Software Economies

David F. Bacon, Eric Bokelberg,
Manu Sridharan

Yiling Chen, Ian Kash,
David Parkes, Malvika Rao
Software Development Challenges

Correctness
- Traditionally: specification, testing, verification
- Intellectual foundation: logics, proof techniques
- Problem: doesn’t scale

Budgeting
- Traditionally: centralized allocation of resources to tasks
- Problem (again): doesn’t scale

Tradeoffs
- Going back to mythical man-month
An Economic Approach

Mechanism Design
Create a market system with rewards for tasks

Game-Theoretic Analysis
Determine equilibria of system

Advantages
- Decentralized and self-regulating
- Explicit (quantitative) tradeoffs
Nascent Software Economies

Vulnerability Markets
- Mozilla Foundation
- Google

Freelance Marketplaces
- TopCoder
- VWorker

App Stores
- App Store
- Android Market
Public Software Economies

**Definition**: large-scale project, direct connection to users

**Key idea**: users bid for bug fixes / features
  - Market aggregates demand for fixes
  - Developers compete to implement profitable fixes

**Correctness equilibrium**: no bugs with sufficient demand to be worth fixing

See “A Market-Based Approach to Software Evolution,” OOPSLA Onward! 2009
Private Software Economies

Definition: smaller projects / user bases, but many projects

Competition Systems (à la Topcoder): reduce idle time, improve quality

Scoring Systems: can incentivize, e.g.,
  – Faster task completion by developers (with sufficient quality)
  – Accurate prediction by managers
  – Component reuse

Current work: game-theoretic re-design of IBM-internal scoring system
Conclusions

Proposing an **economic approach to software development**

- Decentralized and self-regulating
- Via **mechanism design** and **game theory**

Preliminary instantiations in public and private setting

Many open (exciting!) challenges and opportunities