UCI

Design

Human Activity and Software Environments

David F. Redmiles

ICS 221

WQ 2001

Curtis, B., Krasner, H., Iscoe, N. *A Field Study of the Software Design Process for Large Systems*, Communications of the ACM, Vol. 31, No. 11, November 1988, pp. 1268-1287.

Robbins, J., Hilbert, D., Redmiles, D. *Extending Design Environments to Software Architecture Design*, *Automated Software Engineering*, Vol. 5, No. 3, July 1998, pp. 261-290.

Motivation

UCI

- The creation of software becomes more challenging every day because software is applied to increasingly complex problem situations.
- A new kind of complexity is measured in terms of end users and workplace settings.
- The increasing interaction among researchers from many disciplines is revealing the complexity of the new "software crisis" and new aspects to coping.
- Innovative combinations of many research disciplines are essential.

UCI

Part I

Curtis, B., Krasner, H., Iscoe, N. *A Field Study of the Software Design Process for Large Systems,* Communications of the ACM, Vol. 31, No. 11, November 1988, pp. 1268-1287.

Introduction

UCI

"the idea of the programmer as a human being is not going to appeal to certain types of people"

- EXERCISE
  - Describe your ideal software environment.
    (What would you put into it -- tools etc.)
  - What is the basis for this selection?
    What methods are available for verifying these basic assumptions?

Formative Methods in Curtis et al.

UCI

- Critical paper in the literature.
  - one of the first large scale studies
  - in the field, not in the lab
  - real projects, real people
  - numerous publications, e.g. in Empirical Studies of Programmers
  - precedent for methodology

Methodology

UCI

- Experimenters' perspective
  - what were the investigators interested in?
    (what was the real world problem being addressed?)
  - what information was gathered under what circumstances?
    (design, materials, procedure, subject, results, discussion)
  - what was concluded from the data?
  - how much does the reader have to accept on methodological principles
- Readers' perspective
  - how convincing is the evidence? how formal the proof?
  - otherwise, what corroborates the story? other literature? other case studies?

What were the investigators interested in?

UCI

- “dramatically improves software productivity and quality”
- upstream, rather than downstream activities
  (Question: before or after experiment?)
- “problem-driven, rather than technology-driven”

“Our interviews provided detailed descriptions of development problems to help identify high-leverage factors for improving such processes as problem formulation, requirements definition and analysis, and software architectural design.”

What information was gathered under what circumstances?

UCI

- Interviews
- Project participants across the board
- Anonymity of participants

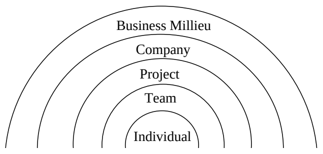
What was concluded from the data?

UCI

- Thin spread of application domain knowledge
- Fluctuating and conflicting requirements
- Communication and coordination breakdowns

Layered Behavioral Model

UCI



How much does the reader have to accept on methodological principles?

UCI

Whatever they choose!

BUT ...

- What corroborates the story?
  - other studies, e.g., Brooks
  - common understanding in the community
  - our own experience

Thin spread of application domain knowledge

UCI

- All the domain knowledge in one head
  (some individuals had knowledge, but only about one aspect)
- The ability to map that knowledge onto systems functionality
- Getting this knowledge isn't cheap! but what is the cost of a system that isn't used?

Fluctuating and conflicting requirements

UCI

- Can change as developers understand more [p. 1275]
- Can change for business purposes
  (interactions of the company with one, than several customers)
- Can change as team gathers more information
  (little access to customers [p. 1277])

Communication and coordination breakdowns

UCI

- Remoteness (organizationally) of different participants
- Information exchanged verbally, but permutations grow quickly with team size
- Negotiating a common [representation] language
- Formal "chain of communication" interfered with customers and developers talking

Conclusions/Recommendations [p.1283]

UCI

- creatively facilitate the staff-wide sharing and integration of knowledge
- must accommodate change as an ordinary process and support the representation of uncertain design decisions.
- must become a medium of communication
- EXERCISE: Hypothesize this software environment!

Hypothetical "Curtis et al." Environment

UCI

- Knowledge Sharing
  - CSCW
  - Hypermedia with suggested links
  - Software or interface agents?
- Accommodating Change
  - Implications for management of a project, assessing progress
  - Assessing fulfillment of a contract
  - Estimating cost? plan for overruns?
- Medium of Communication
  - cost-beneficiary of input of data
  - formal models? leverage from heuristics on informal information?

UCI

Part II

Robbins, J., Hilbert, D., Redmiles, D. *Extending Design Environments to Software Architecture Design, Automated Software Engineering,* Vol. 5, No. 3, July 1998, pp. 261-290.

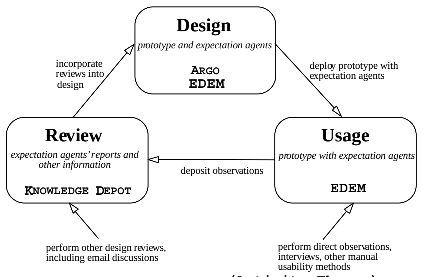
Communication as an Integrative View of the Software Lifecycle

UCI

- A human-centered view of the software lifecycle focuses on software development as a process of communication of knowledge:
  - Knowledge about what constitutes a good design is provided to developers.
  - Knowledge about past designs
  - Knowledge about actual system usage and user commentary is reported.
  - Knowledge about system usage and other related design knowledge is discussed among stakeholders.

