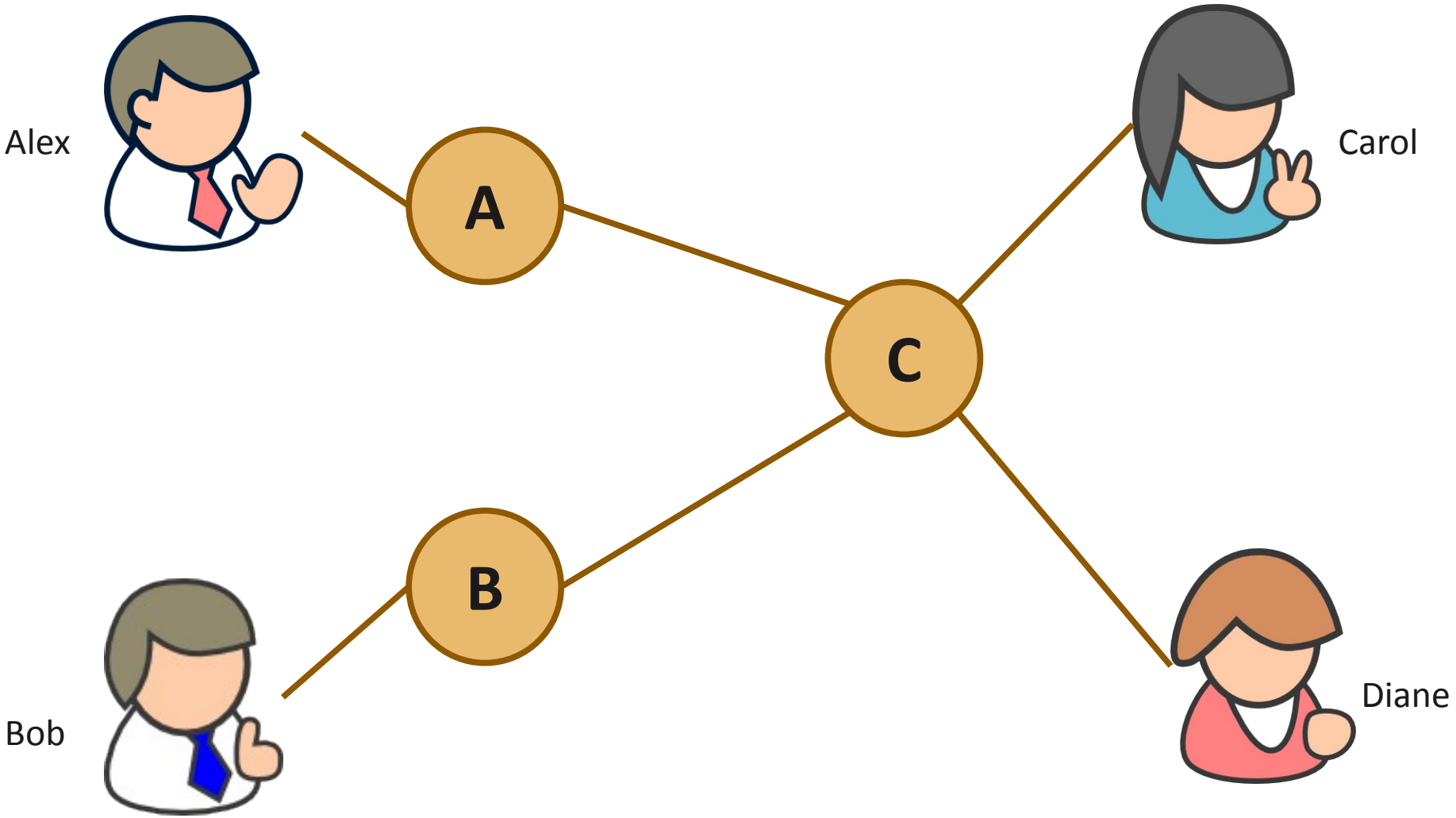
Individualized Socio-Technical Congruence: Metrics, Exploration, and Impact

Patrick Wagstrom
patrick@wagstrom.net
Jim Herbsleb
jdh@cs.cmu.edu
Carnegie Mellon University
October 12, 2008

The Problem



Outline

- Why software engineering?
- Overview of socio-technical congruence
- Implications for tool development
- Individualized socio-technical congruence
- Results

Why Software Engineering?

