Reengineering - Experiences and Some Lessons Learned

Dr. Peter Brössler (SDS GmbH, Vienna)
broessler@sds.at

WCRE 2001
11.10.2001, Stuttgart
Own Background

- SDS: CEO 2001 -
- sd&m: CTO, ... 1994 - 2001
- Computer Science 1981 - 1994
Company Profile

Core competency: Securities & Derivatives

Location: Vienna
Team: roughly 400 people; SW and banking specialists

Software development focus:
STP real-time software products for private & retail banking services
Front-, middle- & back office functionality ECN connectivity

Vision: Expand from Austria, Germany, Switzerland via partnerships with leading global players and international consulting firms to pan-European platform

Acknowledgements:
GEOS: Highlights

- Straight-through processing (STP Award, S.W.I.F.T Gold Label)
- Proven successful operation (4 IT centres with 350 banks in Austria)
- Scalable and platform-independent (NT, OS/2, Unix, MVS)
- Streamlined, multi-level-client design
- Transaction banking & insourcing (of other client institutions) capability
- Multiple legal entities, currencies, languages, ...
- 7 x 24 operation
- high performance

1.100.000 queries / h
100.000 settlements / h
130.000 orders / h
GEOS Development

1. Product Generation
   - Single entity, single country, single platform
   - IT Operation for Schoeller Bank

2. Product Generation
   - Multi-entity, single country, single platform
   - EOS live 11 Austr. Banks & IT-centers

3. Product Generation ongoing renewal
   - Country versions
   - Multi-country
   - Multi-currency
   - Multi-language
   - Multi-entity
   - Multi-platform

Long-term focus & perspective

GEOS Metrics

LOCs measured by Harry Sneed
Theses

1) Software reengineering is necessary
2) Software reengineering projects are not popular and do not “sell” well
3) Long-life software is accompanied by reengineering through its entire life cycle
4) Software reengineering is often carried out in an unprofessional way: technical, organisational, psychological
5) The gap between research and business practice is too wide
Software Reengineering

- Software reengineering =
  + Analysis of existing (legacy) system: conceptual and technical
  + Changes in software design
  + Re-implementation
  + Forward engineering: conceptual and technical

- is often associated with “poor quality” software (maintenance is not efficient)

- is usually akin to archaeology than a methodical, engineering-like process
“Reengineering is like looking at a Picasso and trying to come up with a photograph of the subject.”

(Vaughan Merlyn)
Software Reengineering is necessary

• What are the alternatives for the large number of existing software systems meant to be?
  – Abandon old systems when design is not up-to-date or maintenance becomes too difficult?
  – Stick to old systems and constantly loose quality while treating symptoms and increase costs?
  – Design utterly perfect software (“design for the future“)?

NO!
Software Reengineering takes place

• Almost every large and complex software system that has been in development for a long time is renovated from time to time

• Design mistakes are not necessarily the reason for this kind of reengineering. It is a continuous process of improvement and adaptation

• Of course there is a difference in quality in the sense of how often and for which reasons renovation becomes necessary
Software Reengineering is not popular

• Management does not want to invest money because
  – ROI is difficult to calculate
  – There are no guarantees that the life cycle of a software application can be extended sufficiently through reengineering
  – Reengineering has an air of trying to be perfect (“the application is running, what more do you want?”)
  – Budgets are not set aside in advance or made available for reengineering
  – Reengineering confronts the team with mistakes and omissions of the past
  – Crucial staff with know-how in the legacy system become worried (“will we still be needed after the reengineering?”)
Software Reengineering does not sell well

• Management (customer side) does not want to invest in reengineering
• Developing new systems wins far more prestige than reengineering old systems
• Developers prefer to work on new developments ("it’s more fun …")
• Risks are difficult to assess
• Few software houses are well positioned in this market and bring know-how, tools and responsibility
Software reengineering takes place anyway

- Software systems with a life cycle of 10, 20 or more years *always* encounter:
  - Multiple technological enhancements
  - Multitude of modifications and replacements in co-existing systems
  - Continuous flow of new conceptual requirements
  - New user groups
  - New development teams (loss of know-how!)
  - New development paradigms
  - New business organisations!

