An End-To-End Industrial Software Traceability Tool

Hazeline U. Asuncion – hasuncion@ics.uci.edu
Frédéric François - frederic@uxdesigner.com
Richard N. Taylor - taylor@ics.uci.edu
Sep 5, 2007

Setting

invensys Wonderware
A business unit of Invensys

- Leading supplier of industrial automation and information software
  - 450,000 software licenses
  - 100,000 industrial plants worldwide (~ 30% total)
- Mid-sized globally distributed software development company
  - Based in Lake Forest, CA
  - Development centers: US, Australia, EMEA, India
  - 40+ Individuals projects currently under development
  - 250+ development employees

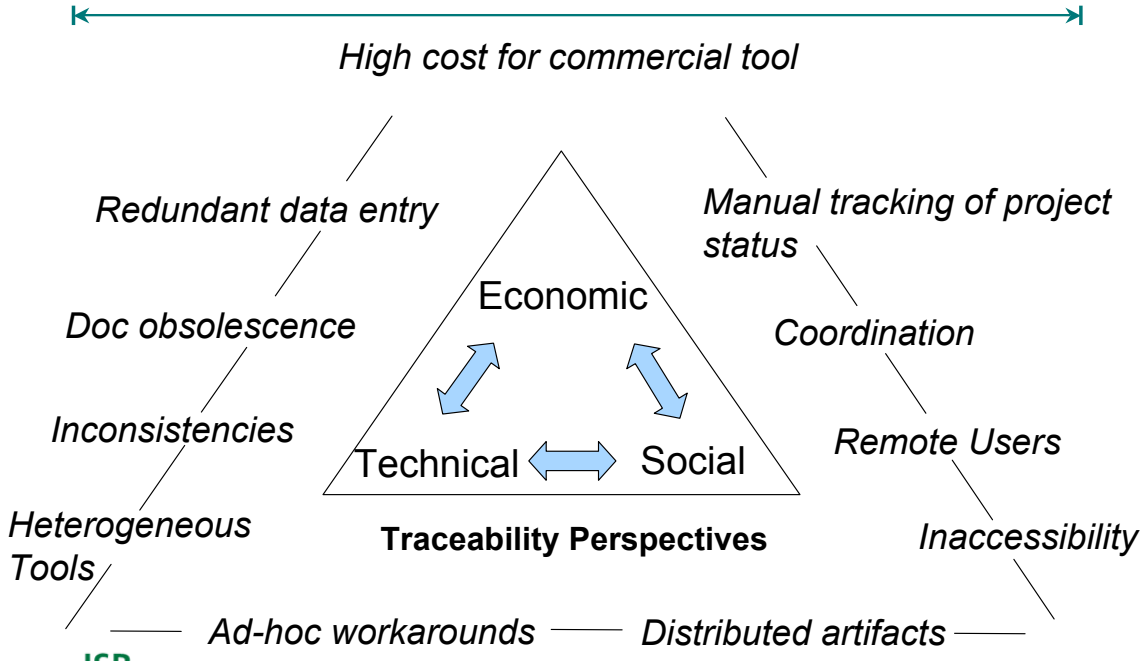
Definitions

- Traceability
  - A mapping between two points to signify a relationship
  - “The degree to which a relationship can be established between two or more products of the development process…” [IEEE]
- Requirements Traceability
  - Artifacts ←→ requirements, Relationship: satisfaction
  - “the ability to describe and follow the life of a requirement, in both forwards and backwards direction” [Gotel & Finkelstein 1994]
- Process Traceability
  - Actual processes ←→ company procedures, Relationship: conformance

Motivation

- Wonderware needs both types of traceability
  - To comply with government regulations
  - To satisfy customer audits
  - To aid in project management
- Need End-to-End Traceability
- But…
  - Traceability is a hard problem
  - Traceability is expensive
  - Many approaches, but rarely adopted in practice [Gotel & Finkelstein 1994][Spanoudakis & Zisman 2005]

Traceability Problem @ WW



Roadmap

- Key Design Decisions (Guidelines)
- End-to-End Software Traceability Model
- Traceability Tool Design & Implementation
- Tool Evaluation & Preliminary Results
- What Not To Do
- Related Work

