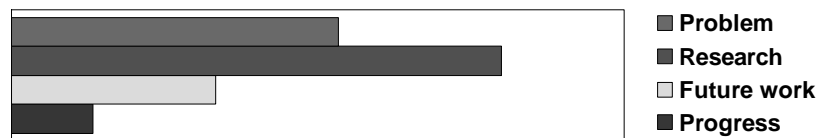


Architecting Requirements

Wendy W.Q. Liu
University of Toronto
2004

Outline

Talk Content Distribution



- Introduction of the Problem
- General Approach, Research Hypothesis, Method and Evaluation
- Prior Work and Further Work
- Current Progress and Potential Contribution

2

Research Questions

- How to produce architectural designs systematically from the requirements?
- How to manage changes effectively as the software evolves in its lifetime?

3

Software Engineering and Challenges

- Software engineering
 - Goal: build useful and robust software systems that meet their intended purposes effectively
 - Means: research better methods and tools
- Software design is an integral part of software engineering
 - It is creative and magical
 - Is there science behind it?
- Challenge: How to design an *architecture* that accounts for all the *requirements* systematically?

4

More Challenges

- Design flaws and requirement misunderstanding contribute to poor adaptability and faulty system behavior
 - Understanding requirements is a key to producing adaptable system architecture
 - Architecture is a key to producing adaptable software systems
- Links between requirements and design must be addressed both at the beginning of development and during evolution
 - Traceability is not enough
 - Derivation and reasoning are required
- Challenge: How to present *requirements* for better understanding and produce adaptable *architecture*?

5

Difficulty

- Difference in perspective
 - Problem vs. solution perspective
- Bridging the gap is about bridging the two perspectives

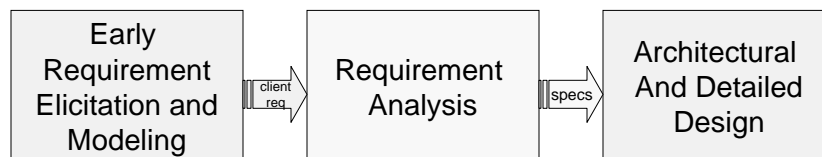
6

Architecting Requirements as a Bridge

- Capture the intent of requirement
 - Use the intent of requirements to drive the architectural designs
 - Refactor the requirements for
 - a degree of separation of concern
 - coupling and cohesion in the problem context
- Manage evolution effectively
 - Provide convenient structure for the factored requirements

11

Fitting in the Development Process



- Requirement analysis on consistency, completeness and correctness
- Architectural design requires understanding and refining requirements

12

Fitting in the Development Process



- Architecting requirements is about “designing” requirements (refinement)
 - to reflect the shape of the problem,
 - to provide better understanding of the requirements
 - to assist the identification of the shape of an effective system solution

13

Quotes from Architects

- Play both the analyst and designer role
- Need to generalize the individual requirements
- Need to consider requirement variance
- Understand the requirements
- Understand and represent the environment
- Realize non-functional requirements
- Anticipate changes that come with using the new system
- Factor out pieces into programmable units

(Selected from interviews with 9 architects in practice)

14

The Intuition

- Good solutions reflect (structural) properties of the problem
 - Example: the design of automobile
 - Maneuver in 2-D space, accelerate/decelerate
- Studying the problem and discovering the underlying structure may reveal properties of the solutions

15

Hypothesis

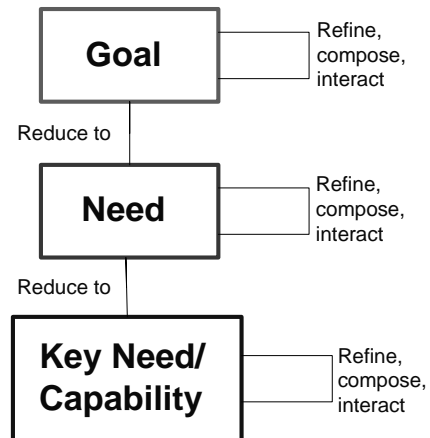
Intent + Refactoring → The Goal

- The goal is to build systems that evolve along the same dimensions as the problems change over time
- Describe the intent of the requirements
 - Overcome the over- and under-specified problem
 - Intent is more resilient to change
- Refactoring and restructuring requirements to reveal the structure of the problem
 - Discover and structure the relationships among the requirements and how they react to environment changes