- Complex non-routine tasks
- Knowledge work – amenable to distributed work
- Development tools capture rich data

STC: The Metric

- Utilize matrix representation of networks

$$T_A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Task assignment network, maps individuals (rows) to tasks (columns)

$$T_D = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Task dependency network, maps tasks to tasks

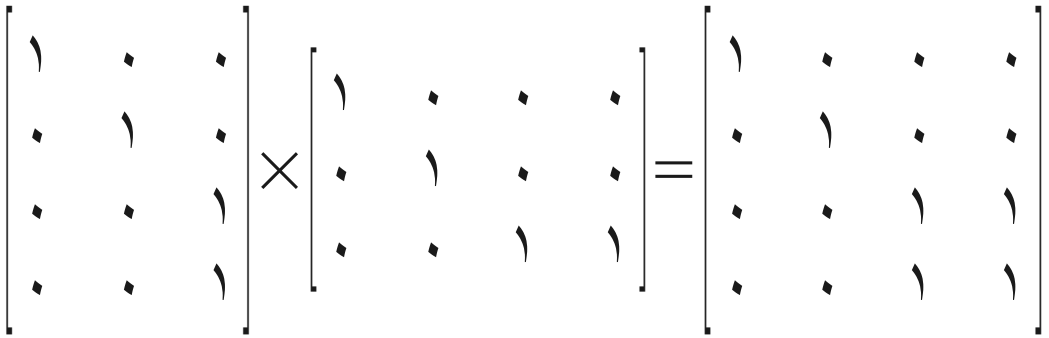
$$C_A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Actual communication network, shows who communicated with whom

STC: The Metric

- Matrix operations inform us of relations

$T_A \times T_A'$ Who works together on the same task



$T_A \times T_D \times T_A'$ Who works on tasks that are dependent on my tasks

Coordination Requirements Network, C_R

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

STC: The Metric

$$C_A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Matrix reflecting actual communication in the organization

$$\frac{\sum \left(C_A \wedge C_R \right)}{\sum C_R}$$

Proportion of coordination requirements that are mirrored in the actual communication network.

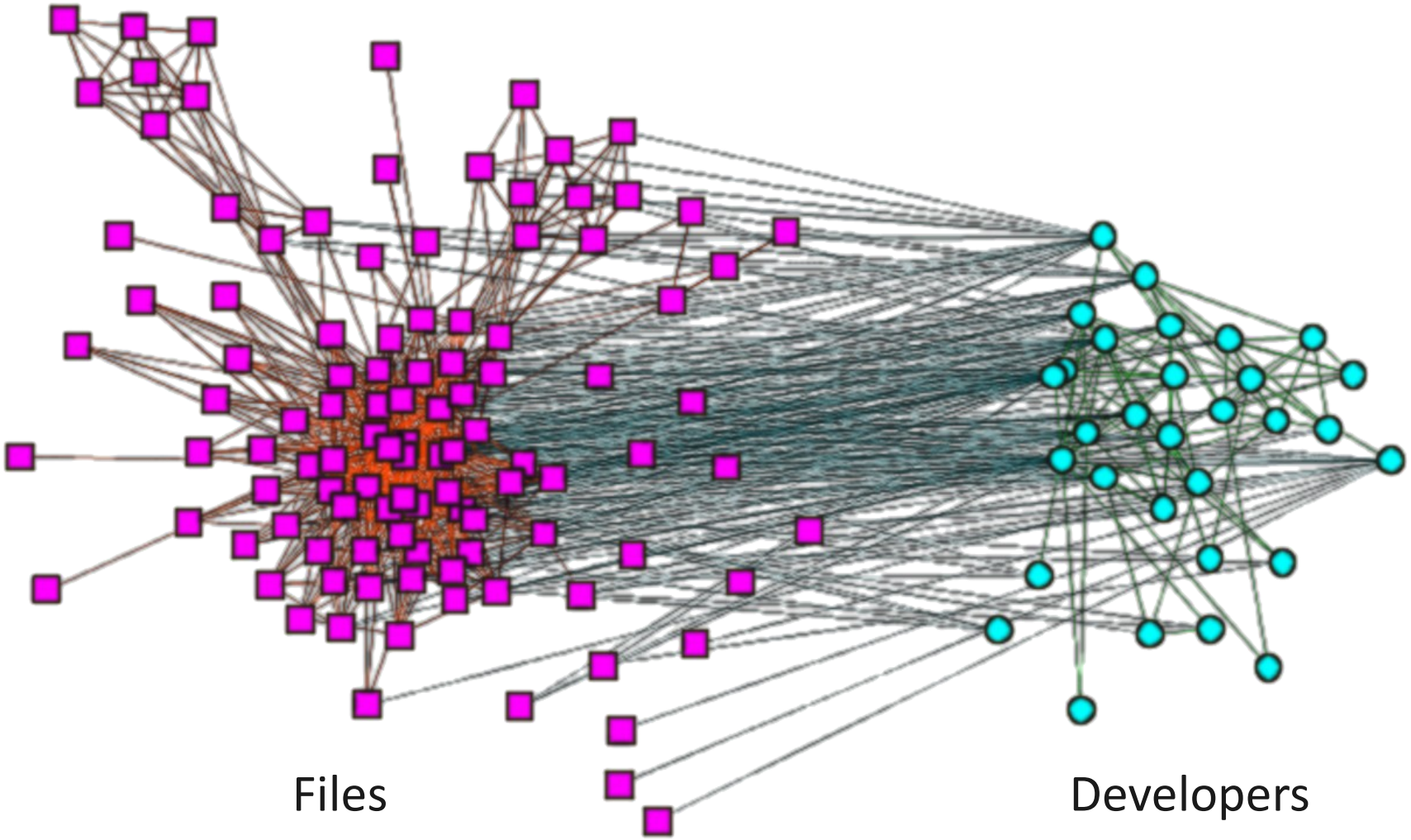
$$\underbrace{\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}}_{C_A} \wedge \underbrace{\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}}_{C_R} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \qquad \frac{6}{10} = 0.6$$

