Understanding the Implementation of System Architectures in the Context of Distributed Cognition

Paper 495

Authored by:

Chris Watkins & Dr. Cihan Dagli

Motivation for Study

- System Architectures attributed to System Architect(s)
- Problem Statement
  - System Architects don't understand entire architecture
  - Architecture diverges from vision of system architects
- Root Cause
  - Distributed Cognition
- Apply Systems Thinking
  - Manage Systems of Knowledge

Objective: Increase probability of success that architecture will meet needs in problem space

Distributed Cognition

- Psychological Theory
  - Thinking extends beyond the individual and is distributed among a collective group
  - Behavior of individual and group may diverge
  - Edwin Hutchins credited for work in mid-1980's
    - "Cognition in the Wild" 1995
    - Uncoordinated system of Navy ships at port

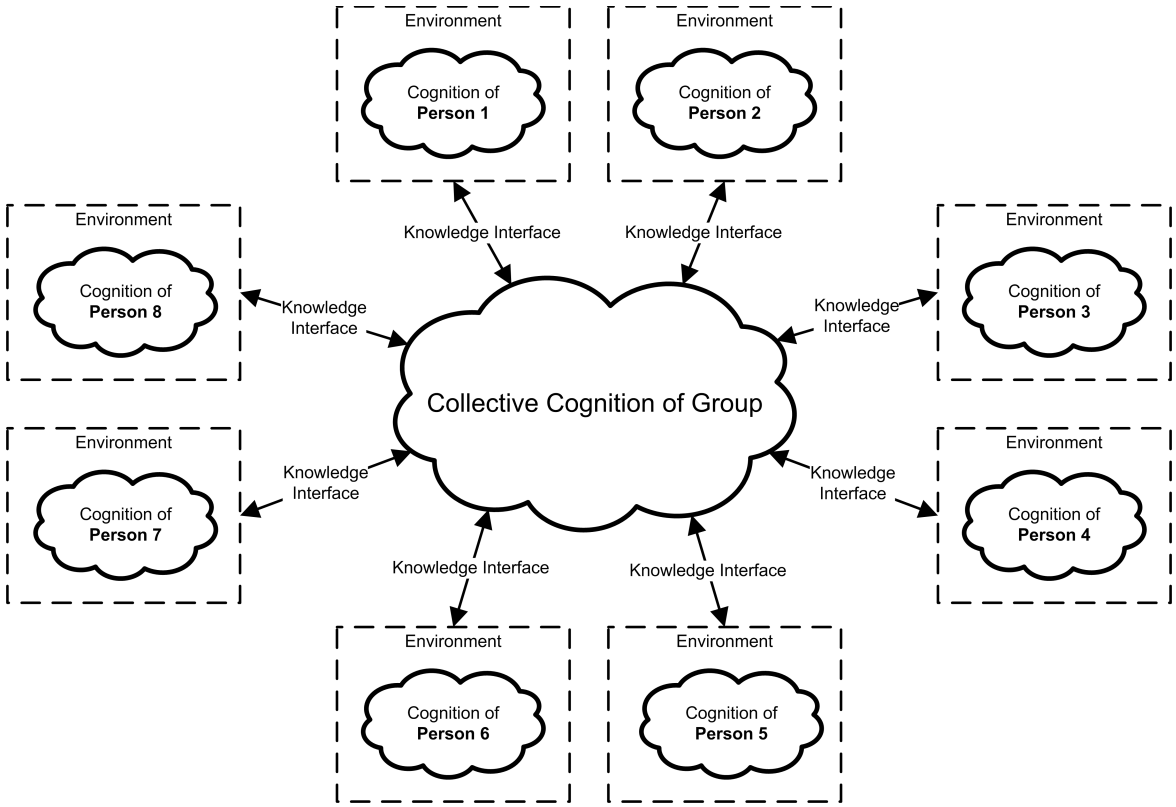
How does "Group Think" influence problem solving behaviors?

Application of Systems Thinking

- Systems of Knowledge
  - Individual's Knowledge interfaces to other knowledge in a group
  - Knowledge system interfaces defined by mental and physical artifacts

Main elements:
Individual Knowledge
Interface
Environment

2 Knowledge Systems:
Problem Space
Solution Space



Recent Study about Group Thinking (Gureckis and Goldstone 2006)

- Authors draw conclusions about conditions under which the behavior of individual agents self-organize into adaptive problem-solving group structures
- Case Studies
  - Group path formation
  - Guess a number between 0 and 100
  - “Pacman” with invisible food
- Lessons learned (aka Heuristics)
  - People are a large part of people’s environments
  - Divide and conquer: Exploration and exploitation in groups
  - More information isn’t always better
  - Influencing groups by bottom-up pressures rather than top-down rules

