

Reengineering

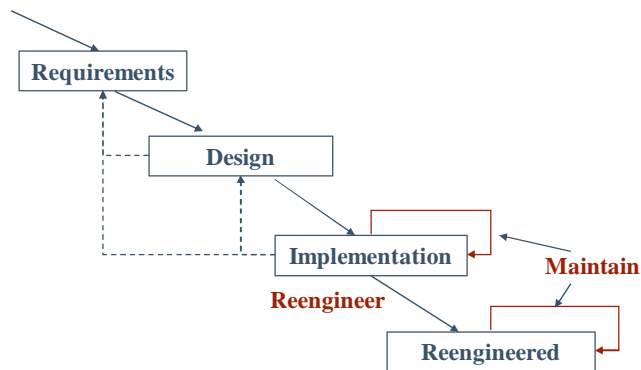
Liam O'Brien

Sponsored by the U.S. Department of Defense
© 2005 by Carnegie Mellon University

Reengineering - Outline

1. Introduction, Context and Strategies
2. Reengineering Techniques
3. Architecture-centric Reengineering
4. Why Reengineering Projects Fail
5. Introduction to OAR

Context: Software Life Cycle



© 2005 by Carnegie Mellon University

Version 1.0

page 3

Reengineering – Definitions - 1

Several definitions of reengineering:

Preparation or improvement to software, usually for increased maintainability, reusability or evolveability

R. S. Arnold, A Roadmap Guide to Software Reengineering Technology, Software Reengineering, 1994.

The examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form

E. Chikofsky and J. Cross, Reverse Engineering and Design Recovery: A Taxonomy, IEEE Software 7(1) Jan 1990: 13-17.

© 2005 by Carnegie Mellon University

Version 1.0

page 4

2

Title
Date

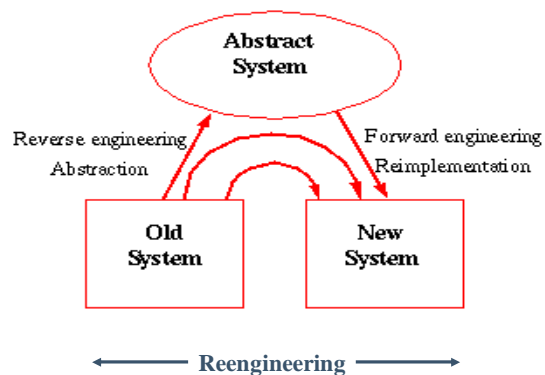
© 2002 by Carnegie Mellon University

Reengineering – Definitions - 2

Reengineering is the systematic transformation of an existing system into a new form to realize quality improvements in operation, system capability, functionality, performance, or evolveability at a lower cost, schedule, or risk to the customer

S. Tilley and D. Smith, *Perspectives on Legacy System Reengineering*, SEI White Paper, 1995

What is Reengineering?



Reengineering – Related Topics - 1

Restructuring

Restructuring is the transformation from one representation form to another at the same relative abstraction level, while preserving the subject system's external behavior (functionality and semantics)

Refactoring

Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure.

Refactoring: Improving the Design of Existing Code – Martin Fowler

Reengineering – Related Topics - 2

Business Process Reengineering (BPR):

- concerned with the conversion of business process and not of computer processes

Data Reengineering

- concerned with conversion of data format and not its content

Why Reengineer?

Why does an organization decide to reengineer one or more of their systems:

- software system is an integral part of the organization
- system must be regularly maintained and maintenance is becoming a large cost factor
- less risk and cost than redevelopment

Reengineering – Difficulties with Legacy Systems

Reasons why legacy systems have not been designed to accommodate change:

- Short life expectancy – not anticipated to last decades when first developed
- Failure of process models and software engineering culture to treat evolution as a first class activity – future requirements ignored
- Satisfying constraints that existed at the time of development – hardware (memory and processing power)

Reengineering Strategies

Reengineering Strategies for legacy applications:

- Ignore – discard them
- Cold turkey – rewrite them from scratch
- Integrate – consolidate them into the current and future applications by access in place
- Data Warehouse – build a “shadow” system to house the frequently accessed data
- Gradual Migration – rearchitect and transition gradually

Factors Influencing Strategy

The following factors will influence what strategy is to be chosen:

- Business value of the legacy systems – support current as well as future business processes
- Flexibility and growth requirements – if legacy system does not need extensive changes for flexibility and growth then minimal effort for integration may be appropriate
- Technical status of the legacy system – represents the quality of the system in terms of modularity, error rates, flexibility and utilization of current technologies.

