



RENAISSANCE

*Method and tool support for the
evolution and re-engineering of legacy systems*

RENAISSANCE METHOD

©RENAISSANCE Consortium 1997

Version 1.0, First published April 1997

The RENAISSANCE project is partially funded by the European Commission under the Framework Initiative (ESPRIT 22010). The objective of the project is to develop a systematic method to support the re-engineering of legacy systems. Further information about the project is available on the World-Wide-Web at URL:

<http://www.comp.lancs.ac.uk/projects/renaissance/>

The members of the RENAISSANCE Consortium are:

CAP Gemini Innovation (Mr Alain Dineur)

Bâtiment Karélian
7, chemin de la Dhuy
38340 Meylan, FRANCE
Tel: +33 476 76 47 47; Fax: +33 476 76 47 48

CAP Gemini IST (Mr Alain Paoli)

Tour Anjou
33 Quai de Dion Bouton
92814 PUTEAUX Cedex, FRANCE
Tel: +33 1 41 26 63 36; Fax: +33 1 41 26 52 17

debis Systemhaus GEI GmbH (Mr Markus Breuer)

Pascalstraße 14
D-52076 AACHEN, Germany
Phone: +49 2408 943 0; Fax: +49 2408 943 119

INTECS Sistemi S.p. A. (Mr Giancarlo Savoia)

Via Livia Gereschi, 32
56127 PISA, Italy
Tel: +39 50 545 111; Fax: +39 50 545 200

Telesoft S.p. A. (Mr Fabio Mungo)

Via Degli Agrostemi, 30
00040 SANTA PALOMBA (Roma), Italy
Tel: +39 6 710 551; Fax: +39 6 710 553 50

Engineering - Ingegneria Informatica S.p. A. (Mr Dario Avallone)

Via dei Mille, 56
I-00185 ROMA, Italy
Tel: +39 6 522 431; Fax: +39 6 522 432 48

Lancaster University (Prof. Ian Sommerville)

Computing Dept,
Bailrigg, LANCASTER LA1 4YR, UK
Tel: +44 1524 593795; Fax: +44 1524 593608

SINTEF (Prof. Reidar Conradi)

O. S. Bragstads plass 2 F
N-7034 TRONDHEM, Norway
Tel: +47 73 593 444; Fax: +47 73 594 466

Executive Summary

The RENAISSANCE method aims at supporting the reengineering of legacy software systems, that is the transformation of valuable software assets which are difficult to maintain toward new systems which are able to evolve both in the short and long term. This report defines such a method.

The report first defines the objectives of RENAISSANCE, and the constraints and benefits of applying this technology. Then it introduces a simple notation for representing the method itself, gives an overview of the method and hints for introducing the technology within an organization and adopting it; and then details both the activities and the control flow among activities in order to provide the user with operative procedures for reengineering systems.

There are two basic ideas behind the RENAISSANCE method:

- reengineering must be *company and project specific*;
- both the reengineering process and the reengineered system must be *continuously refined*.

These are implemented as follows. RENAISSANCE identifies generic activities, which can be specialised and decomposed to address specific needs of the user of the method, coming both from company and project constraints and practices. Activities are identified through their interface, which is defined in terms of Pre, Enabling and Post Conditions. Concerning the implementation of an activity, RENAISSANCE simply describes the actions to be performed, without forcing any implementation approach. Concerning the execution, both the overall control flow and the activation of a single activity can be customized. The cooperation of the activities and the incremental approach to refine the system under reengineering are achieved through a repository containing all the artefacts needed by the process, that is both produced and consumed by activities.

The method is structured on two main phases. The first one, or *what to do*, assesses the organization and the legacy system and identifies both the need and urgency of reengineering and the best candidate strategy to adopt for renewing the system: continue with the current maintenance approach; re-engineer the system from user-interface, structure, architecture, and design point of views; replace the existing system with a new developed one. The second phase, or *how to do*, supports the implementation of the planned transformation.

Since the RENAISSANCE method is an instance of the RENAISSANCE framework, as defined in the corresponding report, the last chapter of this report defines the traceability from that framework to this method. Reading this chapter is optional, and readers who are not interested in pointing out such a traceability can skip it, with no impact at all on the understanding and mastering of the RENAISSANCE method.

Table of Contents

1 Introduction	1
1.1 Introduction.....	2
1.1.1 WHO is this report aimed at.....	2
1.1.2 WHY to read it.....	2
1.1.3 WHAT is covered in this report.....	2
1.1.4 HOW does this report tackle the problem.....	3
1.1.5 Foreword.....	3
1.2 RENAISSANCE Objective and Constraints.....	3
1.3 RENAISSANCE Benefits.....	4
1.4 Outline of the Report	6
2 The RENAISSANCE Method.....	7
2.1 Concepts and Notation	8
2.2 Adoption Hints	10
2.3 Overview.....	11
2.4 Method Description.....	15
2.4.1 Top Level Activities.....	16
2.4.2 Investigate	18
2.4.3 Assess As-Is.....	19
2.4.4 System Modelling	21
2.4.5 Select Assessment Characteristics.....	24
2.4.6 Assess Characteristic #i	26
2.4.7 Global Assessment.....	28
2.4.8 Review Business Goals.....	30
2.4.9 Assess To-Be.....	31
2.4.10 Elaborate Possible Strategies.....	32
2.4.11 Assess Strategies	34
2.4.12 Plan	36
2.4.13 Choose Strategy.....	37
2.4.14 Size & Cost Estimation	38
2.4.15 Risk Assessment.....	40
2.4.16 Benefits Analysis.....	42
2.4.17 Preferred Strategy Assessment.....	44
2.4.18 Strategy Review.....	46
2.4.19 Assess Company Maturity.....	47
2.4.20 Assess Supplier.....	48
2.4.21 Assess Development Environment.....	49

