Software requirement understanding using Pathfinder networks: discovering and evaluating mental models

Udai Kumar Kudikyala and Rayford B. Vaughn

Presented by Sunkyung Lee

© KAIST SE LAB 2008
Contents

- Introduction
- Background
- Experimental design
  - Step 1. Requirement extraction
  - Step 2. Requirement grouping
  - Step 3. Pathfinder network generation
- Analysis of results
- Conclusion
- Discussion
Introduction (1/2)

- Complexity of requirements
  - Poor requirements accounts for 71% of failed software project [CIO Magazine]
  - 40% of effort in an average software project is rework
    - The largest contributor is requirements errors

[Distribution of Defects]
Complexity of requirements (cont’d)

- Ambiguity may exist in SRS
  - Text can be interpreted in different ways
  - Developer and user may have different thoughts about system

- Mental model can be used to check common understanding

1. SRS: Software Requirement Specification
Background (1/2)

- **Mental model**
  - Human’s thought process
    - The way to imagine how something works in the real world

Pick up the phone
Dial the number
Hear the phone on the other end ringing
Hear the other person’s answer

© KAIST SE LAB 2008
**Pathfinder networks**

- A technique to represent a mental model
- Connected graphs
  - Node: concept
  - Edge: relationship between the concepts
  - Weight: strength of the relation

[Example of pathfinder network]
Experimental design (1/7)

Step 1
- Requirement extraction
- Requirement 1
- Requirement 2
- Requirement 3
- Index card

Step 2
- Requirement grouping
- Team organization
- Customer (1)
- Developer (4~6)

Step 3
- Pathfinder network generation
- Index card
- Requirement 1
- Requirement 2
- Requirement 3

1. SRS: Software Requirement Specification
Step 1: Requirement extraction

- Extract all requirements from SRS\(^1\)
- Record on index cards

Diagram:

- **[System 1]**
  - Emailing A Student Resume To A Company
  - Faxing A Student Resume To A Company
  - Faxing A Student Transcript To A Company

- Teams:
  - Team 1
  - Team 2
  - Team 3
  - Team 4
Step 2: Requirement grouping

- Group the index cards into different stacks
  - Based on perceived similarities
- Duplicate a requirement is allowed
- Developers are not allowed to consult with user

Diagram:

- Customer
- Developer
- Related requirements
Experimental design (4/7)

- Step 3: Pathfinder network generation

  - Similarity matrix
  - Dissimilarity matrix
  - Pathfinder network
Step 3: Pathfinder network generation (cont’d)

- Similarity matrix \((N \times N)\)
  - \(N\): The number of requirements
  - Record the co-occurrence count of a pair of requirements

```
     A   B   C   D
A  0   1   1   2
B  0   1   2   
C  0   2   
D  0   
```

[Similarity matrix]
**Experimental design (6/7)**

- **Step 3: Pathfinder network generation (cont’d)**
  - **Dissimilarity matrix**
    - Maximum co-occurrence count – co-occurrence count + 1

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

- The value of 1 means the closest relation

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

© KAIST SE LAB 2008
Step 3: Pathfinder network generation (cont’d)

- Pathfinder network generation using dissimilarity matrix
  - For user and developers

\[
\begin{array}{cccc}
  & A & B & C & D \\
  A & 0 & 2 & 2 & 1 \\
  B & 2 & 0 & 2 & 1 \\
  C & 2 & 2 & 0 & 1 \\
  D & 1 & 1 & 1 & 0 \\
\end{array}
\]

[Dissimilarity matrix]  [Pathfinder network]
Evaluation of requirement analysis

- Check for common understanding
  - Comparison of the resultant Pathfinder networks of user and developers
  - Measure: correlation coefficient (cc)
    - Below 0.4: little or no similarity
    - 0.4 ~ 0.7: moderate similarity
    - Over 0.7: strong similarity

The higher cc, the more similar understanding between user and developers

* 0.4, 0.7: Empirical result of author
Analysis of result (2/7)

- Evaluation of requirement analysis (cont’d)
  - Check for common understanding
    - The result of overall correlation coefficient

<table>
<thead>
<tr>
<th>Software systems</th>
<th>Overall correlation coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>System 1</td>
<td>0.77</td>
</tr>
<tr>
<td>System 2</td>
<td>0.46</td>
</tr>
<tr>
<td>System 3</td>
<td>0.91</td>
</tr>
<tr>
<td>System 4</td>
<td>0.87</td>
</tr>
</tbody>
</table>

low correlation coefficient!
Evaluation of requirement analysis (cont’d)

- Check for common understanding
  - The result of individual correlation coefficient

