Automated Traceability using IRSim: A Preliminary Empirical Study

Dharma Ganesan
Mikael Lindvall
Jan-Philip Quirmbach
Outline

• Problem and Motivation
• IRSim Tool
• Results of Preliminary Empirical Study
Problem

- Many NASA projects require traceability
- Problem: It is tedious to create and maintain traceability
- Traceability between
  - requirement and source code
  - requirement and test cases
- other docs (bug reports, user manuals, etc.)
Key motivations for retrieving traceability links

• Verification and Validation
  – To check completeness of an implementation w.r.t stated requirements
  – To verify whether requirements are correctly implemented, the corresponding source code should be identified

• Program Understanding
  – To understand how a requirement is realized, the corresponding source code should be identified
  – Bug fixes or feature enhancements require tracing from requirements/testcases to code
Retrieving traceability links

Traceability Retrieval Engine

Index

Artifact types

requirements
test cases
source code

Build Index

Output

Traceability matrix among and within artifact types
IRSim Tool
IRSim Tool

• Web application
• Upload archives
  – User can select files for indexing:
    • by extensions
    • by directories
• Automatically produce traceability matrix
• Based on Information Retrieval Methods
  – Vector Space Model and Lucene
Explore Uploads by Extension

<table>
<thead>
<tr>
<th>Extension</th>
<th>Occurrences</th>
<th>Enable(2642/2642)</th>
<th>Example File</th>
</tr>
</thead>
<tbody>
<tr>
<td>html</td>
<td>531</td>
<td>✓</td>
<td>📄</td>
</tr>
<tr>
<td>dat</td>
<td>263</td>
<td>✓</td>
<td>📄</td>
</tr>
<tr>
<td>c</td>
<td>154</td>
<td>✓</td>
<td>📄</td>
</tr>
<tr>
<td>log</td>
<td>145</td>
<td>✓</td>
<td>📄</td>
</tr>
<tr>
<td>doc</td>
<td>142</td>
<td>✓</td>
<td>📄</td>
</tr>
<tr>
<td>loge</td>
<td>126</td>
<td>✓</td>
<td>📄</td>
</tr>
<tr>
<td>logf</td>
<td>126</td>
<td>✓</td>
<td>📄</td>
</tr>
<tr>
<td>logp</td>
<td>126</td>
<td>✓</td>
<td>📄</td>
</tr>
<tr>
<td>logr</td>
<td>126</td>
<td>✓</td>
<td>📄</td>
</tr>
<tr>
<td>h</td>
<td>118</td>
<td>✓</td>
<td>📄</td>
</tr>
<tr>
<td>prc</td>
<td>111</td>
<td>✓</td>
<td>📄</td>
</tr>
<tr>
<td>zip</td>
<td>111</td>
<td>✓</td>
<td>📄</td>
</tr>
</tbody>
</table>
Explore Uploads by Directories
Text Parsing Methods in IRSim

• CamelCase: splitting camel cased words
  – i.e. CamelCase = camel, case, camelcase

• Standard: Using a List of English stop words
  – i.e. and, but, if, not, or, the,…

• Simple: Using a Lower Case Tokenizer
  – i.e. Hello World = hello, world
Matrices, Vector Spaces, and Traceability Retrieval

- In 1960’s, Gerard Salton proposed idea of modeling document collection as a matrix – Vector Space Model

<table>
<thead>
<tr>
<th>Terms</th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$\cdots$</th>
<th>$D_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W_1$</td>
<td>$C_{11}$</td>
<td>$C_{12}$</td>
<td>$\cdots$</td>
<td>$C_{1m}$</td>
</tr>
<tr>
<td>$W_2$</td>
<td>$C_{21}$</td>
<td>$C_{22}$</td>
<td>$\cdots$</td>
<td>$C_{2m}$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\cdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$W_n$</td>
<td>$C_{n1}$</td>
<td>$C_{n2}$</td>
<td>$\cdots$</td>
<td>$C_{nm}$</td>
</tr>
</tbody>
</table>

Typical term-document incidence matrix $C$ ($C_{ij} = n \leftrightarrow$ document $D_i$ contains term $W_i$ exactly $n$ times)

- IRSim tool builds the matrix automatically
- Similarity between documents: cosine of angle between them
Weight options for the Matrix

• Boolean: 1 term present, 0 term not present
• TF: The frequency of the term in the document
• TF-IDF: important of a word to a document in a collection
Geometry and Traceability Retrieval

• If two vectors are nearly orthogonal (90 degree) then they are treated conceptually dissimilar
• If the angle between two vectors are towards 0 degree then they are considered conceptually similar
• User defines a threshold on the angle to select similar vectors of a vector
Vector Space Model Boolean Example

• Document to a vector with terms (Boolean)

\[
\begin{pmatrix}
  d_1 & d_2 & d_3 & d_4 \\
  t_1 & 1 & 0 & 1 & 1 \\
  t_2 & 0 & 1 & 1 & 0 \\
  t_3 & 1 & 0 & 0 & 1 \\
  t_4 & 0 & 0 & 0 & 1 \\
\end{pmatrix}
\]

• Calculate cosine of angle for similarity

\[
\cos \theta = \frac{d_1 \cdot d_4}{||d_1|| \cdot ||d_4||} = \frac{1 \cdot 1 + 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 1}{\sqrt{2} \cdot \sqrt{3}} = 0.8164
\]
Example of generated traceability links

These “raw” traceability links can be used to generate a traditional traceability matrix on any level.
Questions for the Study

• Which is the *best* weight type for the term-doc matrix?