Socio-Technical Congruence

Implications for Tool Development

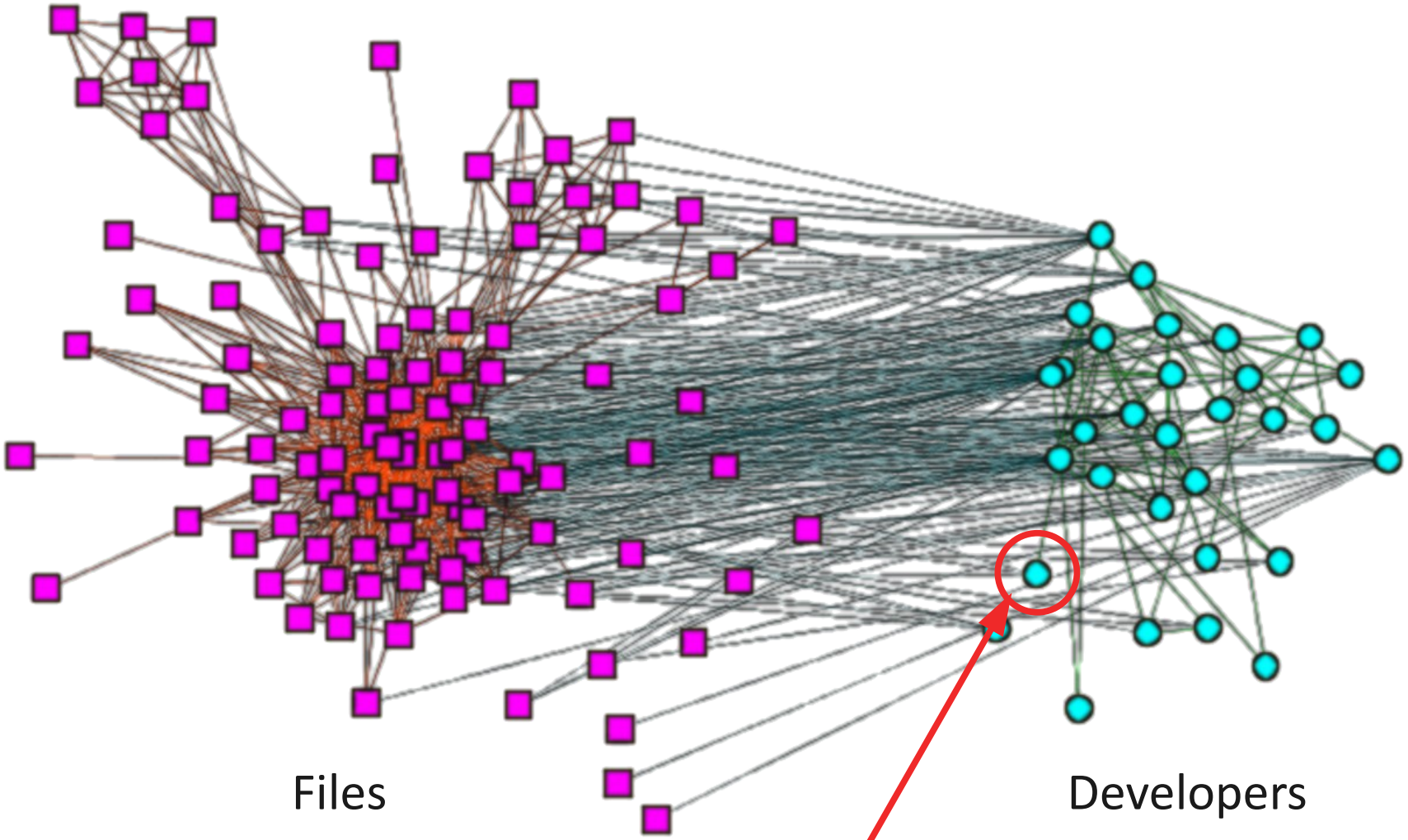
- How do I know I'm talking to the right people?
- If we understand where gaps in communication exist:
  - We can direct communication to address requirements
  - Improve overall team productivity

