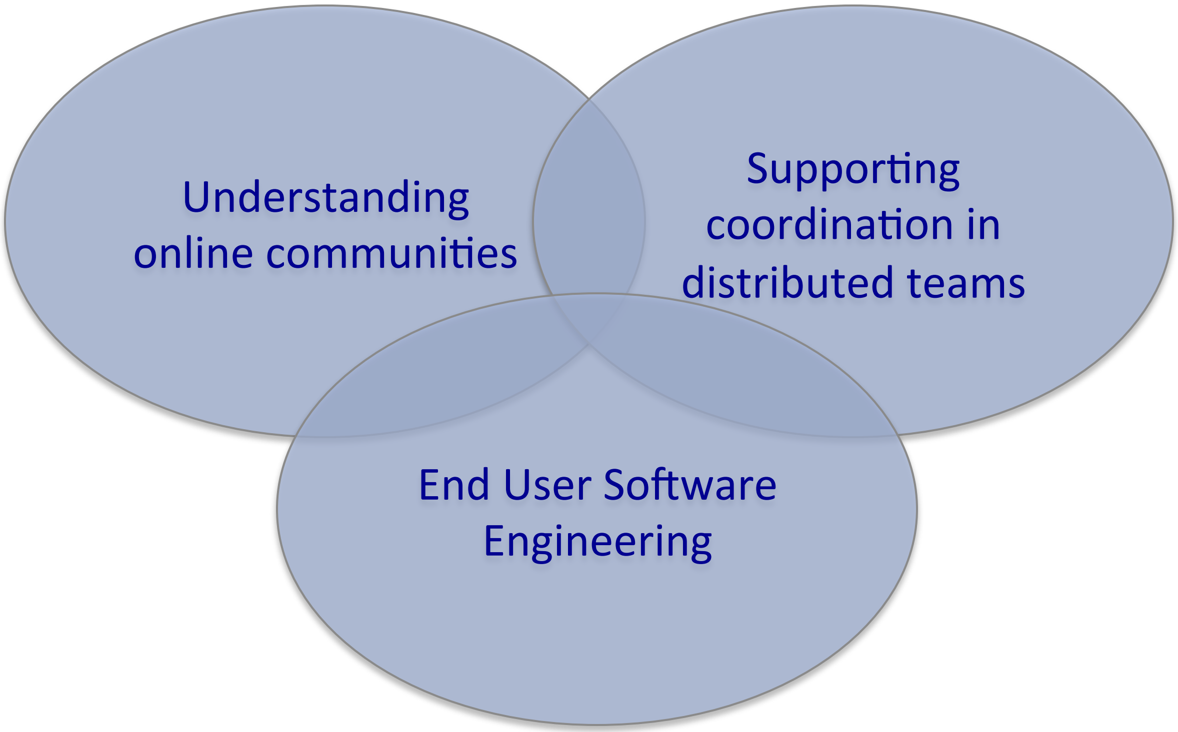
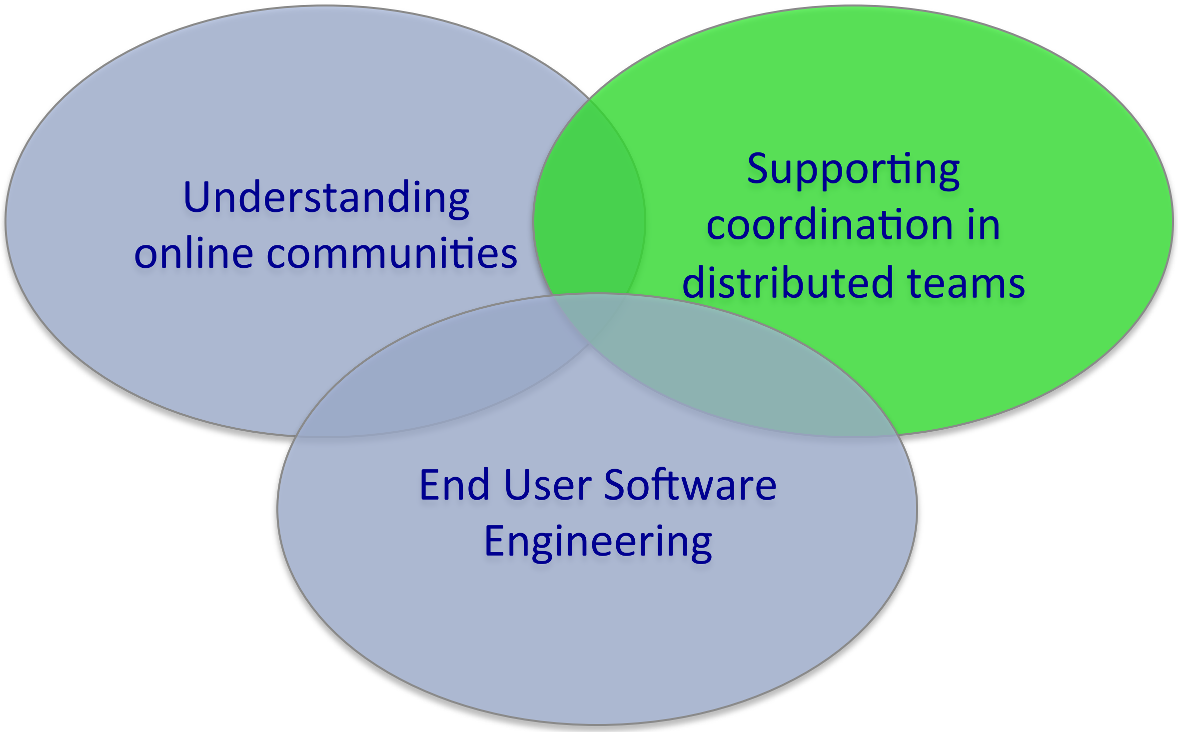
Coordination in Distributed Software Development