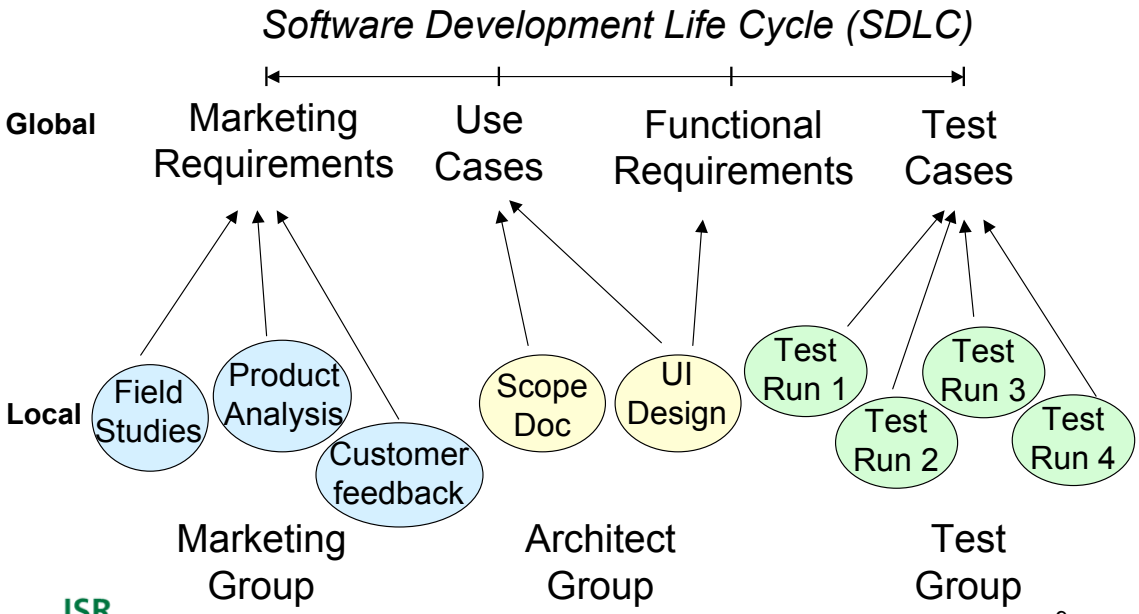
Key Design Decisions (Guidelines)

- Minimize cost
- Bound the problem space
- Enter information once
- Automate only when necessary
- Support existing work practices

Guideline: Minimize Cost

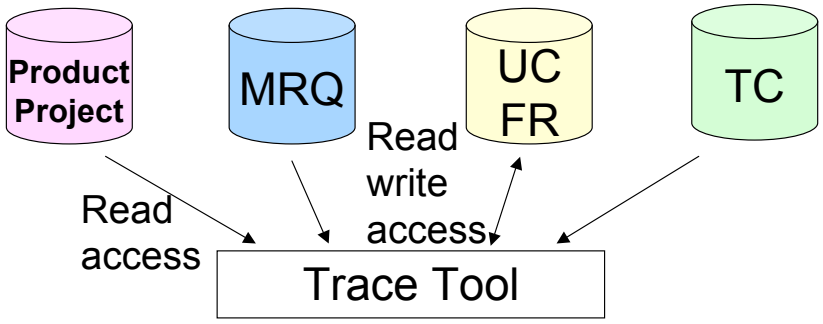
- Minimize labor hours in training (Usability)
- Minimize labor hours in tracing
  - "Just enough traceability"
    - Each trace link should provide a benefit
    - Trace information should aids users in SDLC tasks
- Minimize tool development & maintenance (Use existing tools)

Guideline:
Bound the Problem Space



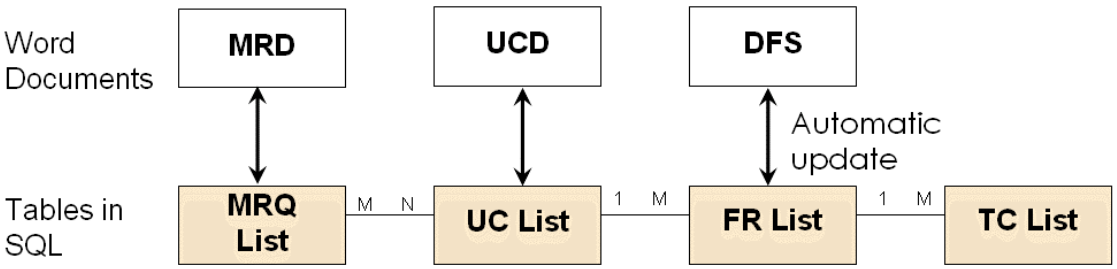
Guideline:
Enter Information Once

- "Distributed Centralization"
  - Maintain group ownership of artifacts
- Shared repository + custom code



