A Highly Selective, Deeply Biased, and Mildly Heretical View of Software Engineering

Jim Herbsleb
School of Computer Science
Carnegie Mellon University

Outline

- Software engineering isn't
- Our conception of software engineering is pathologically narrow
- Where humans fit into the picture
- The data tsunami
- Research examples:
  - Coordination – results and theory
  - Open source ecology

Is “Software Engineering” Really Engineering?

- Engineering: “the disciplined application of scientific knowledge to resolve conflicting constraints and requirements for problems of immediate, practical significance.”
- “In Chem E, when I needed to design a heat exchanger, I used a set of references that told me what the constants were . . . and the standard design equations. . . .”
- “the critical difference is the ability to put together little pieces of the problem that are relatively well known, without having to generate a custom solution for every application . . .”

A New Flavor of Engineering?

- How to advance the field?
  - Should we aspire to be a “typical” engineering discipline?
  - Do we require a different approach?
- “Essential” (as opposed to “accidental”) problems
  - Complexity*
  - Conformity*
  - Changeability*
  - Invisibility*
  - Zero cost reproduction and transmission
  - Design is manufacture

**No Silver Bullet: Essence and Accidents of Software Engineering* by Frederick P. Brooks

Software Is In Everything

- Typical luxury car has 70-80 processors
  - Infotainment
  - Engine function
  - Suspension
  - Brakes
  - Steering
- Increasingly, new features and competitive advantage come from software
- The behavior of the environment is increasingly determined by software

Lessig's Insight

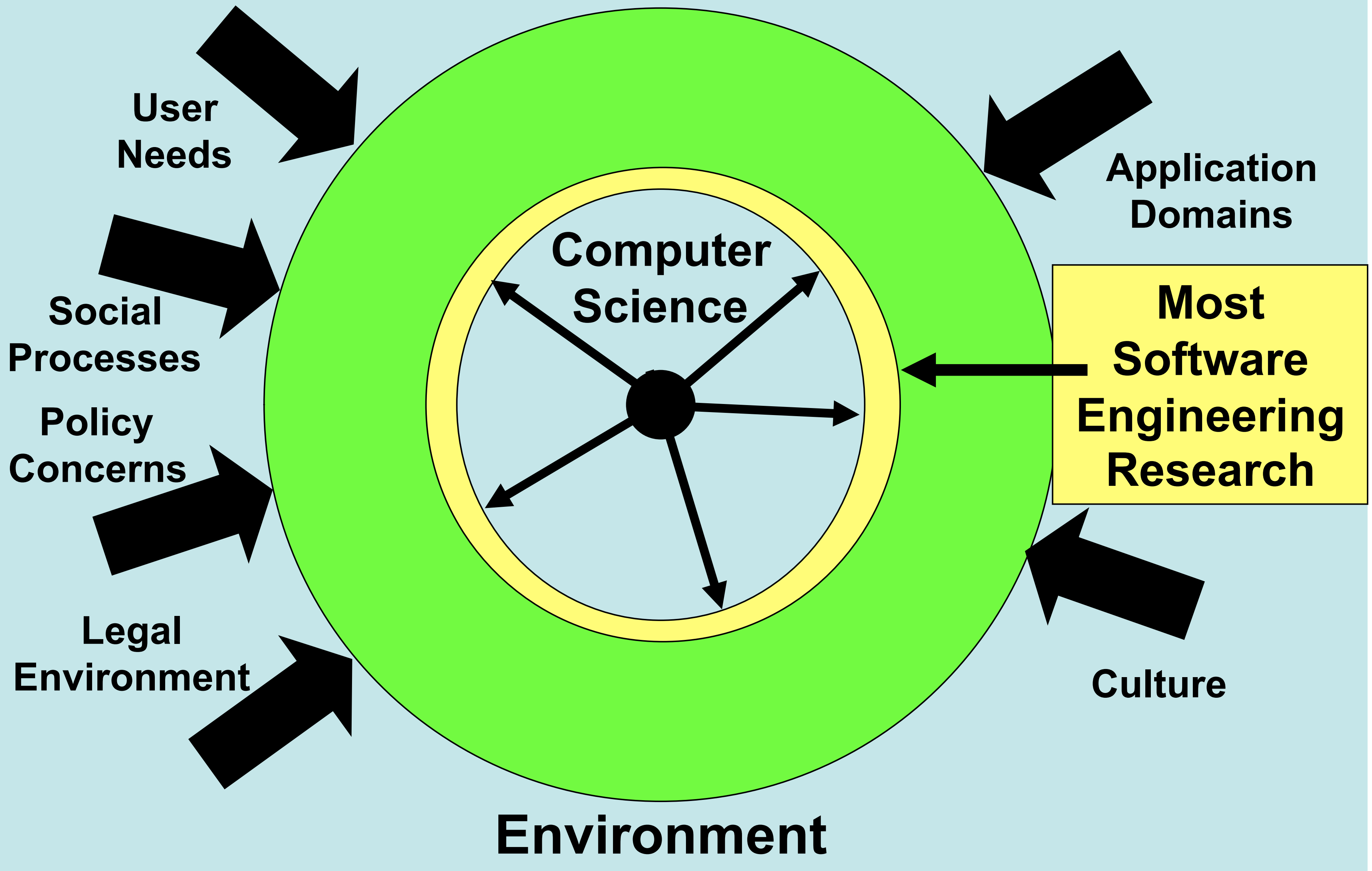
- Four traditional modes of control:
  - Law
  - Norms
  - Markets
  - Architecture
- And now . . . Code
  - Design of code determines possibilities for conduct, commerce, political action, social interaction, creativity . . .
- Many ethical and moral questions
- But also many sociotechnical questions
  - How to design a system to achieve a policy objective?
  - What side effects? (e.g., DRM)
  - What objectives are achievable?