Anita Sarma
University of Nebraska, Lincoln
May 16, 2014

My Research


Understanding online communities
Supporting coordination in distributed teams
End User Software Engineering

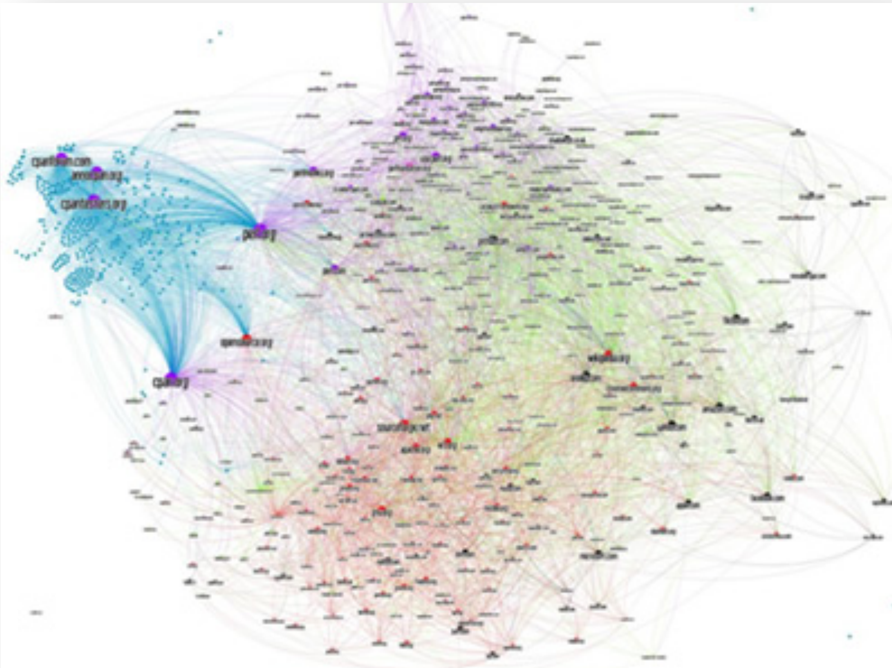
My Research



Understanding online communities
Supporting coordination in distributed teams
End User Software Engineering