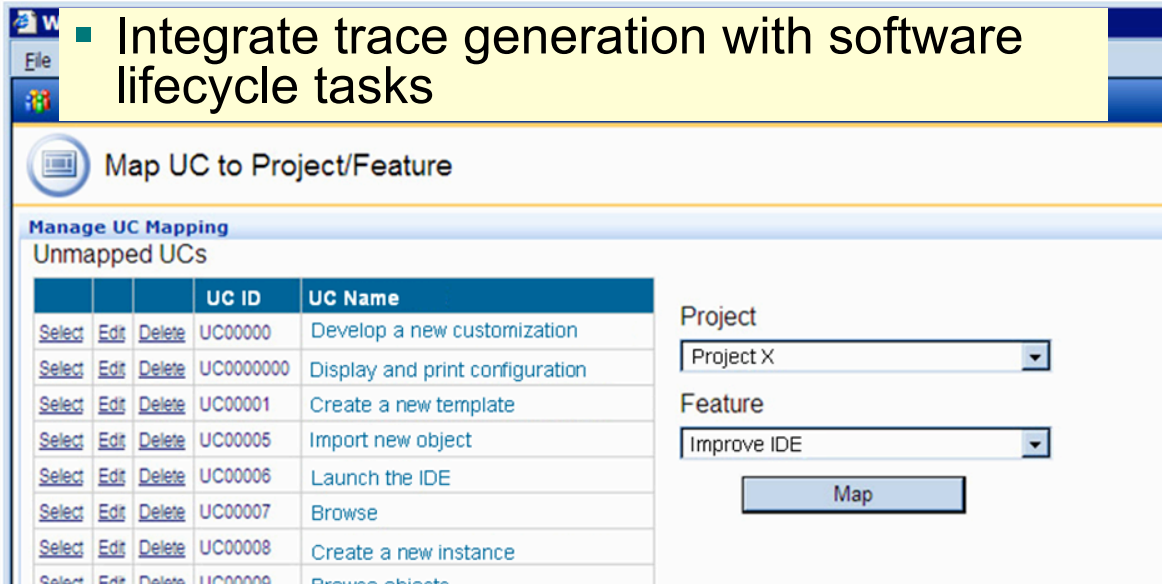
Guideline: Automate Only When Necessary

- Limitations of automation
- Automate the following
  - Migration of artifacts
  - Auto-generation of reports
  - Artifact search across projects
  - Bidirectional updates



Guideline: Automate Only When Necessary

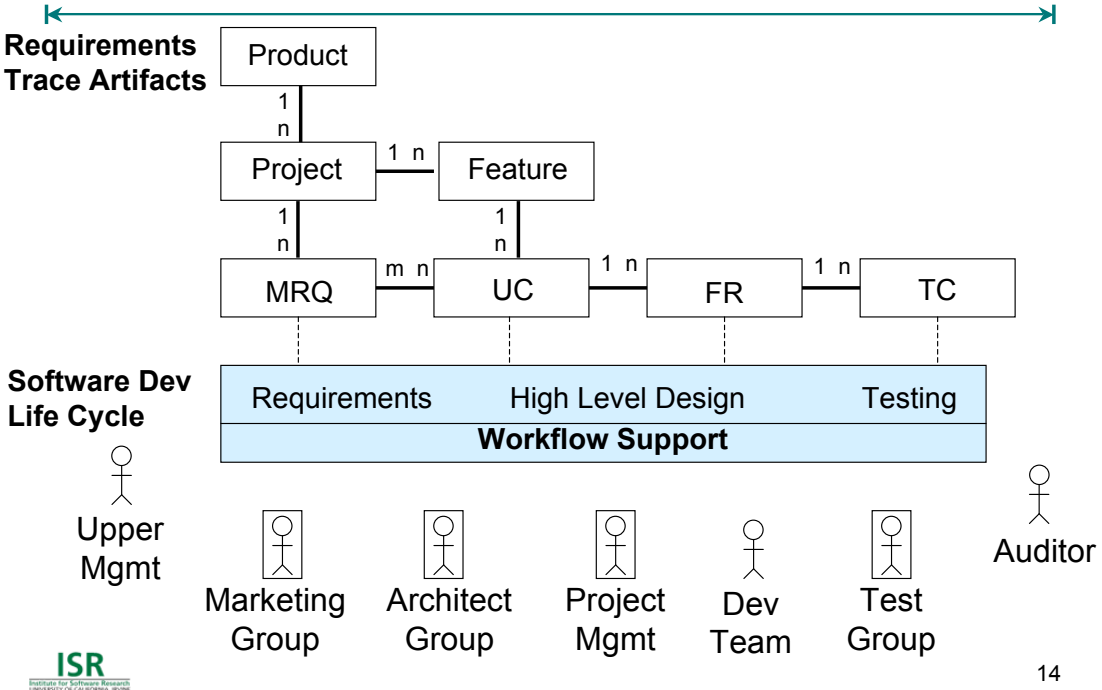
- Trace link generation – not automated
- Integrate trace generation with software lifecycle tasks



