Understanding Individual Contribution and Collaboration in Student Software Teams

Fabio Rocha (fabiorocha@ualberta.ca)
Eleni Stroulia (stroulia@ualberta.ca)
Department of Computing Science
University of Alberta

Outline

- Motivation and Background
- The Tool Architecture
- Empirical Study
- Grounded Theory Study
- Conclusions
- Future Work

Motivation: Collab IDE Data as Answers

- Tools that support software teams should help answer the questions that arise during software development.
- Different stakeholders are interested in different questions:
  - Developers want to know what other developers are currently doing to specific software assets.
  - Managers are interested in higher-level information, such as whether the overall project is on time.
- We are interested in yet another perspective: that of a software engineering course instructor.

Background and Context

- Fritz and Murphy:
  - 78 questions that professional developers ask about source code, change sets, teams, work items, etc.

- Myers and LaToza
  - 67 questions that developers find hard to answer

- We focus on questions regarding team dynamics and individual performance and contribution in software teams

  (encompassed in the high-level categories described by Fritz and Murphy)

The Course Instructor Questions

"How student software teams work".

- What was the role of each member in the project?
  - Was there a clear team leader throughout the project?
- How much effort did each member put into the project?
  - How much time did each team member spend on the project?
  - What work did she do?
- How did the team members communicate with each other?
  - How frequently? How much?
  - Through what channels?
- How was the progress of the team members throughout the project lifecycle?

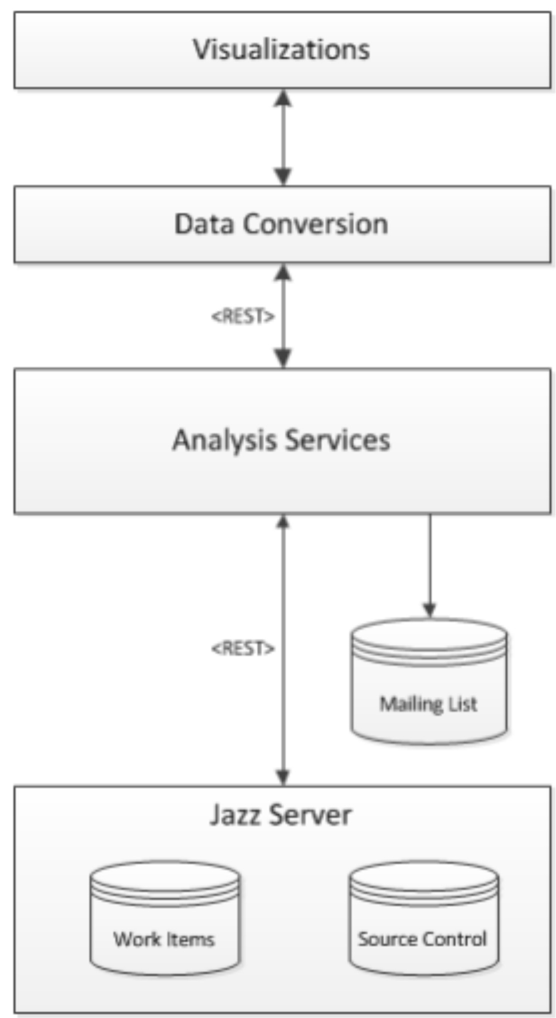
Proposed Solution

The data collected through the integrated collaborative software-development tool **is the spine of the story of how the team works.**

- ***an analysis service*** transforms raw data into aggregations to answer questions of interest for instructors
- ***an interactive visualization*** enable instructors to explore the role and contributions of each individual student to the project and the team dynamics

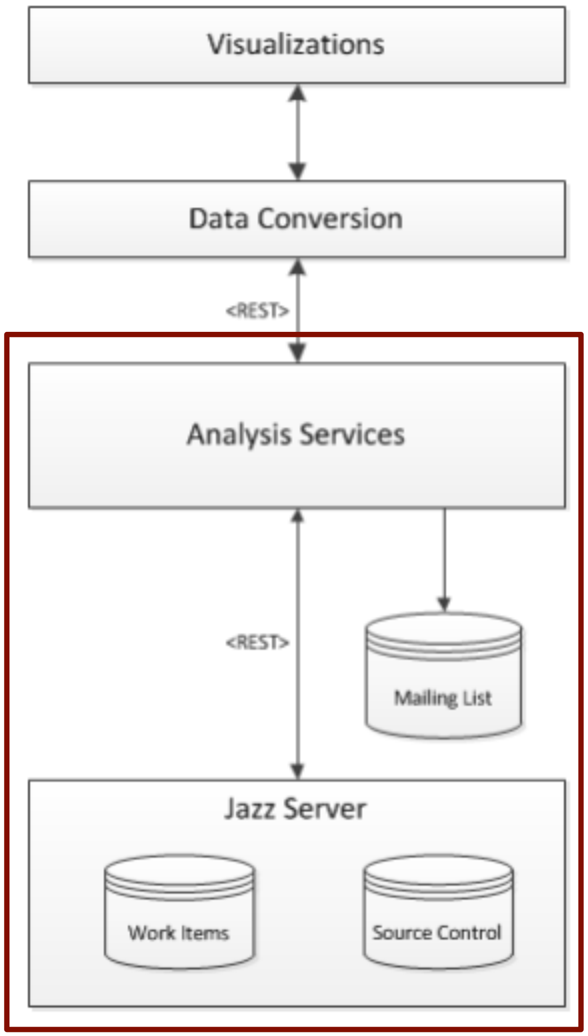
Architecture Overview

- Our framework has a modular architecture and is built with generality and extensibility principles in mind.
- Any collaborative software development IDE that provides similar instances of data works just as well.
- Provides a simple model for developers to:
  - easily extend the default set of operations
  - utilize data from other tools
  - output data for different types of visualizations