Categories of Legacy Applications

1. Completely decomposable – all components are well structured and separable. Friendly to migration efforts
2. Data decomposable – data services well structured and well-defined interfaces. Semi-friendly to migration efforts
3. Program decomposable – program modules are well structured and have well-defined interfaces. Semi-friendly to migration efforts
4. Monolithic – unstructured with intermixed data, program logic and user interface processing. Hostile to migration efforts

Strategies – Ignore - 1

When can the existing system be discarded?:

- The business has changed and new methods of doing business have been developed, existing methods are being phased out
- It is too costly to try to reengineer the current systems
- Current systems are monolithic and not reusable
- New COTS products can be bought with the same or similar functionality for less cost than reengineering

Strategies – Ignore - 2

In most situations it is not practical to ignore legacy systems especially the data.

Some organizations are ignoring legacy systems by outsourcing them to software houses that specialize in operating and maintaining legacy applications.

Strategies – Cold Turkey - 1

When can the existing system be ignored?:

- The business has changed and new methods of doing business have been developed – need to build systems to implement these new methods
- It is too costly to try and understand what you currently have
- It is too costly to try to reengineer the current systems
- Time and resources are available to start from scratch

Strategies – Cold Turkey - 2

In practice this approach may not be viable because:

- Management may not allow such large investments in development just to have a more flexible system – must have a better system produced
- Business does not stand still while new systems are being developed
- Specifications for legacy systems rarely exist and undocumented dependencies frequently exist
- The rewritten system themselves become legacy before completion as development takes a long time

Strategy – Integrate - 1

When can the existing system be integrated?:

- A lot of the business knowledge and rules are built into the current systems and these need to be preserved – high business value
- Costs will be reduced by reengineering the current systems rather than developing from scratch
- Time and resources are not available to start from scratch

Strategy – Integrate - 2

In practice integration is carried out when:

- Access to needed data is urgent and cannot be postponed until completion of migration
- Needed data can be accessed by client applications by employing COTS technologies
- Requirements for queries to the data are low (usually not a good idea for large amounts of queries)

Achieved through the use of mediators (wrappers and gateways)

Strategy – Data Warehouse - 1

When can the data warehouse option be used?:

- The data is the most important part of the existing systems
- There is a need for new ways to access and manipulate the data (web-based, distributed)
- There isn't a need to reimplement existing system functionality and time and resources are not available to do so

Strategy – Data Warehouse - 2

Organization's data is in two formats:

- Operational data which is used for day-to-day transaction processing
- Analysis (decision support) data which are used for business analysis and report generation. This data is extracted from the operational data periodically and downloaded, usually to a separate system, for report generation and analysis.

Strategies - Gradual Migration - 1

When can the gradual migration option be used?:

- Integration is not economical over a long period of time or those legacy systems are to be phased out
- There is an immediate need to update some of the more important components
- Time and resources are not available to do the reengineering all at once
- It is unclear how the reengineering effort will proceed – need for prototyping of different reengineering methods

Strategies - Gradual Migration - 2

The legacy and target systems may coexist during the migration stage.

Challenge will be to design gateways to isolate the migration steps so that end users do not know if the data is coming from old or new systems.

The development of gateways to facilitate migration is usually an expensive undertaking.

Reengineering Issues - 1

- Ease of change
 - how difficult will it be to change the legacy system
- Comprehensibility
 - how difficult is it to obtain an understanding of the systems that will need to be reengineered
- Size
 - what is the size of the systems that have to be reengineering
- Decomposability
 - how easy or difficult is it to decompose the software system
- ...

Reengineering Issues - 2

- Degree of coupling
 - how coupled are the existing components
- Degree of cohesion
 - how cohesive are the components
- Granularity
 - can the system be reused in large chunks or does it need to be decomposed
- Complexity
 - what is the level of complexity of the software
- Degree of documentation
 - what documentation is available
- ...

