



Software Process Improvement through Best and Emerging Requirements Engineering [and other] Practices

Presented at the NYC Spin
November 8, 2007

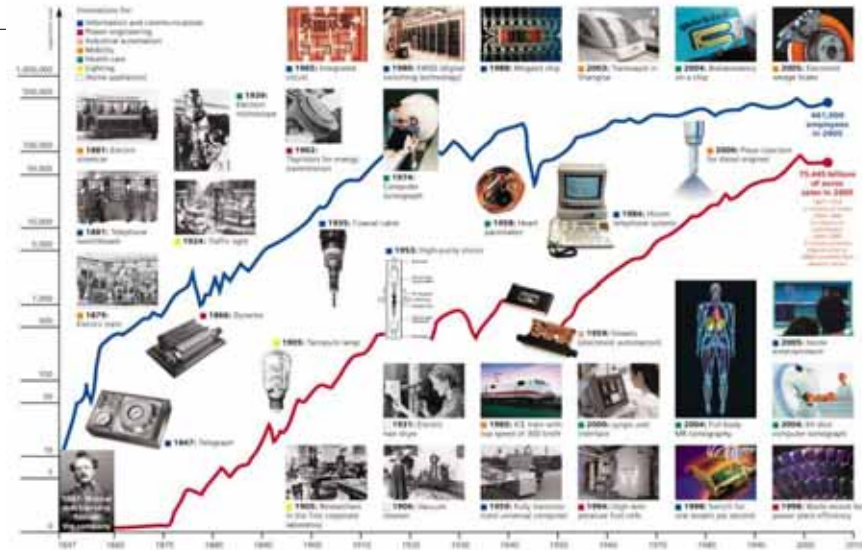
Brian Berenbach
brian.berenbach@siemens.com

Agenda

- Introduction to Siemens Corporate Technology
- Rapid Prototyping
- Dynamic Tracing
- Software Reconnaissance
- Model Driven Requirements Engineering
 - Unified Requirements Modeling Language
 - Heuristics
 - Design Model Quality Assurance
 - Automated Extraction of Requirements from Models
- Model Based Testing
- Video Based Requirements Engineering
- Global Software Development
- Software Factories



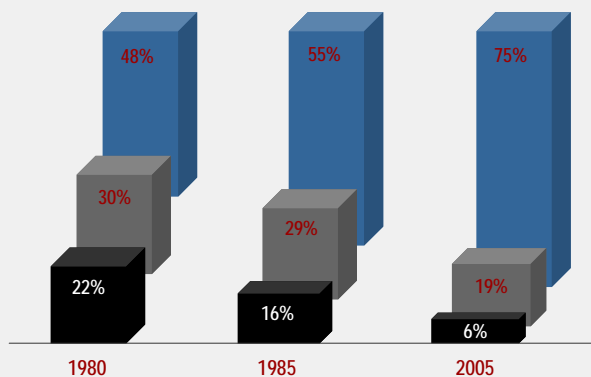
Siemens has a long tradition of technological innovations.



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



Siemens is one of the world's largest software companies.



- ▶ Siemens has more than 30,000 software developers.
- ▶ 60% of Siemens' business is based on software.
- ▶ Siemens spends more than 3 billion euros per year on software development.

Page 5

Copyright 2007 © Siemens Corporate Research

But, Siemens is not recognized as a software company, since most of our software is embedded.

Examples:

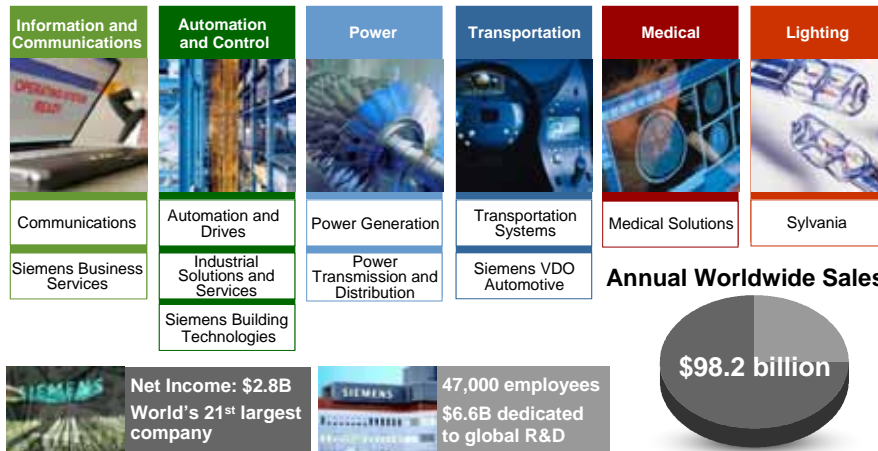
- ▶ Automation devices
- ▶ Industrial control systems
- ▶ Automotive components
- ▶ Communication systems
- ▶ Rail systems
- ▶ Medical devices



Page 6

Copyright 2007 © Siemens Corporate Research

Siemens at a Glance



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

Page 7

Copyright 2007 © Siemens Corporate Research

Global presence of Research and Development

More than 47,000 R&D employees at 150 locations



Page 8

Copyright 2007 © Siemens Corporate Research

SIEMENS

Corporate Technology: About 2,300 Researchers and Developers Worldwide

**Siemens
Corporate Technology**
A Global Network

Page 9 Copyright 2007 © Siemens Corporate Research

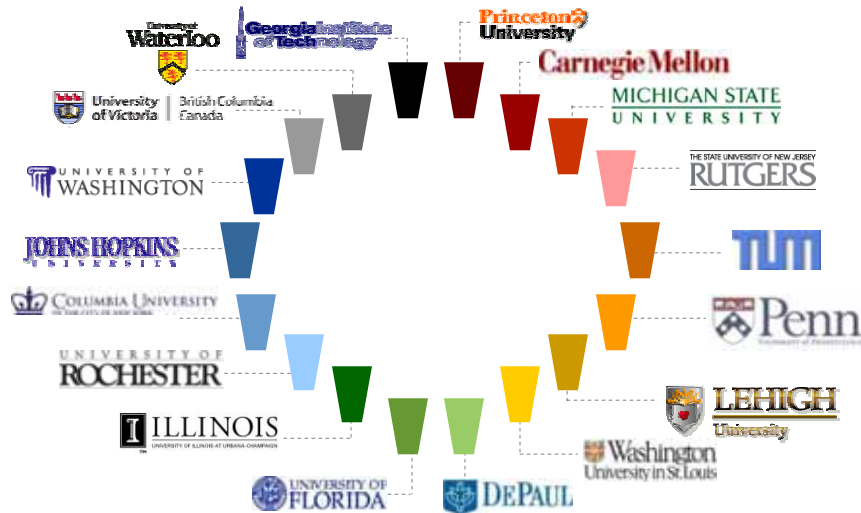
SIEMENS

Siemens Corporate Research

- User Interface Design Center**: Medical user interface for sharing text documents, photos, videos & music.
- Imaging & Visualization**: Computer imaging & vision applied in medicine.
- Real-Time Vision & Modeling**: Real-time vision for safety, security, transportation, automation, automotive, service & maintenance.
- Integrated Data Systems**: Data integration & maintenance informatics.
- Intelligent Vision & Reasoning**: Computer-aided diagnosis for medical & industrial applications.
- Automation & Control**: Software applications and tools for industrial automation and building technology.
- Software Engineering**: Architecture & quality for increased productivity.
- Multimedia Video Technology**: Media processing, management & personalization.

Page 10 Copyright 2007 © Siemens Corporate Research

Much of our research is done in collaboration with universities.



Page 11

Copyright 2007 © Siemens Corporate Research

Software Engineering Challenges at Siemens

- 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 software engineering methods and technologies must address the increasing scale and complexity of emerging software systems.

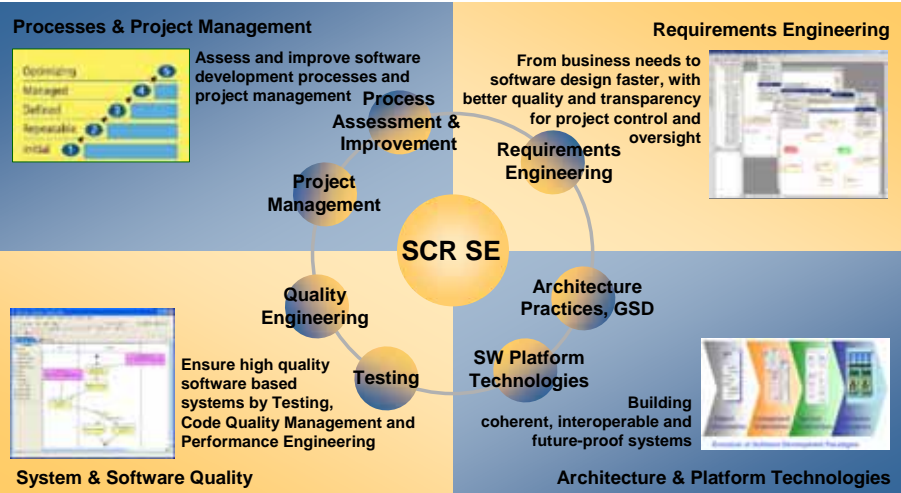
Page 12

Copyright 2007 © Siemens Corporate Research

Time to panic?



We address software engineering challenges through R&D programs and technology transfer



SIEMENS

Our Solutions and practices

Page 15 Copyright 2007 © Siemens Corporate Research

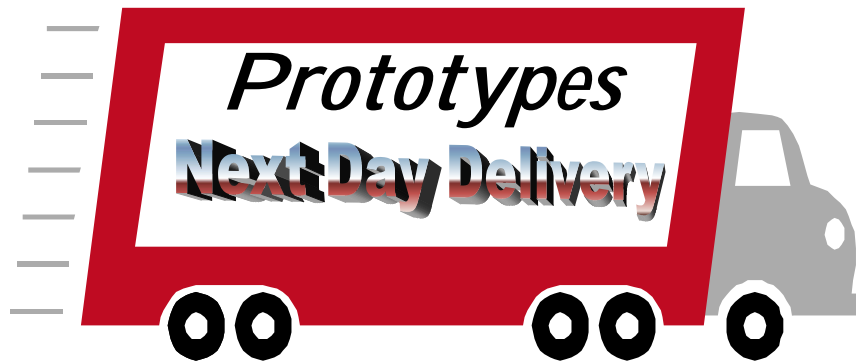
SIEMENS

Agenda

- Introduction to Siemens Corporate Technology
- **Rapid Prototyping**
- Dynamic Tracing
- Software Reconnaissance
- Model Driven Requirements Engineering
 - Unified Requirements Modeling Language
 - Heuristics
 - Design Model Quality Assurance
 - Automated Extraction of Requirements from Models
- Model Based Testing
- Video Based Requirements Engineering
- Global Software Development
- Software Factories

Page 16 Copyright 2007 © Siemens Corporate Research

S-RaP: A Concurrent Evolutionary Software Prototyping Process



Siemens Rapid Prototyping Process

Need to be able to present **new product features** at trade shows or sales presentation.

Need to validate product features with **different** customers who have **different** business environments (e.g., children hospitals vs. general hospitals).

Need to mature new product features through **interactive** prototypes.

Such needs can occur with a **short-time** (a few days) notice.

Such needs occur **regularly** each year.

Requirements on the process

High-concurrency that involves a sizeable development team (20-30 developers, UI designers, testers)

Support high-degree of **user involvement**

Support **iterations** of the process steps to refine the evolving requirements.

Promote the **cooperation** between UI designers and prototype developers

Cost-effective documentation

Compare S-RaP with other agile or prototyping methods (e.g., XP)

Similarities:

- Requirements initially vague
- All support high-degree of user involvement
- Use use-case scenario to drive the development.
- Iterative requirement engineering and implementation

Different emphases:

- Rely on **SIMPLE** tools
- Allow use of **inexperienced** developers (short-term)
- High-degree of **concurrency** to achieve the short-term delivery
- High **visibility** for project progress to mitigate the risk



Copyright 2007 © Siemens Corporate Research



Copyright 2007 © Siemens Corporate Research

A Major Project Experience with S-RaP

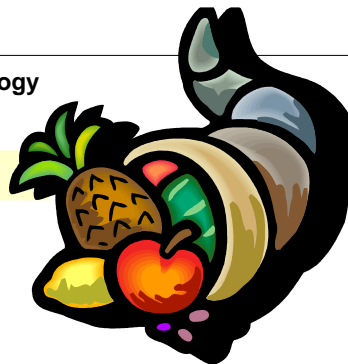
Developed a healthcare information system prototype that included a number of deliveries of different sizes.

Project Data:

- 6-8 Workflows
- Each workflow defined by a storyboard that includes 25 screenshots
- Major component of the prototype was done in 4 months.
- Very high quality of UI look-and-feel and reliability.
- 20 developers on average during the project

Agenda

- Introduction to Siemens Corporate Technology
- Rapid Prototyping
- Dynamic Tracing
- Software Reconnaissance
- Model Driven Requirements Engineering
 - Unified Requirements Modeling Language
 - Heuristics
 - Design Model Quality Assurance
 - Automated Extraction of Requirements from Models
- Model Based Testing
- Video Based Requirements Engineering
- Global Software Development
- Software Factories



Definition of Traceability*

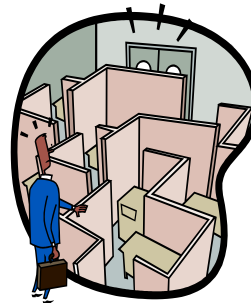
“Requirements traceability is the ability to describe and follow the life of a requirement, in both a forward and backward direction:

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



Why Traceability?

Requirements validation & verification

Impact analysis

Compliance verification

Regression testing

Hazard Tracking

Knowledge management

Process Improvement Initiatives

The Challenge

To provide viable end-to-end traceability solutions when other technologies fail and a manual solution is not cost effective.

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.



Much to learn you have, hmmm? The rebels are using **Dynamic Tracing!**

With over 3000 design documents the rebels will **NEVER** find the flaw in the DeathStar.