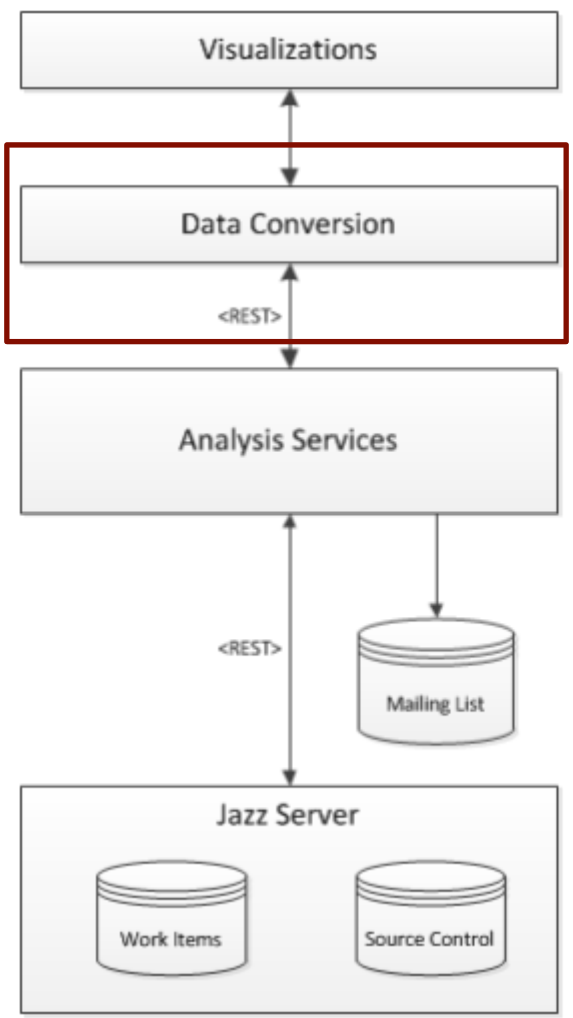
Data Extraction and Analysis

- Two Data Provision services APIs need to be implemented for a tool to be integrated with our framework:
  - a history service
  - a work-item service
- The Analysis Service REST API implements a number of analyses on the collaborative IDE data extracted through the Data Provision Services



Implementation
Data Conversion

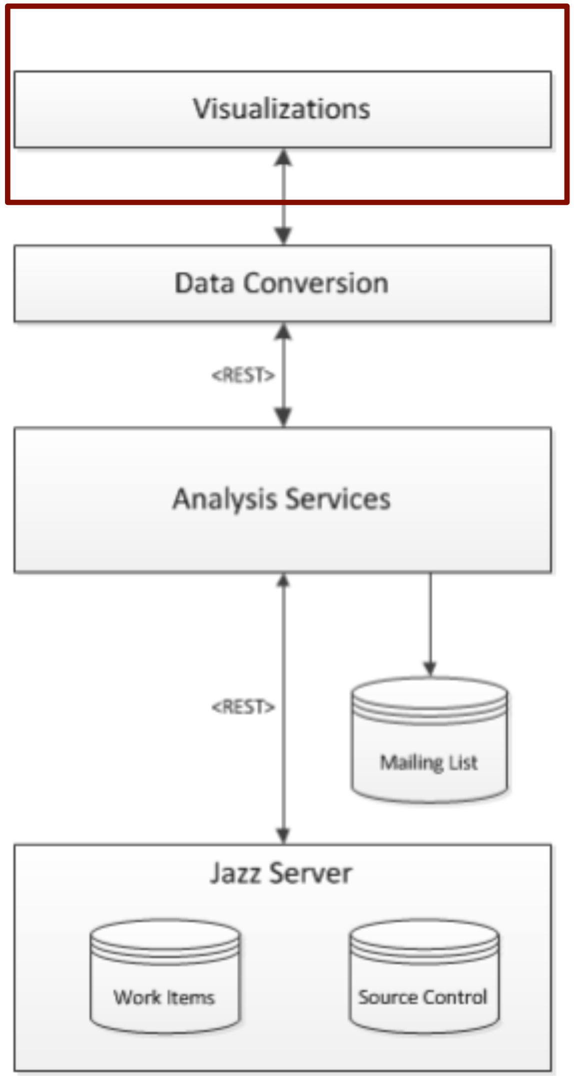
- Maps the Analysis Service JSON output into JSON used by the visualization client
  - Fields from the JSON object are mapped to fields of the visualizations, which translate directly to aspects of the visualization.
- Supports consistent look-and-feel across the various visualizations
  - Consistent color scheme across data elements
  - Global data filtering
  - Tooltips and comments
  - Data audit trail