2.4.22 Select Solution.....	50
2.4.23 Plan Evolution.....	51
2.4.24 Build Plan.....	52
2.4.25 Implement.....	54
2.4.26 Detailed Analysis.....	55
2.4.27 Implement Transformation.....	59
2.4.28 Create Component Batches.....	61
2.4.29 Integration Design.....	63
2.4.30 Test Planning.....	65
2.4.31 Transformation.....	67
2.4.32 Test.....	69
2.4.33 Preparation of Target Environment.....	71
2.4.34 Deliver.....	73
2.4.35 Validate System.....	74
2.4.36 Deploy.....	76
2.4.37 Deployment.....	77
2.5 References to RENAISSANCE Reports.....	79
3 Framework - Method Traceability.....	80
3.1 Introduction.....	81
3.2 Traceability Description.....	81
4 Bibliography.....	85

1 Introduction

Contents

- 1.1 Introduction
- 1.2 RENAISSANCE Objective and Constraints
- 1.3 RENAISSANCE Benefits
- 1.4 Outline of the Report

Summary

This chapter introduces the RENAISSANCE method for reengineering legacy systems. Objectives of the method, as well as constraints and benefits of adopting it are presented and discussed. The outline of the report concludes the chapter.

1.1 Introduction

1.1.1 WHO is this report aimed at

The intended audience are

- senior managers and project managers which are going to start a reengineering project;
- people, both technical and managerial, involved in a reengineering project;
- people interested in reengineering methods.

A background in Software Engineering is recommended, even if not mandatory.

The reading of the "RENAISSANCE Framework" report is recommended, but not mandatory. People interested in learning more on the background of the method can find details in such a report.

Keywords: *application management, software maintenance, software evolution, legacy systems, software reengineering,*

1.1.2 WHY to read it

The objectives of this report are:

- to define a method for reengineering legacy systems; that is to provide support for deciding *if* and *how* reengineering the existing system to migrate toward an evolutionary one;
- to provide senior and project managers with an effective approach for conducting a reengineering project;

1.1.3 WHAT is covered in this report

This report covers the following issues:

- the definition of the method as a collection of activities to perform for conducting a reengineering project; activities are detailed in terms of description of involved actions, input, output, and constraints; activities and products definition will be further detailed in the next version of this report, when the method, refined on the basis of feedbacks from pilot projects, will be fully operational;
- the definition of the operational way to use the method, based on the concept of organization and project specificity;
- adoption hints to successfully introduce RENAISSANCE within an organization and have benefits from its use;
- traceability from the RENAISSANCE framework to the RENAISSANCE method, for people interested in such a mapping.

1.1.4 HOW does this report tackle the problem

A top-down approach has been used in writing this report:

- first, it is presented an overview of the method;
- then, adoption hints are given;
- finally, the method is presented, with a top-down approach again:
 - it is described the top-level view of the method activities;
 - then activities are decomposed and detailed, with the aim of providing operative procedures for reengineering a legacy system.

1.1.5 Foreword

It has to be noted that this version of the method has to be considered as a first incomplete draft, both in terms of scope (only process is described, not e.g. organisation, advice to individuals, etc.), breadth (not all activities are described) and depth (levels of details in descriptions).

The purpose of this document, delivered earlier than scheduled in the project programme, is to prepare and allow starting the evaluation of the method from application providers partners immediately after the end of the first phase of the project. This preliminary version of the method will be the basis for users's experimentation, whose feedbacks will be taken into account for engineering the method during the second phase of the project.

1.2 RENAISSANCE Objective and Constraints

The main objective of the RENAISSANCE project is therefore:

To develop a *systematic method* for system evolution and re-engineering which is geared to the requirements of the commercial systems domain.

This report presents the result of the project, that is the RENAISSANCE method for re-engineering and evolving software systems.

Benefits of reengineering an existing system can be summarised as follows:

- **Lower costs** - there is evidence from a number of US projects that reengineering an existing system costs significantly less than new system development. The figures quoted by Ulrich ('The evolutionary growth of software reengineering' in R.S. Arnold, Software engineering, IEEE Press) showed that reengineering a system cost \$12 million compared to an estimated \$50million cost of redevelopment.
- **Lower risks** - incremental reengineering of a system means that the risks of each improvement are relatively low. It is less likely that the

business will be faced with a system which does not meet its real needs.

- **Better use of existing staff** - existing staff expertise can be used and staff can develop their skills as the system is reengineered. There is less need to bring in new staff from outside the company.
- **Revelation of business rules** - as a system is reengineered, business rules which are embedded in the software may become clear. This is particularly likely when these rules relate to exceptional situations.
- **Incremental development** - reengineering can be carried out in stages as budget and staff are available. The organisation always has a working system available. End-users of the system have time to adapt to system changes and are not faced with a completely new system.