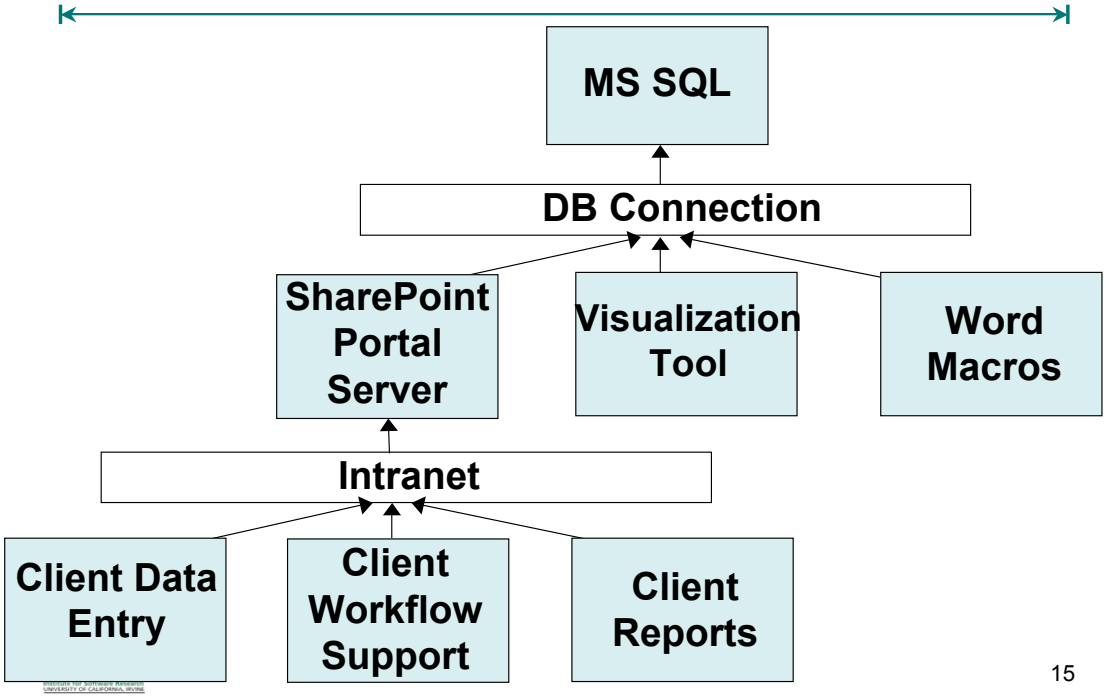
Guideline: Support Existing Work Practices

- Identify key users
  - Producers vs Consumers of trace information
- Provide custom user task list
- Streamline work process
- Respect group ownership structure

