Requirements Engineering Databases: The Good, The Bad, and The Ugly

Brian Berenbach
Siemens Corporate Research
brian.berenbach@siemens.com

New England Database Day (NEDBDay08)
February 4, 2008
Agenda

- Brief Introduction to Siemens (Context)
- What is a Requirements Database?
- Key differentiators of an RE Database
- Industrial Problems
- Some Current Research Areas
Siemens at a Glance

Siemens Business Services

Communications

Siemens Building Technologies

Automation and Control

Automation and Drives

Industrial Solutions and Services

Power

Power Generation

Power Transmission and Distribution

Transportation

Transportation Systems

Siemens VDO Automotive

Medical

Medical Solutions

Lighting

Sylvania

Net Income: $2.8B

World’s 21st largest company

47,000 employees

$6.6B dedicated to global R&D

Annual Worldwide Sales

$98.2 billion

75% of total sales are from products and services developed in the last five years

Siemens AG: Worldwide figures for fiscal 2005¹ (U.S. GAAP²)
¹Fiscal Year October 1 – September 30
²Average annual exchange rate for FY 2005: €1.00 =$1.273
Siemens has a long tradition of technological innovations.
The rate of innovations is increasing.

Share of sales with products…

- 5 years and younger
- 6 to 10 years old
- more than 10 years old

<table>
<thead>
<tr>
<th>Year</th>
<th>5 years and younger</th>
<th>6 to 10 years old</th>
<th>more than 10 years old</th>
</tr>
</thead>
<tbody>
<tr>
<td>1980</td>
<td>22%</td>
<td>50%</td>
<td>22%</td>
</tr>
<tr>
<td>1985</td>
<td>48%</td>
<td>30%</td>
<td>16%</td>
</tr>
<tr>
<td>2005</td>
<td>75%</td>
<td>19%</td>
<td>6%</td>
</tr>
</tbody>
</table>
RE Databases are Important

- Functionality previously realized in electrical or electro-mechanical systems is now being realized in software => bigger, more complex, & more software projects (hundreds of developers, millions of lines of code).
- Meeting functional and non-functional requirements is important to business success => restricted hardware resources, real-time performance, safety critical applications.
- Multisite development projects.
- High quality (i.e., thoroughly tested, reliable) software is important to business success.

Our RE databases must address the increasing scale and complexity of emerging software systems.
Requirements Database

A requirements database is an extension of a standard database that provides key functionality in the area of version control and traceability. That is, the database provides, “off the shelf” the ability to version and track changes to requirements and to trace requirements within and outside of the database, also to create trace matrices and generate formatted reports and specifications.
Attributes of an RE Database

The are many unique attributes of an RE database compared with a traditional database. They include:

- Schema predefined to support the storage of requirements of different kinds
- Version control at the requirement (record) level, with user views of history of a requirement
- Intrinsic support for tracing, that is, a “drag and drop” mechanism that is easy to use and supports creating traces manually between requirements.
- Generation of requirements documents directly from the repository. The preferred method of working is to create and edit requirements in the database, then use the documentation facility of the database to create a filtered and formatted set of requirements in a requirement specification, usually as either a word or pdf document.
There are two typical types of configurations for RE Databases.

- One possible one has the RE management software on the server, using a browser to access it.
- The other configuration, which is faster but requires that software be installed on the client, is to have a client application on the user PC accessing the database on a server.
Scale and complexity make life difficult

Requirements Database with Over 16000*

* We have RE databases with over 2 GB in them
Key Differentiators of RE Databases

- Easy to use, intuitive with minimal need to refer to documentation
- Ability to baseline and perform change control on requirements.
- Support for rich text and graphics in a requirement description
Key Differentiators of RE Databases

- Ability to work offline

  • Take requirements home, review them, change them, roll them back into the database later.

  Good thing I can make those changes at home.
Key Differentiators of RE Databases

- **Product line support**
  - Create subsets of requirements that can be reused for different projects and products.