Reengineering Issues - 3

- Degree of abstraction
 - what abstractions are available
- Language type & choice
 - what language(s) is the existing system written in and what will be used in the new system
- Referential transparency
 - do the components always produce the same output (not dependent on some global state)

Reengineering – Risks - 1

Some of the risk areas in reengineering :-

- Process
 - is there a defined process to follow to successfully reengineer the systems
- Personnel
 - are the necessary resources available
- Skills/Education
 - do the personnel have the right skills or can they be trained
- System/Application
 - difficulties with the legacy system

Reengineering – Risks - 2

Technology

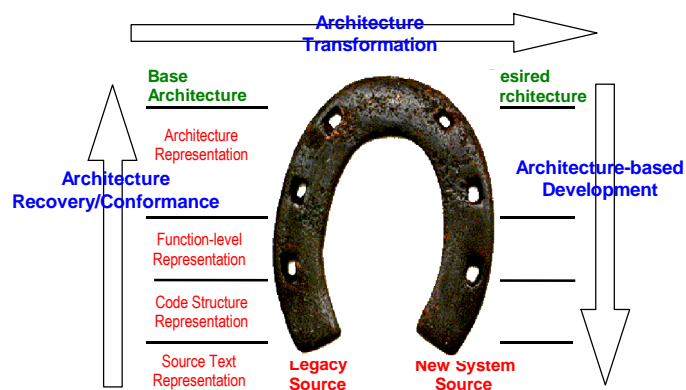
- what technology is used in the target system
- Tools
 - are there tools to assist in the reengineering work
- Strategy
 - has the right strategy been chosen

Business Case for Reengineering

The following are part of the business case:

- Cost-benefit analysis
- Legacy systems constitute massive corporate assets totaling billions of \$\$\$
- Today a large part of the business model of many organizations is implemented in software
- Business rules incorporated in many software systems change at regular intervals (e.g., yearly)
- Implementing a mature software system from scratch could take as long as its entire evolution history
- The average Fortune 100 company maintains 35 million lines of code and adds 10% each year

Architecture-centric Reengineering



Code Level

The source text representation level represents the source text of the programming language or languages.

Code level transformations:

- Language translation
- Representation changes (some fixes for Y2K problems)

Program Structure Level

The code structure representation level represents program artifacts (usually represented as ASTs) constructed from the source code by the application of language semantics (use of grammar and parsing).

The AST can be annotated with control and data flow information.

File/Function/Data Level

The function-level representation represents information constructed from the program structure (ASTs, Control Flow Graphs, Data Flow Graphs) by the application of structural semantics.

Typical information represented include:

- call graphs
- dependency graphs

Architecture Level

The architecture representation level represents information constructed from the File/Function/Data level by application of architectural semantics.

The architectural semantics defines how the system is constructed from the elements in the File/Function/Data level

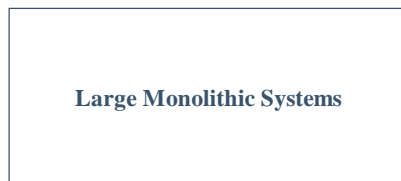
Typical information represented include:

- Components
- Component dependencies

Architecture Transformations - 1

A lot of older IT systems are large monolithic systems which run on mainframe environments.

- Batch type transactions.
- Centralized Data processing.
- Business Logic, UI and Database all run on the mainframe



Architecture Transformations - 2

More recent structure for system is Client/Server.

- Client and Server usually run on different machines (may be many Clients and possibly many Servers)
- Client provides the UI and request services from the Server and may do some processing on results
- Server provides data and possibly some processing capabilities
- Business logic shared between both



Architecture Transformations - 3

New approaches include 3 tier or n-tier systems

- Client layer provides the UI (replicated)
- Data Server provides access to data
- Application Layer provides the business logic. Gets requests from Clients access Data server and provides results. (may be replicated)



Reengineering Projects – Why they fail?

Reengineering projects fail for many reasons.

Most reasons are not specific to reengineering projects but also occur in new development projects.

Usually not just one reason why a project fails but maybe a combination of reasons.

Bergey, John; Smith, Dennis; Tilley, Scott; Weideman, Nelson; Woods, Steven. *Why Reengineering Projects Fail* (CMU/SEI-99-TR-010).

Reengineering Projects Fail - 1

Reason #1:

The organization inadvertently adopts a flawed or incomplete reengineering strategy

This may be caused by poor assumptions or lack of attention to detail. Maybe the wrong problem is being addressed. Possibly not all of the components and steps are considered. If a system is ignored or discarded a lot of corporate knowledge may be lost/abandoned. Maybe a “Big Bang” approach is taken and deployment and transition are ignored.