Implementation

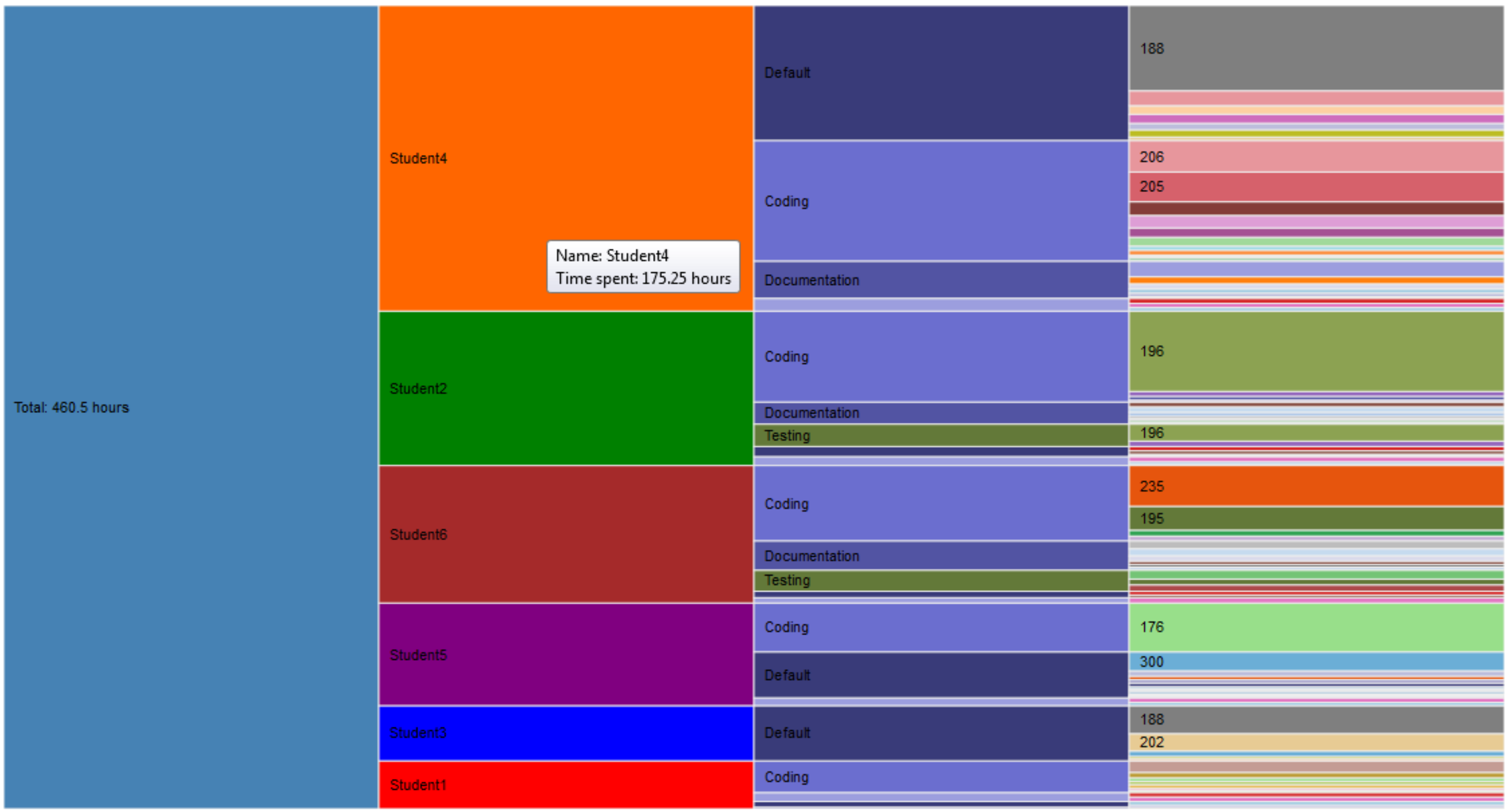
Visualization

- Consumes and displays data created by the analysis service and treated by the data conversion service
- Three different types of interactive visualizations
  - Aggregation Visualizations
  - Temporal Visualizations
  - Relationship Visualizations



Aggregation Visualizations - 1

- How much time on which (types of) work items?