A Real World Network



Congruence = 0.24

A Problem of Individual Motivation



What does all this mean for an individual developer?

Individualized STC

- Calculate C_R as before
- Look only at the communication and requirements surrounding an individual

$$UIC_i = \frac{\sum \left(C_R[i,] \wedge C_A[i,] \right) + \sum \left(C_R[,i] \wedge C_A[,i] \right)}{\sum C_R[i,] + \sum C_R[,i]}$$

Required Communication

Matched Communication

Unweighted Individual Congruence

Weighted Individual Congruence

- STC and UIC use a dichotomized C_R matrix
- Instead, if we choose not to dichotomize in the numerator, we get an idea of the relative significance of the coordination requirements

Evaluating the Metric

- Validate STC metric on a data set featuring a radically distributed team
- Examine the impact of individualized STC on time to resolve software defects
- Better understand the constituent parts of STC

Open Source Software Engineering

- Licenses and work practices characterize open source
- Completely open communities
  - Anyone can join the community and contribute code
- Run as a meritocracy
  - After a period most anyone can become a direct contributor
- Open tracking of software defects

Our Sample

- 1200+ developers
- 800+ projects
- 200,000+ bugs
- 3,000,000+ messages
- 280,000+ files
- 10 years of history

GNOME™

Prediction Model

- Attempting to predict the log of the time to resolve software defects (bugs)
- Use common predictor variables:
  - Developers involved
  - Changes made
  - Comments on bug
- Also include our individualized STC metrics
- Because we want shorter resolution times negative coefficients are beneficial

UIC Prediction Results

	Estimate	Error	T Value	P Value
Intercept	2.2656	0.0589	38.4770	< 0.0001
Developers	0.6110	0.0295	20.7100	< 0.0001
Deltas	0.1499	0.0053	28.0850	< 0.0001
Comments	-0.0017	0.0038	-0.4490	0.0654
UIC	-1.1147	0.0789	-14.1270	<0.001

STC in all forms is a fraction – let's break it apart for me detail

UIC Prediction Results

	Estimate	Error	T Value	P Value
Intercept	1.7809	0.0576	30.9430	< 0.0001
Developers	0.5822	0.0300	19.3870	< 0.0001
Deltas	0.1462	0.0054	26.9010	< 0.0001
Comments	0.0007	0.0039	0.1680	0.8664
Coord Req	0.0285	0.0033	8.5360	<0.001
Matched Comm	-0.0448	0.0058	-7.7280	<0.001
Extra Comm	-0.0114	0.0036	-3.2080	0.0013

Communication that doesn't match up with coordination requirements still has a beneficial effect on the time to resolve software defects!

WIC Prediction Results

	Estimate	Error	T Value	P Value
Intercept	2.0986	0.0510	41.1430	< 0.0001
Developers	0.6220	0.0294	21.1410	< 0.0001
Deltas	0.1457	0.0053	27.3460	< 0.0001
Comments	-0.0016	0.0038	-0.4170	0.6770
WIC	-0.0021	0.0001	-14.5580	<0.001

- Most coefficients are similar to UIC, with exception of WIC
- WIC is unbounded, and frequently scores in 100's or 1000's for developers

WIC Prediction Results

	Estimate	Error	T Value	P Value
Intercept	1.7580	0.0580	30.2830	< 0.0001
Developers	0.6001	0.0301	19.9260	< 0.0001
Deltas	0.1403	0.0055	25.6870	< 0.0001
Comments	0.0012	0.0039	0.3120	0.7553
Coord Req	0.0246	0.0028	0.8663	<0.001
Matched Comm	-9.19E-05	1.09E-05	-8.4280	<0.001
Extra Comm	-0.0060	2.18E-03	-2.7550	0.0059

The magnitudes are dramatically different because there is no way to weigh communications that don't align with coordination requirements.