Reengineering Projects Fail - 2

Reason #2:

The organization makes inappropriate use of outside consultants and outside contractors

Outsiders may bring domain understanding, technical skills, and extra resources. However they may not know the business as well as the insiders. Their role needs to be clearly defined and monitored. Outsiders may have conflicting interests (maximizing cost rather than minimizing). Outsiders take control of the work rather than it being controlled by insiders (results in lack of insight by insiders)

Reengineering Projects Fail - 3

Reason #3:

The work force is tied to old technologies with inadequate training programs

The lack of training can cause project failure. New programming paradigms will be adopted. No possible to continue to do business as usual while at the same time bringing the same workforce up to speed on new technologies. Must be a conscientious and persistent effort to upgrade skills of the existing workforce or there must be a replacement of existing workforce or replacement or some combination.

Reengineering Projects Fail - 4

Reason #4:

The organization does not have its legacy system under control

Before a system can be managed effectively, a system baseline under configuration management should be in place to aid in disciplined evolution. Legacy system needs to be well documented, with an understanding of the priority of change requests and their impact on the system.

Reengineering Projects Fail - 5

Reason #5:
There is too little elicitation and validation of requirements for the reengineering effort

There may be flaws in the requirements elicitation and validation processes. Requirements include functional, non-functional, user and customer requirements.

Reengineering Projects Fail - 6

Reason #6:
Software architecture is not a primary reengineering consideration

Failure can occur when a methodical evaluation of the software architecture of the legacy and target systems is not the driving factor in the development of the reengineering technical approach. Evaluation of the legacy architecture may decide what strategy is undertaken.

Reengineering Projects Fail - 7

Reason #7:

There is no notion of a separate and distinct reengineering process

The means by which a legacy system evolves can have a large influence on the success of failure. The existence of a documented life-cycle process and corresponding work products are often wrongly viewed as being evidence of a sound reengineering process. There needs to be a set of tasks and guidance to perform them and an understanding of how it all fits together.

Reengineering Projects Fail - 8

Reason #8:

There is inadequate planning or inadequate resolve to follow the plans

Projects often get out of kilter by focusing on the low-level "software problems" and neglecting the intermediate-level tactical management planning and systems engineering planning aspects of the job.

Sometimes there may be an absence of a documented project plan that has the buy-in of the key stakeholders (line management, project team, domain and system experts and software engineers).

Reengineering Projects Fail - 9

Reason #9:
Management lacks long-term commitment

Management support means careful monitoring and putting things back on the track when they stray off track. If management gets distracted with other projects during the course of a major reengineering effort, it will not know when things go wrong.

Reengineering Projects Fail - 10

Reason #10:
Management predetermines technical decisions

Mandates or edicts issued by the upper management that predetermine the technical approach or schedule, cost, and performance considerations without sufficient project team input or concurrence are frequently seen to cause reengineering project failure.

Options Analysis for Reengineering (OAR)

OAR is a method developed at the SEI to help organizations develop better estimates of the cost and effort of reengineering and reusing legacy assets in the development of new systems.

OAR focuses on making initial decisions on rehabilitating specific candidate code components to serve as assets for a specific new system or product line.

Bergey, J., O'Brien, L. and Smith, D., *Options Analysis for Reengineering (OAR): A Method for Mining Legacy Assets* (CMU/SEI-2001-TN-013), Software Engineering Institute, 2001.

<http://www.sei.cmu.edu/publications/documents/01.reports/01tn013.html>

Why Mine Existing Components?

Few systems or product lines start from “green fields”.

Many organizations have large investments in their existing systems and they want to make use of that investment.

Problem of mining existing components:

- under what conditions should components be extracted
- what types of components are worth extracting
- issues in extraction:
 - existing components are often poorly structured and poorly documented
 - existing components differ in levels of granularity
 - organizations lack clear guidance on how to perform the salvaging

Bottom line: it is difficult to get off the dime!

Addressing Reuse Issues

For meaningful mining it is necessary to:

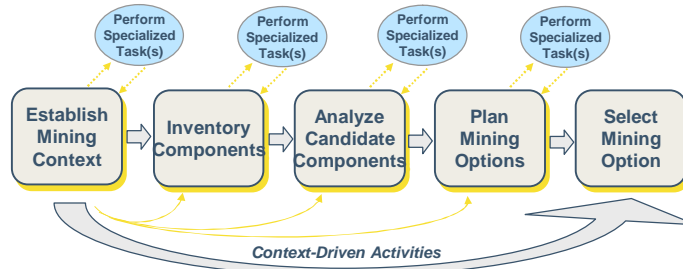
- Identify relevant and non-relevant legacy assets
- Make decisions based on “hands-on” analysis
- Identify target needs that can be satisfied and those that cannot be satisfied
- Estimate the cost, risk and confidence of estimates of changes required to each legacy component