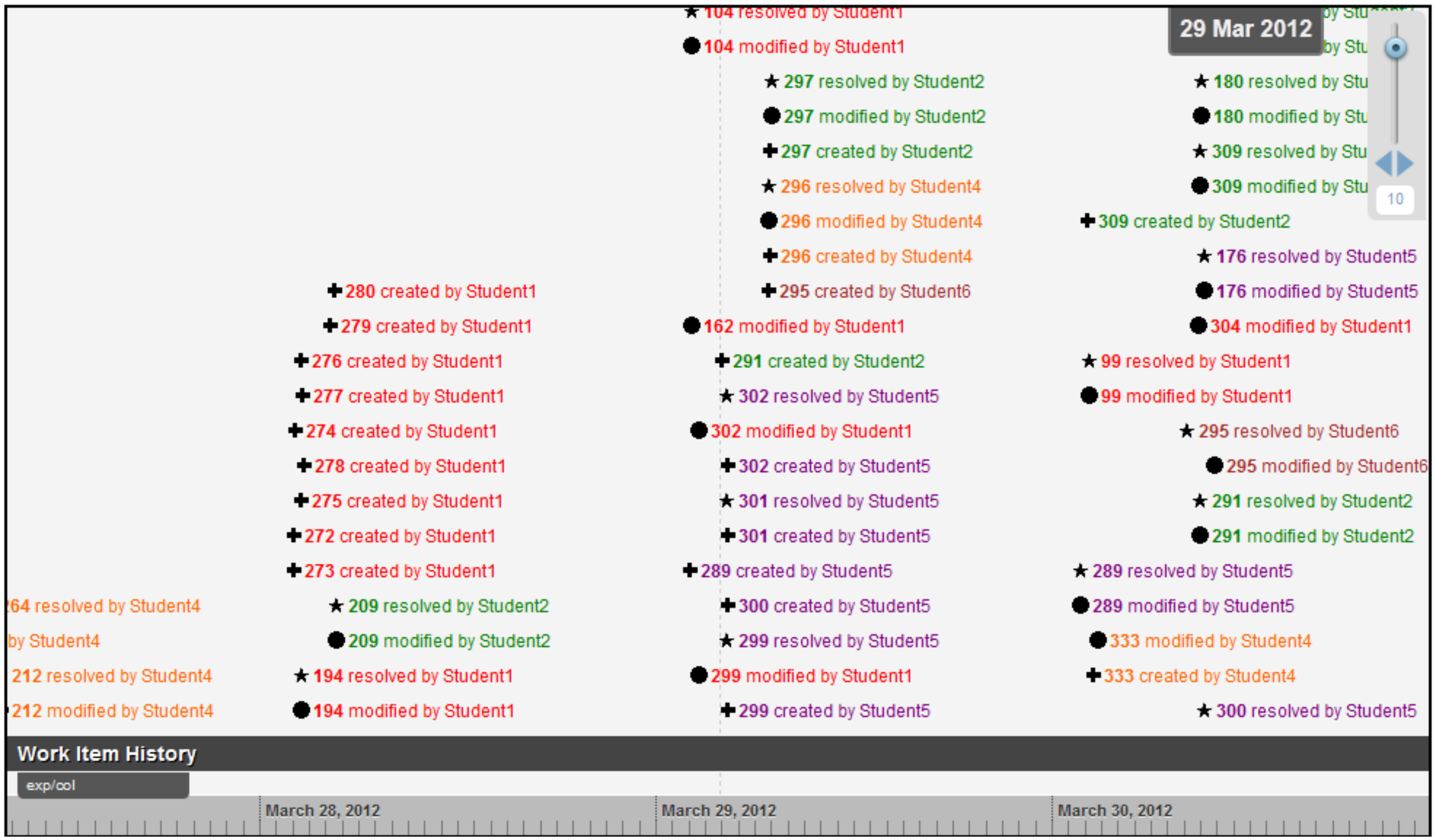


Aggregation Visualizations - 1

- How much time on what types of communications with whom?

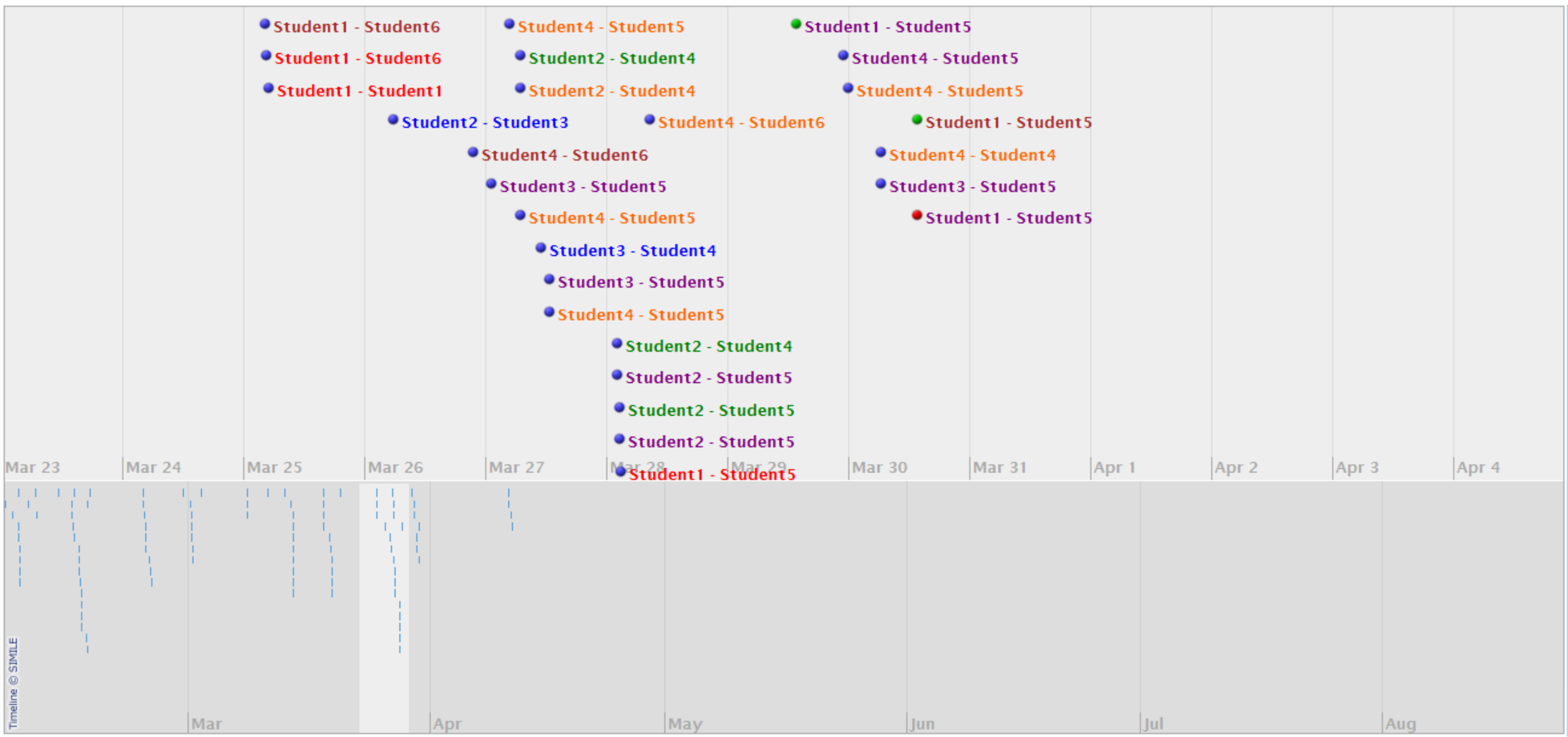
Temporal Visualizations - 2

- Who did what to the project plan and when?