- **Global Support.**
  - The ability to have distributed requirements analysis is more than just the ability to fold in rules to determine routing, review procedures (e.g. workflow), and scripting for user guidance and quality assurance.
Key Differentiators of RE Databases

All fields/attributes should be definable on a per company or per project basis, such that the same corporate data dictionary is used on multiple projects for consistency, but each project can still have its own glossary of terms, keywords, etc.
Key Differentiators of RE Databases

Parent child relationship - it should be possible to have a parent child relationship, and where there is such a relationship, it should be possible to have parent and child of a different core requirement type.

FEAT101
SECRQT101.5
PERFRQT 101.33

• Note that some commercial databases do not allow parent-child requirements to be of different types.
Key Differentiators of RE Databases

- **Traceability** - the tool used shall enable end-to-end traceability, vertical and horizontal traceability. This may require integration with other tools such as IDEs and/or modeling and testing tools.

- **Mandated by law**, including Sarbanes-Oxley, the FDA and the FAA.

- **Traceability required by most demanding safety-critical process standards** such as: DO178B, IEC 61508, EN50128*.”
Traceability and The FDA*

A software requirements traceability analysis should be conducted to trace software requirements to (and from) system requirements and to risk analysis results.

In addition to any other analyses and documentation used to verify software requirements, a formal design review is recommended to confirm that requirements are fully specified and appropriate before extensive software design efforts begin.

Requirements can be approved and released incrementally, but care should be taken that interactions and interfaces among software (and hardware) requirements are properly reviewed, analyzed, and controlled.”

*General Principles of Software Validation; Final Guidance for Industry and FDA Staff, Jan 11, 2002
Traceability and the FAA

FAA's Advisory Circular AC20-115B established DO-178B as the accepted means of certifying all new aviation software:

“The stringent and internationally accepted DO-178B process standard that governs the development of civilian avionics embedded software is primarily requirements driven. ...at each stage ...software developers must be able to demonstrate traceability of designs against requirements; i.e. are the high-level requirements I am specifying traceable to the system requirements? Are the low-level requirements traceable to the high-level requirements? ....

*Esterel Technologies
Multiple Layers

Product lines makes this worse!!!
Databases do not scale

• Significant performance Degradation
• Lack of functionality
• Not user friendly
• Obsolete scripting languages
• Inefficient query mechanisms
• Poor merge functionality
• Traces are unmanageable and break
Time to panic?
Captain! There are no viable solutions on the horizon
The Bad

An unsolved computer science problem
The Good

some advanced techniques that SCR/SE has piloted may help in the future

• Unified Requirements Model
• Dynamic Tracing
The Unified Requirements Modeling Language

- The Unified Requirements Modeling Language (URML) is a standard notation for the modeling of requirements artifacts as first step in developing a product or performing business modeling.
- It incorporates elements of:
  - Use cases
  - Feature Modeling
  - Hazard Analysis
  - Threat Modeling
  - Goal Modeling
  - Other requirements related visual modeling techniques
A unified requirements model is a model that has been created using the URML.

- It has been found empirically that text specifications are inherently ambiguous. The ambiguity is exacerbated with global software development, different languages and cultures. Graphical models are inherently unambiguous.
Requirements Relationships are naturally organized in directed graphs, whereas traditional databases support a tree structure. The use of a tree structure forces the creation of an inordinate number of traces and does not work well with cross-cutting requirements. As a typical project grows to N requirements, the number of traces needed is often greater than N^2.
A Visual Language Provides Views into the Model

- Faster than text (up to 70%)
- Verifiable
- Easier to understand than text.

I finally understand why I should use childproof door locks!

New Symbols

- Functional Requirement
- Hazard
- Non-Functional Requirement
Definition of Traceability*

- "Requirements traceability is the ability to describe and follow the life of a requirement, in both a forward and backward direction:
  - from its origins, through its development and specification
  - to its subsequent deployment and use
  - through periods of ongoing refinement and iteration in all of the project phases."