SIEMENS



Page 29

Copyright 2007 © Siemens Corporate Research

SIEMENS

A Typical Traceability Matrix

Tag	Name	TracedFrom	Tracedto
SUC101	Filter View in IET Central	CSC306, CSC397, CSC153, CSC154, CSC155, CSC241, CSC297, CSC378, CSC379, CSC390, CSC381, CSC382, CSC383, CSC484	BUC17, BUC18, BUC22, BUC201
SUC107	Change Active Project in MSD	CSC226, CSC40, CSC188, CSC189, CSC191, CSC248, CSC244, CSC302	BUC196, BUC201
SUC111	View Concepts in MSD	CSC44, CSC195, CSC284, CSC387, SUC116, SUC121	BUC17, BUC18, BUC22, BUC201
SUC112	View Drawing in MSD	CSC226, CSC269, CSC338, CSC340, SUC120, SUC121	BUC6, BUC18, BUC23, BUC25, BUC124, BUC125, BUC131, BUC135, BUC145, BUC146, BUC201, BUC202
SUC113	View Material Flow Diagram in IET Central	CSC230, CSC231, CSC242, SUC122	BUC3
SUC114	View Projects in IET Central	CSC298, CSC305, SUC122	BUC17, BUC18, BUC22, BUC201
SUC115	View Properties in IET Central	CSC74, CSC101, CSC102, CSC136, CSC138, CSC218, CSC227, CSC228, CSC300, CSC385, SUC122	CSCOMP121
SUC116	Manage Concepts in MSD	CSC192, CSC193, CSC194, CSC195, CSC196, CSC197, CSC208, CSC283	BUC6, BUC17, BUC22, BUC35, BUC36, BUC37, BUC201, SUC26, SUC29, SUC34, SUC40, SUC55, SUC111, SUC132, SUC133, SUC134
SUC117	Manage Concepts in IET Central	CSC216, CSC217, CSC218, CSC219, CSC220, CSC221, CSC226, CSC235, CSC295, CSC297, CSC309	BUC47, BUC48, SUC92, SUC94, SUC97, SUC99
SUC119	Manage Groups in IET Central	CSC32, CSC54, CSC96	SUC15, SUC24, SUC30, SUC35, SUC36, SUC38
SUC122	View Objects in IET Central	CSC213, CSC226, CSC227, CSC230, CSC231, CSC242	SUC63a, SUC64, SUC65, SUC113, SUC114, SUC115
SUC123	Manage Collections (Dashboards) in IET Central		SUC39, SUC63b, SUC74, SUC76, SUC77
SUC124	Promote Objects in IET Central	CSC75, CSC308, CSC309	SUC48, SUC49, SUC50
SUC125	Manage System Elements in MSD	CSC391	BUC6, BUC17, BUC22, SUC17, SUC18, SUC19, SUC23, SUC27, SUC32, SUC36, SUC52, SUC53, SUC61, SUC68, SUC83, SUC84
SUC131	Manage Drawing Content in MSD	CSC412, CSC413, CSC414, CSC415, CSC416, CSC417, CSC418, CSC419, CSC420, CSC421, CSC422, CSC423, CSC424, CSC425	BUC18, BUC201
SUC136	Refresh Drawing in MSD	SUC120	BUC17, BUC18, BUC22, BUC201
SUC137	Open Drawing in MSD	SUC120	BUC6, BUC18, BUC23, BUC25, BUC124, BUC125, BUC131, BUC135, BUC145, BUC146, BUC201, BUC202
SUC138	Check in Drawing in MSD	SUC120	BUC6, BUC18, BUC23, BUC25, BUC124, BUC125, BUC131, BUC135, BUC145, BUC146, BUC201, BUC202

Page 30

Copyright 2007 © Siemens Corporate Research

Problems of Classic Traceability

Structures such as matrices are **very hard** to maintain.

Too many traceability links result in an **unwieldy tangle** of useless information.

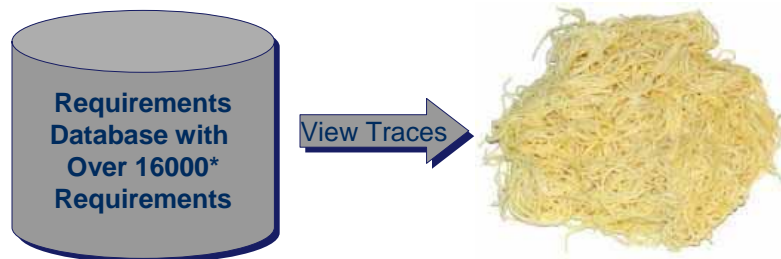
Often no sufficient support for automation provided →
Reduction of the traceability effort.

Traceability is a **hard sell** because it is often perceived to have an insufficient ROI.

Often insufficient support for non-functional requirements.

Often time consuming and error-prone.

Scale and complexity make life difficult



* We have RE databases with over 2 GB in them

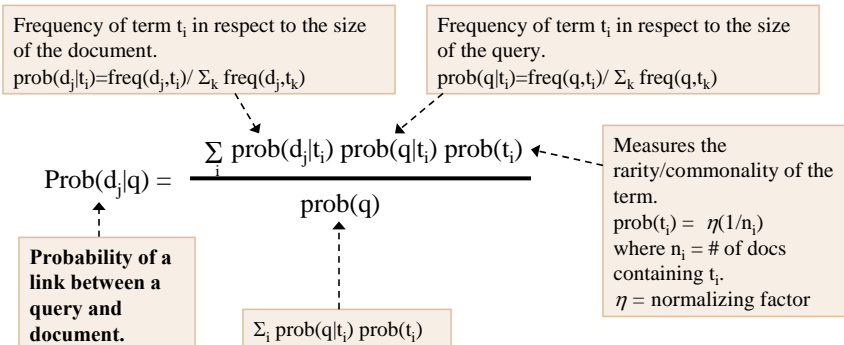
Automated Traceability Definition

Automated traceability is
"the automated generation of candidate traceability links".

Prof. Jane Huang, DePaul University

Information Retrieval

Inference Network Model (probabilistic approach)



Recall & Precision

Metrics used to measure and evaluate the quality of retrieved trace links.

$$\text{Precision} = \frac{\text{Number of relevant traces retrieved}}{\text{Total number of relevant traces}} = \frac{\text{Correct links} \cap \text{Retrieved links}}{\text{Retrieved links}}$$

$$\text{Recall} = \frac{\text{Number of relevant traces retrieved}}{\text{Total number of traces retrieved}} = \frac{\text{Correct links} \cap \text{Retrieved links}}{\text{Correct links}}$$

Recall & Precision (cont.)

Example:

10 possible trace links (correct, incorrect):
 { A, B, C, D, E, F, G, H, I, J }

→ 5 correct links

→ 5 incorrect links

6 trace links found by the trace tool:
 { A, D, E, F, G, H }

→ Recall = 4 / 5 = 0.80 = 80%

→ Precision = 4 / 6 = 0.66 = 66%

Recall & Precision (cont.)

Recall

High value

- Most of the correct trace links were **retrieved**.
- Analyst can find the trace links in the returned result set.

Low recall value

- Many of the correct trace links were **not retrieved**.
- Analyst is unable to find the missing trace links in the result set.

Precision

High value

- Most of the retrieved trace links in the result set are **correct**.
- Analyst can find relevant trace links easier.

Low value

- Most of the retrieved trace links in the result set are **incorrect**.
- Analyst has to reject many incorrect trace links and finding the relevant trace links is more difficult.



Indicator Terms

The terms (words) used in a query that indicate a trace link to another artifact are called **indicator terms**.

The indicator terms in a query determine if and how similar two artifacts are to each other.

Each indicator term can be given a weight by the trace tool, which determines how important this term is in relation to another artifact or the complete data set.

Indicator Terms (cont.)

Examples

- The weather monitor shall accept readings from valid WeatherStations.
- Product shall determine number of trucks needed for a given schedule.

Stop words list

- Common terms like "the", "shall", "from", "of", "a", "by", "as", "and", "or", etc. are ignored.

Other techniques to improve use of indicator terms

- **Stemming**: use of root of term (e.g. "drive", "driving", "driven" → "driv")
- **Aliases / Glossary**: different terms have the same meaning (e.g. to show = to display)

Applications of automated tracing

- ❖ Dynamically establish trace links and thus save time and prevent errors.
- ❖ Provide traceability if no traceability scheme exists yet.
- ❖ Help in reconstructing a requirements trace matrix.
- ❖ Support supplemental tracing in documents not included in the current trace scheme yet.
- ❖ Support traceability of Requests For Change or new requirements.
- ❖ **Cross-cut many different media to retrieve traces** including drawings, models, documents and code.
- ❖ **Handle traces through large amounts of legacy and third party material automatically**, where manual traces would just not be feasible.

Poirot

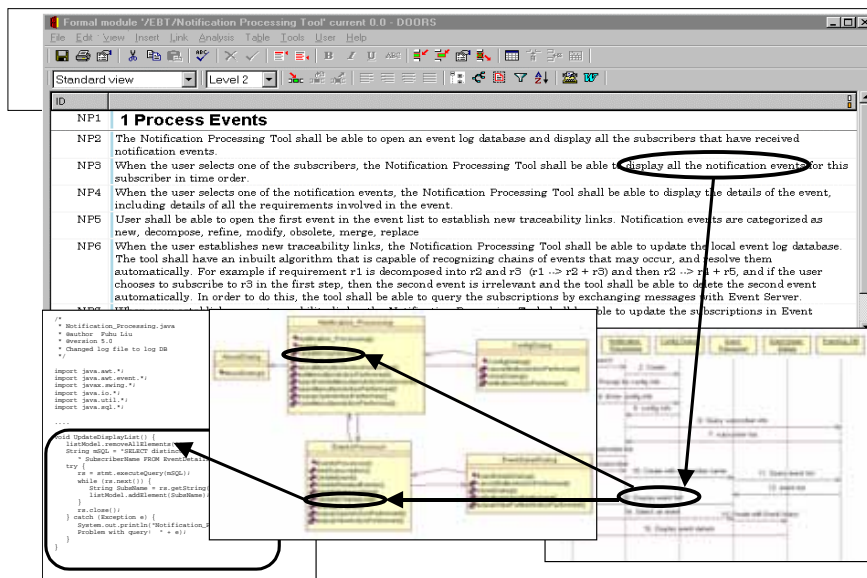
Developed at DePaul University's Center for Requirements Engineering and funded by Siemens Corporate Research.

Web-based tool written in Java.

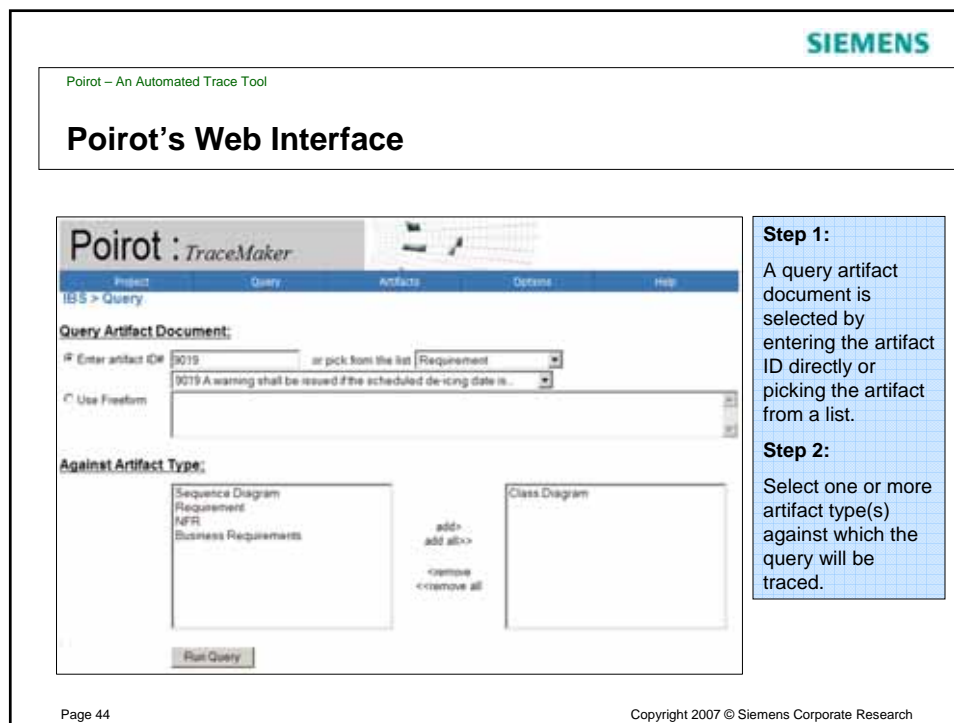
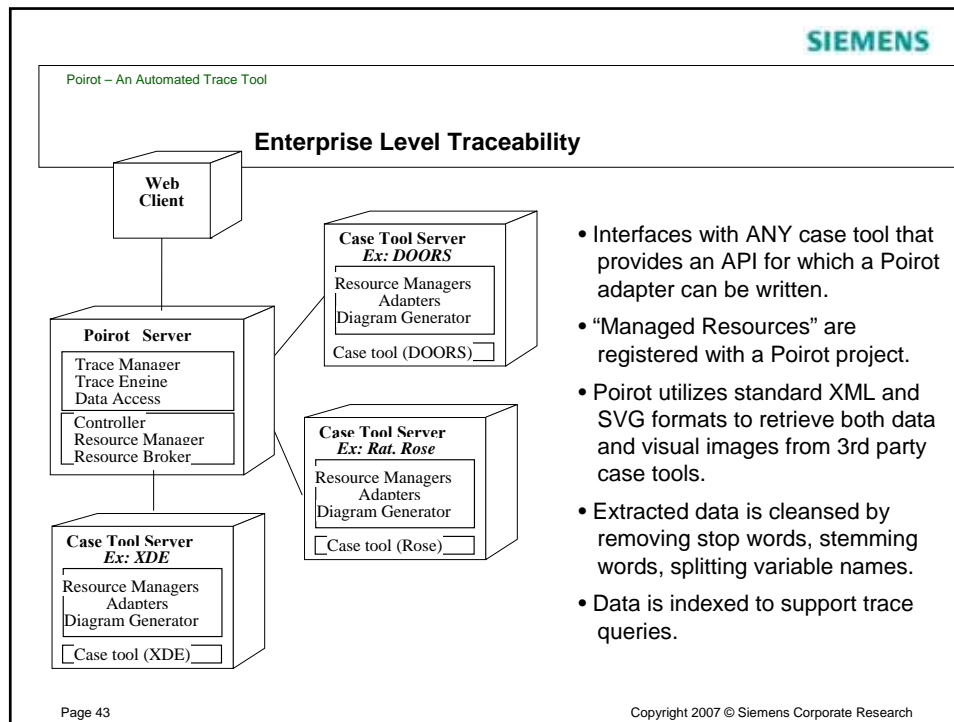
Supports traces between distributed heterogeneous software artifacts.