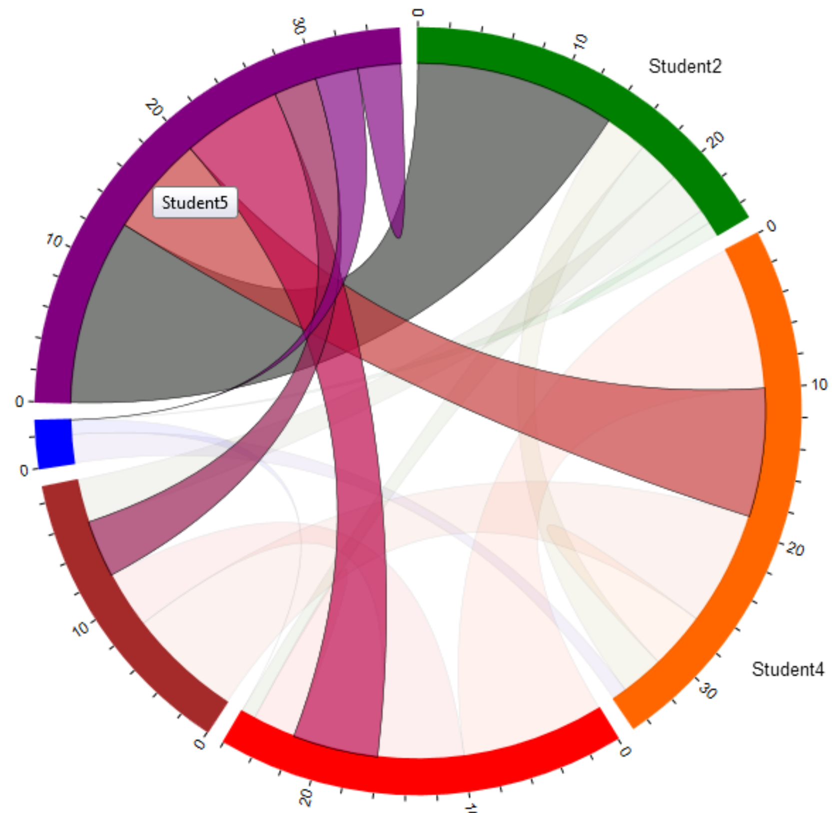
Temporal Visualizations - 2

- Who communicated with whom and when? How much over time?



Relationship Visualizations - 2

- For a given individual, who is her most frequent collaborator? Over what communication channel?

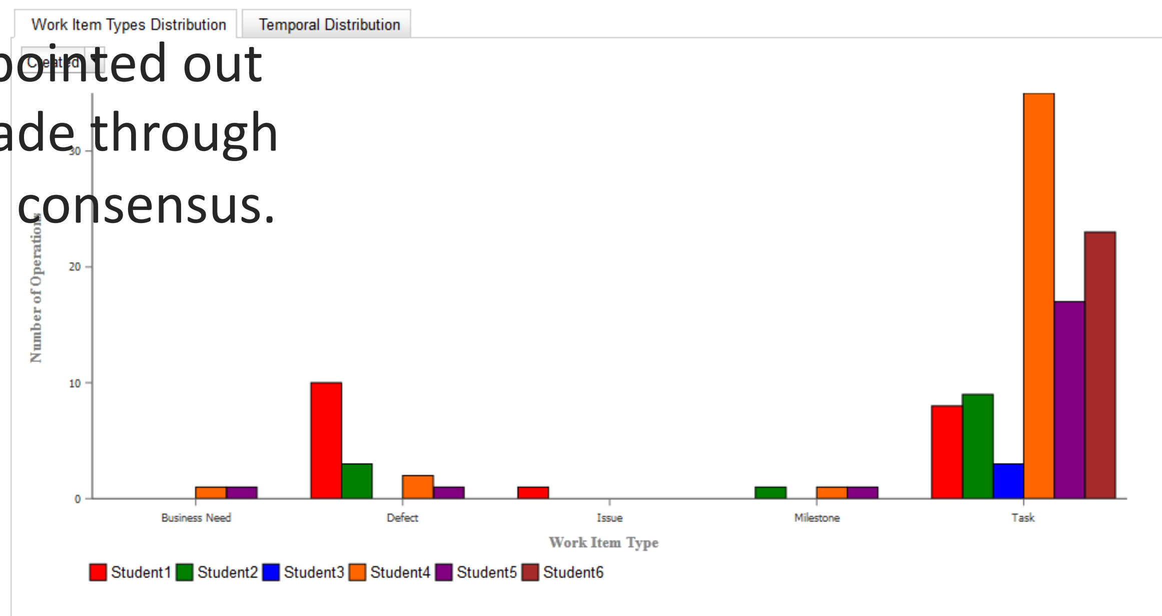


Empirical Study

- We used our toolkit in the context of a senior undergraduate software engineering course (i.e., CMPUT401).
- We studied a six-member student team that committed to using Jazz and its many functionalities to manage their collaborative software development.
- We also collected perceived data through questionnaires and interviews.
- We compared the ***information reported*** by the students against the ***information observed*** through our visualization toolkit.

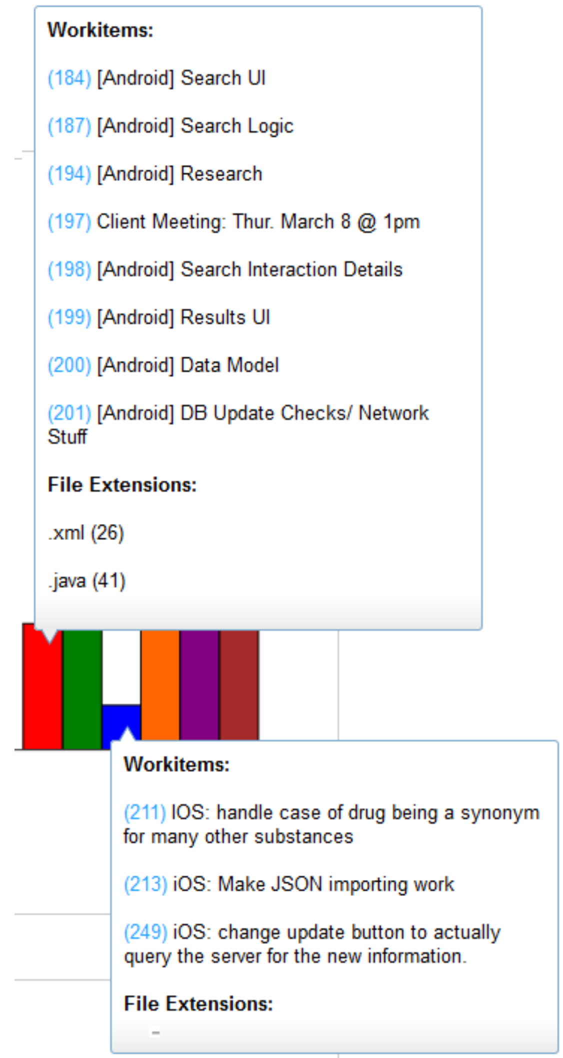
Self Perceptions vs. Observations -1

- **Was there a clear team leader?**
- ***Student4*** appears to contribute more to project management
  - He created most of the tasks and assigned them to team members
- BUT, the team members pointed out that all decisions were made through a democratic process and consensus.