It has to be noted that even if the RENAISSANCE method defined in this report is intended to be a general method of wide applicability, techniques required by the pilot applications that are going to be used for the assessment of the method itself are analysed in more details. The method focuses on a subset of the possible evolution strategies.

Table 1 shows the reengineering strategies supported by RENAISSANCE to achieve the above mentioned objective (details can be found in the RENAISSANCE documentation, and especially in the report named D3.3 "Evolution Strategies"), which can be summarised as follows:

- to continue with the current maintenance approach;
- to reengineer the system from user interface, structure, architecture, design point of views;
- to replace the system with a new developed one.

Strategy	Characteristics
Maintenance	Bug-Fixing, adaptations, etc. without changing the structure or architecture.
Re-vamp	The user interface of a system will be updated to a more advanced technology, i.e. character mode to GUI. The general structure of the software will not be changed.
Re-structure	The structure of the software will be changed. The underlying hardware will not be changed.
Re-architecture	The structure of the system will be changed as well as the underlying hardware.
Re-design with re-use	A new system will be created integrating re-usable assets of the existing system.
Re-placement	The existing system will be replaced by a new developed system without regarding the existing one.

Table 1 : Evolution Strategies supported by RENAISSANCE

1.3 RENAISSANCE Benefits

Error! Style not defined.

A concrete and motivated identification of the specific benefits brought from the application of the RENAISSANCE method will be possible only after the experimentation of the method itself.

In the following, an initial list of expected benefits is reported:

Enable Business Process Change

- *Functionality and Flexibility*
- *Speed to market and response time*
- *Re-engineer or replace obsolete processes*

Improve Productivity

- *Employees and end-users*
- *Processes*
- *Interfaces to Processes*

Support Corporate Initiatives

- *Customer services and satisfaction*
- *Competitive response to existing and new markets*
- *Re-organisation, downsizing and decentralisation*
- *Alignment of corporate and IT goals*
- *Improve revenues and profits*

Control Costs

Improve Information Access

- *Improved access and usability of corporate data*
- *Improved accuracy of process data*

It must be noted that the motivations that drive RENAISSANCE method adoption fall in two category:

- *enhancement of the market competitiveness,*
- *improvement of the IT performances.*

As enabler for *business process change*, RENAISSANCE defines a way to assess a quality factor and a business value for the existing system within the current organization. Using such values and the business goals, the method provides support for identifying the best kind of intervention to be done on the existing system.

It is not always the case that reengineering is the best solution: to continue with the current maintenance approach or even replace the legacy system with a new developed one without regarding the existing system could be more convenient than reengineering the system itself.

This is why RENAISSANCE uses a two-steps approach:

- first identifies the best approach for evolving the system and justifies the choice on the basis of the assessment done both on the existing and desired systems; and
- then implement the transformation.

The categories of activities supported by RENAISSANCE can be hence summarized as follows:

- **Justifying the evolution project** - involving the investigation of an existing system to identify it as candidate for evolution, the assessment of alternative evolution strategies, the estimation of risk, effort and resources necessary for the transformations, the costing of these and a cost-benefit analysis exercise to decide whether the project can be justified and, if so, to identify the best alternative.
- **Planning the evolution project** - involving the use of the estimate of risk, effort, resources and associated costs as a base from which to plan the evolution project. Planning should address all identified areas of risk and to consider all aspects of the evolution project both from the technical and human viewpoints during the transformation and for the foreseeable life of the new system.
- **Executing the plan** - involving the creation of the environment in which the transformation will take place, the performance of all planned tasks, the monitoring of progress against the project plan, and the integration of the new system with existing systems.

1.4 Outline of the Report

Chapter 1 defines the objectives of the method, as well as constraints and benefits of using RENAISSANCE.

Chapter 2 defines the RENAISSANCE method for evolving legacy software systems, using a notation, both graphical and textual, which is introduced at the beginning of the chapter. An approach for adopting RENAISSANCE is suggested as well.

Chapter 3 presents the traceability from the RENAISSANCE framework to the RENAISSANCE method, which is an instance of the framework.

Chapter 4 contains the bibliography of this report.

2 The RENAISSANCE Method

Contents

- 2.1 Notation
- 2.2 Adoption Hints
- 2.3 Overview
- 2.4 Method Description

Summary

This chapter defines the RENAISSANCE method for reengineering legacy software systems. The first section introduces concepts and the notation of the method, both textual and graphical. Then, it is suggested a strategy for adopting RENAISSANCE within an organization, and an overview of the method is presented, with particular emphasis on the concept of organization and project specificity supported by the method itself. The last section defines the method using the introduced notation and details the activities to perform for conducting a reengineering project using the RENAISSANCE approach.

2.1 Concepts and Notation

In order to facilitate readers, the RENAISSANCE method is defined with a very simple notation, both graphical and textual, which comprises the intuitive concepts of

- activity,
- activity decomposition,
- inputs, outputs and enabling conditions

Anyway, for readers which are already familiar with a process modelling language, the adopted representation can be easily mapped to it, whatever the language is.

The adopted graphic notation is summarized in the following.