Major Contributions

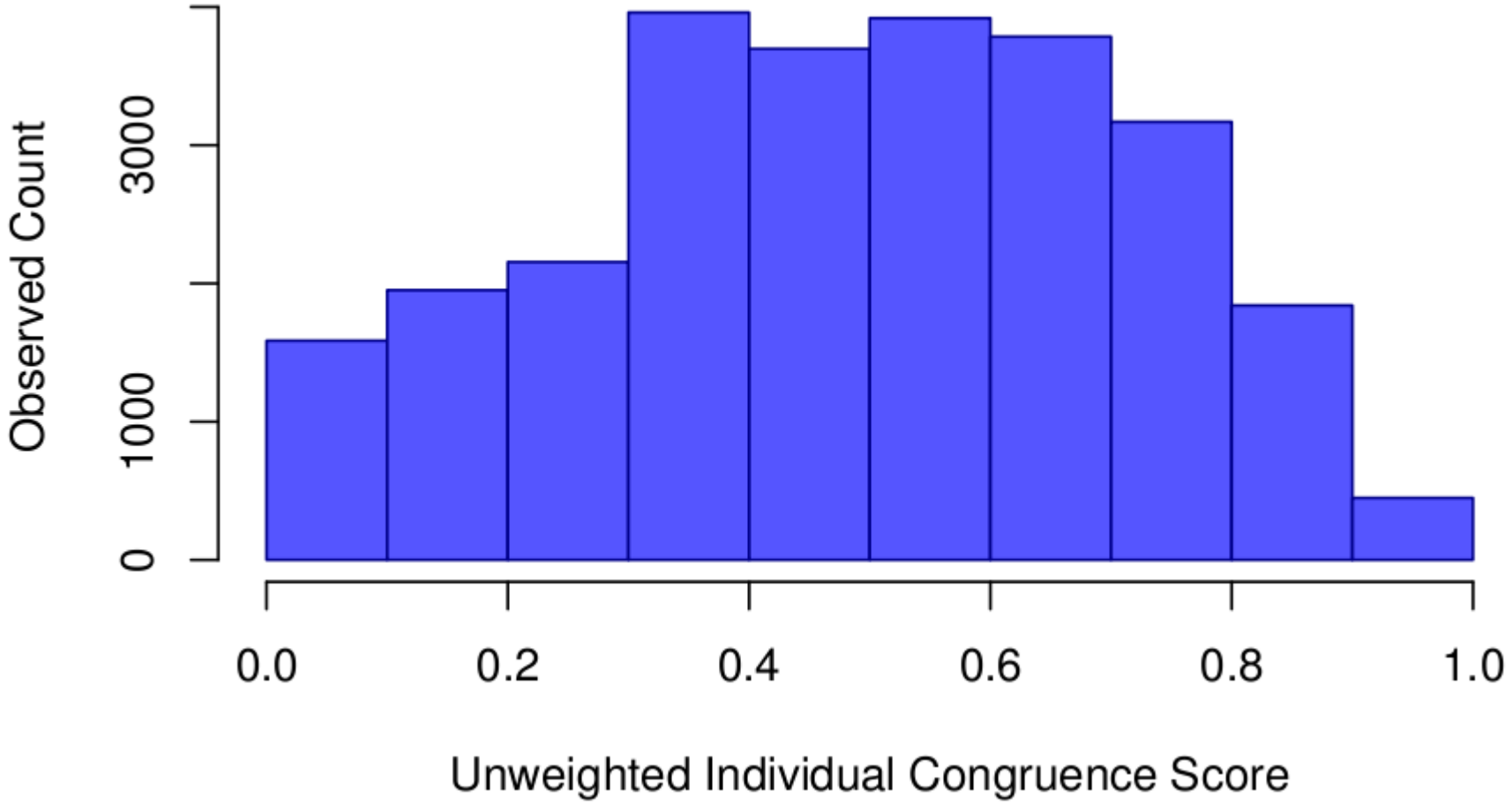
- Replicated original* on a widely distributed open source software development team
- Formalized the calculation of congruence on an individual level
- Shown that extra communication has a beneficial relation to time to solve software defects
  - However, it does not appear that communication through the bug tracker is beneficial

* Cataldo, Wagstrom, Herbsleb and Carley. Identification of Coordination Requirements: Implications for the Design of Collaboration and Awareness Tools. In Proceedings of 2006 Conference on Computer Supported Cooperative Work. November 2006, Banff, AB, Canada. ACM Press.