Based on the probabilistic Inference Network Model

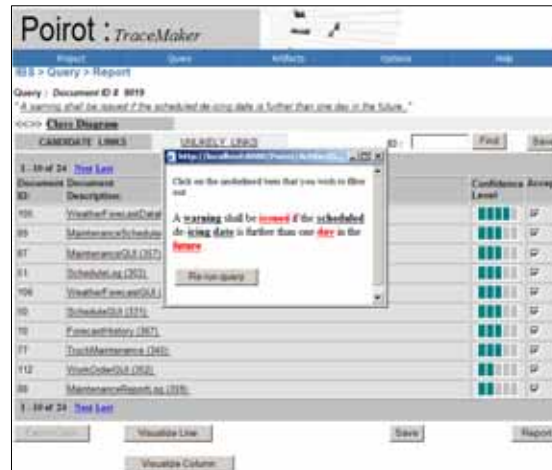
Piloted on several **large** Siemens projects.



An example of traceable artifacts and selected links



Poirot's Web Interface



Step 4:

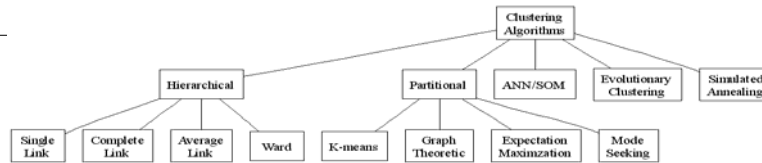
Refine the query by using the filtering feature of Poirot. Unimportant terms in the query can be marked and the query can be run again; the marked terms are ignored. Term filtering enables the analyst to refine and improve the results.

Recall the definition of “Recall”

High value

- Most of the correct trace links were **retrieved**.
- Analyst can find the trace links in the returned result set.
- How to improve RECALL??

Choose Proper Clustering Algorithms



Algorithm	Time complexity	Shape of discovered clusters	Quality of clusters	Ease of tuning
K-means	$O(n)$	spherical	varied	Hard
Agglomerative Hierarchical Clustering	$O(n^2)$	versatile	prone to bloated clusters	Easy
Bisect Hierarchical Clustering (using square-error objective function)	$O(n)$	spherical	good	Easy

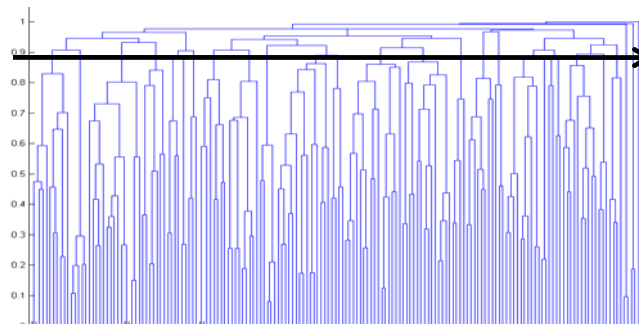
Agglomerative Hierarchical Clustering

Initialization Each requirement is assigned to an individual cluster.

Iterations

- Merge the most similar pair of clusters
- Calculate the similarity between the new cluster and each existing cluster.

Termination K (desired number of clusters) is met



Possible cut leads to 20 clusters in EBT data set

Poirot : TraceMaker

Project Query Artifacts Options Help

IBS > Query > Report

Query : Document ID # 12
 "Temperature, wind chill, and precipitation data will be received from external weather stations and roadside sensors."

Dominant Cluster Cross-cutting Cluster ID : Find

Next Last

Document ID:	Document Description:	Confidence Level:	Accept
Temperature readings			
Updates			
9404	Road maps shall be updated by importing data from an external source.	■■■■■	<input type="checkbox"/>
9016	Data received from the road sensors shall be updated regularly	■■■■■	<input checked="" type="checkbox"/>
9012	Weather forecasts shall be updated as received by the weather bureau.	■■■■■	<input type="checkbox"/>
9140	When new road sensors are added, the thermal map shall be updated to reflect the new weather data.	■■■■■	<input type="checkbox"/>
9007	Weather forecast update	■■■■■	<input type="checkbox"/>

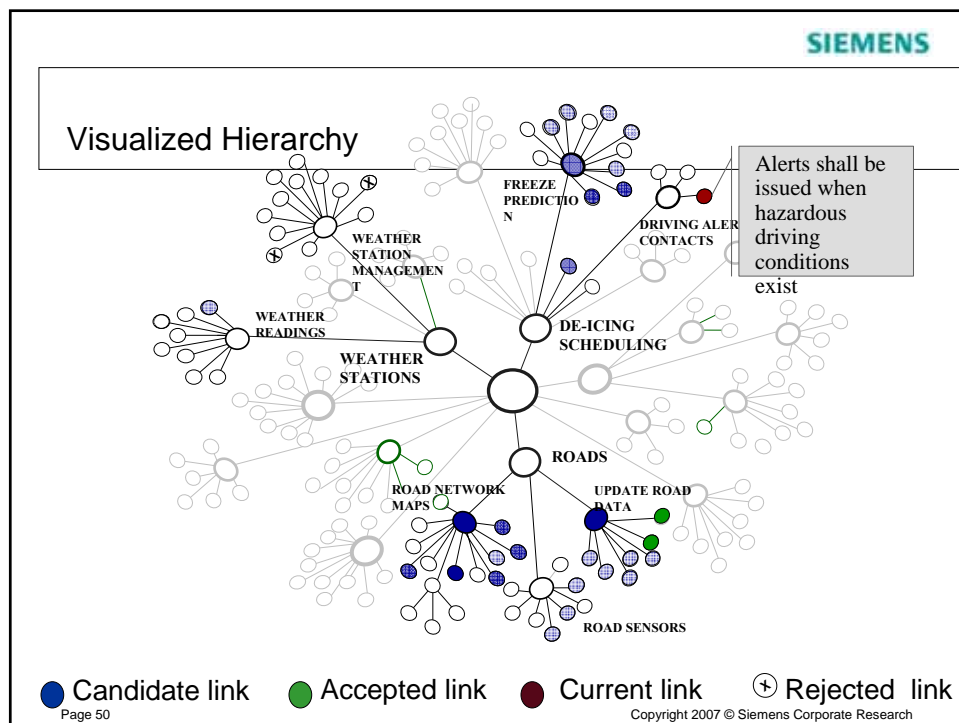
Show Context Select All Clear All

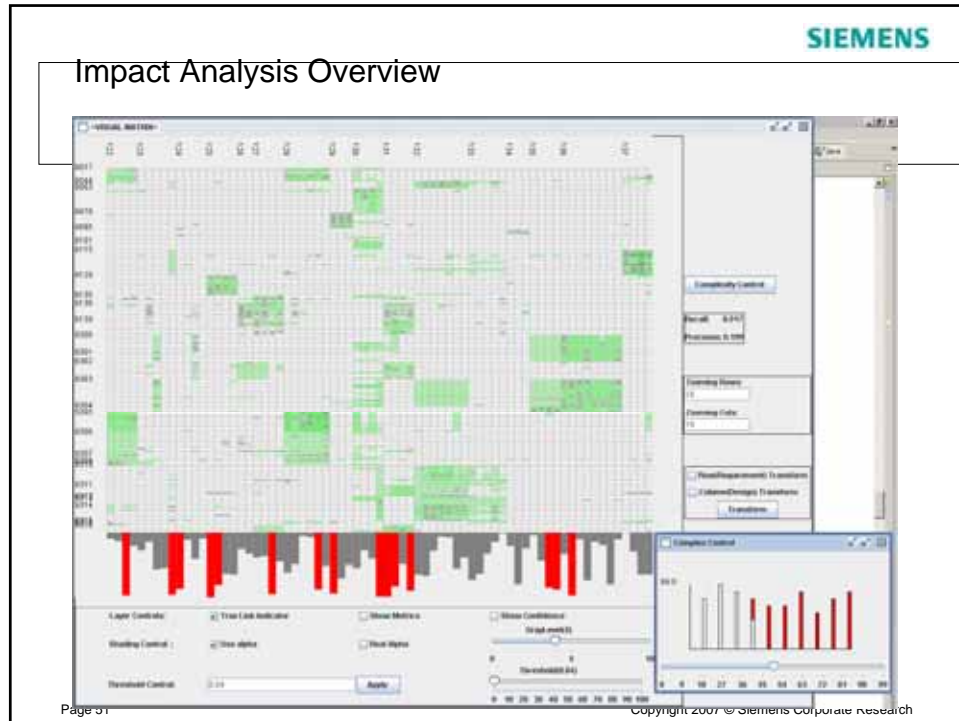
Transmission

Weather data

Next Last

Export Data Save Report esearch





SIEMENS

Siemens Pilot Results

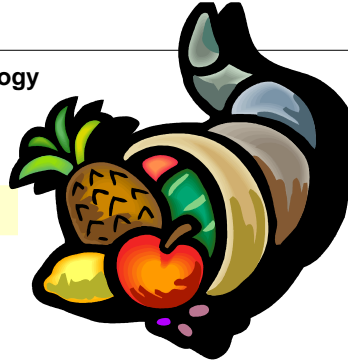
"Industrial applications of automated traceability can only be considered successful when high recall levels are achieved, we report precision results at recall levels fixed close to 90%" - Professor Jane Huang

Data Set	Requirements	Traceable artifact	Recall
1. Ice Breaker System (IBS)	Text	UML Classes	90.48%
2. Event-Based Traceability System (EBT)	Text	UML Classes	90.87%
3. Siemens Automated Warehouse Design Tools (IET)	Text	Text, UML Models	90%

Page 52 Copyright 2007 © Siemens Corporate Research

Agenda

- Introduction to Siemens Corporate Technology
- Rapid Prototyping
- Dynamic Tracing
- **Software Reconnaissance**
- **Model Driven Requirements Engineering**
 - Unified Requirements Modeling Language
 - Heuristics
 - Design Model Quality Assurance
 - Automated Extraction of Requirements from Models
- Model Based Testing
- Video Based Requirements Engineering
- Global Software Development
- Software Factories



Captain! There are too many features. I can't see the impact of this change request.



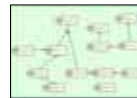
The Challenge

To determine the impact of a feature change on large, complex software systems with many features.

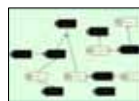
Software Reconnaissance: Where in this program is feature X implemented?

To modify an old program safely, a software engineer often needs to understand how particular **features** are implemented in the current code.

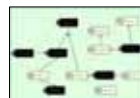
But how to **find** that code scattered through the hundreds of thousands of lines making up the program?



WITH



WITHOUT



COMPARE



Solution, track execution

1. Run tests with the feature
2. Run tests without the feature

Compare what was executed

- Components executed in the first group of tests, but not in the second group
- **LOOK HERE FIRST!**

Software Reconnaissance Tools: *TraceGraph gives the broad picture*

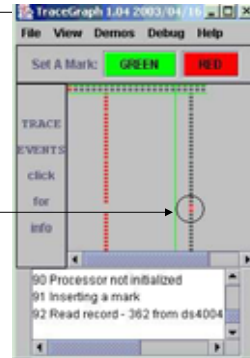
The rectangles show what is executed

- Each column is a test
- Each row is a component

First columns are tests without the feature,
last column is the test with it

- Components that only show in the LAST COLUMN are the ones to investigate
- Rectangles are only a few pixels, so you can see the broad picture on one screen

You can try an on-line demo to see
TraceGraph analyzing a web server as it
serves up different kinds of web page



<http://www.cs.uwf.edu/~recon/recon3/r3wDemo.htm>

Slide courtesy of Norman Wilde, Sharon Simmons, Dennis Edwards, University of West Florida

Page 57

Copyright 2007 © Siemens Corporate Research

Software Reconnaissance Status:

Developed for > 10 years at University of West Florida

- Support from the *Software Engineering Research Center* and the *Air Force Office of Scientific Research*
- Many published case studies have established benefits and limitations
- Current projects ongoing with Motorola and Northrop Grumman



Benefits

- Finds a small number of good places to start efficient code exploration
- Requires only the "as-built" system and a few test cases

Drawbacks

- You need to instrument the system to track what components are executed
- You need to understand the code after you have located it!