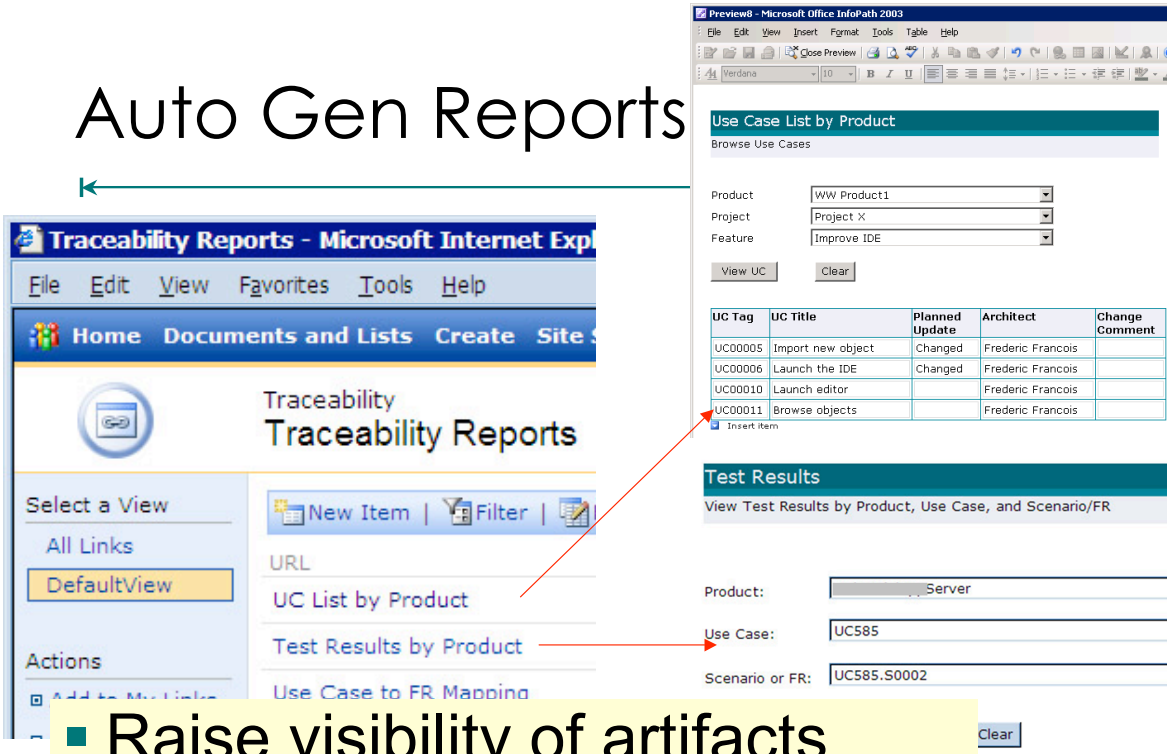
End-To-End Traceability Model



Traceability Tool Design

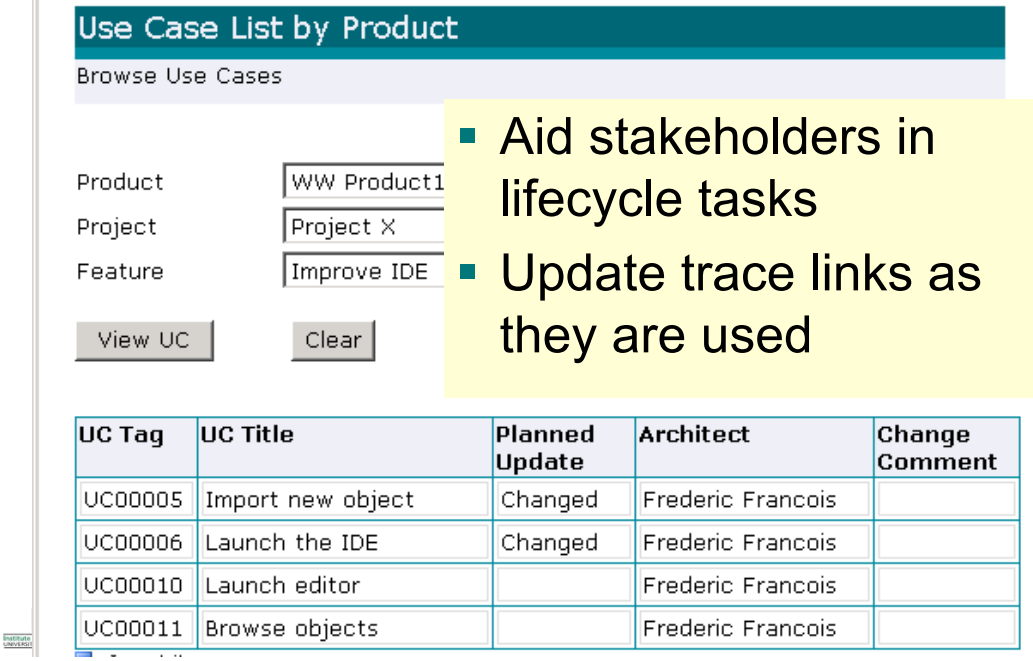