Code and Other Laws of Cyberspace by Lawrence Lessig

Humans in SWE: Role and Scale

	Individual	Group/Team	Organization	Business Milieu
Human as User	HCI	CSCW	IT	Supply Chains Web Services
Human as Designer/ Developer				

The Intellectual Challenge

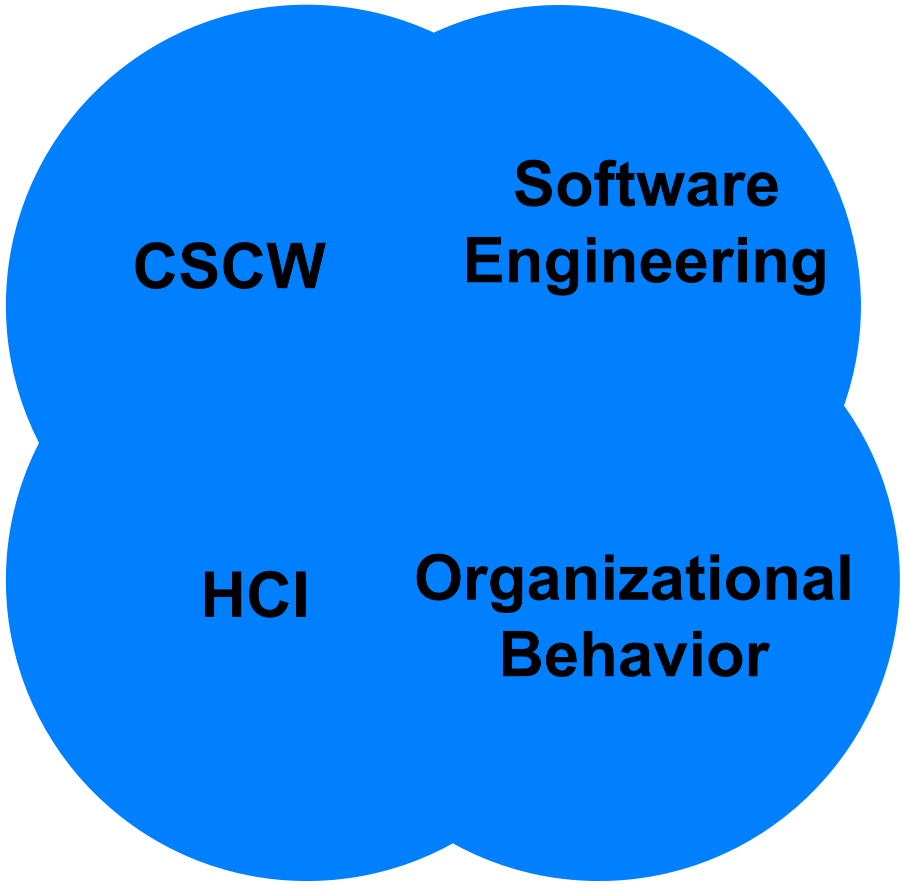


Humans in SWE: Role and Scale

	Individual	Group/Team	Organization	Business Milieu
Human as User	HCI	CSCW	IT	Supply Chains Web
Human as Designer/ Developer	ESP Psych of Prog.	Software Teams IPD Teams	IT Groups Product Custom SaS	Open Source Ecologies

End User Programming

Four Disciplines?



Carnegie Mellon
School of Computer Science

The Data Tsunami

- Software projects typically keep a very detailed record of human activity
- Version control (VC) system
  - Maintains all changes to all files – each checkin is a “delta”
  - For each delta, it records
    - Login of the person submitting the code
    - Date and time
    - Size
    - Actual code submitted (“diff”)
- Modification request (MR) system
  - Users, testers, developers request changes
  - Records who, when, what about the request