Interactive on-line demo at:

<http://www.cs.uwf.edu/~recon/recon3/r3wDemo.htm>

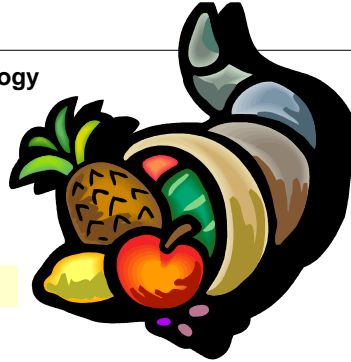
Slide courtesy of Norman Wilde, Sharon Simmons, Dennis Edwards, University of West Florida

Page 58

Copyright 2007 © Siemens Corporate Research

Agenda

- Introduction to Siemens Corporate Technology
- Rapid Prototyping
- Dynamic Tracing
- Software Reconnaissance
- Model Driven Requirements Engineering
 - Unified Requirements Modeling Language
 - Heuristics
 - Design Model Quality Assurance
 - Automated Extraction of Requirements from Models
- Model Based Testing
- Video Based Requirements Engineering
- Global Software Development
- Software Factories

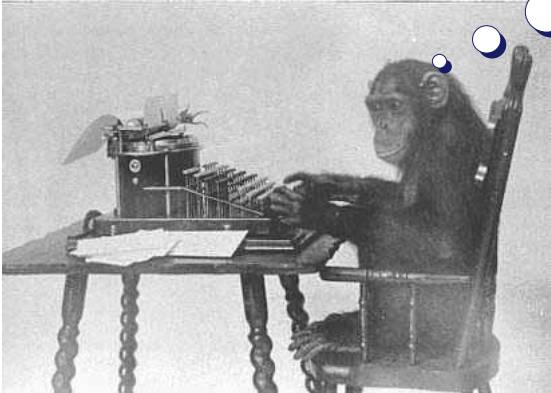


The Challenge

*To rapidly create a product definition
using verifiable, precise, scalable easy
to understand methods.*

SIEMENS

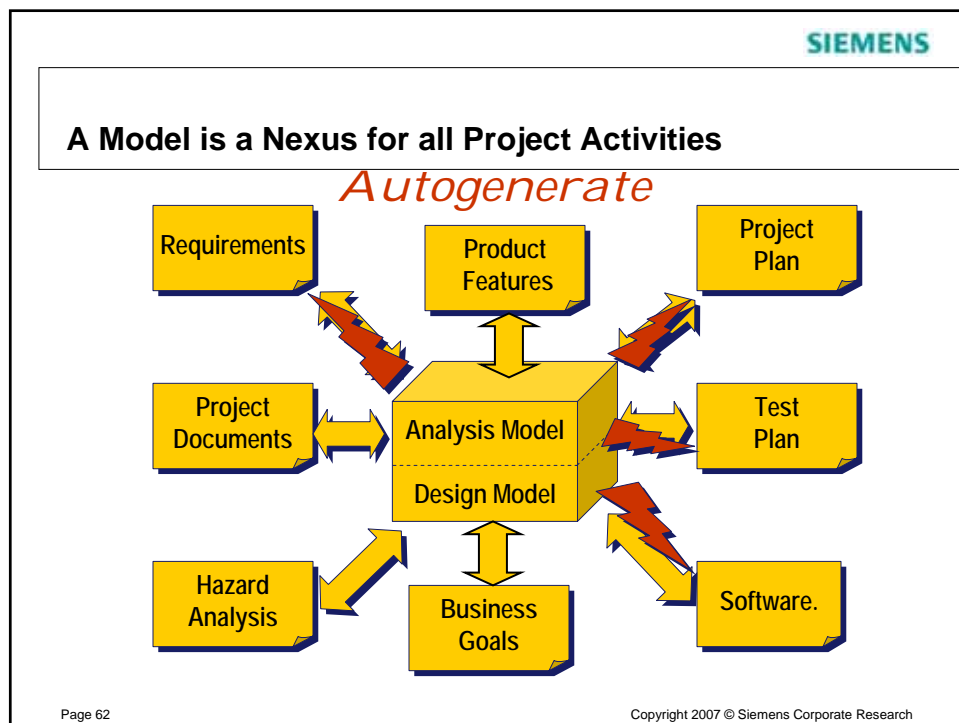
Words, words, words, I'm so sick of words



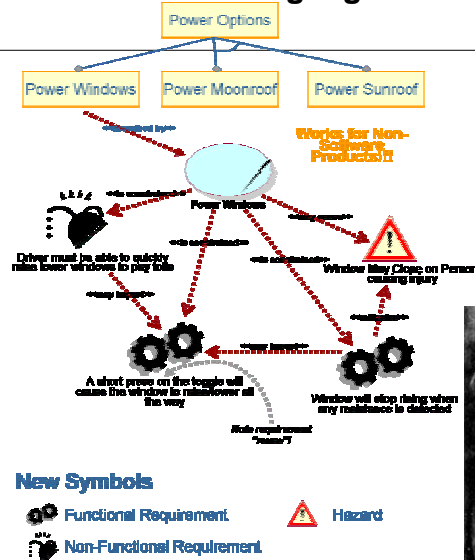
The Requirements Analyst

Page 61

Copyright 2007 © Siemens Corporate Research



A Model is a Visual Language



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

What is a Unified Requirements Model?

The Unified Requirements Modeling Language (URML) is a standard notation for the modeling of requirements artifacts as a first step in developing a product or performing business modeling. It incorporates elements of:

- Use cases (including UML related artifacts)
- Feature Modeling
- Hazard Analysis
- Threat Modeling
- Other requirements related visual modeling techniques

A **unified requirements model** is a model that has been created using the URML.

Rationale for a Unified Model

It has been found empirically that text specifications are inherently ambiguous. The ambiguity is exacerbated with global software development, different languages and cultures.

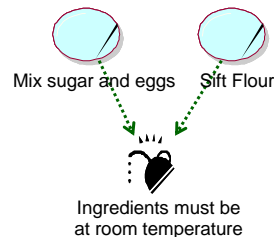
Graphical models are inherently unambiguous

“Mix the sugar and eggs.
Sift the flour. The
ingredients must be at
room temperature”

Mix what? – the sugar, the
eggs the flour or some
combo?

Page 65

VS.



Unambiguous!

Copyright 2007 © Siemens Corporate Research

Rationale for a Unified Model (2)

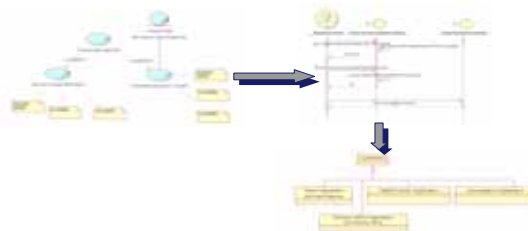
It is extraordinarily difficult to get stakeholders across multiple time zones to effectively review complex requirement specifications

Graphical models can be reviewed by all stakeholders together with minimal language or cultural issues.

This



VS.



Page 66

Copyright 2007 © Siemens Corporate Research

Rationale for a Unified Model (3)

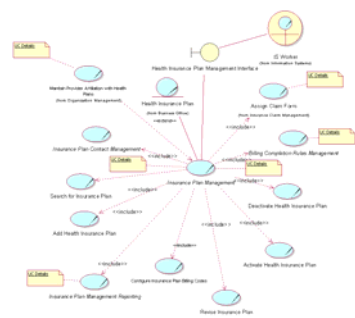
Hierarchical Storage Mechanisms for requirements breakdown with cross-cutting requirements and scale.

Graphical models size nicely and manage crosscutting requirements without difficulty

Req. Name	Trace Map	Trace Map
REQ-1001: The car shall have a power window.	REQ-1001: The car shall have a power window.	REQ-1001: The car shall have a power window.
REQ-1002: The car shall have a power window.	REQ-1002: The car shall have a power window.	REQ-1002: The car shall have a power window.
REQ-1003: The car shall have a power window.	REQ-1003: The car shall have a power window.	REQ-1003: The car shall have a power window.
REQ-1004: The car shall have a power window.	REQ-1004: The car shall have a power window.	REQ-1004: The car shall have a power window.
REQ-1005: The car shall have a power window.	REQ-1005: The car shall have a power window.	REQ-1005: The car shall have a power window.
REQ-1006: The car shall have a power window.	REQ-1006: The car shall have a power window.	REQ-1006: The car shall have a power window.
REQ-1007: The car shall have a power window.	REQ-1007: The car shall have a power window.	REQ-1007: The car shall have a power window.
REQ-1008: The car shall have a power window.	REQ-1008: The car shall have a power window.	REQ-1008: The car shall have a power window.
REQ-1009: The car shall have a power window.	REQ-1009: The car shall have a power window.	REQ-1009: The car shall have a power window.
REQ-1010: The car shall have a power window.	REQ-1010: The car shall have a power window.	REQ-1010: The car shall have a power window.
REQ-1011: The car shall have a power window.	REQ-1011: The car shall have a power window.	REQ-1011: The car shall have a power window.
REQ-1012: The car shall have a power window.	REQ-1012: The car shall have a power window.	REQ-1012: The car shall have a power window.
REQ-1013: The car shall have a power window.	REQ-1013: The car shall have a power window.	REQ-1013: The car shall have a power window.
REQ-1014: The car shall have a power window.	REQ-1014: The car shall have a power window.	REQ-1014: The car shall have a power window.
REQ-1015: The car shall have a power window.	REQ-1015: The car shall have a power window.	REQ-1015: The car shall have a power window.
REQ-1016: The car shall have a power window.	REQ-1016: The car shall have a power window.	REQ-1016: The car shall have a power window.
REQ-1017: The car shall have a power window.	REQ-1017: The car shall have a power window.	REQ-1017: The car shall have a power window.
REQ-1018: The car shall have a power window.	REQ-1018: The car shall have a power window.	REQ-1018: The car shall have a power window.
REQ-1019: The car shall have a power window.	REQ-1019: The car shall have a power window.	REQ-1019: The car shall have a power window.
REQ-1020: The car shall have a power window.	REQ-1020: The car shall have a power window.	REQ-1020: The car shall have a power window.

Trace Map

VS.

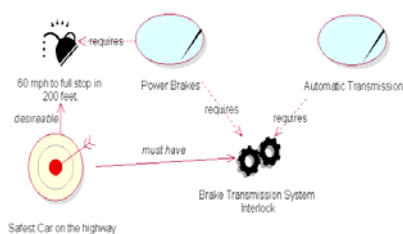


Rationale for a Unified Model (4)

Requirements Relationships are naturally organized in directed graphs

Traditional RE databases support a tree structure
Requirements database

URML



FEAT 3.5 Transmission
SAFE 3.5.11 Shall Have a Brake Transmission System Interlock
FEAT 4.2 Braking System
SAFE 4.2.44 Shall go from 60 mph to full stop in 200 feet.

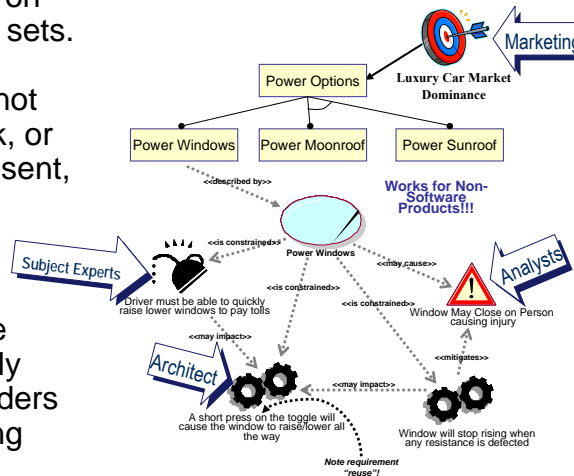
➤Where did the business requirement go?
➤Where did the interlock requirement go for the braking system?

ANSWER: They can only be found by looking at traces
IF THE TRACES WERE MANUALLY ADDED!

Rationale for a Unified Model (5)

Different teams work on different requirement sets. With a conventional approach, they may not see each others work, or the traces may be absent, or may break across different media.

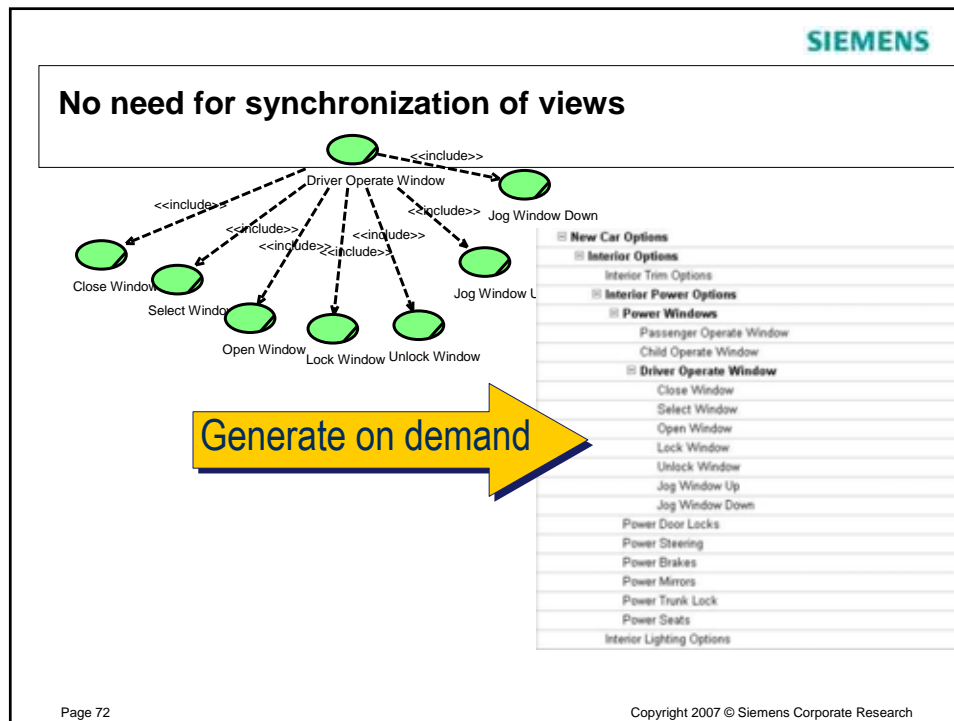
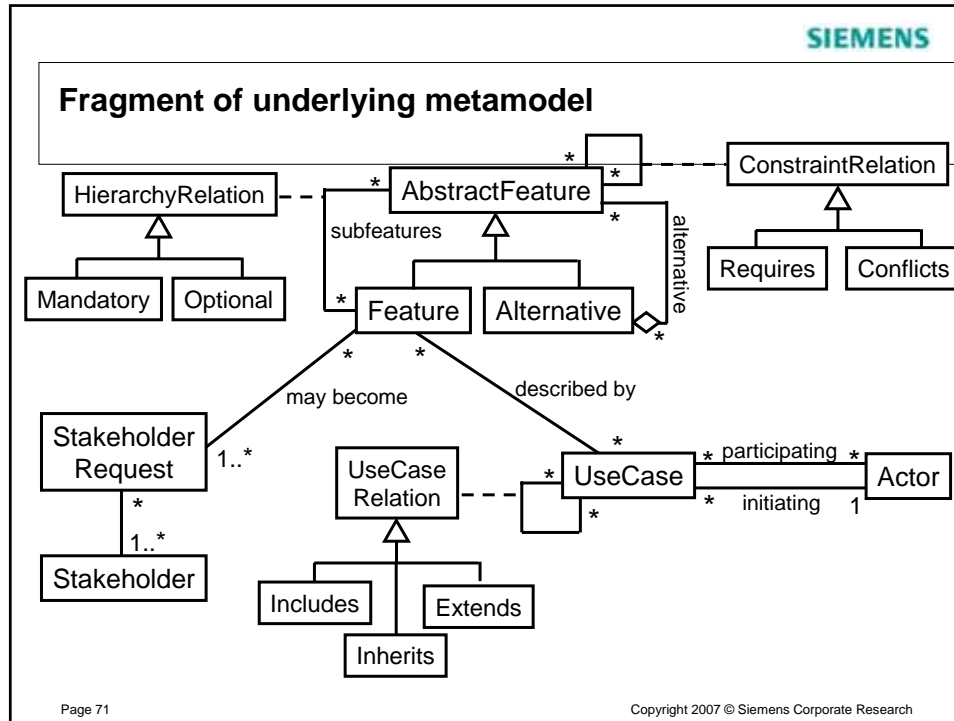
With a URM all requirements that are related are intrinsically visible to all stakeholders with no manual tracing required.



