Extending the Discipline:
Beyond Computer Science

James D. Herbsleb
Carnegie Mellon University
International Conference on Software Engineering
May 19, 2005

Fundamental problem of software engineering:
How to make good technical decisions.

Computer science provides a necessary
— but not a sufficient —
intellectual foundation for understanding how to make good technical decisions.

In order to make progress, we need

Theories that will inevitably cross disciplinary boundaries

Theory-driven empirical research

Why Theory-driven Research?

- Medicine
  - Try what seems sensible (e.g., blood-letting)
  - Clinical trials: test outcomes
  - Theory-based research on fundamental processes
- Software Engineering
  - Try what seems sensible
  - Clinical trials: test outcomes
  - Theory-based research on fundamental processes

This talk will focus on two illustrative examples.

Fundamental problem of software engineering: How to make good technical decisions.

Example 1: Focus on one important property of the decision-maker

Example 2: Focus on concurrent, distributed nature of the decision-making

Properties of the decision-maker . . .

The human brain was "designed by natural selection to solve adaptive problems faced by our hunter-gatherer ancestors" (Cosmides & Tooby)

This does not include writing software.

This does not include mentally simulating run-time behavior of software.

This does include navigating a complex social world.

A Puzzle . . .

Edsger Dijkstra

On the Cruelty of Really Teaching Computer Science

“I have now encountered programs wanting things, knowing things, expecting things, believing things, etc.”

“is an enormous handicap for every computing community that has adopted it”

A Puzzle . . .

A few more quotes . . .
. . . from experienced gurus

"The service is going to try to send the message, so the protocol needs to understand the state of the transaction."

"The application understands [a particular protocol], and it knows where to put operations in the message."

Why do gurus speak this way?

Why speak of software as if it were human?

"Anthropomorphic semantics" is not part of most computer science curricula . . .

Just sloppiness?

But that implies that anthropomorphic description is easier for some reason – not really an explanation . . .

Why do we prefer to manipulate objects with our hands . . .
. . . rather than with our feet?

We're taking advantage of a powerful built-in capability

Manual dexterity: structure of hands, allocation of cerebral cortex

Anthropomorphic explanation: powerful cognitive capability understand, explain, predict behavior

Naïve Psychology Capability

- Built on specialized brain structures
- Specific cognitive capacity for explaining and predicting complex intelligent behavior
- Explains behavior in terms of beliefs and desires
  - E.g., "I went to the workshop because I believed I would hear about the latest research, and I want to stay up to date in my field."
- Naïve psychology capability, e.g.,
  - Recall information person has been exposed to
  - Infer what a person must have known
  - Current state of person's beliefs
  - Infer what a person must be trying to do
  - Current desires, goals
  - Explain behavior
  - Predict behavior

Naïve Psychology and Software

- Need to understand highly complex run-time behavior of interacting components
  - Understanding dependencies among components
  - Debugging
- Highly abstract representational medium
  - E.g., "Component *A* wants to send a message to component *B*"
- Beliefs and desires express complex relationships
  - E.g., "Ah! Component *X* thinks file *Y* is corrupted!"
- Virtually universal human capacity
  - Vocabulary not universal, but the way it shapes perception is universal, like 3-D vision
  - Interdisciplinary teams may have little else to share

Theory-driven Research

- Evidence that the specialized cognitive capacity is used, at least in some tasks, to reason and talk about software
- Theory-driven research to understand and exploit this cognitive system, e.g.,
  - When is this capacity used?
  - Semantics of naïve psychology: explore by translating expressions into conventional terminology
  - Analysis of software errors -- when is naïve psychology misleading?
  - Try using naïve psychology for global views, adhering as closely as possible to intuitive meanings

Concurrent, distributed decision-making

Technical decisions constrain other technical decisions in complex ways that are difficult to fully understand or fully anticipate.