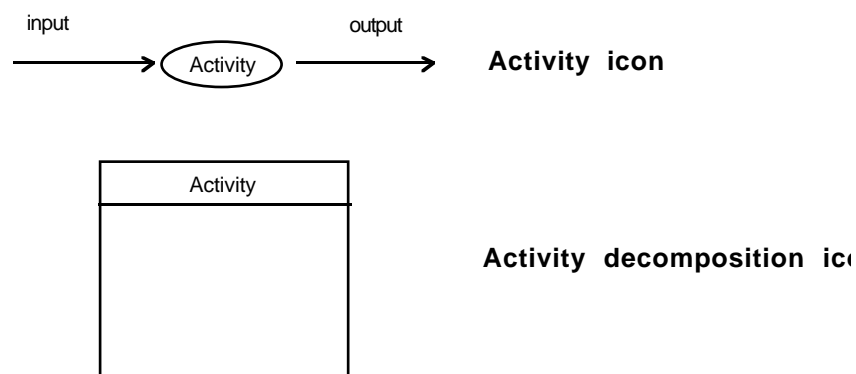


Figure 1 : The RENAISSANCE Notation for Activities

As depicted in the above picture, the following mapping rules hold:

- the intuitive notion of *activity* is mapped into a bubble with labels for inputs, outputs and activity name;
- the *control flow* among activities is represented by arrows from sources to targets of the flow;
- an activity can be decomposed into *sub-activities*; the corresponding icon is a square with a label for the name of that activity; the square contains the sub-activities graph;
- the word *task* denotes an activity which is not further decomposed (at the level of detail we have, arbitrarily, chosen to study a process)..

While a bubbles-and-arrows graph defines the control flows among activities, tasks describe the actions which have to be done in order to

perform the intended objective, and also the inputs and the output products, as well as the conditions, if any, which regulate the execution of the task itself. Conditions represent more than the *presence* of elements; they are generic logical conditions, including qualitative statements on elements.

This means that RENAISSANCE defines a general reengineering method, which can be customized by users of the method in the two following ways:

- activities can be decomposed and specialised on the basis of the user company/project;
- tasks can be specialised on the basis of the user company/project.

Since tasks detail actions to perform, they need a specialised notation. The proposed template for the description of the tasks identified in the method is based on two representations:

- a graphical one which can be used to build sample graphical synopsis of the method, or of method parts;
- a textual one, describing more precisely the different facets of a terminal activity.

The idea behind the chosen formalism is to facilitate the instantiation of the method for specific companies/projects, the control flows between the different tasks being not defined at the current description level (the task level), describing only the conditions to be full-filled to enable task execution

The graphical representation of a task is shown in Figure 2.

Task Name		
Task description		
Roles involved		
Pre Conditions	Enabling Conditions	Post Conditions

Figure 2 : The RENAISSANCE Notation for Tasks

The following is the template for the textual description of a task:

Task description:	a description of the actions performed within the task.
Pre conditions:	the list of elements that must be available in order to allow to start task execution, and, if any, logical conditions, including qualitative statements, on those elements.
Post conditions:	the list of elements that must be available at the end of the task execution, and, if any, logical conditions, including qualitative statements, on those elements.
Enabling conditions:	the list of conditions describing in which case the task has to be executed; if empty, the task is mandatory.
Roles involved:	the list of roles involved to perform the task.

Because *Role Involved*, *Enabling Conditions*, *Pre Conditions* and *Post Conditions* can be free text of arbitrary length, they are described in detail in the textual part, and they will be identified in the graphic form simply using a 3 or 4 letters acronym.

2.2 Adoption Hints

As with any new technology, reengineering and reengineering methods should be inserted cautiously within an organization.

Before starting a strategic reengineering project, we suggest to familiarize with the RENAISSANCE method by a set of pilot projects. A pilot project should be small and isolated, that is with few or no impact at all on other software components and data files.

Such pilot projects should enable an organization to test the new technology and to develop a background. Lessons learnt from pilot projects can then be applied to larger, more complex reengineering projects.

The suggested adoption approach, based on trial use and pilot projects, can be represented by a curve, which shows how an organization react to new working methods. One of the possible curves, known in literature as Conner and Patterson's Adoption Curve, is shown in Figure 3 and detailed in the following.

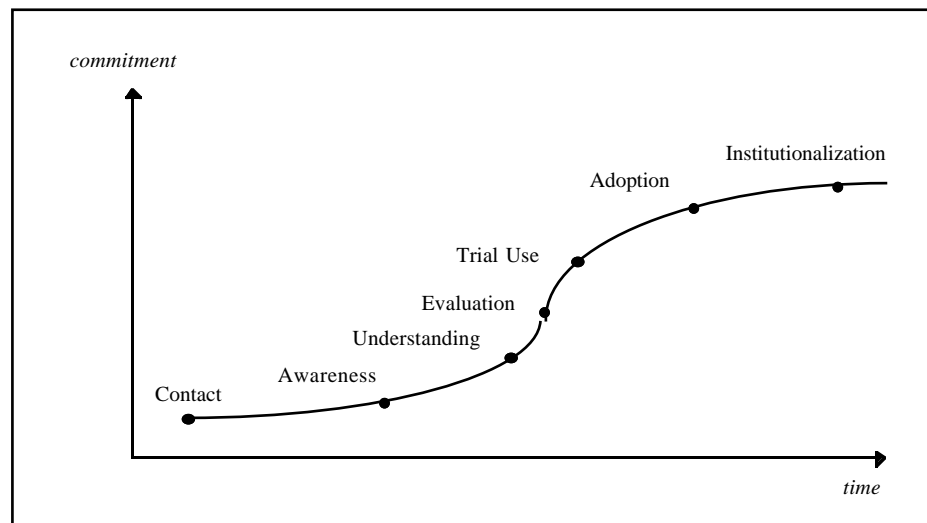


Figure 3 : Conner and Patterson's Adoption Curve for a new technology

After encountering a new process or technology, potential customers of that technology increase their awareness of its usage, maturity, and application. If the process or technology is promising, then customers try to better understand its strengths, weaknesses, costs, and applicability. The first activities in the adoption curve take a significant amount of time. Promising processes and technologies are then evaluated and compared. To reduce risk, customers usually try new processes or technologies on a limited scale through beta tests, case studies, or pilot projects. A customer then adopts processes or technologies that prove effective. Finally, refined processes and technologies become essential parts of an organization's daily process: we name this as institutionalisation.