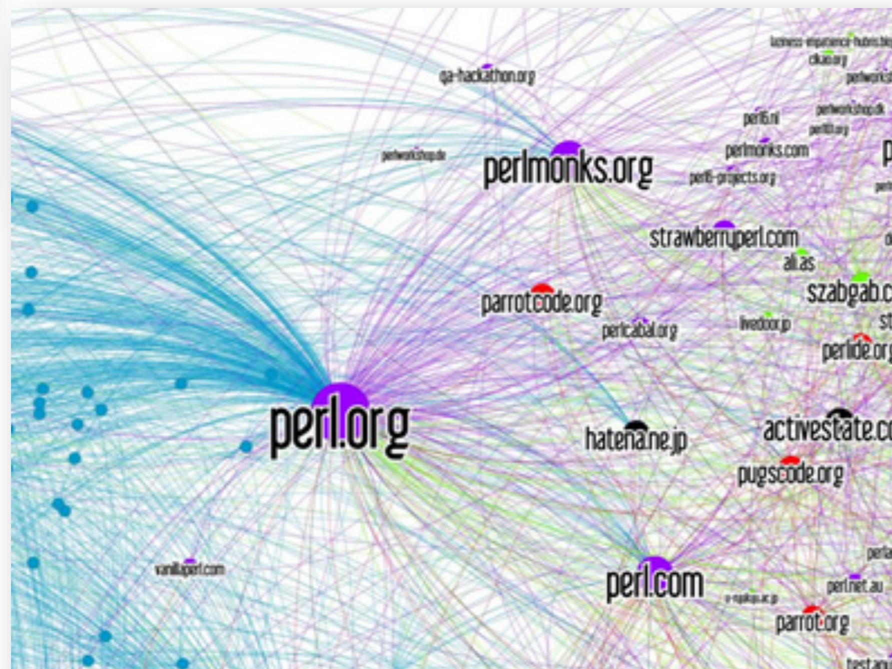
Software Development

Project Dependencies



Dependencies among packages in PERL language

Relationships among developers in CPAN



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

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

- 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

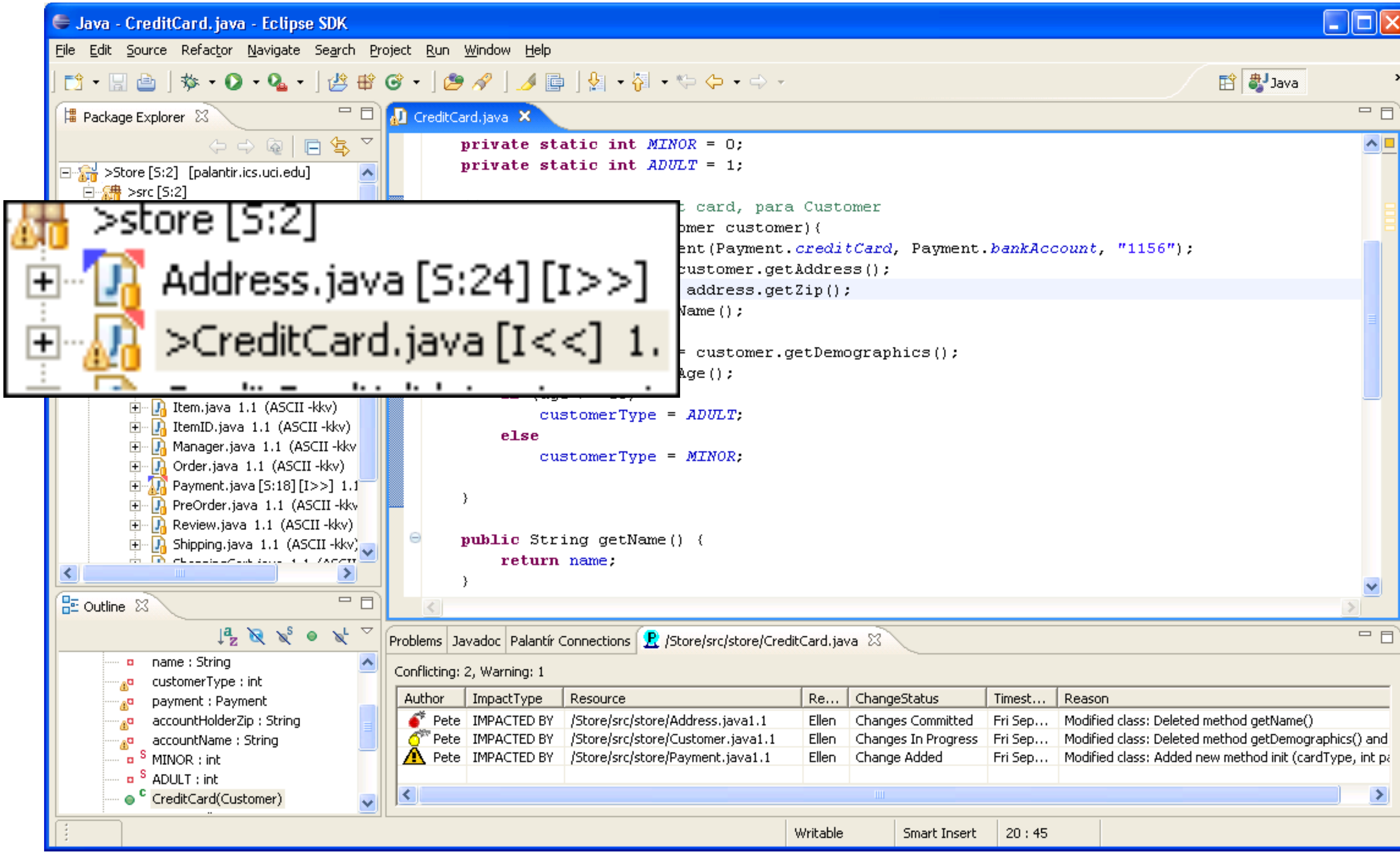
- How can we
  - **identify emerging conflicts?**
  - predict the severity of conflicts?
  - be proactive and avoid conflicting situations?