Human-Centered Software Development

UCI


Design
prototype and expectation agents
ARGO
EDEM
incorporate reviews into design
deploy prototype with expectation agents
Review
expectation agents' reports and other information
KNOWLEDGE DEPOT
deposit observations
Usage
prototype with expectation agents
EDEM
perform other design reviews, including email discussions
perform direct observations, interviews, other manual usability methods

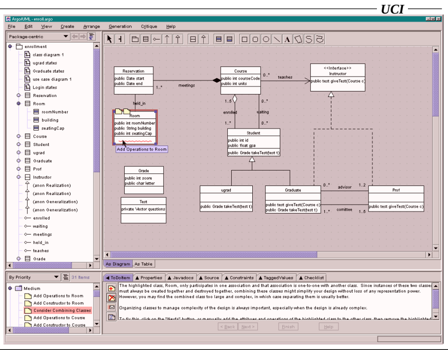
(Activity Theory)

Design — Argo/UML

UCI

Designers use Argo/UML much like they would use other object-oriented design tools: they place class, state, and use case icons in diagrams and draw relationships between them (upper right).

While designers work, design critics analyze the design and provide helpful advice. The "to do" list (lower left) presents and organizes advice about pending design changes.



The highlighted class, Room, only participates in one association and that association is one-to-one with another class. Since instances of these two classes must always be created together and destroyed together, combining these classes might simplify your design without loss of any representation power. However, you may find the combined class too large and complex, in which case separating them is usually better.

Organizing classes to manage complexity of the design is always important, especially when the design is already complex.

To fix this, click on the "Next>" button, or manually add the attributes and operations of the highlighted class to the other class, then remove the highlighted

Underlying Theories of Designers' Cognitive Needs

UCI

- We have tried to identify what makes design challenging for people
  - Design feedback: Reflection-in-Action [Schoen]
  - Design process support: Opportunistic Design [Guindon]
  - Design perspectives: Comprehension and Problem Solving [Kintsch]
  - Design visualization [Green]
- We have specialized and applied these theories to software architecture and object-oriented specification.

Design Feedback — Reflection-in-Action

UCI

- Observation: Designers "work through" the design
  - Designers rarely produce a design fully formed
  - Incrementally frame a partial design; evaluate it; revise
- Suggested requirements on design environments
  - Support the designer in framing a partial solution
  - Support the designer in evaluating it
  - Integrate manipulation and analysis
- Support in Argo: Editors, Critics, To Do List, Wizards

Design Process Support - Opportunistic Design

UCI

- Observation: Designers deviate from prespecified processes (even their own!)
  - Emerging issues change the design process
  - Cognitive costs explain some task re-orderings
- Suggested requirements on design environments
  - Provide process models as resources
  - Use process models to keep design feedback timely
- Support in Argo: First-class decision model, To Do List

Design Perspectives — Comprehension and Problem Solving

UCI

- Observation: Designers use multiple mental models
  - Work through successive levels of refinement
  - Restructure their thinking to address different issues
- Suggested requirements on design environments
  - Present multiple perspectives to match mental models
  - Help maintain mappings between mental models
- Support in Argo: Multiple coordinated design perspectives, customizable visualizations

Visual Clarifiers — Design Visualization

UCI

- Observation: visualizations support design, but
  - Communities develop visualization conventions
  - Within bounds, secondary notations improve communication among workers
- Suggested requirements on design environments
  - Support conventional notations
  - Support extensions to optimize tasks
- Support in Argo: multiple views, extensions to views, visual clarifiers

Argo/UML Features and Benefits

UCI

- Strict adherence to the UML standard
- Design checklists help resolve common design problems early, and make design reviews more effective.
- Design critics help catch design errors immediately, improve design decisions, and reduce re-work.
- Design history tracks criticism and resolutions.
- Navigational perspectives define alternative views of the design that make specific design issues visible.
- Customization features help adapt Argo to specific organizations practices.

Evaluation

UCI

- Argo Design Environment
  - Rockwell/Collins CORE Requirements Method
  - Over 500 registered users
  - UCI classroom use UCI Software Projects Course

Future Work

UCI

- Argo/UML Design Environment
  - Evaluation with respect to emerging research in software architecture.
  - Support for UML standard extension mechanisms.
  - End user authoring of critics.
  - Support for code generation.

Future Work (cont.)

UCI

- Knowledge Depot
  - Create a framework for ubiquitous awareness based on event and notification server architectures.
- Activity Theory
  - Explore the application of the theory to workflow process design.
  - Refine the theory to be more adaptable by software practitioners.

Long-Term Direction

UCI

- Human augmentation through software tools and cognitive and social theories.
- Accessible, pragmatic approaches to software tool deployment and software development processes.

Summary

- The changing nature of software places new demands on software engineering.
- An empirically-driven, evolutionary and human-centered model is necessary.
- The human-centered lifecycle interprets software development as a process of communication:
  - What constitutes a good design is provided to developers.
  - Actual system usage and user commentary is reported.
  - Developer discussion forums
- The activities are supported in a way compatible with software developers' work styles.