What OAR Provides

OAR provides:

- quick identification of relevant and non-relevant components
- decision making based on consensus and “hands-on” analysis
- identification of target system component needs that can be satisfied and those that cannot be satisfied
- estimates of the cost and risk of changes required to each legacy component to satisfy a target system need
- a scalable architecture-centric approach an organization can use for follow-on mining efforts of greater scope

Options Analysis for Reengineering (OAR) Method



© 2005 by Carnegie Mellon University

Version 1.0

page 53

for Software Components

OAR Asset Table – Composite

Target System Asset Needs		Corresponding Legacy Assets and Basic Characteristics				SEGMENT • Header • LSI • Mining Team • Unique • Common			
Asset Type	Specific Asset Need	Legacy Asset	Language	Age	Size				
Identify	LSI's Asset Screening Characteristics								
	Stability & Volatility	Number of Modules	Structured Code	Level of Documentation	Coupling	Cohesion			
Identify	<Qualifier>	Mining Team's Asset Screening Characteristics							
		Number of Interfaces	Number of Modifications	Programming Standards Compliance	Mining Team Familiarity	Compilation Environment			
Identify	Asset Rehabilitation Characteristics			Asset Rehabilitation Estimates					
	Black Box White Box Suitability	Scale of Changes Required	Level of Granularity	Other Software Dependencies	Level of Difficulty	Level of Risk	Mining Effort Required Mining Cost Estimate		
Identify	<AS-is>	External Development Estimates				Mining Comparison Relative to Development			
		Effort	Cost	Level of Difficulty	Level of Risk	Relative Effort	Relative Cost	Relative Difficulty Relative Risk	
		<MM>	<\$K>	<Qualifier>	<Qualifier>	<MM>	<\$K>	<Qualifier>	<%>

© 2005 by Carnegie Mellon University

Version 1.0

page 54

OAR Options Table

Legacy System Software										
Option No.	Legacy System Software Components	Support Software Required	Level of Risk	Level of Difficulty	Mining Effort ¹ (mm)	Mining Cost ¹	New Development Effort (mm)	New Development Cost	Comparative Cost of Mining	Comparative Effort of Mining
1	Event Track	S&D Files	Low	4	8	\$80,000	18.1	\$180,567	44%	44%
	Option Summation			Low	4	8	\$80,000	18.1	\$180,567	44%
2	Objects	S&D Files	High	2	2.2	\$22,000	30.7	\$306,733	7%	7%
	Event	S&D Files	High	2	1.3	\$13,000	14.7	\$147,067	9%	9%
	Option Summation			High	2	3.5	\$35,000	45.4	\$453,800	8%
3	Sim1 Gateway	S&D Files	Low	1	0.1	\$1,000	7.7	\$76,567	1%	1%
	Sim2 Gateway	S&D Files	Low	1	0.1	\$1,000	16.3	\$162,567	1%	1%
	C-Source	S&D Files	Low	1	0.1	\$1,000	14	\$139,533	1%	1%
	Option Summation			Low	1	0.3	\$3,000	38	\$378,667	1%

¹ Note: Mining Effort and Cost do not include effort and cost to convert scripts and data files that are part of the support software

OAR Summary

OAR is a systematic, architecture-centric means for mining existing components for a product line or new software architecture

OAR can be used to analyze the reuse potential of components from an organization's legacy systems.

OAR provides:

- management visibility into this highly complex analysis activity
- insights into implicit stakeholder assumptions, constraints, and other major drivers that impact the mining of components

LSI OAR can be used to provide an objective analysis of supplier reuse estimates.

Baseline OAR capability can be subsumed by LSI OAR model since it is equivalent to having one supplier and the LSI and supplier organizations being one and the same.

Value-Added of OAR

OAR provides:

- With LSI OAR, a sanity check for credibility of reuse estimates
- An inventory of existing legacy components and related documentation
- Identification of components that can fill identified product line needs, their characteristics, types of changes needed, and estimated cost and effort (Component Table)
- A viable mining approach (Options Table) that reflects the elicited needs, priorities and concerns of the organization
- Identification of product line (or new system) component needs that can and cannot be satisfied through mining

Summary

In this lecture we talked about Reengineering

Outlined some approaches, issues and risks.

Outlined reasons why reengineering projects fail

Introduction to Options Analysis for Reengineering (OAR)