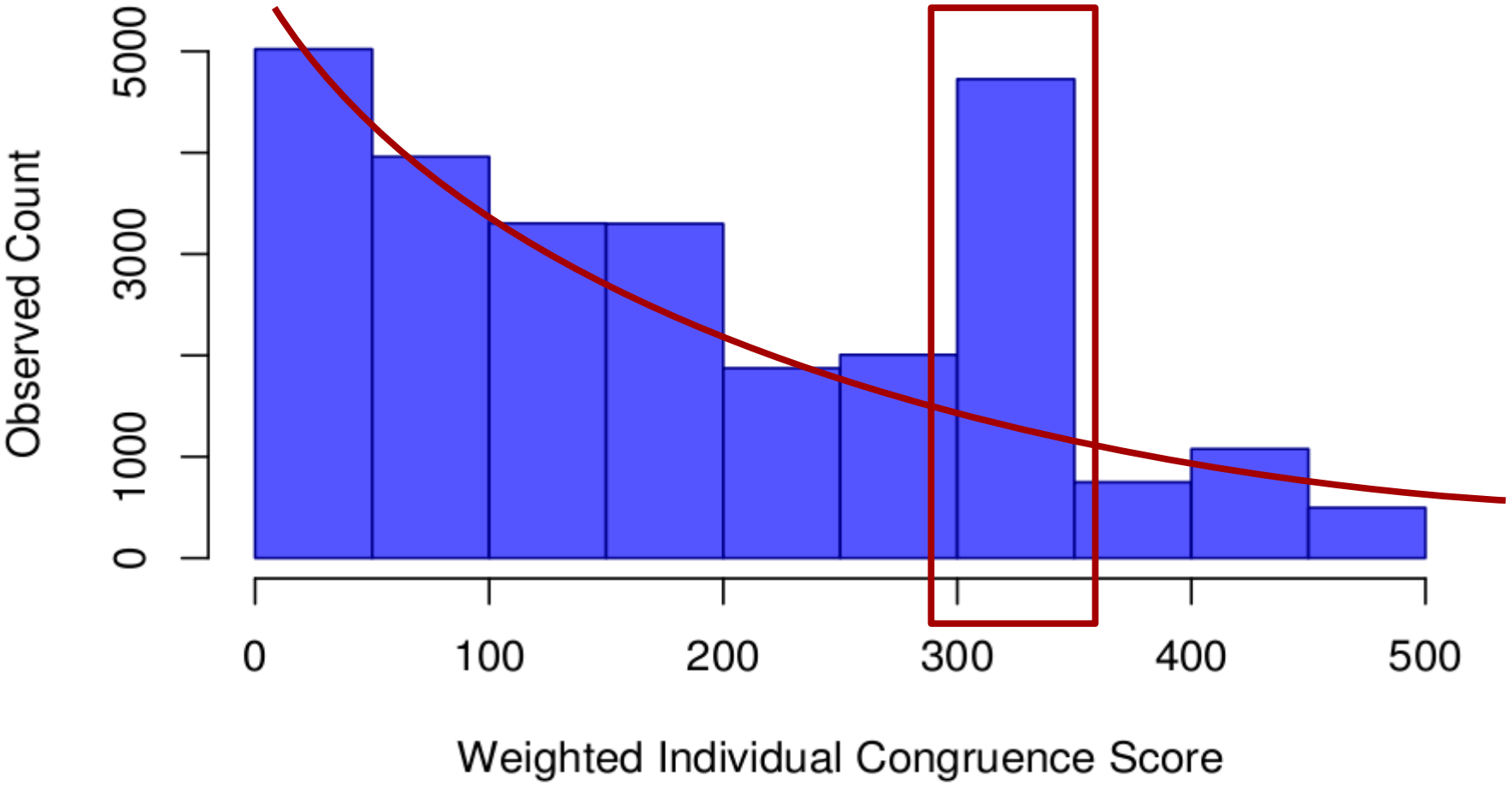
Questions/Comments?

This slide intentionally left blank – old backup stuff follows

Distribution of UIC



Distribution of WIC



Why did I leave in comments?

- Comments was not significant in the models
  - Why was it left in?
- Conventional wisdom in software engineering – document everything related to the bug in the bug tracker