16

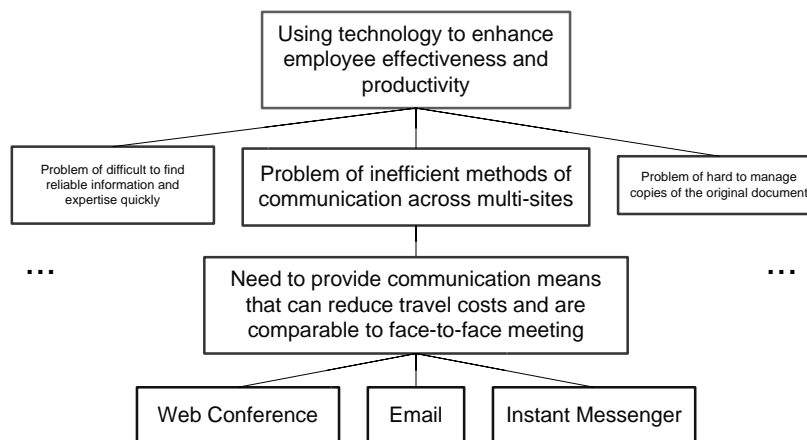
Capture Intent in Refinement Hierarchy

- *Goal*: business vision and opportunity
- *Need*: problems that may hinder the achievement of the goal and their potential resolution strategies
- *Key Need and Capability*: refactored *need* relating to the system



17

Example Hierarchy



18

Refactoring and Restructuring

- From within the requirements
 - Lifecycle analysis
 - Describe and analyze the projected *lifecycles* of requirements
 - Coupling and cohesion
- From the external perspectives
 - Environment factor analysis
 - Identify the *environment factors*
 - Determine the dimensions of change, and
 - Refactor the dimensions for more concise description

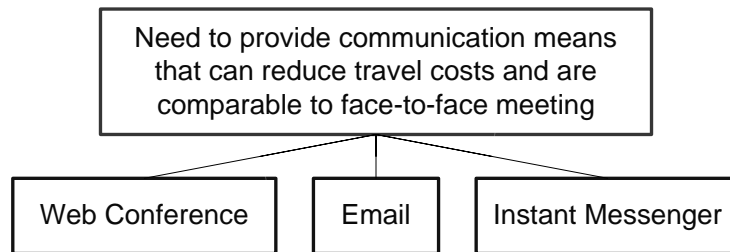
19

Refactoring: Lifecycle Analysis

- Project requirements lifecycle based on their temporal properties
 - Frequency, temporal coupling, speed, concurrency
- Use the temporal relations to
 - Discover the dependency, and
 - Restructure the requirements by split and join actions

20

Lifecycle Analysis Example



- Off-line
 - Email
- On-line
 - Message exchange: Instant Messenger
 - Slide show/teleconference/data exchange: Web conference

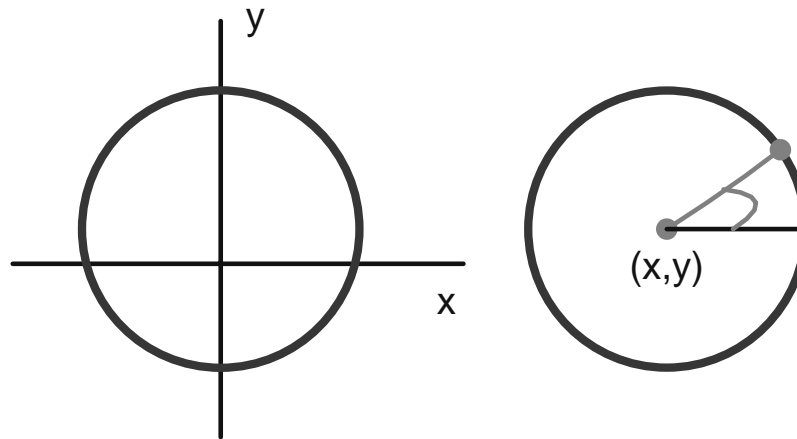
21

Refactoring: Environment Factor Analysis

- Identify the initial environment dimensions from environment factors that may introduce change
- Evaluate how *needs* respond to changes in each dimension and introduce new system dimensions
- Refactor both types of related dimensions for each requirement to achieve simplicity and conciseness in the requirement description