Requirements Analysis

A Unified Requirements Model can enable process:

- ✓ Force correct choices
- ✓ Preemptively find errors
- ✓ Simplify the complex



Experiences Piloting the URML*

- Piloted on a project to create control systems for airport baggage handling
- Analysts were in Michigan, Princeton, Vienna and Bratislava
- URML used to define high level requirements and very complex control algorithms
- Using diagrams resulted in catching mistakes that were in the first textual description of the algorithms
- Project completed on-time and in-budget
- Analysts at all locations very much preferred visual reviews to reviews of text



Experiences Piloting the URML* (2)

- Piloted on a project to create a mail sorting system for the U.S. Postal Service (USPS)
- Analysts were in Texas, Princeton, Florida and Australia
- URML used to define high level USPS requirements
- Using diagrams resulted in effective reviews with minimal rework
- Project completed on-time and in-budget
- The USPS management appreciated the graphical approach and were enthusiastic reviewers
- Follow-on \$40 Million contract awarded



Conclusions

The URML has worked on pilots

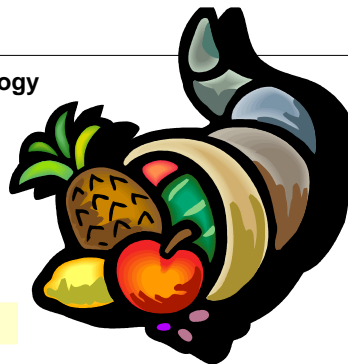
It needs more investigation and the completion of a prototypical tool

It has proved to be superior to natural language, or a combination of pure UML and a requirements database (traces less fragile)

It is superior to text for large projects in terms of scalability, crosscutting and navigation.

Agenda

- Introduction to Siemens Corporate Technology
- Rapid Prototyping
- Dynamic Tracing
- Software Reconnaissance
- Model Driven Requirements Engineering
 - Unified Requirements Modeling Language
 - Heuristics
 - Design Model Quality Assurance
 - Automated Extraction of Requirements from Models
- Model Based Testing
- Video Based Requirements Engineering
- Global Software Development
- Software Factories



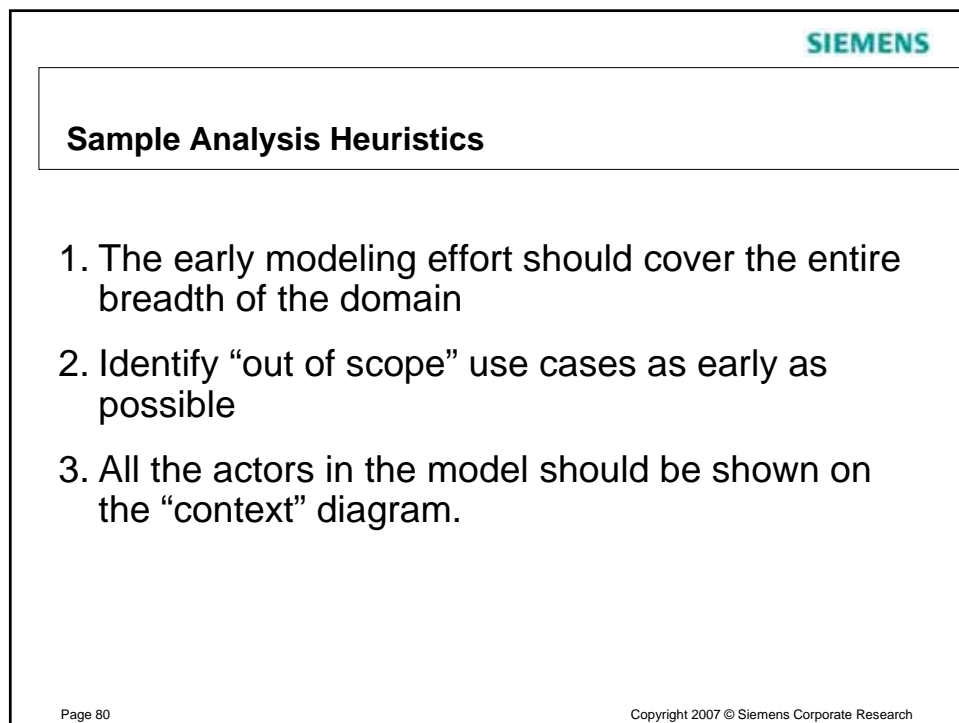
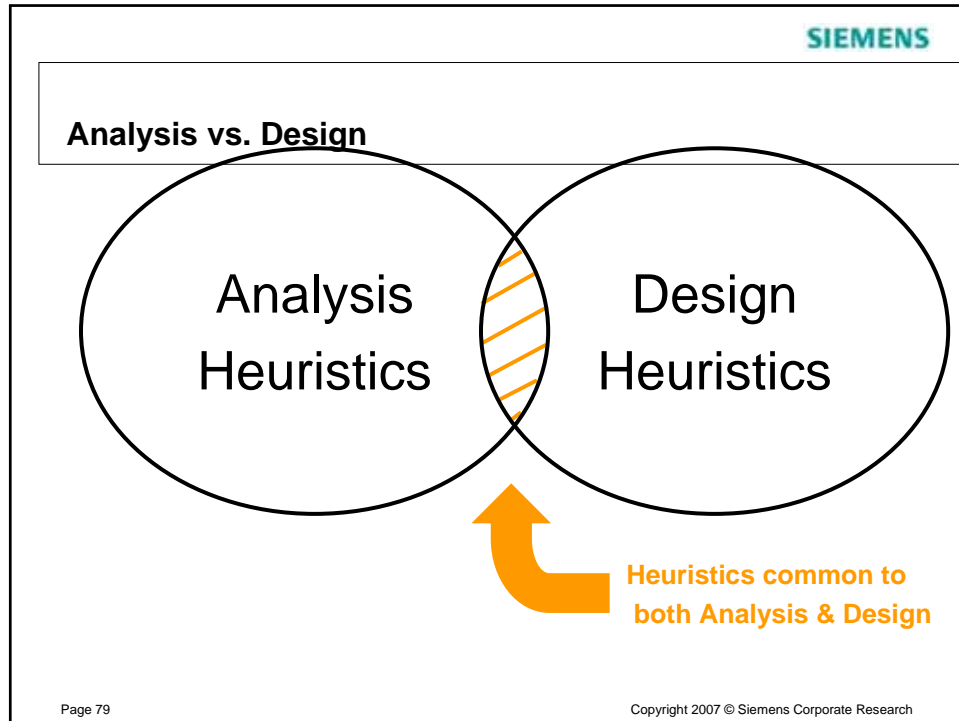
A Challenge

*To provide RE processes and tools that support
VERY LARGE projects with a variety of
work products.*

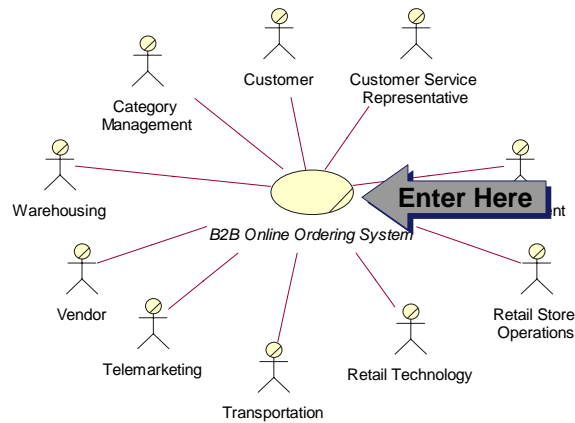
Analysis & Design Heuristics

Heuristics are needed for large models:

- Programmatically Verifiable
- Require Human Cognition

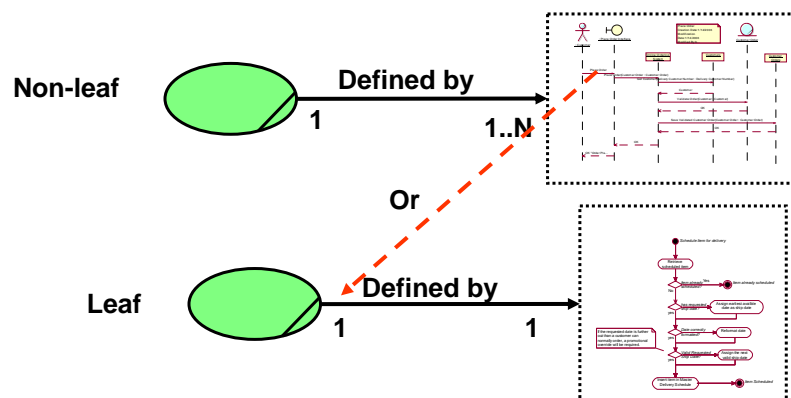


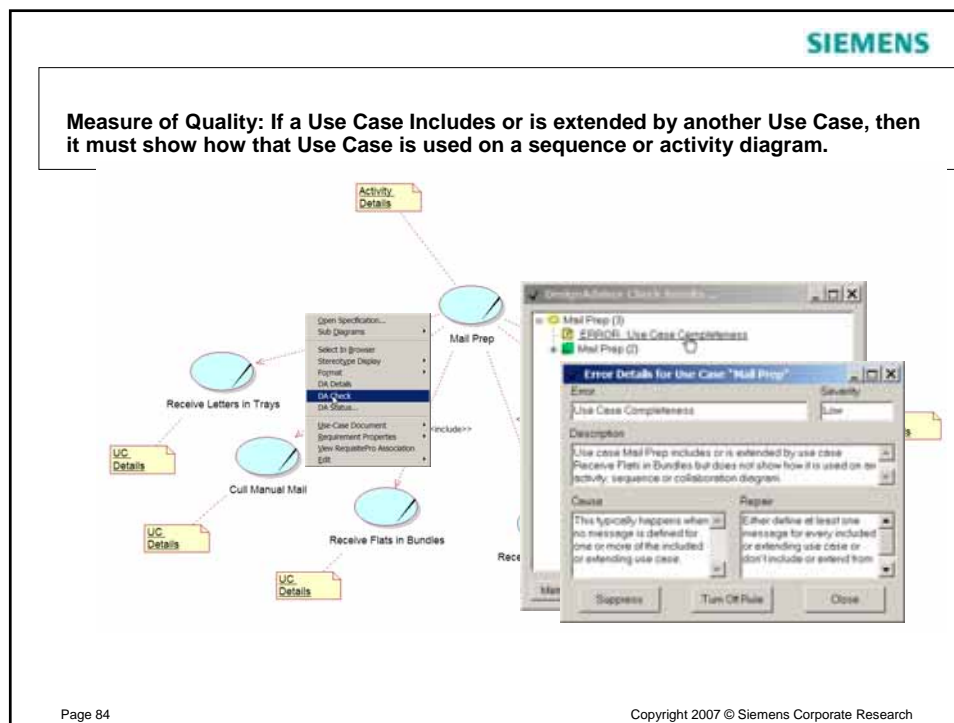
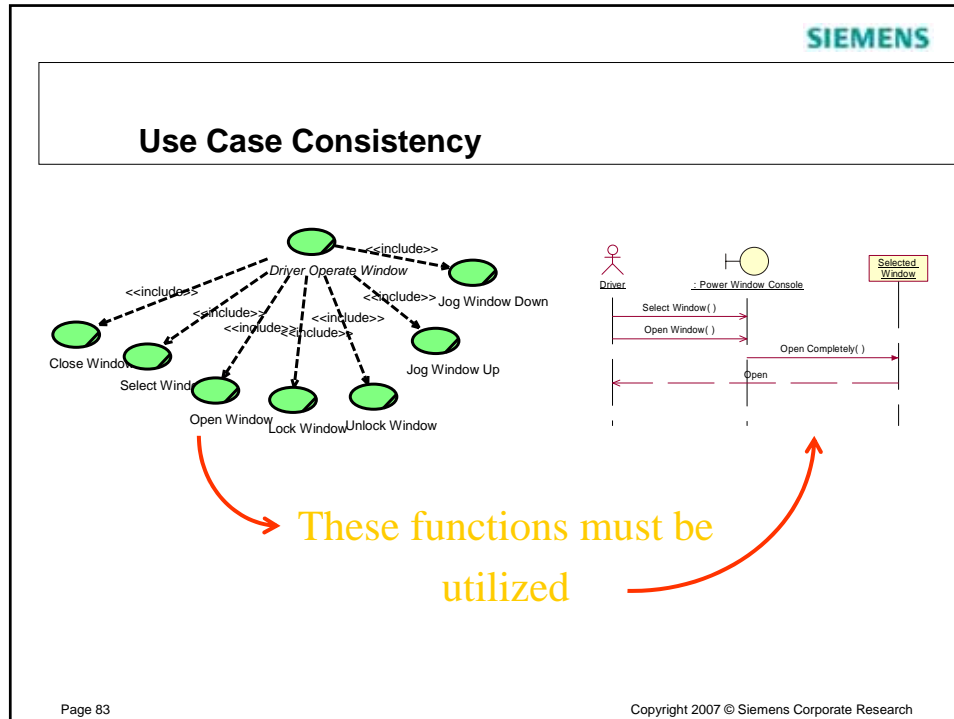
Model Entry Point