2.3 Overview

The overview of the RENAISSANCE method is shown in Figure 4.

The method defines:

- a set of activities and tasks (as defined in the "Concepts and Notation" section) whose execution supports the overall reengineering project; and
- the control flow for the identified activities, which drives the cooperation among tasks.

The idea behind the method is that Reengineering must be *company and project specific*. This is supported in the following way:

- activities identified by RENAISSANCE are generic activities, which can be specialised and decomposed to address specific needs of the user of the method, coming both from company and project constraints and practices;

- tasks are identified by RENAISSANCE through their interface, which is defined in terms of Pre, Enabling and Post Conditions. Concerning the implementation of a task, RENAISSANCE simply describes the actions to be performed within the task, without forcing the user to adopt a particular approach for implementing such actions; in other words, the method is *goal-oriented*, in that it specifies goals for tasks rather than how to accomplish them;
- the execution of tasks can be customized, by using the Enabling Conditions lists, as well as the overall control flow.

The implementation of this concept, depicted in the figure below, can be described as follows:

Reengineering a system S within an organization O using RENAISSANCE means

1. to instantiate the RENAISSANCE method for that system and that organization, and
2. to apply that instance of the method to S within O .

The instantiation of RENAISSANCE to different reengineering projects and different organizations is not a trivial job.

The level of difficulty for a company to instantiate the method in an effective manner for a given project decreases with the level of confidentiality the company has on the method itself. And the level of confidentiality increases when applying the method more and more times.

In order to achieve the best result in using the RENAISSANCE approach, method and techniques, it is strongly recommended to adopt the method as described in section 2.2 "Adoption Hints". This should prevent false starts, which could compromise the return of investment in choosing RENAISSANCE as reengineering methodology.