22

Environment Factor Analysis Example



23

Managing Evolution

- Enhance system adaptability
 - Preparation: project the evolution path
 - Anticipate changes
 - Design for evolution
- Manage change during operation
 - In-action: adjust the evolution path
 - React to changes
 - Relate to the high-level goals
 - Identify the impact more easily

24

Evaluation

- On the validity of the hypothesis
 - Identify classes of problems that exhibit the claimed characteristics
 - Validate the classification on industrial case studies
- On the effectiveness of the proposed method
 - Examine how well the problem structure, defined by applying the method, assists in finding good solutions
 - Summarize and classify the characteristics of the problems where the method is effective or non-effective
- Cost and benefit
 - Cost of change in perception and learning curve
 - Extra effort for analysis and information gathering and maintenance
 - Explicit capture of the intent and rationale
 - Adaptability and evolution management

25

Prior Research

- Goal-oriented Approach – KAOS [Lamsweerde 1993]
 - Philosophy: driven by goals
 - Meta-model, goal patterns, and acquisition strategy
 - Extension links to architecture
- Problem Frames [Jackson 2000]
 - Philosophy: problem before solution
 - Five basic problem frames
 - Required behaviour, commanded behaviour, information display, simple workpieces, transformation
 - Make use of known solutions
- Intent Specification [Leveson 2000]
 - Philosophy: design for evolution
 - Cognitive process from system design theory
 - A specification methodology:
 - process, content, structure, form
 - Five-level intent dimension:
 - system purpose, system design principles, black-box behavior, physical and logical function, physical realization
- STRAW'01 and STRAW'03

26

Further Work

- Theory
 - requirement
 - architecture
- Representation language
 - intent specification
- Refactoring methods and tool support
- Empirical validation

27

Further Work: Theory

- A theory of requirement is needed to
 - Model the reality of requirements
 - Explain what they are
- A corresponding theory of architecture is also required to
 - Explain what is expected from the requirements in constructing an architecture
- As the foundation of method and tool design in *architecting requirements*
- Empirical validation of the theory

28

Further Work: Method and Tool Support

- Use the theory as a foundation for the methods
- Provide tool support for
 - Constructing requirement hierarchy
 - Flexible internal representation that supports customizable view and manipulation
 - Assisting the two analysis methods
 - Generating graphical and textual documentation

29

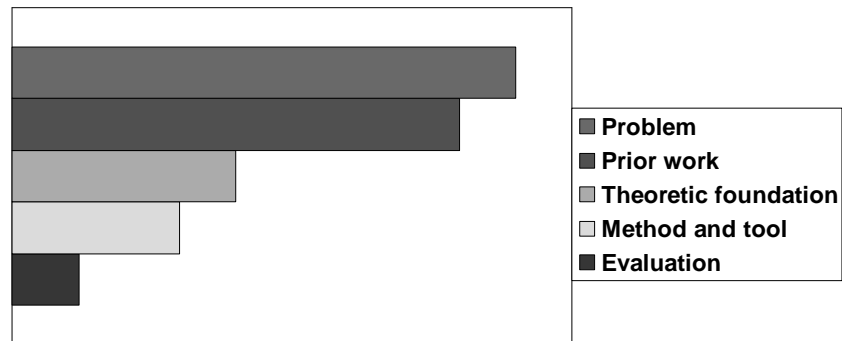
Further Work: Representation Language

- Intent specification
 - Goal
 - Need
 - Key Need / Capability

30

Current Progress

How much has been done?



31

Potential Contribution

- Theoretic framework for software development
 - Theory of requirements
 - Theory of architecture
 - Empirically validation
- Techniques for analyzing requirements and moving towards architectural design
 - Requirement refactoring methods
 - Intent capturing
 - Links to design
 - Handling adaptability and evolution

32

Discussions and Questions

<http://www.cs.utoronto.ca/~wl>

Need Help

- Weakness and strength of this approach
- Evaluation means
- Additional related work
- Intent representation

34