Auto Gen Reports



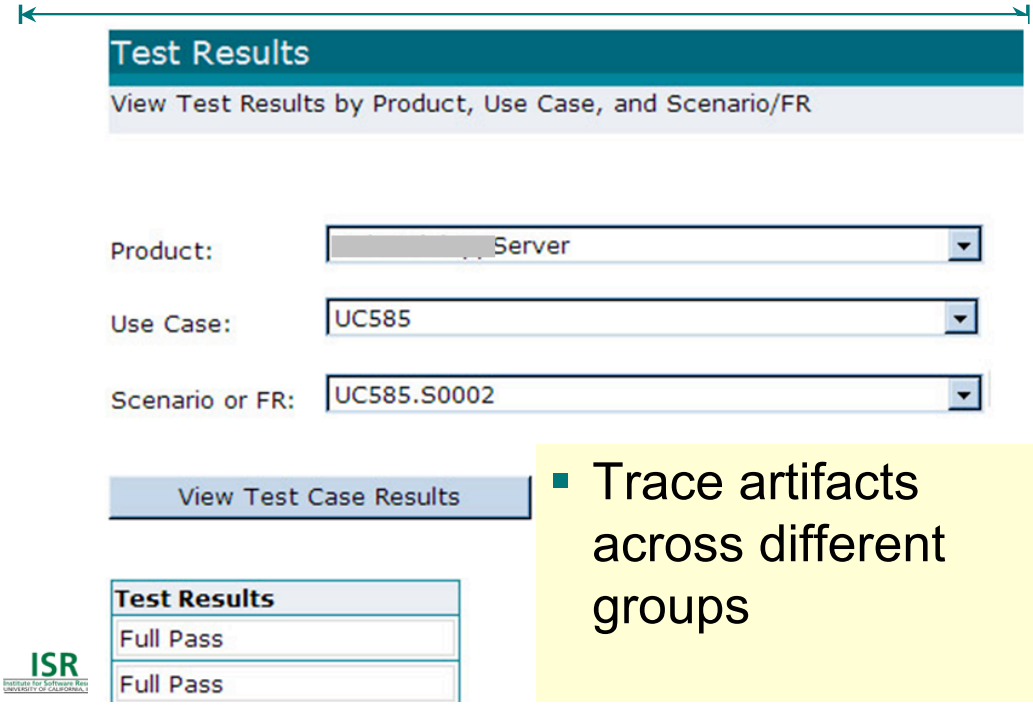
- Raise visibility of artifacts
- Verify trace links