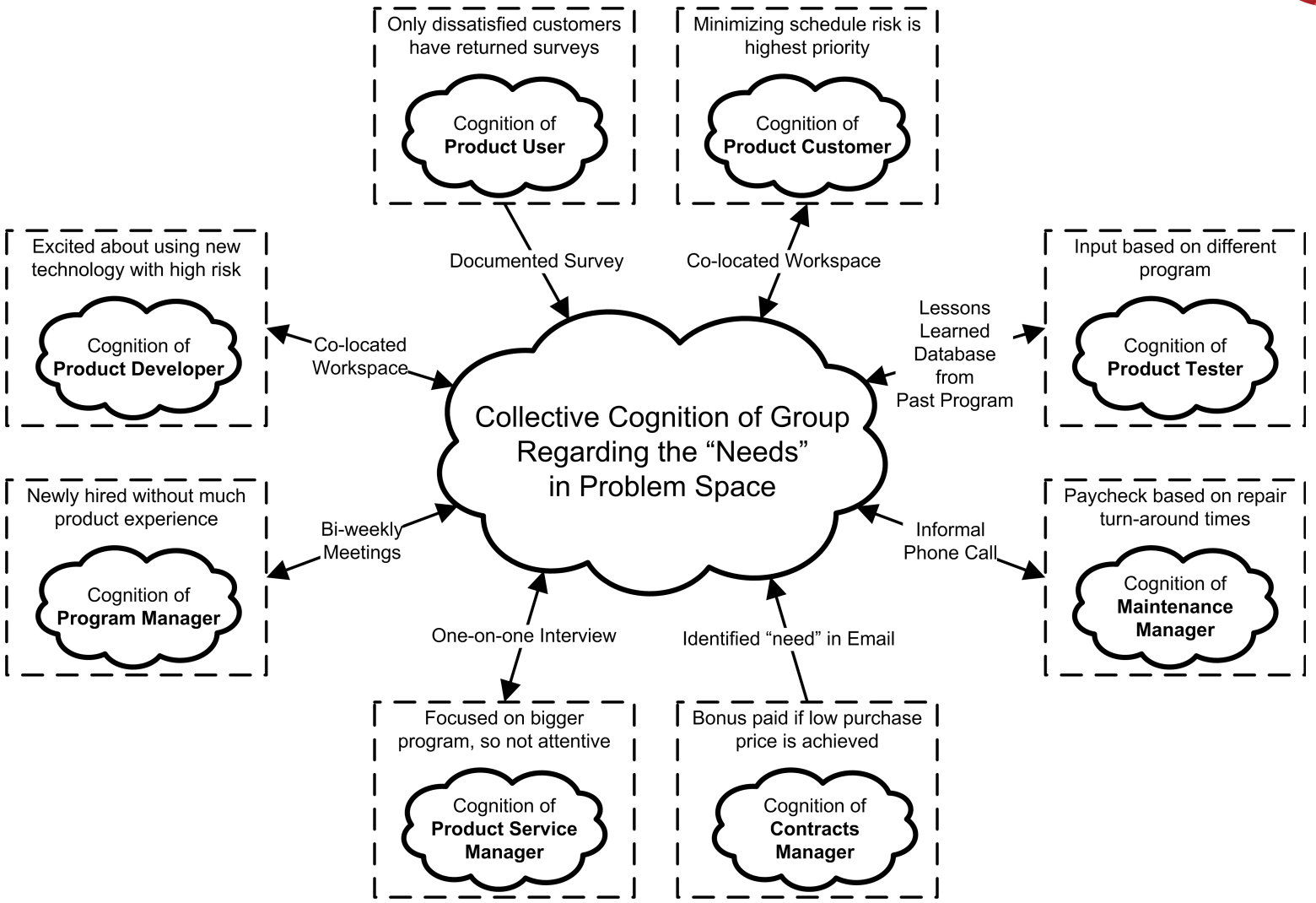
Application within Systems Engineering

- Implemented architecture is result of collective understanding of everyone that can affect architecture
- Who can affect a system architecture?
  - System Architects
  - System Designers
  - System Implementers
- Is distributed cognition beneficial or destructive?
  - Depends on how it is managed
  - Knowledge systems can be unintentional (wild) or intentional
- Requirements decomposition is not a sufficient knowledge system
- Recognizing the cost of a knowledge system
  - System costs increase proportionally to the increase in system complexity

Distributed Cognition in Problem Space

- (Saunders 2007) Biggest value in systems engineering is to sufficiently characterize the problem space
  - SCARIT process model
- (Trainor and Parnell 2007) Initial problem statement is rarely the full statement of the problem
- Distributed Cognition:
  Problem not fully understood by any single person
  - System of Knowledge