  - Records all steps in workflow
  - May have link to deltas that implement change
  - Generally support asynchronous discussions

In the Best Case

- Data creates a very detailed record of
  - Precisely what was done
  - Who did what when
  - What were the dependencies of the work
  - Why was it done
  - Discussions about each unit of work
- May have similar record for all phases
  - Requirements and design often put under change management and version control
- Lends itself to network analyses
  - Nodes: people, files, MRs, deltas, etc.
  - Links: task assignment, dependencies, things used together, etc.

Research Examples

- Coordination and Congruence
- Theory of coordination
- Open source ecology

Measuring Coordination Requirements

- Dependencies among tasks:

  matrix D where $d_{ij} \neq 0$ means that task i and task j are dependent **Files changed together**

- Assignments of workers to tasks:

  matrix A where $a_{kl} \neq 0$ indicates that worker k is assigned to task l **Developer modified file**

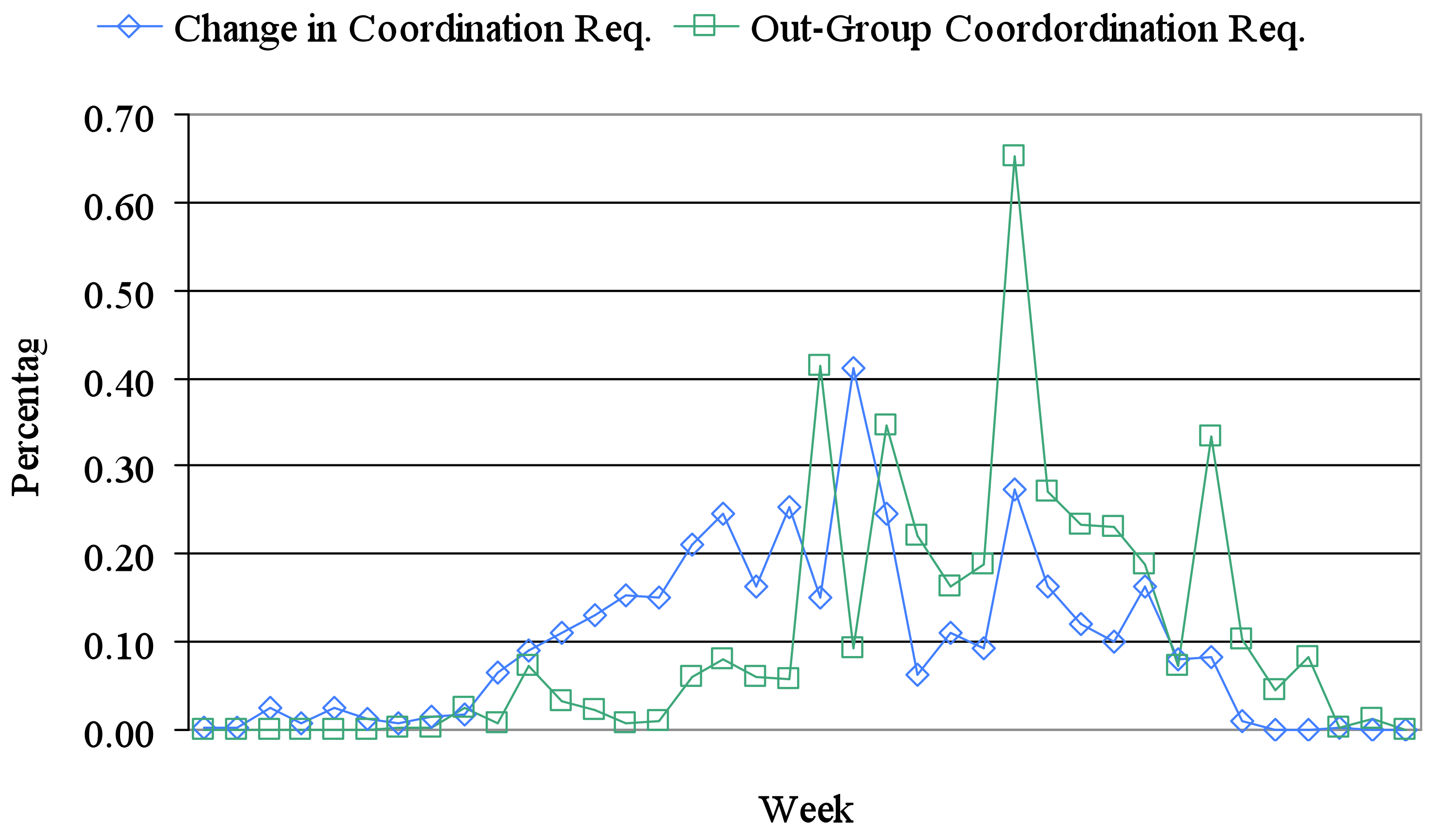
- Coordination requirements:

  $ADA^T = R$, where $r_{mn} \neq 0$ indicates that worker m and worker n have dependencies in their tasks

Coordination Requirements for some unit of work or period of time

From Cataldo, et al, 2006

Volatility in Coordination Requirements



From Cataldo, et al, 2006

Measuring Congruence

Coordination Requirements (R)

Coordination Behavior (B)

$$\begin{vmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \longleftrightarrow \begin{vmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

- Team structure
- Geographic location
- Use of chat
- On-line discussion in MR system

From Cataldo, et al, 2006

Summary of Findings

- Each type of congruence is associated with shorter development times
- We can measure coordination requirements and congruence
- Coordination requirements are volatile and extend beyond the team

What kind of theory can account for these results?

From Cataldo, et al, 2006

Theoretical Views of Coordination

- Coordination theory (Malone & Crowston)
  - Match coordination problems to mechanisms
  - E.g., resource conflict and scheduling
- Distributed Cognition (Hutchins, Hollan)
  - Computational process distributed over artifacts and people
- Distributed AI (Durfee, Lesser)
  - Partial global planning
  - Communication regimens
- Organizational behavior
  - Stylized dependency types, e.g., sequential, pooled
  - Coordination regimens that address each type

Technical Coordination Modeled as CSP

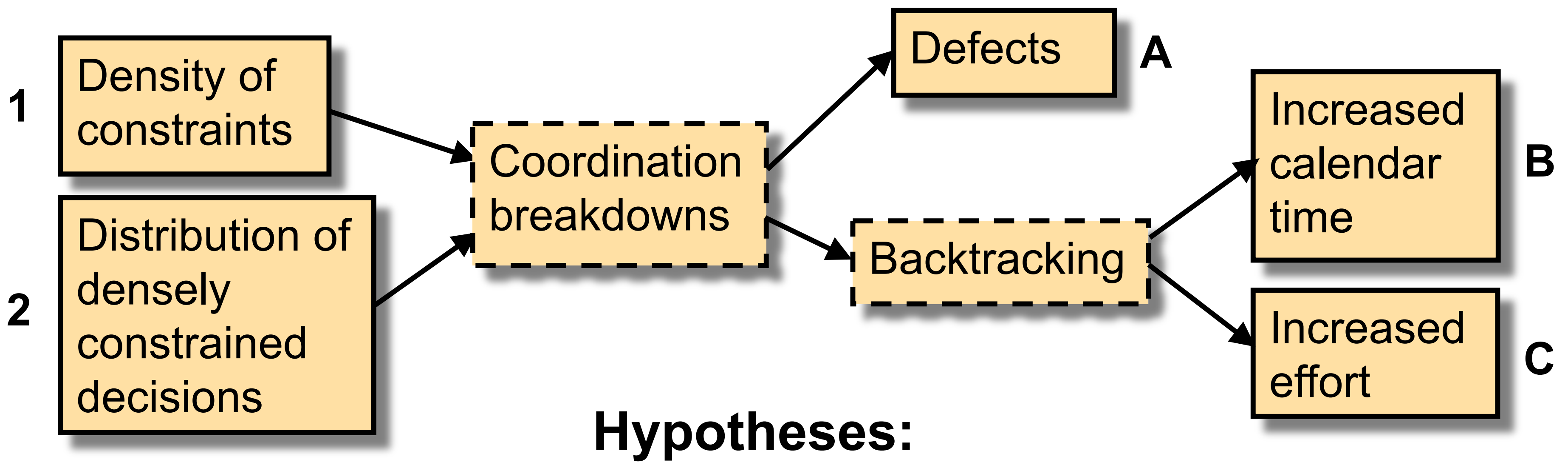
- Software engineering work = making decisions
- Constraint satisfaction problem
  - a project is a large set of mutually-constraining decisions, which are represented as
  - n variables $x_1, x_2, . . . , x_n$ whose
  - values are taken from finite, discrete domains $D_1, D_2, . . . , D_n$
  - constraints $p_k(x_{k1}, x_{k2}, . . . , x_{kn})$ are predicates defined on
  - the Cartesian product D_{k1} x D_{K2} x . . . x D_{kj}.