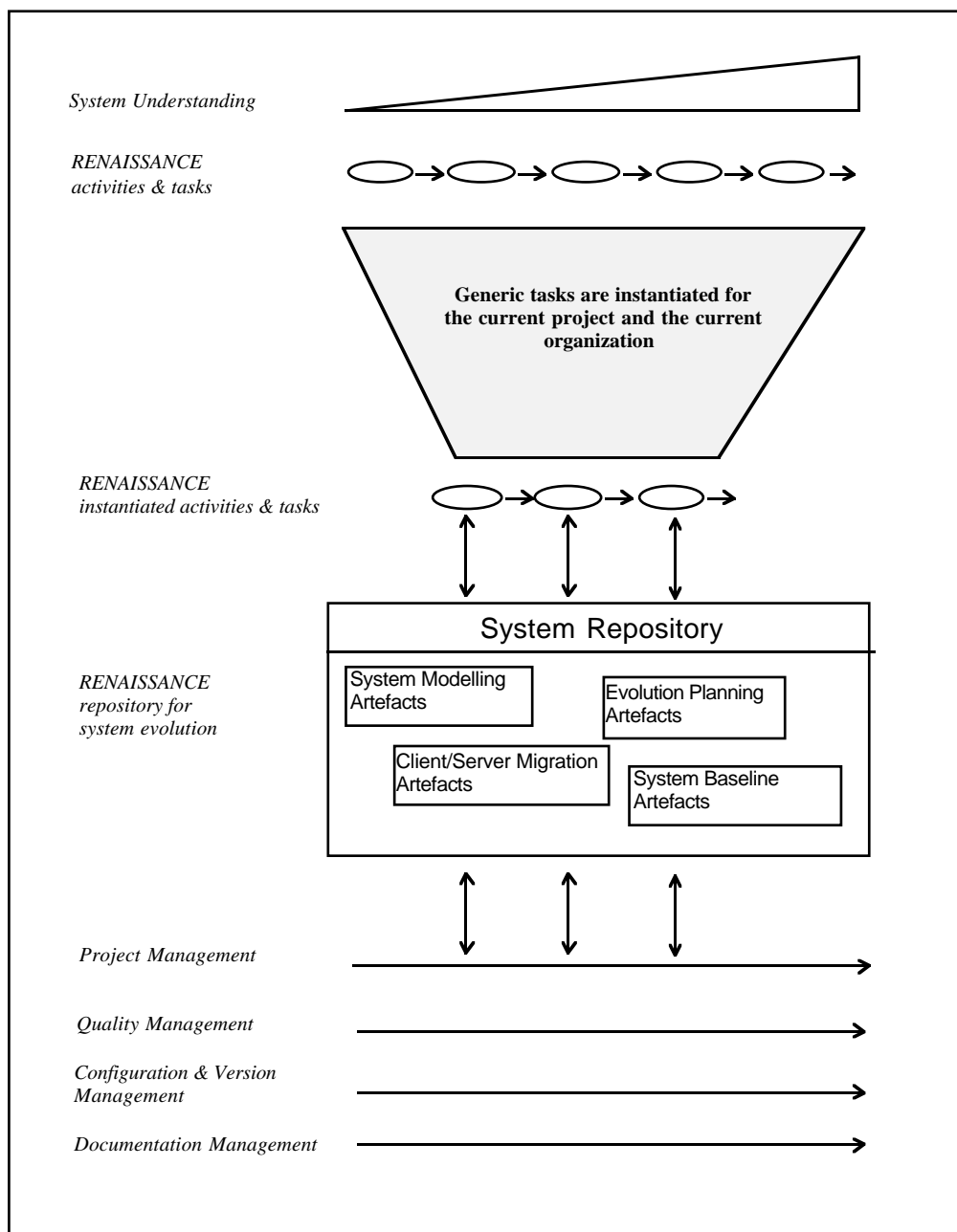


Figure 4 : Overview of the RENAISSANCE Method

RENAISSANCE enforces the concept that reengineering is a project, which means that:

- it must be managed, and
- it must be conducted

according to the standards of the company (or department, or group, etc.) standards.

This means that the method suggests to manage:

Error! Style not defined.

- the resources and timescale of the project,
- the quality of the project and its products,
- the documentation of the project and its products,
- the versioning and configuration of the project and its products,

but it does not force any way to do that: users are completely free to use their own approach, or even any approach at all, if this is compliant with the standards of their company for that specific project.

This means that *Project Management*, *Quality Management*, *Documentation Management*, and *Configuration and Version Management* are activities which cross all the activities defined by the method, and the user is responsible for using and implementing them.

When we proceed in applying the method, that is in transforming the existing legacy system, we need more and more information, or at least more and more detailed information, on the existing system. Hence, the understanding of the legacy system must be pervasive of the overall reengineering process, and the level of detail which is necessary depends on the step of the process. Each activity, or step, defines which information it needs, and suggests the way to collect them. The figure explicitly represents this by the triangle named *System Understanding*, which pervades all the RENAISSANCE activities.

All the information concerning the system under reengineering are collected in a *System Repository*, which is accessible by all the activities of the process. The repository, which grows as the process goes on, contains the following four categories of information:

- *System Baseline Artefacts*: they identify the legacy system to reengineer.
- *Evolution Planning Artefacts*: they support the building of a plan for evolving the subject legacy system.
- *System Modelling Artefacts*: they describe the model of the legacy system, and the desired model.
- *Client/Server Migration Artefacts*: they support the migration to a client/server n-tiers architecture.

The repository also holds information about best practices and benchmarking, which are necessary to evaluate the impact of the small enhancements involved in the continuous evolution of the system.

Customization of the process to specific reengineering projects can involve to insert new chunks of information in the system repository. In that case, new and specialised tasks will be written by the user to produce and consume such information.

From the logical point of view, RENAISSANCE is a two-phases method, as shown in Figure 5:

- the *What To Do* phase (or *Planning* phase) concerns the decision of the kind of intervention the legacy system needs. It identifies which

reengineering strategy must be applied to the system to evolve it, based on a detailed assessment of the system itself in terms of business goals, technical factors and economic indicators. The result is a Go/NoGo decision on starting that reengineering project, and, in case of Go, a plan which drives the project of transforming the system.

- the *How To Do* phase (or *Doing* phase) concerns the implementation of the planned transformation. It supports the process of implementing the identified migration path, ranging from the building of the new model of the system to its deployment in the user environment.