Self Perceptions vs. Observations -2

- **What was the role of each member in the project?**
- A number of work items and artifacts were associated with pairs of team members ➔ We inferred that the team was organized in pairs
- ***Student2:*** "We have divided our group into 3 pairs based on the development platforms. We have one group doing iOS, one for Android, and one group working on the web-based administration."



Self Perceptions vs. Observations -3

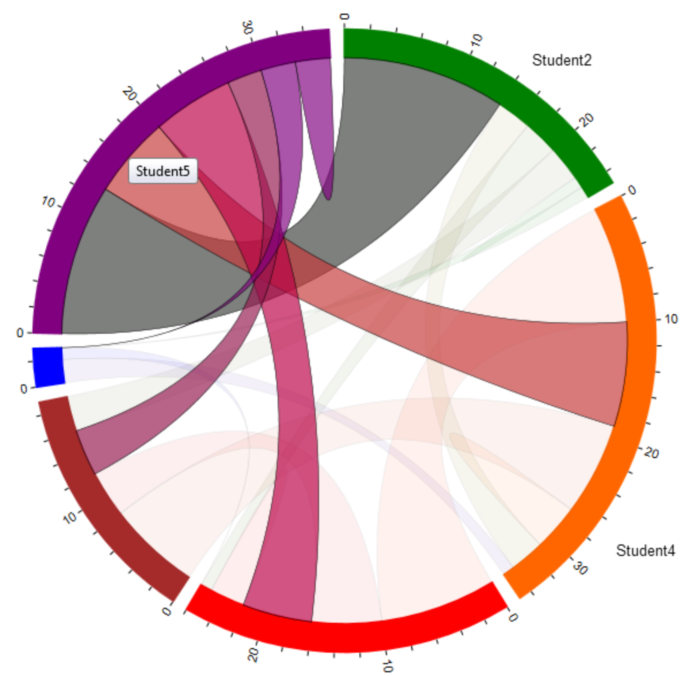
- **How much effort did each member put into the project?**
- ***Student4*** put more hours (double as much as the second student)
- ***Student2:*** I don't know how productive people actually were, but if I were asked to guess... student4 had the most groundwork to cover.
- ***Student5***: I wouldn't say that student4 worked the hardest necessarily, but it took him the most time to wrap his head around the system.
- ***Student4:*** I was the most productive team member for I had the most to research, learn and did the most of the programming on …

Self Perceptions vs. Observations -3

- **How much effort did each member put into the project?**
- ***Student4*** put more hours (double as much as the second student)
- ***Student2:*** I d but if I were groundwork
- ***Student5***: I w necessarily, b around the s
- ***Student4:*** I w most to rese ...

Self Perceptions vs. Observations -4

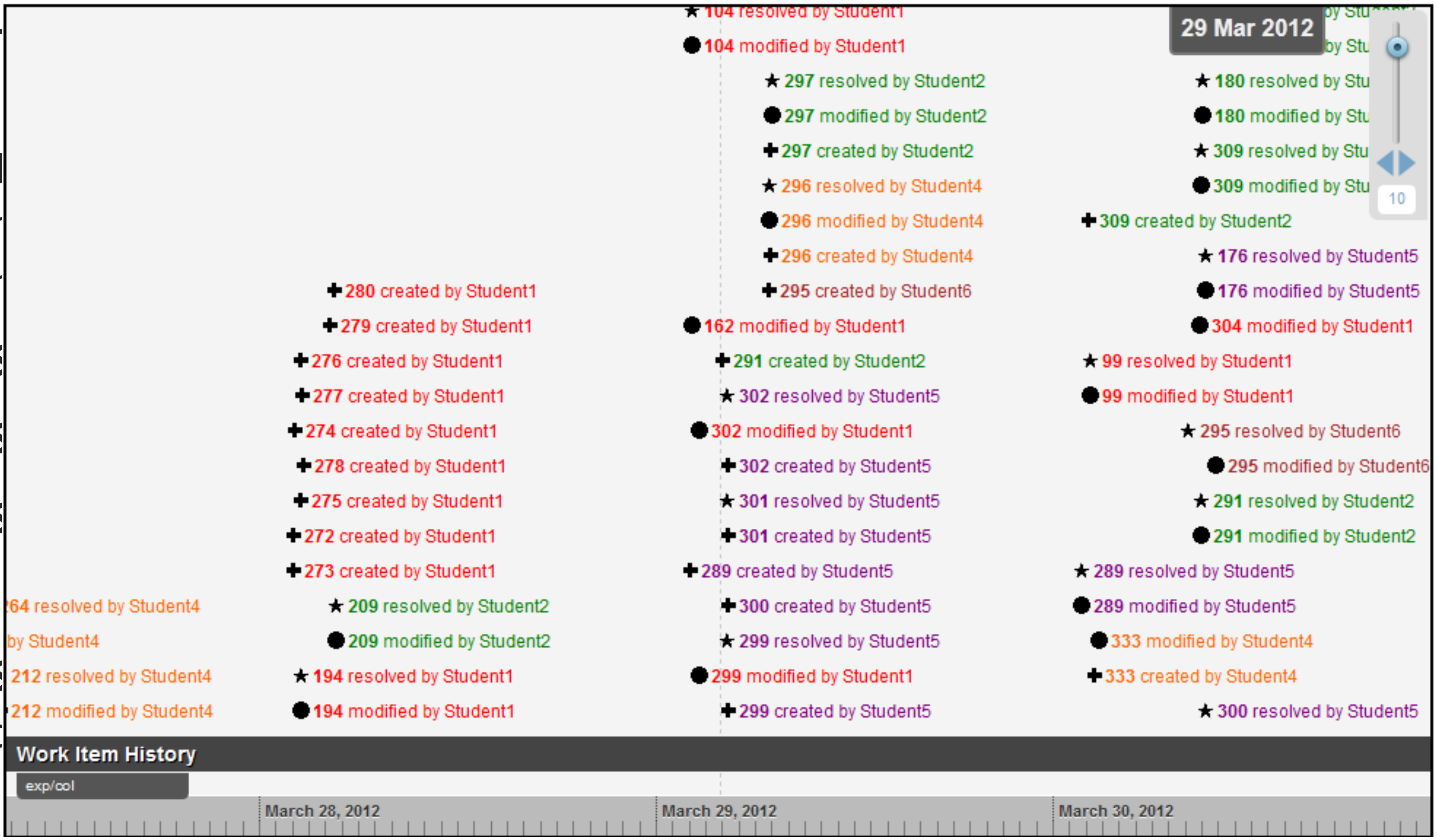
- **How much and how did the team members communicate with each other?**
- Consistent communication via email, and work-item comments; more email between non pair members
- ***Student4 :*** "Email was by far much more used than work items."
- ***Student2:*** "As a group we used the weekly meetings and the team's email account to communicate. As for communication between me and student1 it was done with skype and calling each others cell phones"