System of Knowledge: Problem Space



Example

- Proposal submitted to geographically separated, cross-cultural customer
  - RFP issued (no RFI)
  - RFP assumed to fully document problem space
  - Virtually no communication about RFP
  - Proposal Rejected
- Proposal submitted to co-located customer
  - Recognized that neither supplier, nor customer were fully cognizant of entire need in problem space
  - Co-wrote RFI
  - Responded to RFP
  - Proposal successfully met needs of customer

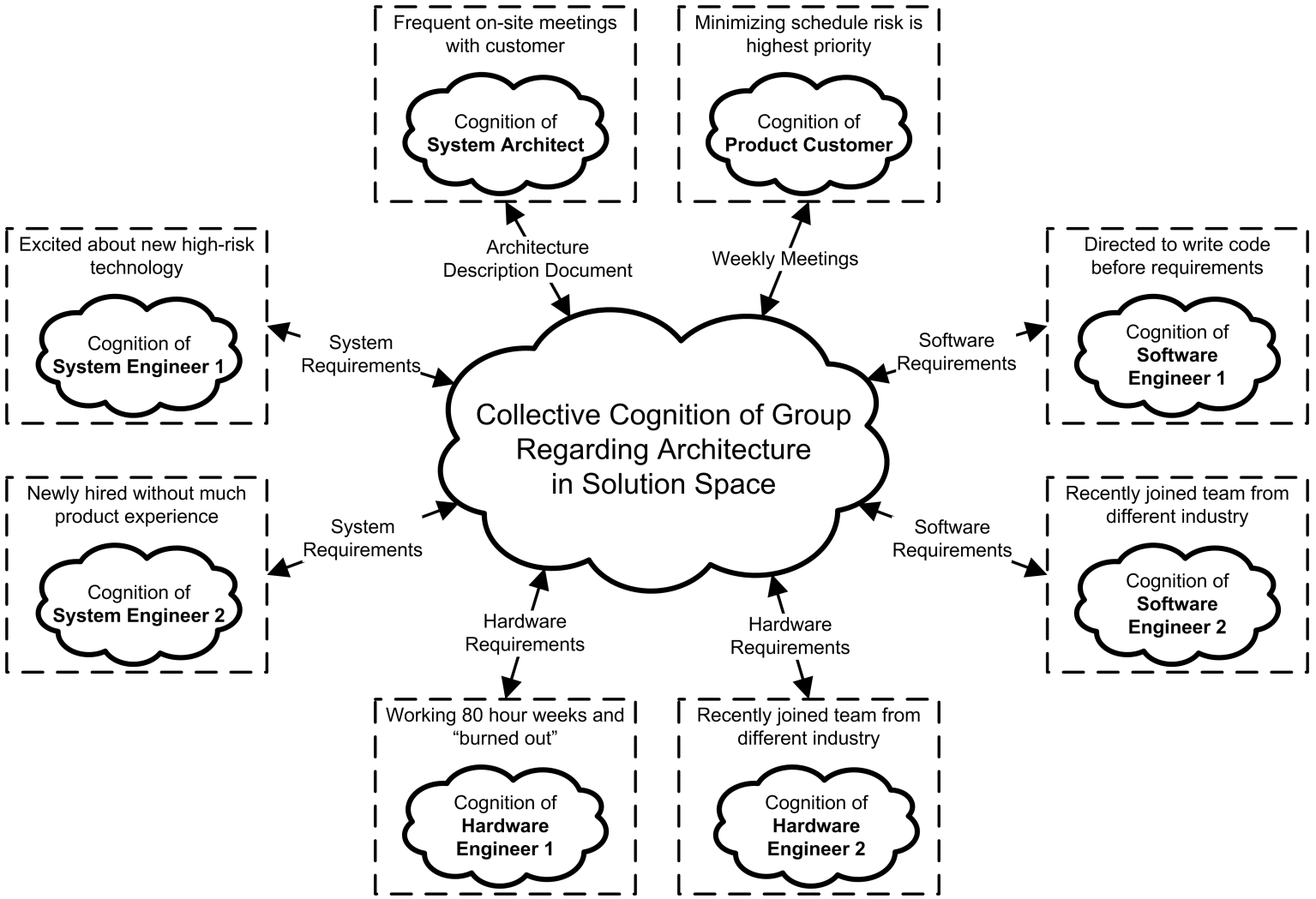
Architecting Systems in the Problem Space