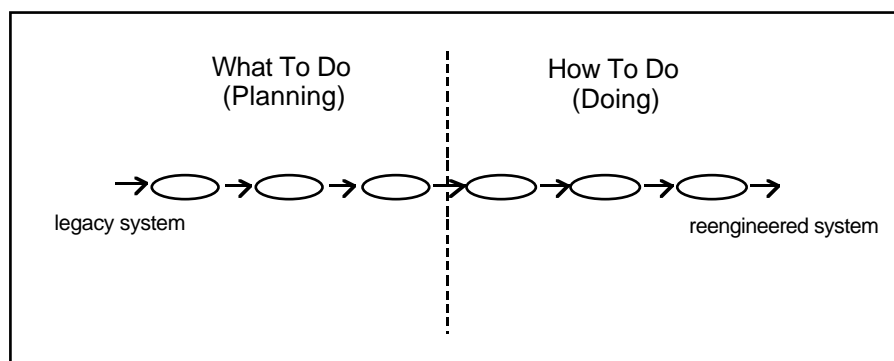


Figure 5 : The Logical View of the RENAISSANCE Method

2.4 Method Description

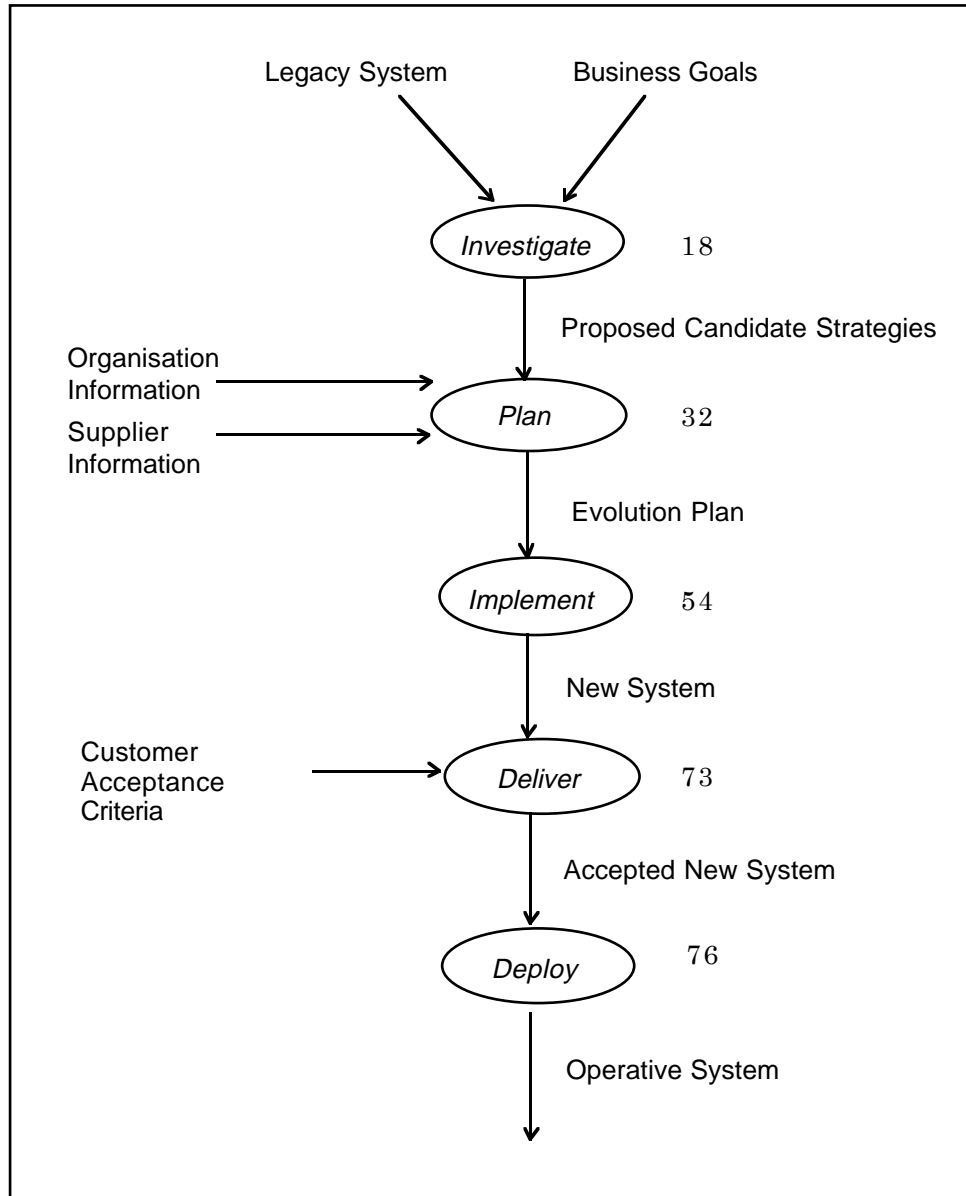
This section presents the RENAISSANCE method, as a collection of activities and tasks: activities are depicted in graphical diagrams, while tasks are described both with textual and graphical representation.

In order to facilitate reading, each diagram contains page references to the involved activities.

The last subsection points out references to other RENAISSANCE reports.

The starting point for browsing the method activities is the section "Top Level Activities", which defines the top-level activities of RENAISSANCE.

2.4.1 Top Level Activities



Activity Description

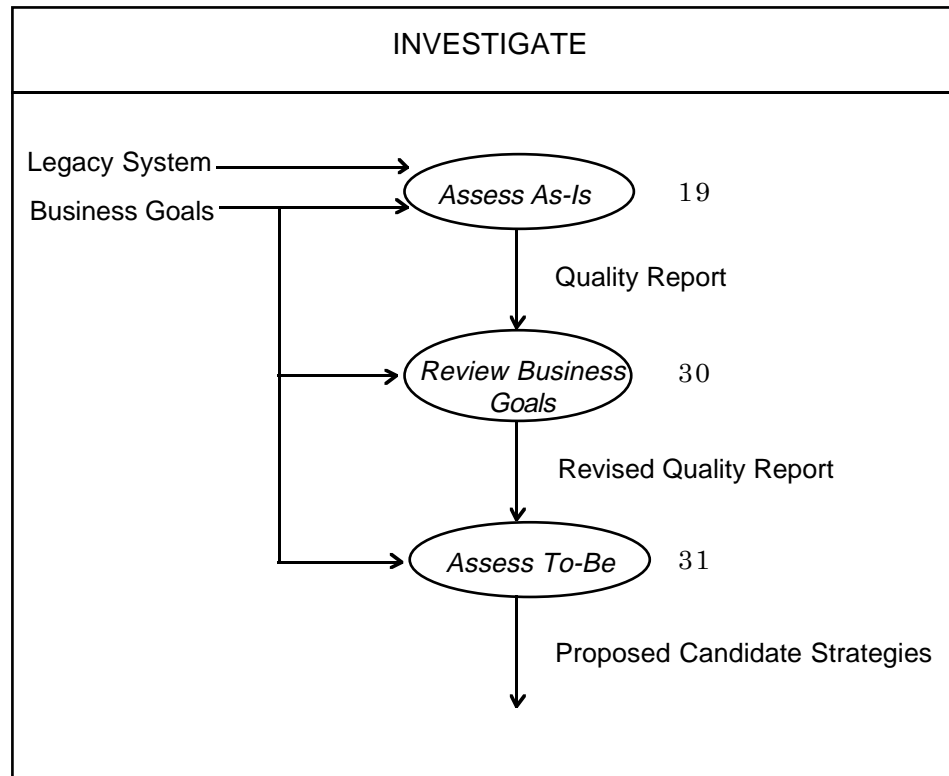
With the business goals in mind, the selected legacy system is investigated with the aim of assessing its current quality and its business values, and its needs for evolution.