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

Palantír

[ICSE 2003, ASE 2007, FSE 2008, TSE 2012]

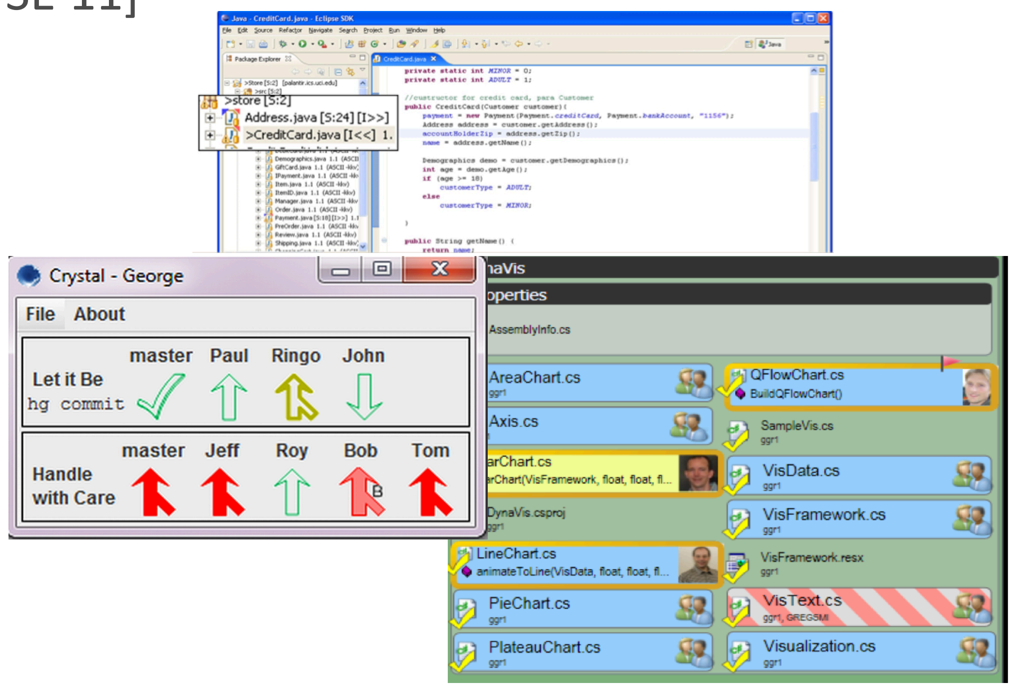


Results

- 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

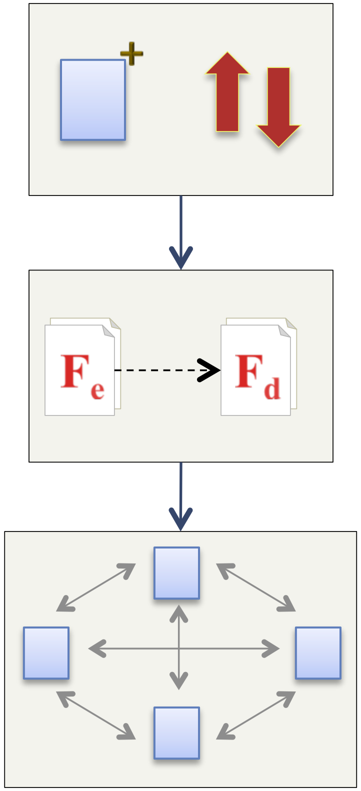
Cassandra

[ICSE 2013, USER 2013]