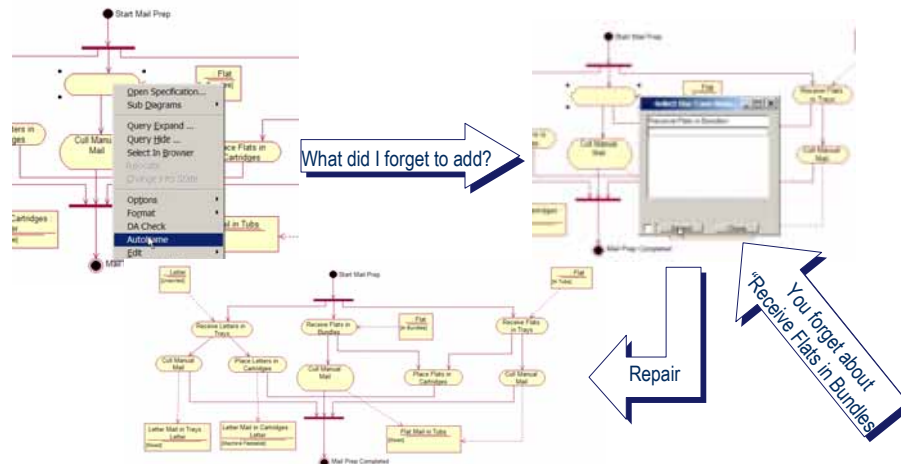
The early modeling effort should cover the entire breadth of the domain

Every concrete use case must be defined





Measure of Quality: If a Use Case Includes or is extended by another Use Case, then it must show how that Use Case is used on a sequence or activity diagram – tool support



Page 85

Copyright 2007 © Siemens Corporate Research

Agenda

- Introduction to Siemens Corporate Technology
- Rapid Prototyping
- Dynamic Tracing
- Software Reconnaissance
- Model Driven Requirements Engineering
 - Unified Requirements Modeling Language
 - Heuristics
 - Design Model Quality Assurance
 - Automated Extraction of Requirements from Models
- Model Based Testing
- Video Based Requirements Engineering
- Global Software Development
- Software Factories



Page 86

Copyright 2007 © Siemens Corporate Research

Evaluating the Quality of a Design



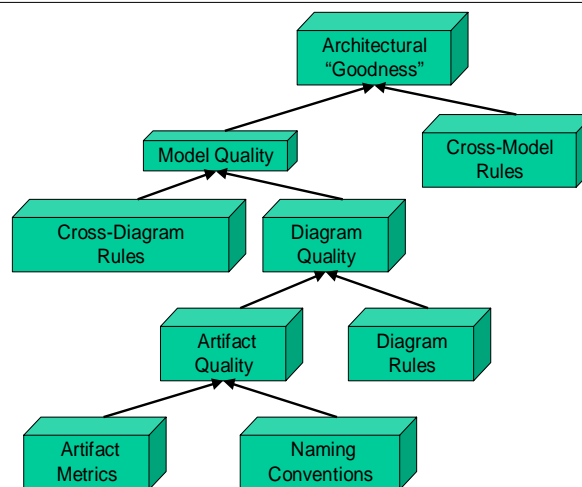
Oh dear, I found a base class using a method from a derived class. I must disintegrate all the developers!

Darth Quality

Page 87

Copyright 2007 © Siemens Corporate Research

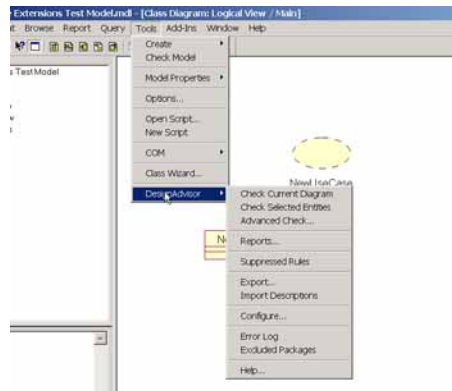
Evaluating the Quality of a Design



Page 88

Copyright 2007 © Siemens Corporate Research

Siemens DesignAdvisor – Automated Error Checking



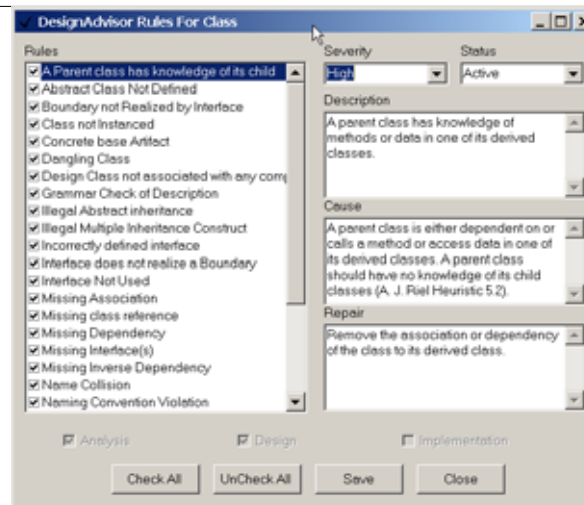
CASE Tool Plug-In

Configurable Error & Metric Checks

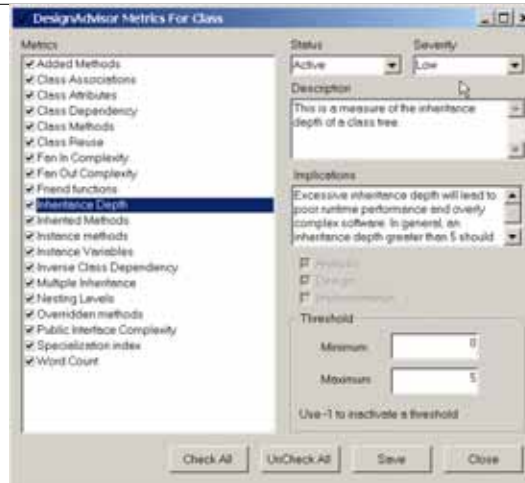
Export Requirements, Project Tasks & Test Cases

Round Trip-Descriptions

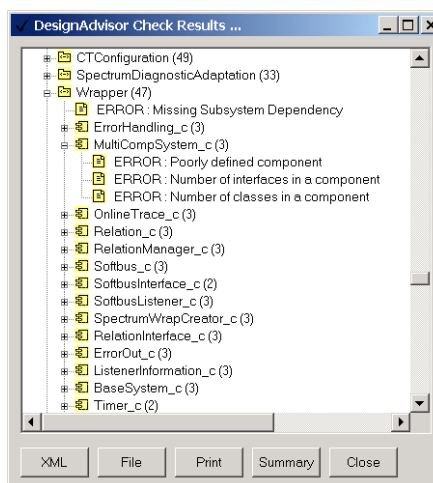
Configuring Rules



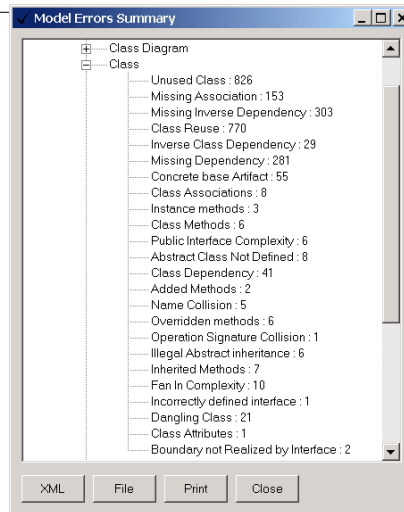
Configuring Metrics



Making Sense of the Results



Summarizing the Results



Model Comparison

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6
Type	Analysis	Analysis & Design	Analysis	Reversed Code	Design	Reversed Code
Domain	Health	Trans.	Chemicals	Trans.	Power	Health
Classes	384	1105	243	1570	1104	1268
Use Cases	1121	35	86	0	0	0
Total Errors	7014	10742	2243	5319	11733	18677

Reversed Code Model Typical Errors

Class Associations - 42	Public information in base Class - 12
Name Collision - 30	Circular Associations - 8
Concrete base Artifact - 165	Subsystem Depth - 4
Incorrectly defined interface - 97	Empty Component - 444
A Parent class has knowledge of its child - 32	Poorly defined component - 1323
Abstract Class Not Defined - 26	Interfaces in a component - 1572
Illegal Multiple Inheritance Construct - 8	Number of classes in a component - 1507
Multiple Inheritance - 10	
Illegal Abstract inheritance - 34	
Instance Variables - 1	
Class Attributes - 1	
Public Interface Complexity - 3	

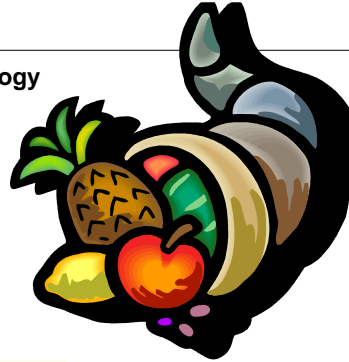
Interesting Metrics -> Reversed Model

Total classes1267	Total attributes10587
Abstract classes..... 72	Average attributes per class..... 8
Concrete classes..... 1195	Maximum attributes per class. 2119
Total methods22785	Public attributes 6647
Average methods per class... 17	Protected attributes 147
Maximum methods per class.698	Private attributes 3738
Public methods 21182	
Protected methods 651	
Private methods 885	

**GOD
Class**

Agenda

- Introduction to Siemens Corporate Technology
- Rapid Prototyping
- Dynamic Tracing
- Software Reconnaissance
- Model Driven Requirements Engineering
 - Unified Requirements Modeling Language
 - Heuristics
 - Design Model Quality Assurance
 - Automated Extraction of Requirements from Models
- Model Based Testing
- Video Based Requirements Engineering
- Global Software Development
- Software Factories



Acyclic Directed Graph

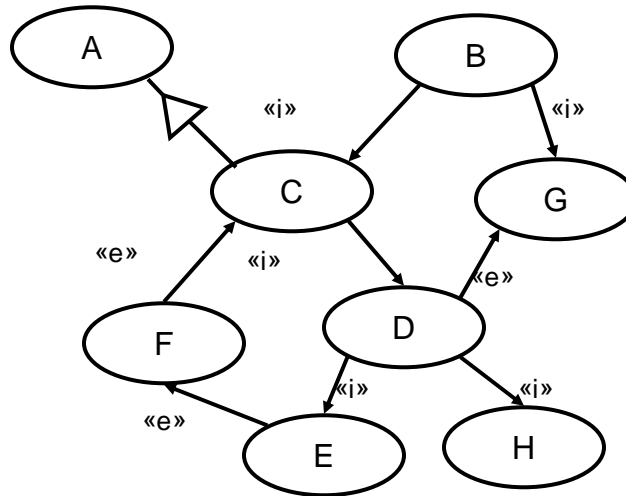
Definition:* A graph whose edges are *ordered* pairs of vertices. That is, each edge can be followed from one vertex to the next.

Formal Definition: A graph G is a pair (V, E) , where V is a set of vertices, and E is a set of edges between the vertices $E = \{(u, v) \mid u, v \in V\}$. If the graph does not allow self-loops, adjacency is irreflexive, that is $E = \{(u, v) \mid u, v \in V, u \neq v\}$.

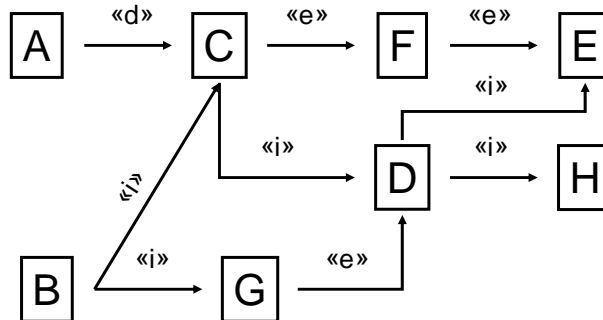
Also known as digraph, oriented graph.

*National Institute of Standards and Technology

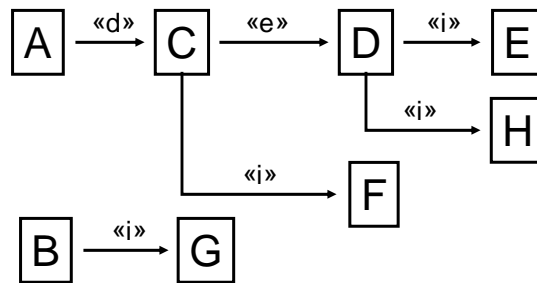
Sample Use Case Model



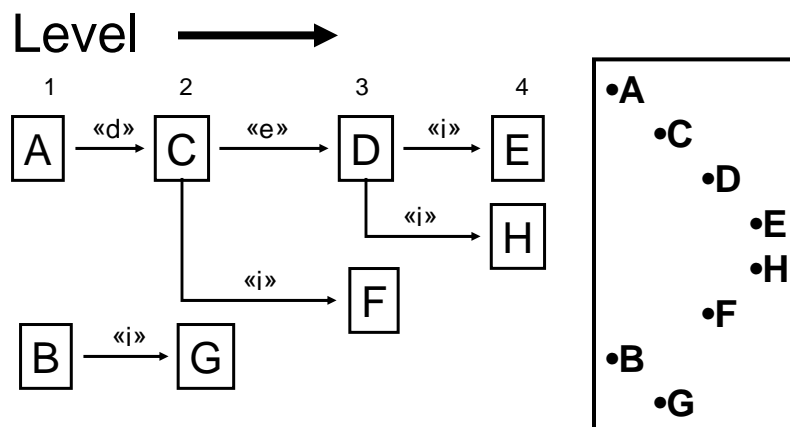
As a Directed Graph

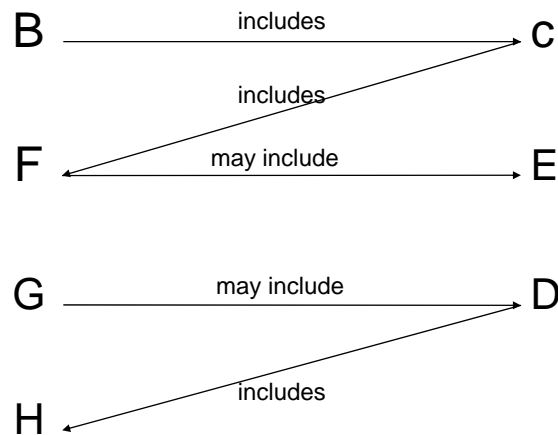


As a Set of Trees
























Depth-First Search to Extract Requirements



Complete by Adding Traces**The extraction is scalable**

The requirements extraction process has worked successfully on models with about a thousand use cases!