- All of this is not possible without reengineering; Software reengineering does take place!!!
Major obstacles associated with reengineering projects

- Lack of know-how and awareness for reengineering
- Reengineering is often understood as “emergency surgery” instead of a continuous process in long-term software development
- Reengineering needs top specialists for redesign … but where do you find them?
- A pure technical analysis is not sufficient. For design recovery a number of top specialists in the old system are necessary … but where do you find them?
- Reengineering technology leaves a lot to be desired and existing tools are not very widespread yet
Implementation of standard software

• The implementation of large standard software packages often implies complete or partial reengineering of existing neighbour systems systems
  – Poorly or undocumented interfaces have to be used
  – All processes have to be analysed and - if necessary- to be reengineered
  – existing systems have to be modified that nobody wants to touch any more
  – Even if old systems are replaced by standard software they have to be analysed and documented beforehand

• Awareness of these aspects rarely exists!!
Organisational aspects of Software Reengineering

• An isolated reengineering team alongside the development team (project) often fails
• Reengineering activities have to be included in project planning and therefore belong to the overall project management
• Specialists in the old system have to be closely involved in reengineering activities
• Reengineering needs good project management, good design and good developers … just like a new development!
Product Evolution and Reengineering

- Long-term product evolution means multiple (or even continuous) software reengineering
- Example: GEOS
- Additional complexity: Release compliancy, release management and reengineering at the same time
Tools

- Good reengineering technology is badly needed but rare
- Good reengineering tools are often also good development tools
- Example: Repositories, e.g. SHORE by sd&m
Tools: Analysis and Documentation

- Central storage of all relevant documents for SW development or reengineering projects

![Diagram showing Central Repository with connections to Specification, Sources, JCL, Database tables, Documentation, User interviews, and Results of reengineering activities.]
Tools: SHORE

SHORE® stands for
s d & m
H ypertext
O bjekt
Re pository

® registered Trademark of sd&m AG
SHORE: Objectives

• Document management
  – Easy access to project documents
  – No restrictions for developers
• Navigation between “objects”
• Comprehensive query possibilities
• Flexibility and adaptability through meta-model and parsers (e.g. for Cobol and Java, also UML)
• Management and evaluation of a large amount of documents (> 10,000) in large projects
SHORE at work

import de.sdm.shoreparser.utilities.*;
import java.io.*;

class Java2Shore {
    BlockCounter bc;
    de.sdm.shoreparser.utilities.SymbolTable st;
    ImportTable it;
    XMLBridge bridge;
    Configuration config;
    JavaParser parser;

    Java2Shore(String args[]) {
        st = new SymbolTable();
        bc = new BlockCounter(0);
        bridge = new XMLBridge();
        init();
        parser = null;
    }

    Objekt JavaAttribut de.sdm.shoreparser.java2shore.Java2Shore.bridge

    - JavaMethode de.sdm.shoreparser.java2shore.Java2Shore.Java2Shore(java.lang.String[]) -- verwendet --> (Java2Shore.java)
    - JavaMethode de.sdm.shoreparser.java2shore.Java2Shore.parseFile(java.lang.String) -- verwendet --> (Java2Shore.java)
    - JavaKlasse de.sdm.shoreparser.java2shore.Java2Shore -- verwendet --> (Java2Shore.java)
    - gehoert_zu --> JavaKlasse de.sdm.shoreparser.java2shore.Java2Shore
    - hat --> JavaKlasse de.sdm.shoreparser.java2shore.Java2Shore

WCRI 2001 (3-10-2001): Dr. Peter Brössler (SDS): Reengineering: Experiences and some lessons learned
SHORE: Model levels

Meta-meta-model
(fixed)
document type, object type, relation type

Meta-model
(configurable)

Task
Function
Test case

implemented
tests
calls

defines types for instances for
defines types for instances for

Print addresses
printAddr(…)
printName(…)
Test 44

WCRE 2001 (3-10-2001): Dr. Peter Brössler (SDS): Reengineering: Experiences and some lessons learned
SHORE: Document links

Analysis

Design

Programming

- Document
- Object
- Relation
- Cross-reference (Hyperlink)
- Hotspot (Hyperlink)
SHORE: Functionality

Meta-model

Project documents

Parser

Parser (per document type)

XML documents

OODB

P(X) :-
A(X,Y),
B(Y,X).

SHORE server

Prolog queries

XSB

Documents and query results

Hypertext

SHORE

per Project configurable
SHORE: Further information

- In-house development of SHORE was necessary because there is no comparable product on the market
- Used in sd&m projects and by selected customers
- Further information:
  - OBJEKTspektrum march/april 2000
  - mailto:olaf@sdm.de (Olaf Deterding)
Research and business practice

• Some questions:
  – How many presentations address subjects covered in this presentations and actually offer solutions?
  – How many universities deal with software reengineering?
  – How many software reengineering cooperation projects exist between universities and industry?