• Which is the *best* text parsing strategy?
Results of a preliminary empirical study using NASA's Core Flight Software
NASA's Core Flight Software

• Core Flight Executive
• Used as basis for satellite data system and instruments and for embedded systems
• Written in C with a software library called Operating System Abstraction Layer (OSAL)
# Requirements

<table>
<thead>
<tr>
<th>Module</th>
<th># of Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Service</td>
<td>148</td>
</tr>
<tr>
<td>Software Bus</td>
<td>34</td>
</tr>
<tr>
<td>Time Service</td>
<td>38</td>
</tr>
<tr>
<td>Table Service</td>
<td>52</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>272</strong></td>
</tr>
</tbody>
</table>

## Requirement Text

<table>
<thead>
<tr>
<th>ReqID</th>
<th>Requirement Text</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>cSB4000</td>
<td>Upon receipt of a NOOP command, the cFE shall increment the command counter.</td>
<td>Useful for verifying communication</td>
</tr>
</tbody>
</table>
## Source Code

<table>
<thead>
<tr>
<th>Module</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Service</td>
<td>2478</td>
</tr>
<tr>
<td>Software Bus</td>
<td>589</td>
</tr>
<tr>
<td>Time Service</td>
<td>574</td>
</tr>
<tr>
<td>Table Service</td>
<td>1344</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>4985</strong></td>
</tr>
</tbody>
</table>
Golden Model

• Manually traced Requirements to the Source Code (.h,.c,.mak)
• Used as basis for the Traceability (Compare with IRSim)
• Goal is to evaluate the precision of IRSim
• Fraunhofer has analyzed the code for several years
  – Golden model was validated
Golden Model

- ID, Requirement, File, Function / Name

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement</th>
<th>File</th>
<th>Function / Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>cSB4000</td>
<td>Upon receipt of a NOOP command, the cFE shall increment the command counter.</td>
<td>cfe_sb_task.c</td>
<td>void CFE_SB_ProcessCmdPipePkt(void)</td>
</tr>
<tr>
<td>cSB4001</td>
<td>Upon receipt of Command the cFE shall set to zero the following counters in housekeeping telemetry: - Valid command counter - Invalid command counter - No subscriptions counter - Message send error counter - Message receive error counter - Create Pipe error counter - Subscribe error counter - Pipe Overflow error counter - MsdlID-to-pipe limit error counter</td>
<td>cfe_sb_task.c</td>
<td>void CFE_SB_ResetCounters(void)</td>
</tr>
</tbody>
</table>
Evaluation procedure

• Looked at the Top 5 Results of IRSim

• If the expected functions are present, then it is considered as success

• Otherwise as failure
Precision

- **Precision** is the ratio of the Top 5 traced requirements with the IRSim Tool to the total manually traced requirements. It is expressed as a percentage.

- \[ \text{Precision} = \frac{\# \text{ of Results in Top 5}}{\# \text{ of Requirements}} \]

- Example: Requirements: 4, Results in Top 5: 3
  \[ \text{Precision} = \frac{3}{4} = 0.75 = 75\% \]
CamelCase Analyzer

![Bar chart showing precision with TF-IDF, TF, and BOOL categories for SB, ES, TIME, and TBL categories with values: SB (85), ES (60), TIME (74), TBL (87), SB (26), ES (30), TIME (29), TBL (58), and SB (33), ES (42), TIME (42), TBL (55).]
Standard Analyzer

![Precision Chart]

- SB: TF-IDF 56, TF 15, BOOL 11
- ES: TF-IDF 58, TF 5, BOOL 38
- TIME: TF-IDF 65, TF 10, BOOL 52
- TBL: TF-IDF 87, TF 55, BOOL 74

© 2014 Fraunhofer USA, Inc.
Center for Experimental Software Engineering
Simple Analyzer

Precision

<table>
<thead>
<tr>
<th></th>
<th>TF-IDF</th>
<th>TF</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB</td>
<td>41</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>ES</td>
<td>55</td>
<td>8</td>
<td>19</td>
</tr>
<tr>
<td>TIME</td>
<td>70</td>
<td>6</td>
<td>23</td>
</tr>
<tr>
<td>TBL</td>
<td>81</td>
<td>13</td>
<td>29</td>
</tr>
</tbody>
</table>

© 2014 Fraunhofer USA, Inc.
Center for Experimental Software Engineering
Why IRSim is not tracing all requirements?

- Typo in Requirement or Source Code
- Requirement is not well enough worded
- Chopping of Sub-Requirements
  - Parent-child requirement hierarchy
- Heavy use of acronyms in the source code
Summary

- IRSim can be used to generate Traceability links
- Depends on the wording of the Requirement
- Combination of CamelCase and TF-IDF is the best one
- IRSim is completely independent of programming language and can be used on any text
- Need to replicate on other systems
- Need further validation
Acknowledgements?

• NASA CFS/CFE team
• Fraunhofer Interns
Thank you for your Attention

dganesan@fc-md.umd.edu