- Solving CSP is equivalent to finding an assignment for all variables that satisfies all constraints

Formulation of CSP taken from Yokoo and Ishida, Search Algorithms for Agents, in G. Weiss (Ed.) *Multiagent Systems*, Cambridge, MA: MIT Press, 1999.

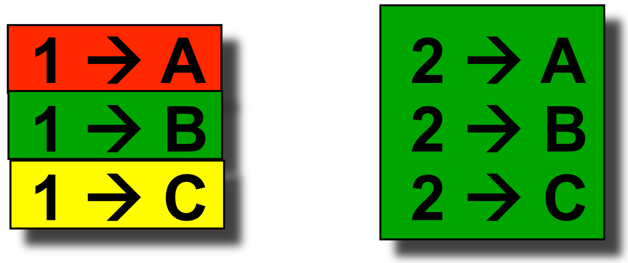
Distributed Constraint Satisfaction

- Each variable x_j belongs to one agent *i*
- Represented by relation *belongs*(x_j,*i*)
- Agents only know about a subset of the constraints
- Represent this relation as *known*(P_l, *k*), meaning agent *k* knows about constraint P_l
- Agent behavior determines global algorithm
- For humans, global behavior emerges

Model, Hypotheses, and Results



Hypotheses:



Carnegie Mellon
School of Computer Science

From Micro to Macro: The Eclipse Ecology

- Integrated Development Environment
- Plug-in architecture
- History
  - Initially developed by OTI group at IBM for internal use
  - Intent to provide to a few partners as well
- Decision to open source
  - More competition among vendors
  - Anyone could get in the game
  - Offload some development effort
- Organization
  - Consortium, IBM still in control
  - Foundation, IBM just one member

Eclipse Ecology

- Collaboration on commodity software
- Minimal centralized functions
  - Process
  - Membership
  - Infrastructure
- Member decisions
  - What to open source
  - Where and how to participate in community
- How you collaborate and where you compete depends on software architecture
  - Change framework: community decision
  - Create plug-in: part or all can be proprietary
  - Architecture shapes community and markets

Conclusions

- Four disciplines, or blind men and the elephant?
- Important effects exist at the micro level, and software engineering is uniquely positioned to explore them
- Technical characteristics of software also influence shape and relationships of organizations, businesses, and markets