Auto Gen Reports



- Aid stakeholders in lifecycle tasks
- Update trace links as they are used

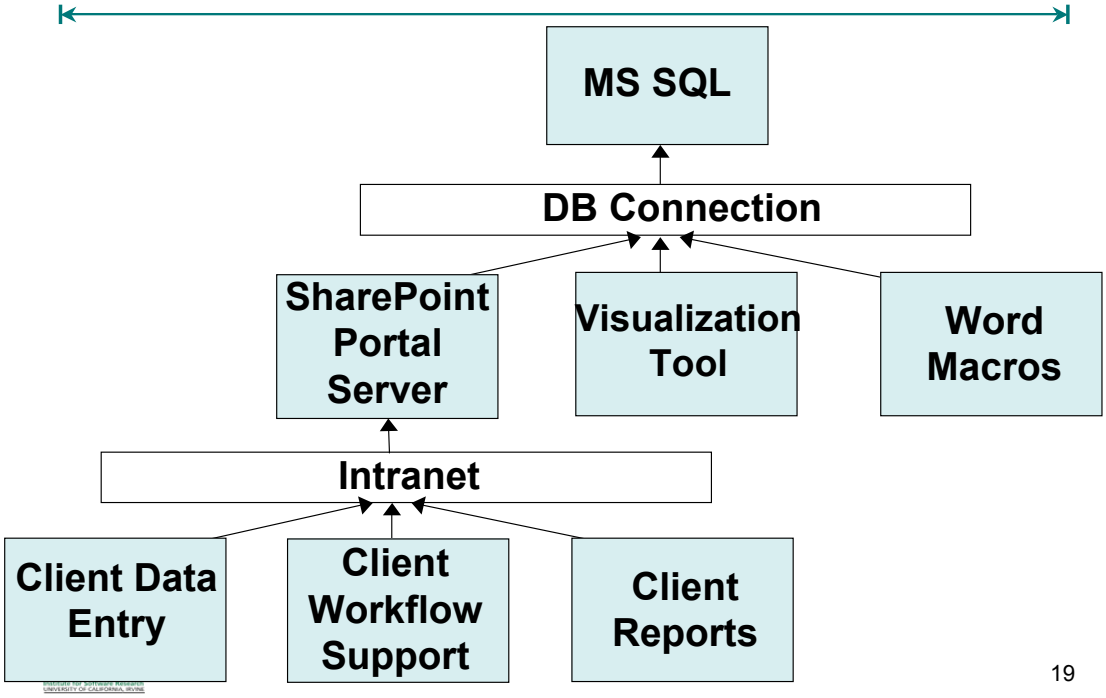
Auto Gen Reports



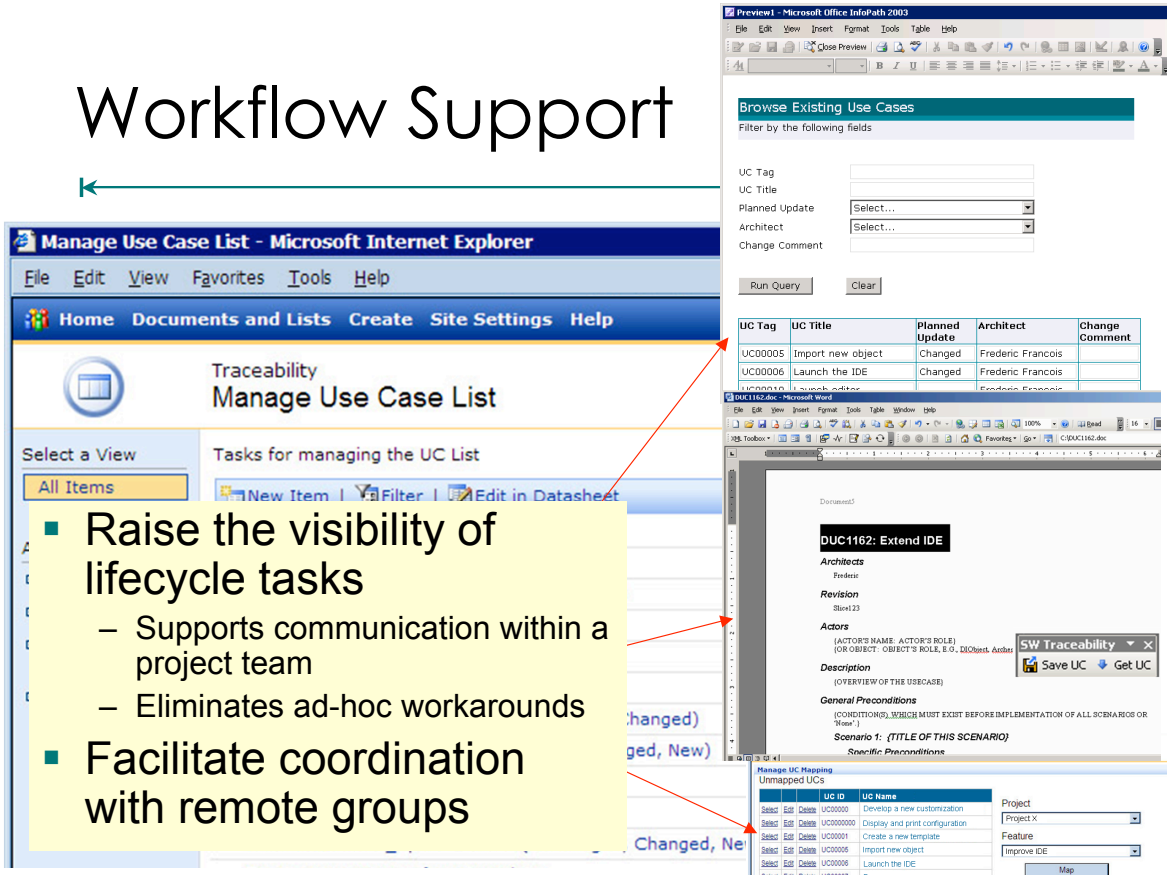
- Trace artifacts across different groups

