruption
(some) Solutions

- Proactive conflict prediction among tasks
- Predicting conflict complexity from project history
- Using development context to scope impact analysis
(some) Solutions

- Proactive conflict prediction among tasks
- Predicting conflict complexity from project history
- Using development context to scope impact analysis
Schedule independent tasks to minimize conflicts arising because of concurrent software development

- proactive instead of reactive
- solutions at the task level
- avoid individualistic solution e.g. race conditions
Cassandra Approach

- Obtain task context (task – files)
- Order of tasks (Developer preferences)

- Identify edited files ($F_e$)
- Identify dependent files ($F_d$)

- Analyze tasks for conflicts
Cassandra Approach

- Obtain task context (task – files)
- Order of tasks (Developer preferences)

- Identify edited files ($F_e$)
- Identify dependent files ($F_d$)

- Analyze tasks for conflicts

- Formalize constraints
  - hard constraints ($>$)
  - soft constraints ($\neq$)
Constraint Evaluation

Evaluate Constraints (Z3)

- Optimize Solution
- Match developer preferences
- Display conflict information
- Display recommended task order

Re-evaluate constraints

SAT

Matrix:

\[
\begin{bmatrix}
4 & 2 & 3 & 1 \\
1 & 2 & 3 & 4
\end{bmatrix}
\]
Constraint Evaluation

Evaluate Constraints (Z3)

- Re-evaluate constraints
  - SAT
    - [ 4 2 3 1 ]
    - [ 1 2 3 4 ]
      - Optimize Solution
      - Match developer preferences
  - UnSAT
    - >
    - ≠
      - Relax constraints
- Display conflict information
- Display recommended task order

Re-evaluate constraints

Empirically-based Software Quality Research
Computer Science and Engineering, UNL
Results, Ongoing Work

- Cassandra successful in
  - scheduling conflict minimal tasks
  - 50%-97% conflicts avoided
  - optimizing based on developer preferences
  - 2-3 seconds; Maximum (6 months data): 3 min

- Ongoing work
  - sensitivity of task context precision
  - unSAT heuristics: automatically predict conflict complexity
  - consider task duration as a constraint
  - deployment
Some Solutions

- Proactive conflict prediction among tasks
- Predicting conflict complexity from project history
  - Use Machine Learning to predict severity of merge, build, test conflicts
  - Features selected: # files, file names, configuration files
  - F measures (merge conflict - 0.92, build - 0.87, test – 0.84)
- Using development context to scope impact analysis
  - Change of interest: the single change set and at a set granularity
  - Region of interest: active workspace, public API, specific developer changesets
Contributions

- Eliminate seclusion, while maintaining insulation
- Early detection of conflicts to proactive detection
- Granularity of conflict notification at the level of tasks
- Analyze repositories to identify conflict complexity
- Use development context to scope change impact analysis
Thank you!

This work is supported by:

- NSF CCF -1016134, IIS-1110916, IIS-1314365, CCF-CAREER
- AFOSR - 9550-10-1-0406

Interaction Design and Coordination Lab & Collaborators
Contributions

- Eliminate seclusion, while maintaining insulation
- Early detection of conflicts to proactive detection
- Granularity of conflict notification at the level of tasks
- Analyze repositories to identify conflict complexity
- Use development context to scope change impact analysis