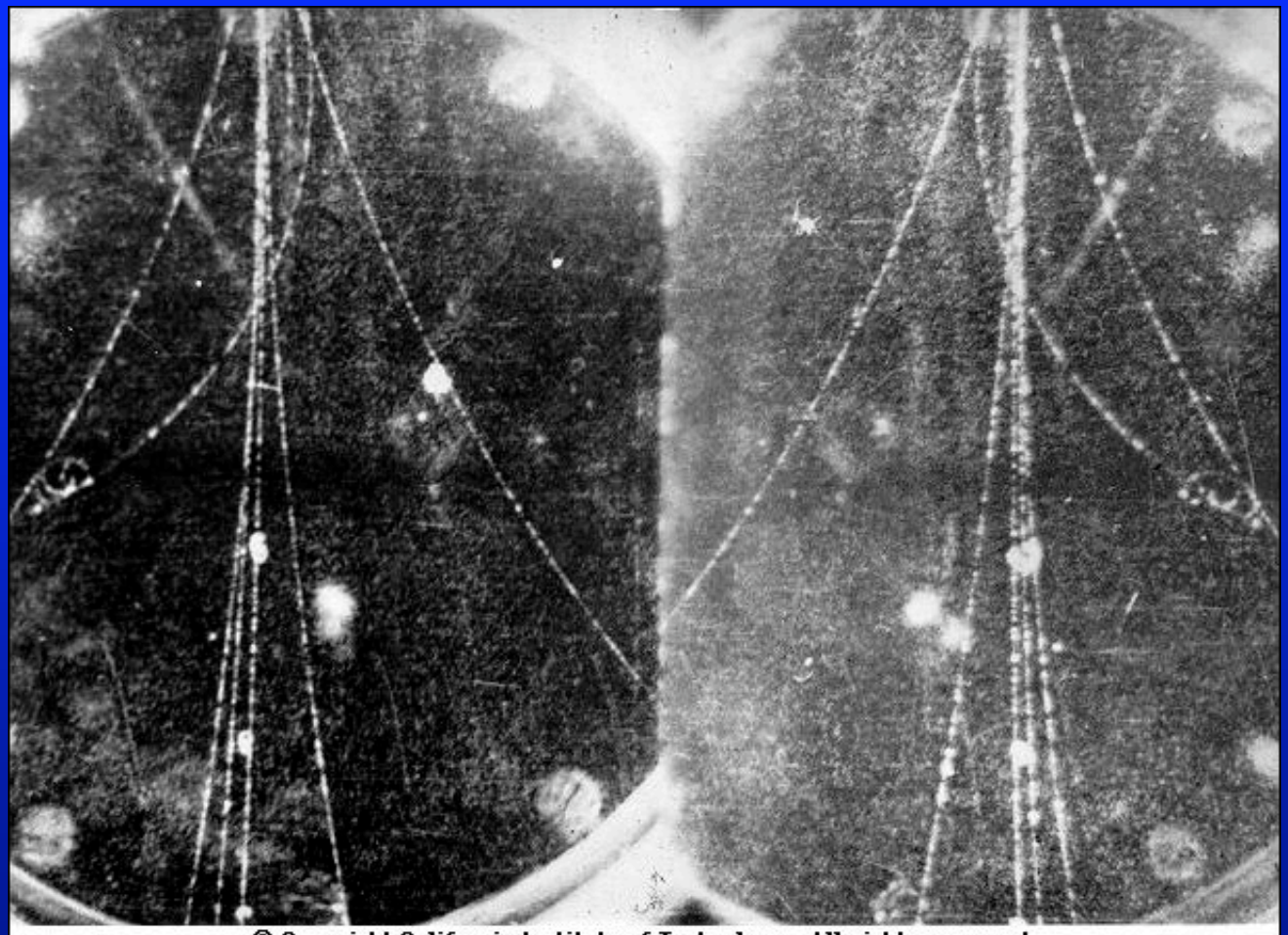
It is hard to be fully aware of all the constraints that limit my options in making technical decisions.

It is hard to be fully aware of all the ways my technical decisions constrain others, or even myself.

This is the problem of coordination: "Managing dependencies between activities" (Malone & Crowston)

Concurrent, Distributed Decision-making

How to study coordination?