ISR

Traceability Tool Design



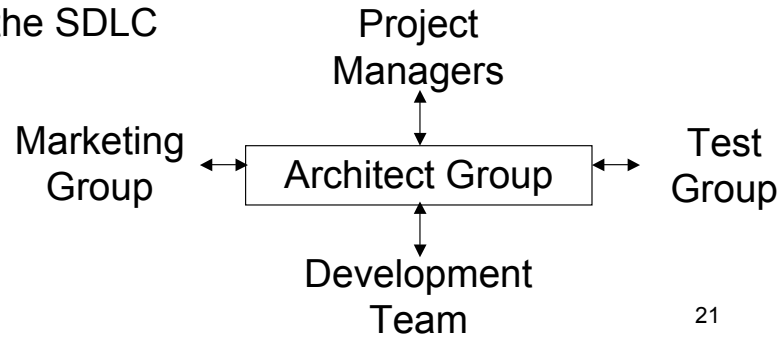
Workflow Support



- Raise the visibility of lifecycle tasks
  - Supports communication within a project team
  - Eliminates ad-hoc workarounds
- Facilitate coordination with remote groups

Tool Evaluation

- Test subjects: Architect Group
- Repository populated with live data
- Tested the following functionality:
  - Mapping between trace artifacts (Projects, Features, Use Cases, Functional Requirements)
  - Maintaining document integrity
  - Supporting the SDLC



Preliminary Results

- Results
  - Time spent in traceability tasks cut in half
  - Decrease in required support staff to maintain traceability tool
  - Low cost of deployment
  - Running successfully for over a year
- Feedback from Architect Group
  - Easy to use
  - Minimal training required
  - Aids architects in high level design tasks
- All active projects migrated to trace tool

What Not to Do

- ✗ Focus solely on automating trace generation
- ✗ Use an expensive commercial trace tool to solve an organization's traceability problem
- ✗ Delegate all traceability tasks to a third party entity to minimize cost
- ✗ Generate traces long after artifacts have been created
- ✗ Invest much time in training users on how to use a trace tool
- ✗ Require users to abandon their existing toolset and current work practices to use a trace tool

Related Work

- Closest match: Case Study on a US DoD project [Ramesh 1995]
  - Comprehensive view of traceability
  - Documents difficulties that reflect the three key traceability perspectives
  - Does not report on the success of trace approach
- Smaller scale:
  - Limited success in tracing between requirements and use cases [Alexander 2002]
  - Success in implementing traceability within one group [Arkley 2006][Neumuller 2006]

Conclusion

- Traceability is a hard problem
- Our solution: End-to-end traceability via a lightweight approach
- Traceability Perspectives
  - Economic, Technical, Social
- Key Design Decisions (Guidelines)
  - Minimize Cost
  - Bound the problem space
  - Enter information once
  - Automate only when necessary
  - Support existing work practices

Citations

Alexander, I. Towards Automatic Traceability in Industrial Practice. In *Proc. of the 1st Int'l Workshop on Traceability,* Edinburgh, Scotland. p. 26-31, 2002.

Arkley, P. and Riddle, S. Tailoring Traceability Information to Business Needs. In *Proc. of the 14th IEEE Int'l Conf on Reqts Engr.* Minneapolis, St. Paul, MN, Sep 11-15, 2006.

Gotel O. and Finkelstein A., An Analysis of the Requirements Traceability Problem. In *Proc. of the 1st Conf on Reqts Engr*. Los Alamitos, CA, 1994.

IEEE Standard Computer Dictionary, IEEE, New York, NY, Jan 1991.

Neumuller, C. and Grunbacher, P. Automating Software Traceability in Very Small Companies - a Case Study and Lessons Learned. In *Proc. of the 21st IEEE Int'l Conf on Automated Soft Engr.* Tokyo, Japan, Sep 18-22, 2006.

Ramesh, B., Powers, T., et al. Implementing Requirements Traceability: A Case Study. In *Proc. of the 1995 Intl. Symp on Reqts Engr.* York, UK, Mar 27-29 1995.

Spanoudakis, G. and Zisman, A. Software Traceability: A Roadmap Advances in Software Engineering and Knowdledge Engineering. Chang, S.K. ed. 3, World Scientific Publishing, 2005.

Thank you