<table>
<thead>
<tr>
<th>Systems</th>
<th>Percentage of requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cc ≥ 0.9</td>
</tr>
<tr>
<td>System 1</td>
<td>43.75</td>
</tr>
<tr>
<td>System 2</td>
<td>0.0</td>
</tr>
<tr>
<td>System 3</td>
<td>50.0</td>
</tr>
<tr>
<td>System 4</td>
<td>50.0</td>
</tr>
</tbody>
</table>

System 2: ambiguous requirement!
Evaluation of requirement analysis (cont’d)

- Check for common understanding
  - Pathfinder network for System 2

1. PFNet : PathFinder Network

[PFNet\textsuperscript{1} for developer]  [PFNet for user]
Evaluation of requirement analysis (cont'd)

- The requirement has low correlation and tend to get poor satisfaction of user

- The result of correlation and interviews

<table>
<thead>
<tr>
<th>Req.</th>
<th>Correlation (analysis phase)</th>
<th>Customer (after implementation)</th>
<th>Comments of developers (after implementation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.87</td>
<td>WU</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.64</td>
<td>WU</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.51</td>
<td></td>
<td>Customer and time</td>
</tr>
<tr>
<td>8</td>
<td>0.74</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.58</td>
<td>U*</td>
<td>Customer and time</td>
</tr>
<tr>
<td>11</td>
<td>0.62</td>
<td></td>
<td>Customer, technical and time</td>
</tr>
<tr>
<td>12</td>
<td>0.72</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0.62</td>
<td>U</td>
<td>Customer and time</td>
</tr>
</tbody>
</table>
Evaluation of requirement analysis (cont’d)

- Check for duplicates of requirement
  - Clustering information (cliques) can identify suspect duplicates

1. Clique: Fully connected graph
<table>
<thead>
<tr>
<th>Software systems</th>
<th>Original requirement</th>
<th>Seeded requirement</th>
<th>cc</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1-30 : Sending A Student Resume To A Company By Electronic Mail</td>
<td><strong>1.0</strong></td>
</tr>
<tr>
<td></td>
<td>1-13 : Emailing A Student Resume To A Company</td>
<td></td>
<td><strong>1.0</strong></td>
</tr>
<tr>
<td></td>
<td>1-16 : Faxing A Student Resume To A company</td>
<td>1-30 : Sending A Student Resume To A Company By Electronic Mail</td>
<td><strong>0.96</strong></td>
</tr>
<tr>
<td></td>
<td>1-12 : Emailing A Student Transcript To A Company</td>
<td>1-31 : Sending A Student Transcript To A Company By Electronic Mail</td>
<td><strong>1.0</strong></td>
</tr>
<tr>
<td></td>
<td>1-15 : Faxing A Student Transcript To A Company</td>
<td>1-31 : Sending A Student Transcript To A Company By Electronic Mail</td>
<td><strong>0.96</strong></td>
</tr>
<tr>
<td></td>
<td>1-11 : Emailing A Letter Of Recommendation to A Company</td>
<td>1-32 : Sending A Letter Of Recommendation To A Company By Electronic Mail</td>
<td><strong>1.0</strong></td>
</tr>
<tr>
<td></td>
<td>1-14 : Faxing A Letter Of Recommendation To A Company</td>
<td>1-32 : Sending A Letter Of Recommendation To A Company By Electronic Mail</td>
<td><strong>0.96</strong></td>
</tr>
</tbody>
</table>
Conclusion

- Summary
  - Proposing a new technique for better understanding of requirement
    - Check for common understanding between user and developer
    - Check for duplication of requirements

- Future work
  - Support for medium-scale and large-scale software project
  - Automated tools to make network and analysis
Discussion

- In my opinion,
  - This is a simple and easy way for user
    - User can participate in requirement analysis
  - This is fast and cheap
    - Making a prototype is expensive and takes long time
  - But, this technique has these disadvantages
    - Focus on preference rather than what works best
    - Don’t reveal performance
    - Reliability may be questionable
Pathfinder network

- PFnet(q, r)
  - Two parameters
    - q: limit on the number of links in the paths
      - Integer: between 2 and (n - 1)
    - r: metric used for computing the distance of paths
      - Minkowski r metric
      - Real number: between 1 and infinity
  - Triangle inequality

\[
W(P) = \left( \sum_{i=1}^{k} w_i^r \right)^{\frac{1}{r}}
\]

© KAIST SE LAB 2008
Related Issue

- User centered design (programming)
  - How to get the user’s requirement
    - Understanding about user
      - User model
      - Knowledge sharing
  - How to manage user’s requirement
    - Automated tool