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

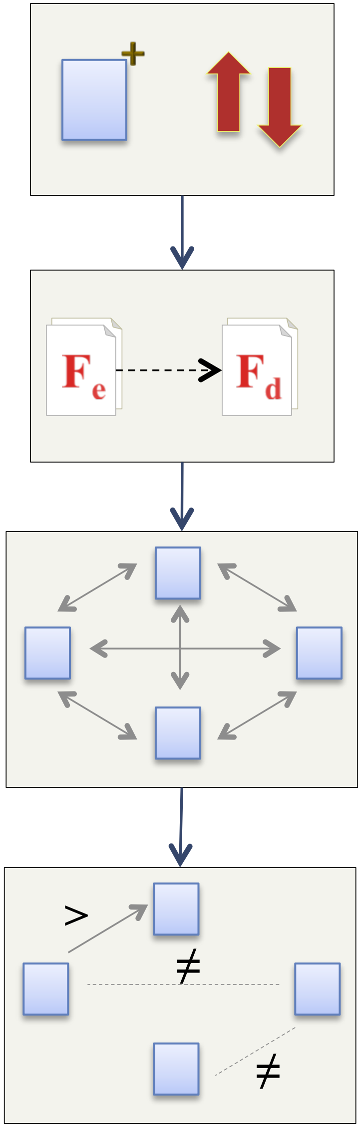


- Obtain task context (task – files)
- Order of tasks (Developer preferences)

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

- Analyze tasks for conflicts

Cassandra Approach



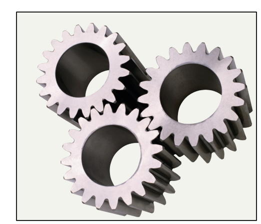
- Obtain task context (task – files)
- Order of tasks (Developer preferences)

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

- Analyze tasks for conflicts

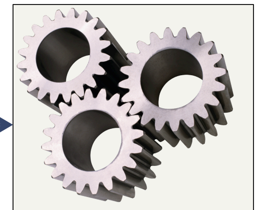
- Formalize constraints
  - hard constraints (>)
  - soft constraints (≠)

Evaluate Constraints



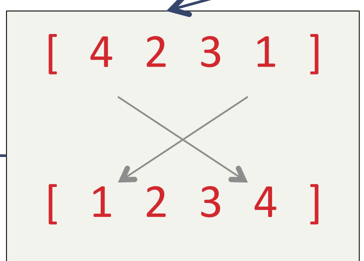
Constraint Evaluation

Evaluate Constraints (Z3)

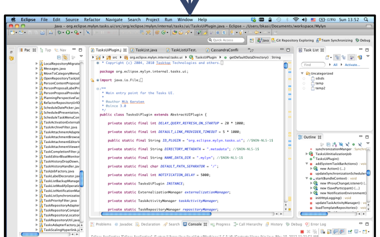


Re-evaluate constraints

SAT



- Optimize Solution
- Match developer preferences



- Display conflict information
- Display recommended task order

Constraint Evaluation

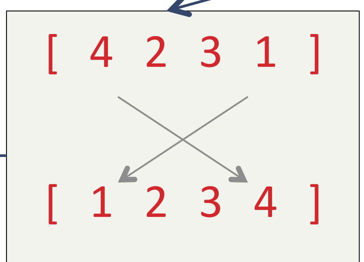
Evaluate Constraints (Z3)

