Opportunities for Extending Activity Theory for Studying Collaborative Software Development

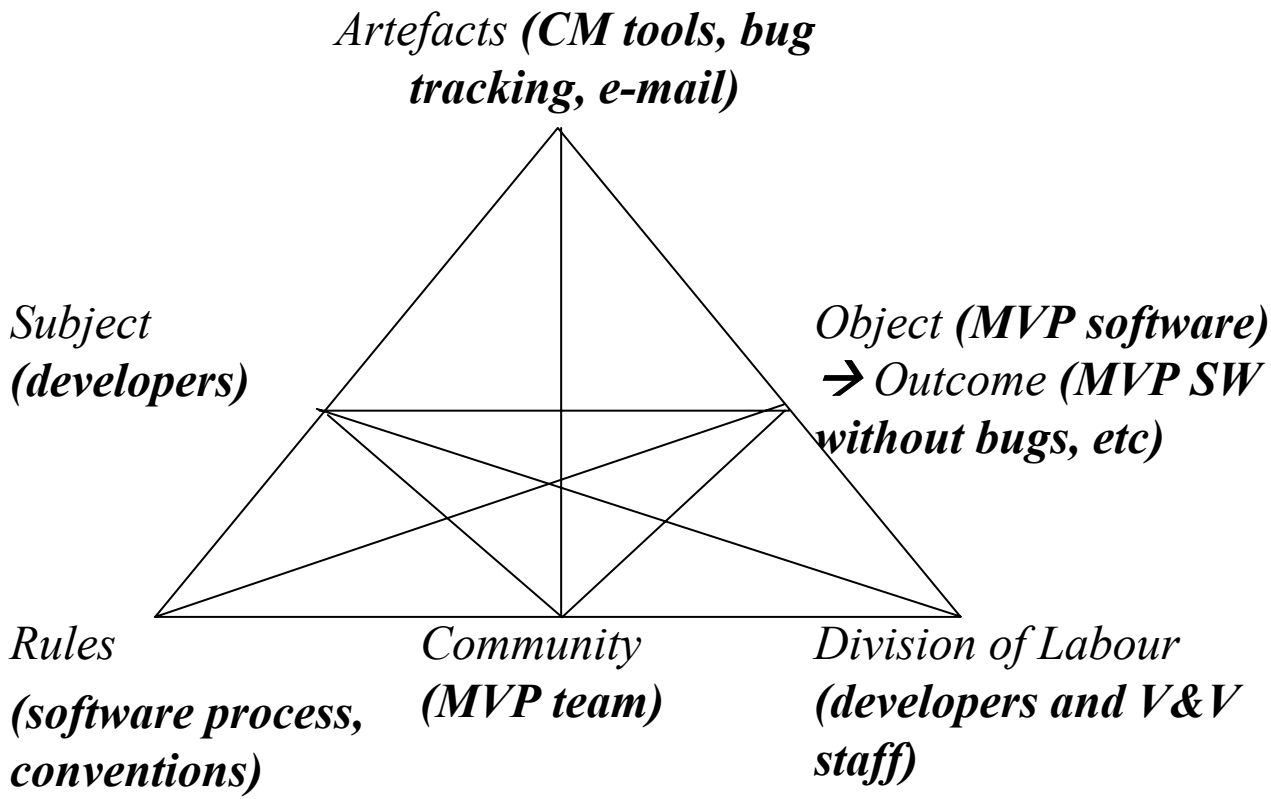
Cleidson R. B. De Souza and David F. Redmiles

Department of Informatics
School of Information and Computer Science
University of California, Irvine
{cdesouza, redmiles}@ics.uci.edu

Orientation

- One author did a student internship in a sizable software organization.
  - "Sizable" = 1M lines of code; 10 subsystems; ~35 members
- The organization's objective was to evolve a software system ("MVP") and leave no "bugs."
- Even with good software tools and methods, problems impeded the ease of achieving the objective.
- As researchers, our objective was twofold
  - Make suggestions about tools and methods
  - Learn about methods to best analyze the situation and make suggestions

AT Analysis


Artefacts ***(CM tools, bug tracking, e-mail)***
Subject ***(developers)***
Object ***(MVP software)***
→ *Outcome* ***(MVP SW without bugs, etc)***
Rules ***(software process, conventions)***
Community ***(MVP team)***
Division of Labour ***(developers and V&V staff)***

Observed Tensions and Conflicts

- Between the object and the community
  - E.g., an individual's objective is a modified module or subsystem, but the modification could adversely affect a) the modules of others in the community and b) the overall object, a working system.
- Between rules and the community
  - E.g., one rule suggests that a developer should perform a specific action (check-in), but he or she does not want to perform that action out of concern for the effects of the action on the rest of the community.

Observations

- Recommendations to the organization
  - Be aware of workarounds and fixes
  - Augment current tools and methods with greater awareness
- Observations on using activity theory
  - An analysis or model evolves through iterations and stops when fidelity matches observations and experiences.
    - Is this stopping function sufficient?
    - Is the similarity to object-oriented modeling an advantage to adoption of activity theory by software organizations?
  - A single activity might be consistent when observed as a single instance, but might be a source of tension when there were multiple instances of that activity
    - Does this arise with finer granularity of study?
  - Activities, especially multiple instances, share many dependencies
    - Could techniques of dependency analysis in software testing be adopted to help activity theory analysts?
    - Would such an analysis identify potential hazards for organizations adopting new software tools or methods?
  - A detailed understanding of an organization's collaborative work may be developed
    - What are the effects of reflecting this information back to the organization?
    - What if software tools could dynamically reflect this information back to the organization?

Selected Observations for Discussion

- A single activity might be consistent when observed as a single instance, but might be a source of tension when there were multiple instances of that activity
  - Does this arise with finer granularity of study (e.g. software engineers working amongst themselves)?
- Activities, especially multiple instances, share many dependencies
  - Could software dependency analysis techniques identify potential hazards for organizations adopting new software tools or methods?
- A detailed understanding of an organization's collaborative work may be developed
  - What are the effects of reflecting this information back to the organization through interventions and computer visualizations?

A Final Question

- Why am I so fascinated by tensions and conflicts?
  - Because this is where human creativity takes place (and humanity revealed)????
  - Because this is where tools and methods may be improved?