- Heuristic #1: People are a large part of people's environment
  - Shared environment for stakeholders fosters better understanding
- Heuristic #2: People divide and conquer. As a group they explore the unknown while exploiting the known.
  - Shared environment for stakeholders fosters additional contributions
- Heuristic #3: More information isn't always better
  - Drowning stakeholders in information will give false sense that problem space is already fully explored and can lead to less contributions
- Heuristic #4: Groups are best influenced by bottom-up pressures rather than top-down rules
  - Personal benefit of solving problem should be clearly defined for each stakeholder up-front so that they are motivated to contribute
- Heuristic #5: System cost increases proportionally to the increase in system complexity
  - Interface with minimum number of people from each stakeholder group as necessary to capture stakeholder needs

Distributed Cognition in Solution Space

- Important to manage knowledge system in the solution space
- (Dagli, Miller and Abbott 2007) Practical techniques for managing knowledge interfaces in a geographically separated development environment
  - Two-way video over internet
  - Shared information databases
  - Affect positive group behaviors through joint venture business model

Distributed Cognition in Solution Space



Example

- Toyota's "Big Room" concept
  - Leverages communication interfaces
  - Closed offices and cubicles replaced with large open spaces
  - Team leaders located in center of the room
  - Outer wall hosts visuals that support development process

Toyota's development process is **4x shorter** than typical North American Company (Cleveland 2006)

Architecting Systems in the Solution Space

- Heuristic #1: People are a large part of people's environment
  - Share working environment with development team
- Heuristic #2: People divide and conquer. As a group they explore the unknown while exploiting the known.
  - Don't separate teams in the same sub-solution space
  - Avoid work duplication by communicating what people are working on between teams of sub-solution spaces
- Heuristic #3: More information isn't always better
  - Architects should not stifle creativity by micro-managing implementation team
  - Recognize that implementation team can positively affect architecture
- Heuristic #4: Groups are best influenced by bottom-up pressures rather than top-down rules
  - Evaluate individual/team performance based on how their dependent's are able to use their work product
- Heuristic #5: System cost increases proportionally to the increase in system complexity
  - Minimize complexity of the knowledge system by minimizing the number of people required to implement the architecture

Conclusions:
Don't Neglect Your Knowledge Systems

- Document knowledge systems along with architecture
  - Problem Space Knowledge System validates that problem space was thoroughly searched
  - Solution Space Knowledge System used to manage and explain architecture development in solution space
    - Example: 80-hour work weeks
- Document changes to knowledge systems throughout project lifecycle
  - Helps explains why changes are made to systems during development
  - Identify (and agree) if change was positive or negative
  - Identify (and agree) if change should be repeated in future projects based on same architecture

Knowledge Systems: How to Get Started

- Heuristic #6: It is better to design how a system will fail as opposed to invent how it will perform or operate. (David Stanislaw, FAA DER)
  - Identify failure conditions before designing a knowledge system
  - Knowledge System design should address how it will work in midst of failure
    - We don't live in a utopia
    - Be prepared for high risk failures

References

- Brooks, F. P., Jr. 1995. The Mythical Man-Month: Essays on Software Engineering. Reading, MA: Addison-Wesley.
- Cleveland, John. 2006. The Toyota Product Development System's Implementation Challenges. Field Guide to Automotive Technology. Website accessed on 20 March 2010: http:// www.autofieldguide.com/columns/0506insight.html
- Dagli, Dr., Dr. Ann Miller, and Russell Abbott. 2007. An Approach to a Network Centric Product Development System. In Proceedings of the 17th Annual International Symposium of the International Council on Systems Engineering (San Diego, CA). Seattle: INCOSE.
- Gureckis, Todd M., and Robert L. Goldstone. 2006. Thinking in Groups. Pragmatics & Cognition. 14:2:293-311. Amsterdam: John Benjamins Publishing Company.
- Hutchins, Edwin. 1995. Knowledge in the Wild. Cambridge: MIT Press.
- Leuf, Bo, Ward Cunningham, 2008, What is Wiki. Website accessed on 7 December 2008: http:// www.wiki.org/wiki.cgi?WhatIsWiki.
- Saunders, Steven. 2007. Promoting the Real Value of Systems Engineering using an Extended SCARIT Process Model. In Proceedings of the 17th Annual International Symposium of the International Council on Systems Engineering (San Diego, CA). Seattle: INCOSE.
- Tausworthe, R. C. 1976. Simple Intuitive Models of Programming. Deep Space Network Progress Report 42-33. Jet Propulsion Laboratory. Pasadena, CA.
- Trainor, Timothy E. and Gregory S. Parnell. 2007. Using Stakeholder Analysis to Define the Problem in Systems Engineering. In Proceedings of the 17th Annual International Symposium of the International Council on Systems Engineering (San Diego, CA). Seattle: INCOSE.
- Zhong, Yang. 2003. Local Government and Politics in China: Challenges from Below. Armonk, NY: M.E. Sharpe.

Don’t Neglect Your

Knowledge Systems