  *Gotel & Finklestein, 1994
Dynamic Tracing

- **Problem:** Manual tracing is subject to errors, omissions, and decay as artifacts are changed and the lack-of-benefit-to-tracer problem.
- **One solution:** Dynamic tracing

Dynamic tracing is the automated generation of candidate trace links.
With over 3000 design documents the rebels will **NEVER** find the flaw in the DeathStar.
Much to learn you have, hmmm? The rebels are using Dynamic Tracing!
## A Typical Traceability Matrix

<table>
<thead>
<tr>
<th>Tag</th>
<th>Name</th>
<th>Traced To</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUC101</td>
<td>Filter View in IET Central</td>
<td>BUC101, BUC111, BUC112</td>
</tr>
<tr>
<td>SUC107</td>
<td>Change Active Project in MSD</td>
<td>BUC101, BUC111, BUC112</td>
</tr>
<tr>
<td>SUC111</td>
<td>View Concepts in MSD</td>
<td>BUC101, BUC111, BUC112</td>
</tr>
<tr>
<td>SUC112</td>
<td>View Drawing in MSD</td>
<td>BUC101, BUC111, BUC112</td>
</tr>
<tr>
<td>SUC113</td>
<td>Central</td>
<td>BUC101, BUC111, BUC112</td>
</tr>
<tr>
<td>SUC114</td>
<td>View Projects in IET Central</td>
<td>BUC101, BUC111, BUC112</td>
</tr>
<tr>
<td>SUC115</td>
<td>View Material Flow Diagram in IET Central</td>
<td>BUC101, BUC111, BUC112</td>
</tr>
<tr>
<td>SUC116</td>
<td>Manage Concepts in IET Central</td>
<td>BUC101, BUC111, BUC112</td>
</tr>
<tr>
<td>SUC117</td>
<td>Manage Groups in IET Central</td>
<td>BUC101, BUC111, BUC112</td>
</tr>
<tr>
<td>SUC118</td>
<td>Promote objects in IET Central</td>
<td>BUC101, BUC111, BUC112</td>
</tr>
<tr>
<td>SUC119</td>
<td>Manage System Elements in MSD</td>
<td>BUC101, BUC111, BUC112</td>
</tr>
<tr>
<td>SUC120</td>
<td>Manage Collections (Tags/Revisions) in IET Central</td>
<td>BUC101, BUC111, BUC112</td>
</tr>
<tr>
<td>SUC121</td>
<td>Manage Design Content in IET Central</td>
<td>BUC101, BUC111, BUC112</td>
</tr>
</tbody>
</table>
Information Retrieval*

- Inference Network Model (probabilistic approach)

\[
\text{Probability of a link between a query and document:}
\]

\[
\text{Probability of a link between a query and document:}
\]

\[
\text{Frequency of term } t_i \text{ in respect to the size of the document.}
\]

\[
\text{Frequency of term } t_i \text{ in respect to the size of the query.}
\]

\[
\text{Measures the rarity/commonality of the term.}
\]

\[
\text{Probability of a link between a query and document.}
\]

\[
\text{Measures the rarity/commonality of the term.}
\]

\[
\text{Probability of a link between a query and document.}
\]

\[
\text{Probability of a link between a query and document.}
\]

\[
\text{Probability of a link between a query and document.}
\]

\[
\text{Probability of a link between a query and document.}
\]

*Slide Courtesy of Prof. Jane Cleland-Huang, DePaul University
```java
void UpdateDisplayList() {
    listModel.removeAllElements();
    String mSQL = "SELECT distinct SubscriberName FROM EventDetails";
    try {
        rs = stmt.executeQuery(mSQL);
        while (rs.next()) {          String SubsName = rs.getString(1);
            listModel.addElement(SubsName);
        }
        rs.close();
    } catch (Exception e) {
        System.out.println("Notification_Processing: Problem with query: "+ e);
    }
}
```

• An example of traceable artifacts and selected links
Alerts shall be issued when hazardous driving conditions exist.

*Slide Courtesy of Prof. Jane Cleland-Huang, DePaul University*
Questions?

Am I supposed to disintegrate the database or the audience???

Zappppp