Re-evaluate constraints

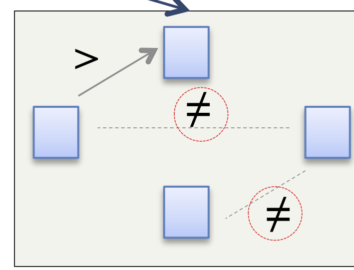
Re-evaluate constraints

SAT

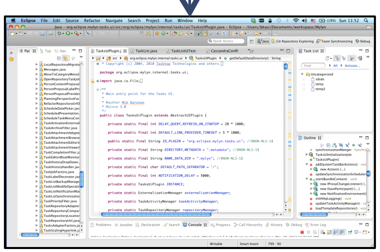
UnSAT



- Optimize Solution
- Match developer preferences



- Relax constraints



- Display conflict information
- Display recommended task order

Results, Ongoing Work

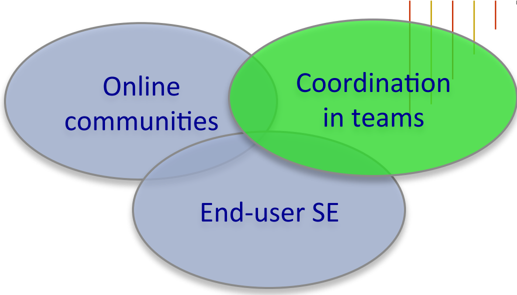
- Cassandra successful in
  - scheduling conflict minimal tasks
  - 50%-97% conflicts avoided
  - optimizing based on developer preferences
  - 2-3 seconds; Maximum (6 months data): 3 min

- 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

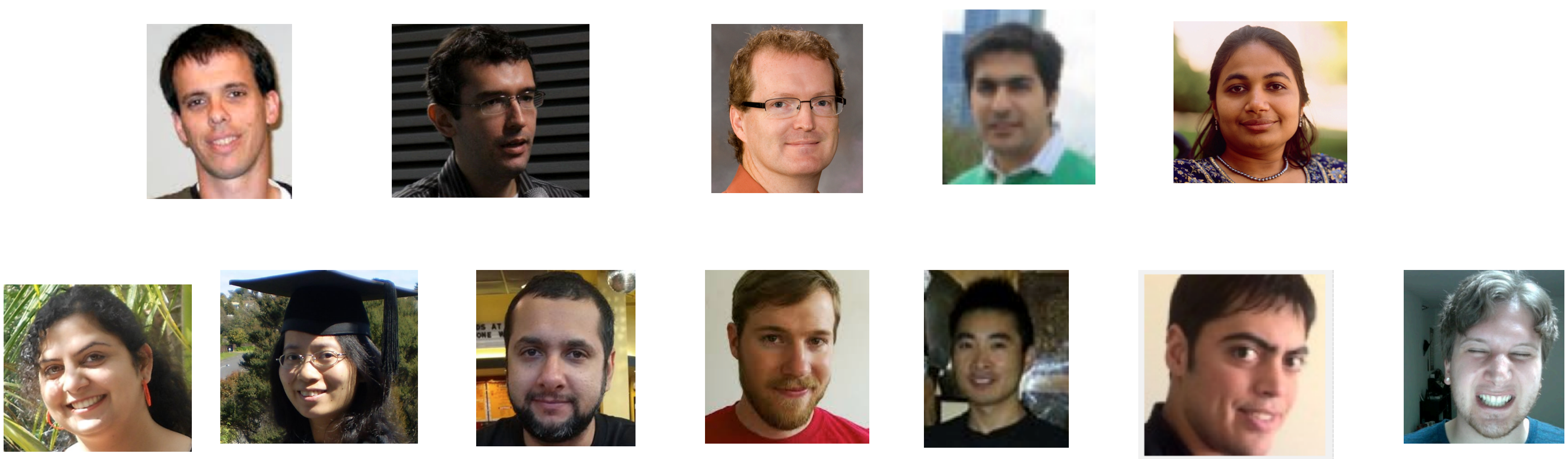


- Eliminate seclusion, while maintaining insulation
- Early detection of conflicts to proactive detection
- Granularity of conflict notification at the level of tasks
- Analyze repositories to identify conflict complexity
- Use development context to scope change impact analysis

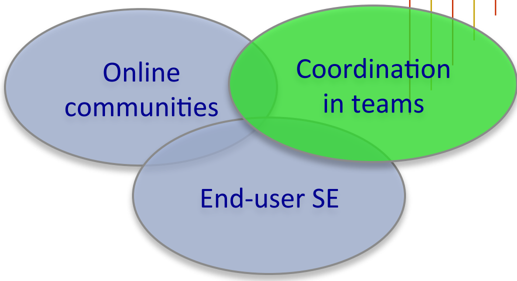
Thank you!

- 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