Self Perceptions vs. Observations -5

- **How was the progress of the team members throughout the project lifecycle?**
- The team exhibited a fairly consistent activity throughout the project; however, the activity-visualization widget made evident that considerably more effort was devoted to the project just before deadlines.
- ***Student2:*** Yeah. The 30th was the last week of code.
- ***Student5:*** 30th was when it was due, yeah.
- ***Student6:*** That reflected the deadline. The 30th right there.
- Interviewer: And this one.
- ***Student4:*** That's going to be a deadline, original documentation deadline. This definitely reflects how our deadlines work.

Self Perceptions vs. Observations -5



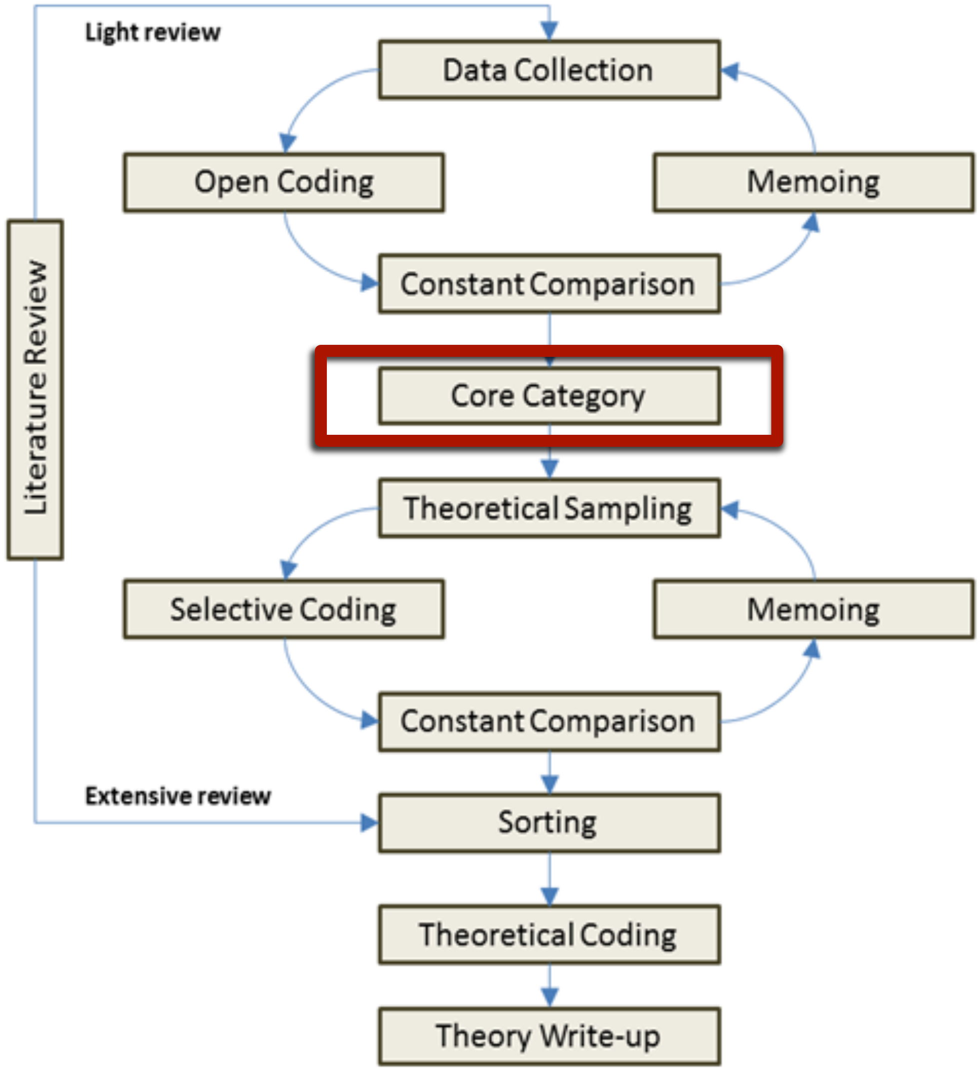
Empirical Study: Conclusions

- Our visualizations toolkit can help the instructor
  - gain valuable insights about the individual contribution and role played by each member of the team.
  - make inferences on how the team communicated.
  - collect evidence to consider in making decisions about individual performance/marks