Cloud Chamber

Distributed Development as Cloud Chamber

- Splitting projects apart makes coordination mechanisms visible by disrupting them
- Observing differences between co-located and distributed projects
  - Helps you infer what is happening in co-location
  - Helps find ways to assist distributed projects
- Need a theory
  - To clarify the underlying mechanisms
  - To go beyond specific observations and point solutions

Technical Coordination Modeled as CSP

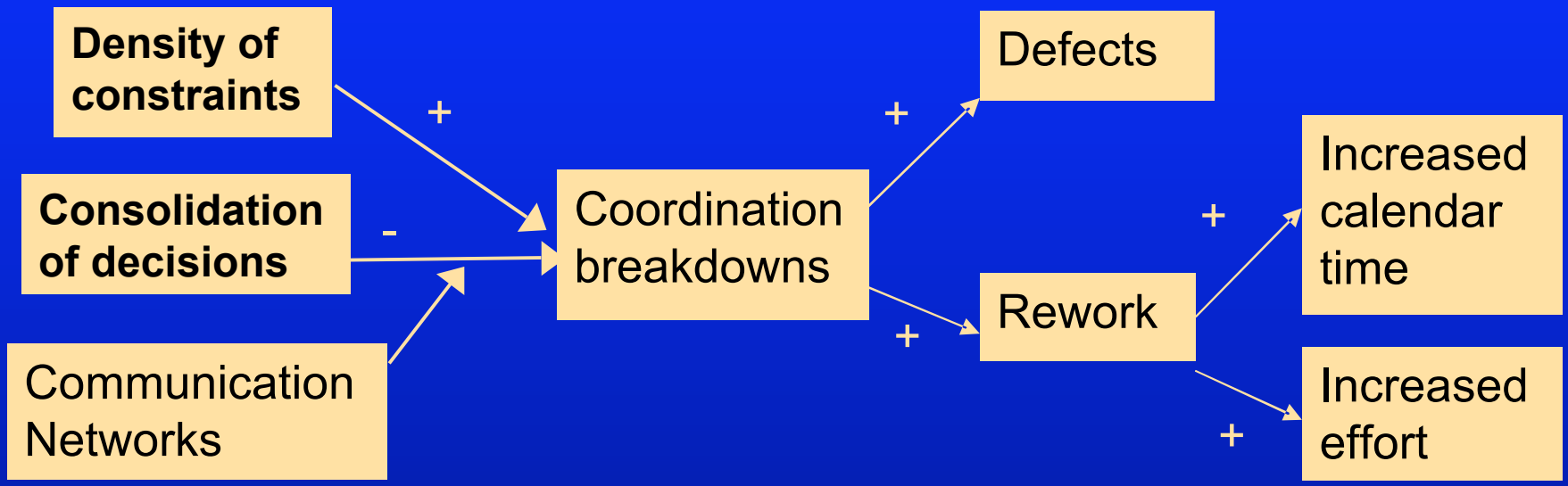
- Constraint satisfaction problem
  - a project is a large set of mutually-constraining decisions, which are represented as
  - n variables $x_1, x_2, \ldots, x_n$ whose
  - values are taken from finite, discrete domains $D_1, D_2, \ldots, D_n$
  - constraints $p_k(x_{k1}, x_{k2}, \ldots, x_{kn})$ are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

Distributed CSP formalism 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
  - Agents make decisions
  - Agents have communication networks
  - Agents communicate decisions and constraints
- For humans, global behavior emerges

Basic Hypotheses



Work with Audris Mockus

Extending the Discipline

- Need to investigate everything that importantly influences our ability to make good technical decisions.
- Crossing disciplinary boundaries is inevitable.
- We have much to learn from other disciplines.
- We will eventually develop our own style of theorizing, our own approach to theory-driven research.

Barriers to Interdisciplinary Culture

- The universal principle of interdisciplinary contempt
- The universal management principle: everything I don't understand is simple
- Administrivia
- Border defense
- Practical application on a per-paper basis