Requirements Extracted from Large Model

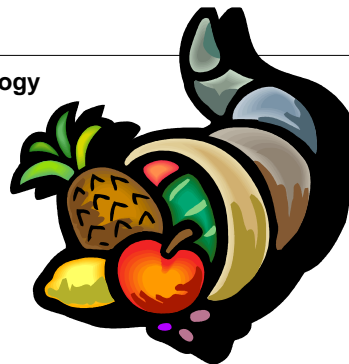
634		Facilities Management
635		Location Management
636		Add Location
637		Revise Location
638		Location Structure Management
639		Create Location Structure
640		Copy Existing Location Structure
641		Revise Location Structure
642		Add Location to Location Structure
643		Remove Location from Location Structure
644		Activate Location Structure
645		Deactivate Location Structure
646		Validate Location Structure
647		Maintain Location Floorplan
648		Location Availability Management
649		Deactivate Location
650		Activate Location
651		Modify Location Occupancy
652		Update Patient Location Condition
653		Assign Location Usages
654		Search for Location

Page 105

Copyright 2007 © Siemens Corporate Research

Agenda


- Introduction to Siemens Corporate Technology
- Rapid Prototyping
- Dynamic Tracing
- Software Reconnaissance
- Model Driven Requirements Engineering
 - Unified Requirements Modeling Language
 - Heuristics
 - Design Model Quality Assurance
 - Automated Extraction of Requirements from Models
- Model Based Testing
- Video Based Requirements Engineering
- Global Software Development
- Software Factories



Page 106

Copyright 2007 © Siemens Corporate Research

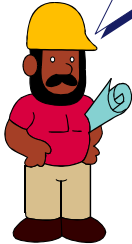
SIEMENS



Page 107

Copyright 2007 © Siemens Corporate Research

SIEMENS



If only they had used automatic test case generation!

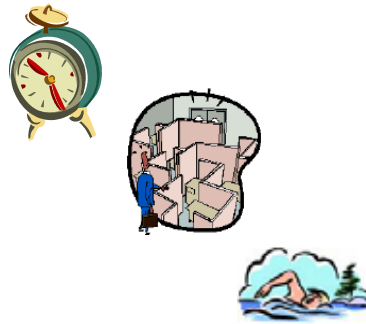
Page 108

Copyright 2007 © Siemens Corporate Research

Testing Overview

What are typical problems on testing software?

- Time is limited
- Applications are complex
- Requirements are fluid



Need to Increase Productivity!

Some Insights on Testing

- No matter how rigorous we are, software is going to be faulty
- Testing represent a substantial percentage of software development costs and time to market
- Impossible to test under all operating conditions
 - Based on incomplete testing, we must gain confidence that the system has the desired behavior
 - Testing cannot show the absence of bugs
- Testing large systems is complex
 - Testability is a design issue
 - It requires strategy and technology- and is often done inefficiently in practice

Model-Based Testing

MBT offers an opportunity to increase tester productivity

- Aims to automatically generate test cases using models created during analysis & development process or extracted from artifacts of that process

Average testers are already using models.

- All testing creation activities are based (at least) on mental model

MBT occurs when the model is:

- Recorded in some form, formalized, and used for generating test cases and/or oracles

Modeling for TDE/UML

TDE/UML Workflow

Modeling

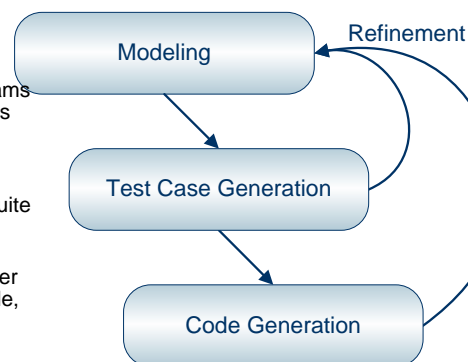
- Focus on UML Use Case Diagrams refined by UML Activity Diagrams

Test Case Generation

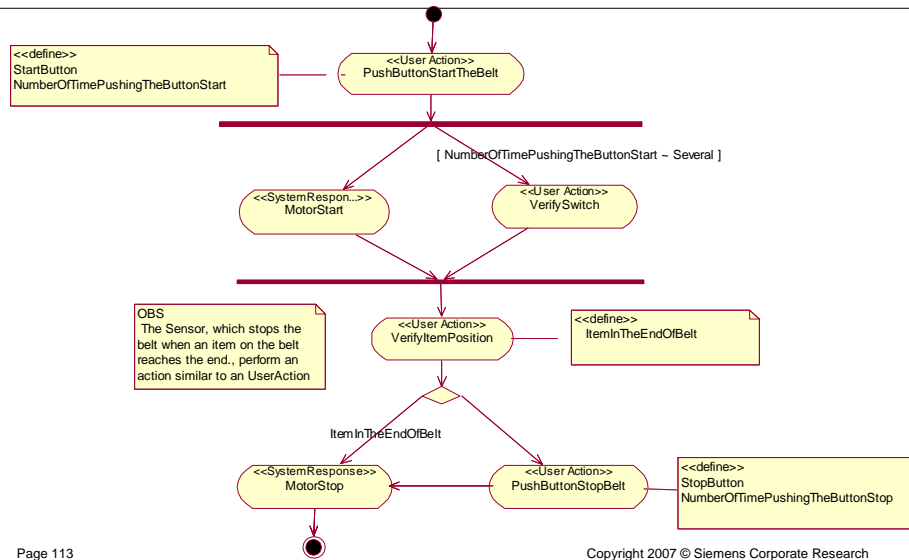
- Combines Graph Coverage and Data Coverage to form a Test Suite

Code Generation

- Transforms a Test Suite into other formats, such as executable code, reports, xml...



Testing Conveyor Belt Control – Activity Diagram



Test Case Generation for the Example

Generation Results:

- Total of 14 test cases.
- 2 Paths to be covered and 14 different data set to be used to cover those paths

Generation Summary :

Results:

- Number of Paths: 2
- Number of Non-Completed Paths: 0
- Activities Covered: All activities were covered by this test suite
- Transitions Covered: All transitions were covered by this test suite
- Test Data Generated: The following number of test data were generated for the test paths:
 - 4 test data for Generation1_TP_1
 - 10 test data for Generation1_TP_2

Statistics:

- Use Case: Controlling a Conveyor Belt
 - PushButtonStartTheBelt: in 2 test case (100%)
 - MotorStart: in 2 test case (100%)
 - VerifySwitch: in 2 test case (100%)
 - VerifyItemPosition: in 2 test case (100%)
 - MotorStop: in 2 test case (100%)
 - PushButtonStopBelt: in 1 test case (50%)

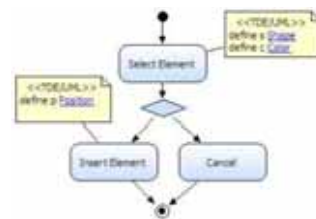
Test Case Generation

Data Coverage Criterion

- sampling
- choice-per-suite
- choice-per-variable
- choice-per-path
- **exhaustive**

Color	Shape	Position
black	circle	Center
blue	square	Left
green	triangle	Right
yellow		

2 Tests, 48 Procedures



Page 115

Copyright 2007 © Siemens Corporate Research

Code Generation

TDE/UML comes with pre-packaged code generators:

- Text file
- CSV file
- Word report
- Custom code

Each project can implement its own code generator

Code generations can use:

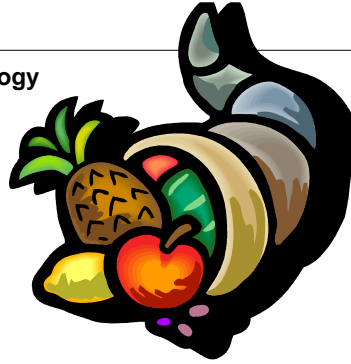
- properties defined in notes
- configuration parameters defined in code generator configuration file
- properties of choices defined by attributes of categories

Page 116

Copyright 2007 © Siemens Corporate Research

Agenda

- Introduction to Siemens Corporate Technology
- Rapid Prototyping
- Dynamic Tracing
- Software Reconnaissance
- Model Driven Requirements Engineering
 - Unified Requirements Modeling Language
 - Heuristics
 - Design Model Quality Assurance
 - Automated Extraction of Requirements from Models
- Model Based Testing
- Video Based Requirements Engineering
- Global Software Development
- Software Factories



The Challenge

To understand the exact context or meaning of a requirement as the customer understands it.

Real-time meeting capture

Focus Group Meeting in a hospital

If somebody with a serious injury is hospitalized, we immediately have to take care of that



When updating a patient record, do nurses get interrupted? And why?

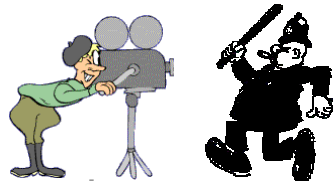
Captured meeting record, split into requirement clips



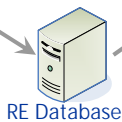
Developer in China

Real-time Customer Context

Following a 'blue collar' worker



That's what we have identified as requirements

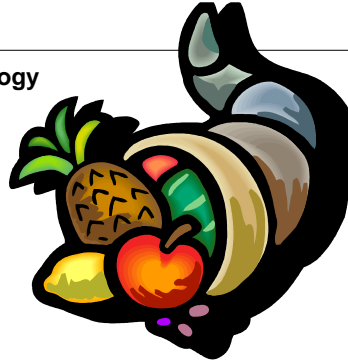


Meeting with subject matter experts



Agenda

- Introduction to Siemens Corporate Technology
- Rapid Prototyping
- Dynamic Tracing
- Software Reconnaissance
- Model Driven Requirements Engineering
 - Unified Requirements Modeling Language
 - Heuristics
 - Design Model Quality Assurance
 - Automated Extraction of Requirements from Models
- Model Based Testing
- Video Based Requirements Engineering
- Global Software Development
- Software Factories



Global Software Development (GSD): Motivation

Lower development costs

- *Low-wage countries*

Capitalizing on a pool of trained workforce

- *Quest for talent*

Increased output, reduced time

- *Improve time-to-market*
- *Round-the-clock development*

Market proximity

- *Specific local expertise*
- *Market acquisition effort*

Governmental policies and incentives

Original motivation was reduced development cost, but there are other reasons for GSD.

SIEMENS

But, Global Software Development is Difficult.

The diagram shows a world map with two locations marked: one in North America (blue dot) and one in Asia (green dot). A vertical double-headed arrow between the dots is labeled 'Physical distance'. A horizontal double-headed arrow below the map is labeled 'Time Difference: 11.5 hours'.

Page 123 Copyright 2007 © Siemens Corporate Research

SIEMENS

Global Software Development

Siemens has more than 15 years experience developing software products globally.

Today, most Siemens business groups develop software in low-cost countries.

Organization and process model for global development divides responsibility between a central headquarters organization & remote development teams.

The diagram illustrates the organization and process model for global software development. It shows a 'Central team' on the left and 'Remote development teams' on the right, both containing 'Domain expertise', 'Management & Process expertise', and 'Software engineering expertise'. The 'Central team' also includes 'Supplier Managers'. The process flow is as follows: 'Requirements' and 'Architecture & design' flow from the Central team to the Remote teams. 'Development plan' and 'Acceptance tests' flow from the Remote teams back to the Central team. 'Component requirements' and 'Component design' flow from the Central team to the Remote teams. 'Code' and 'Tests' flow from the Remote teams back to the Central team. Verification and integration steps are shown at the bottom: 'Verified by' (Central to Remote), 'Integrated by' (Remote to Central), and 'Verified by' (Central to Remote).

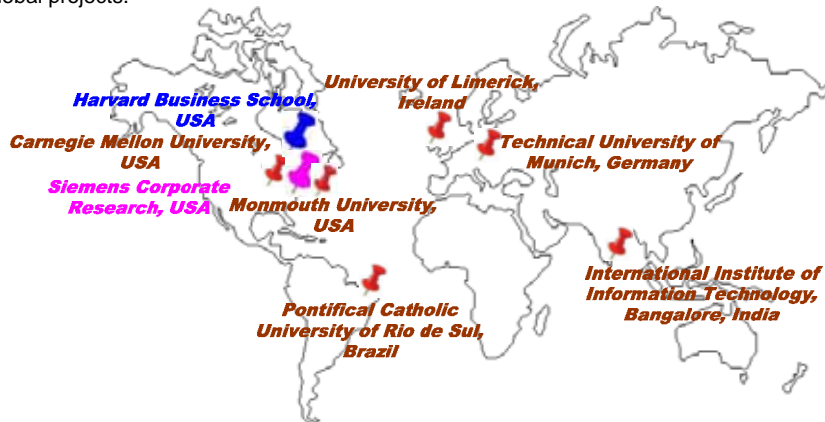
Page 124 Copyright 2007 © Siemens Corporate Research

Global Studio Project (GSP)

Experimental global software development project using university student teams and researchers.

Shadows real Siemens global development project, process, & organization.

Document processes, best practices, and understanding of how to successfully execute global projects.