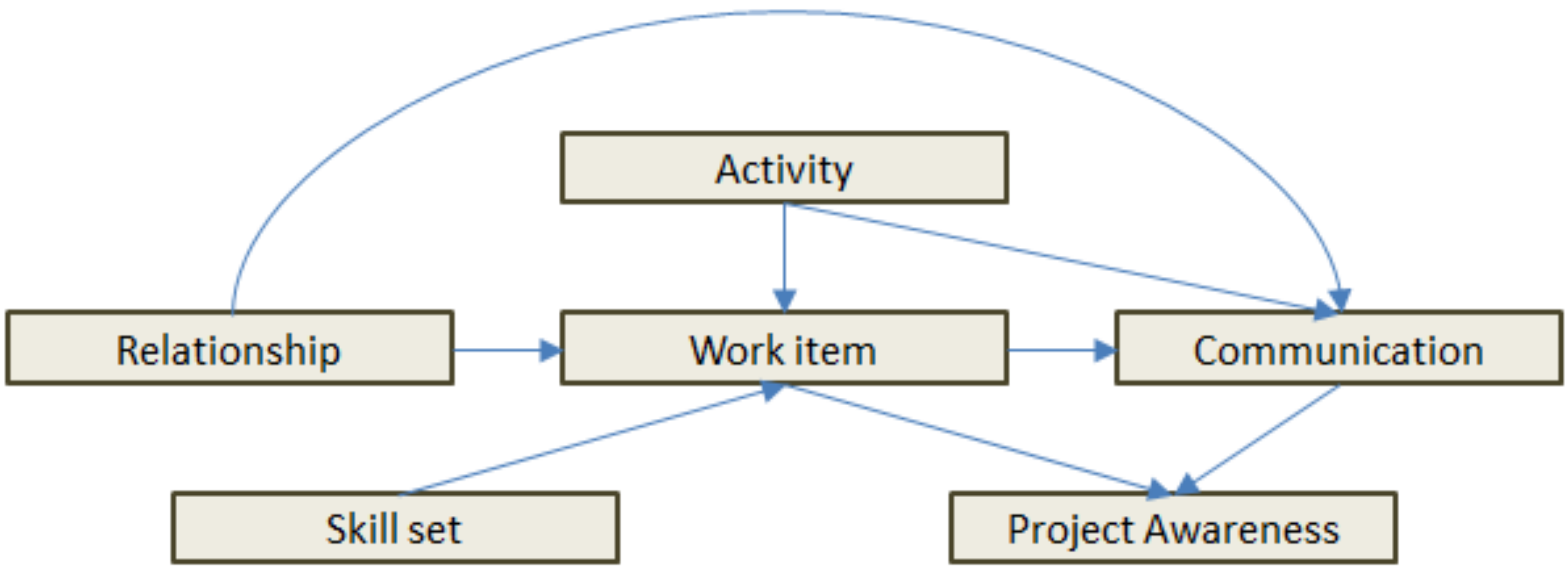
Grounded Theory Study

- A systematic process for generating theory from data, developed by Glaser and Strauss in the late 60's
  - It is a complete research method: it provides procedures that cover every stage of research, sampling participants, data collection and analysis, use of literature and writing.
- Grounded Theory is a useful research method to study the human and social aspects of Software Engineering
  - It guides researchers to investigate social interactions and behavior through interviews with the study participants.
  - It is useful for research in areas where a new perspective would be beneficial.

Grounded Theory Study



Grounded Theory Study



Grounded Theory Study
Categories

Category	Description	Codes
Activity	Common activities throughout a software engineering course	Management, Code, Documentation, UI, Design, Test, Learning, Process, Deployment
Communication	Communication channels among team members	Face-to-face, Email, Tickets
Project Description	Opinions or facts about the project and its members	Who, When, How, How much
Relationship	Interaction among participants	Student-Student, Student-Manager, Student-Client, Student-Team
Skill set	Experience and competence on a particular subject or technology	Skills, Knowledge
Work Item	Process components	Task, Defect, Milestone, Issue, Business Need, Risk

Grounded Theory Study

Category Relationships

- Activity and Work Item
  - Activities define which work items will be created.
- Activity and Communication
  - Activities define how people communicate.
- Relationship and Communication
  - Different relationships imply different forms of communication.
- Skill set, Relationship and Work Item
  - Skill set and relationships define the tasks of team members.
- Project Awareness, Work Item and Communication
  - Students often know about what other students are working on if they are working on the same or related work items.

Findings

- Students adapt their process to the activity being carried out at each point during the course.
- This modified process influences how students communicate, contribute and collaborate as a team.
- Students adapt to their natural process as soon as they find themselves under pressure.
- The process is not valued as much as the process outcome, a working piece of software.
- Students also care in performing the activity currently required by the instructor, in detriment of whatever process they decided to follow in the first place.

Guidelines for Instructors

- Emphasize the importance of the process.
- Do not undervalue activities like documentation and testing.
- Ask students to specifically define their roles upfront.
- Make evaluations mandatory.
- Make the use of collaborative software development tools mandatory

Conclusions

- We developed a toolkit for analyzing and visualizing data collected by collaborative IDEs.
  - Its modular architecture makes it easy to integrate with other IDEs.
  - It helps instructors answer questions about the role and performance of students through the use of visualizations.
  - It supports instructors to explore the IDE data through useful aggregate views.
- We also conducted a grounded theory study of how a team of software engineering students worked on a course project.

Future Work

- Provide a wider range of visualizations.
  - By using code-quality metrics, we will be able to provide not only visualizations about the amount of work a student has done but the quality of the work.
- Make the visualizations available for students
  - Study how the analytics services we have developed may or may not increase team productivity through increasing team awareness.
- Expand our experiments to an industry setting
  - See how our visualizations behave with larger teams and consequently, larger data sets. Experiment with distributed teams.

Q & A

Eleni Stroulia (stroulia@ualberta.ca)
http://ssrg.cs.ualberta.ca