GSP Research Goals

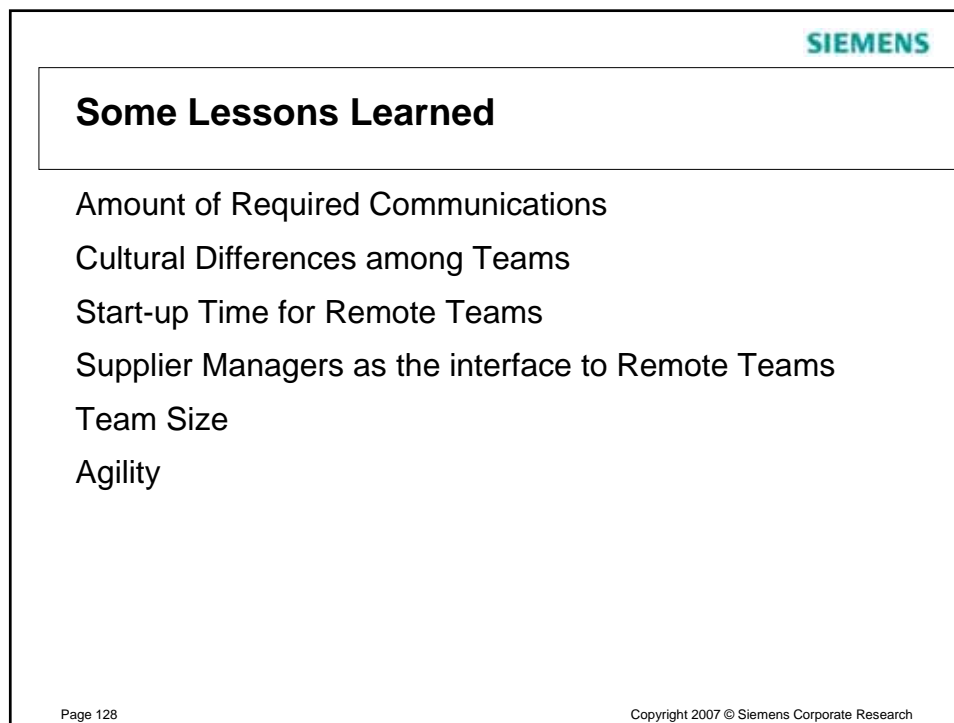
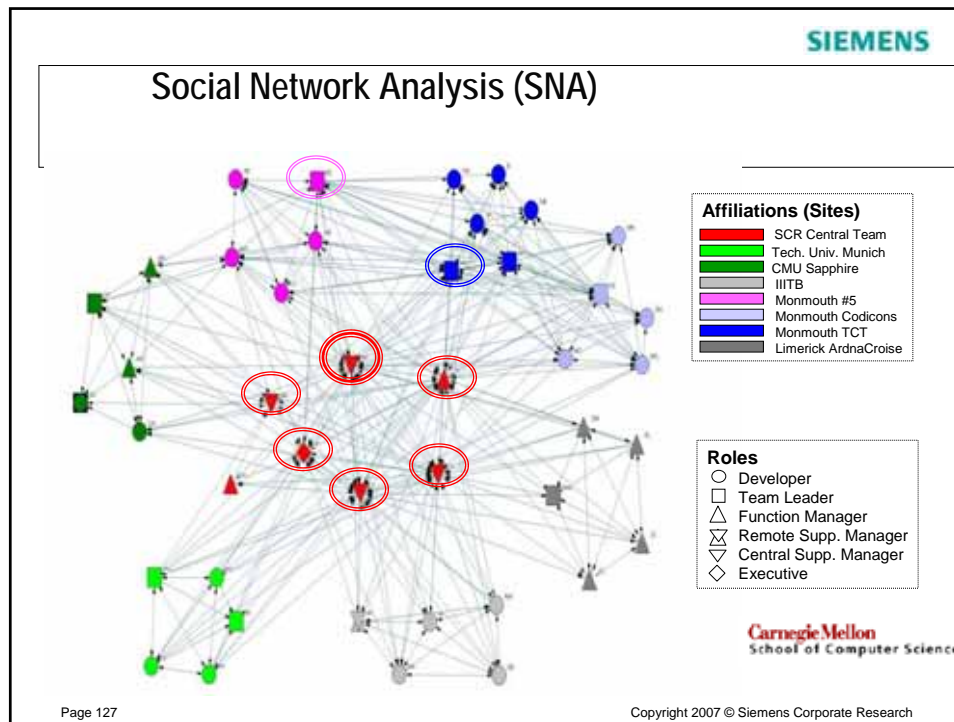
Identify and document **best (and worst) practices** for global software development.

Identify the **prerequisites** for **successful global software development**.

Test a global software development **reference process**.

Determine **artifacts** for **commissioning a remote development team**.

Identify **communications necessary** for effective global development.



Open Issues and Research Questions

1. Given that the technical artifacts that are delivered to the remote component development teams are not adequate in specifying the precise work to be done, in what ways are they deficient?
2. What strategies do the remote teams employ to compensate for the deficiencies found in the received technical artifacts?
3. What are the early warning signs that an issue is imminent? Can communication patterns, for example, between the central and remote teams be used to predict future component integration problems?

Agenda

- Introduction to Siemens Corporate Technology
- Rapid Prototyping
- Dynamic Tracing
- Software Reconnaissance
- Model Driven Requirements Engineering
 - Unified Requirements Modeling Language
 - Heuristics
 - Design Model Quality Assurance
 - Automated Extraction of Requirements from Models
- Model Based Testing
- Video Based Requirements Engineering
- Global Software Development
- Software Factories



Software Factories

**Enable higher efficiency
in planning, development,
and maintenance.**



Job Management Software Factory – Prototype Definition

Goal 1: Tools

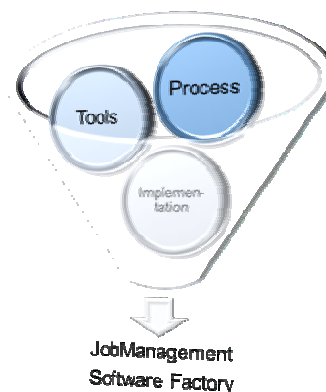
- DSL
- Guidance Packages
- Code & Artifact Generators

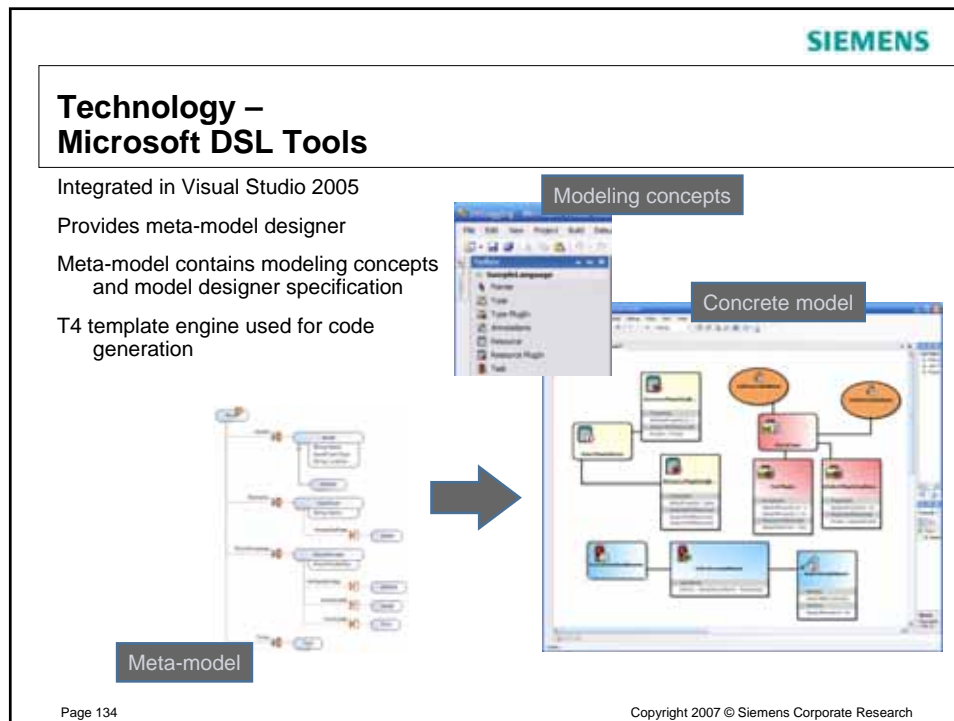
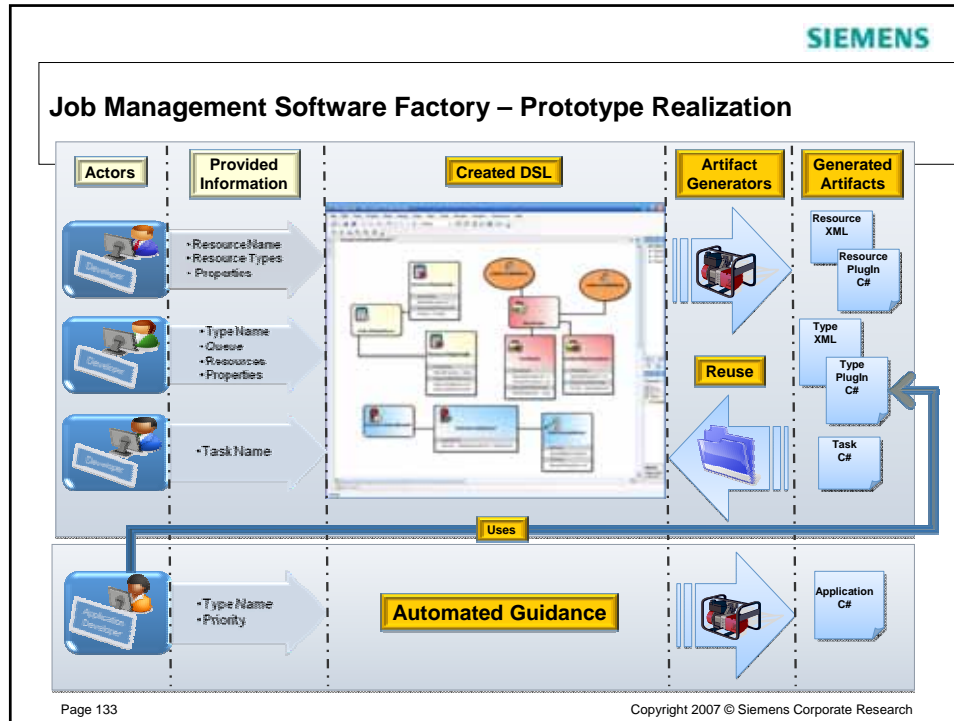
Goal 2: Implementation

- Job Types
- QSelect Plugins
- Resource Types
- Resource Plugins
- Executors
- Job Enqueue

Goal 3: Process

- Standardize
- Automate



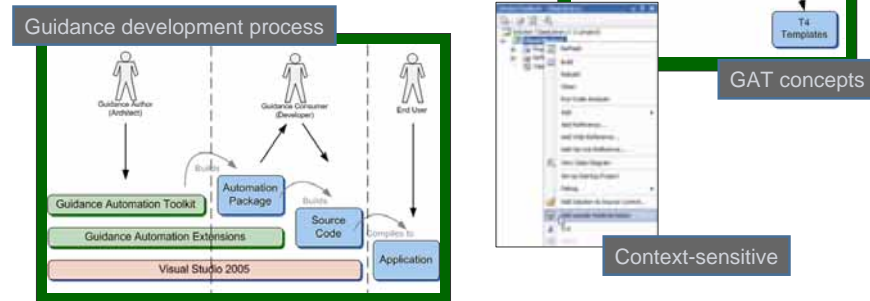


Technology – Microsoft Guidance and Automation Toolkit (GAT)

GAT directly plugs into the development environment, Visual Studio 2005

Provides context- and content-sensitive recipes, actions, templates, and others

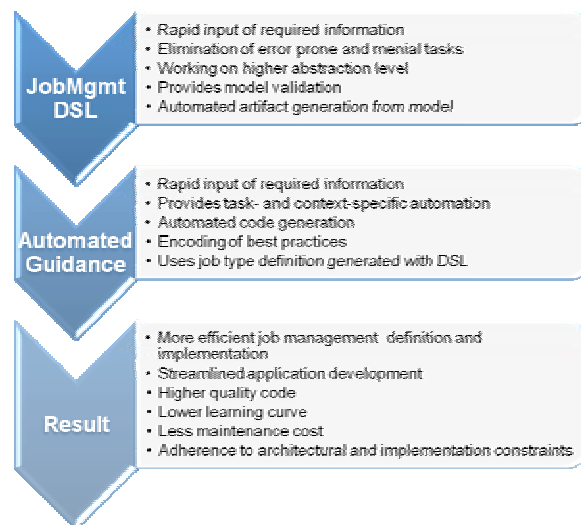
Helps automate menial tasks and providing guidance at the right place and time



Page 135

Copyright 2007 © Siemens Corporate Research

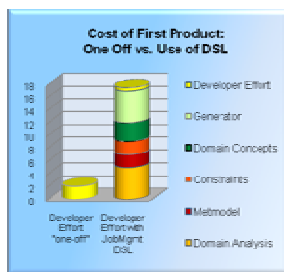
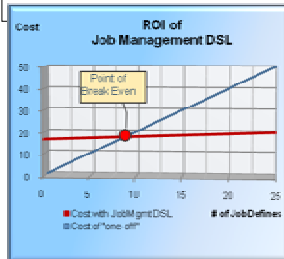
Job Management Software Factory – That's how it helps



Page 136

Copyright 2007 © Siemens Corporate Research

ROI: Job Management Domain Specific Language (DSL)



The JobManagement DSL increases the level of abstraction beyond the level of 3rd generation programming languages such as Java, C#, or VB.NET. This enables a tremendous increase in productivity as well as quality through the elimination of error prone and menial tasks and the systematic reuse of software components and generators.

Challenges

- Time constraints and therefore scenario with limited variability
- Analysis of the domain and the existing software development environment to identify best practices applicability of DSL
- Define domain concepts and software development process
- Define, identify and implement DSL that offer best ROI

Benefits

- Elimination of error prone, menial, and repetitive development tasks
- Enforcement of architecture and development constraints
- Automated code generation from models
- Reduced complexity for developers
- Faster development times and increased reuse
- Estimate: Break even point at ~7 job management definitions

Potential

- Work on higher level of abstraction
- Hidden complexity
- Less of a learning curve
- Enforce architectural constraints and encode of best practices

Questions?



Did Darth
QUALITY say
to disintegrate
the developers or
the audience???