As a result, a set of candidate strategies for transforming, if necessary, the system is proposed and then refined taking into account other constraints, like the availability within the company of the necessary expertise for implementing the proposed technologies.

The final result of this technical and economical assessment is a detailed plan for driving the evolution of the subject system, that is to implement the necessary transformation.

The new system, after implementation, must be delivered and accepted by the final user. The deployment of the accepted system into the target operational environment concludes the intervention.

2.4.2 Investigate



Activity Description

The scope of this activity is to determine a set of candidate reengineering strategies needed by a legacy system to address the business goals of the company.

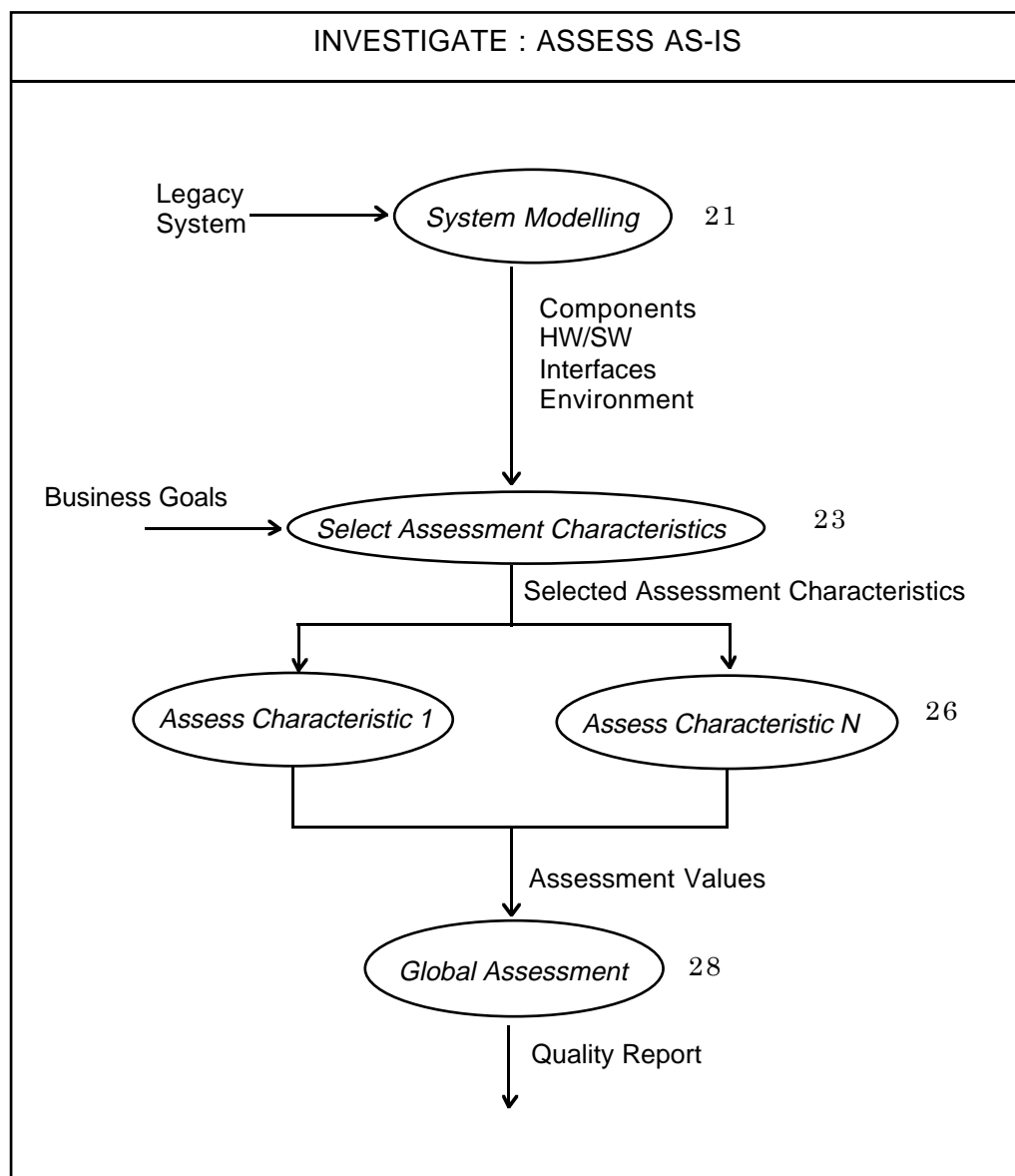
The process of determining such strategies consists in:

- assessing the current quality and business value of the existing system;
- refining such a value in function of the business goals; and
- assessing the desired system and providing a transformation mapping between the existing system and the desired one.

Such a mapping identifies a set of possible strategies to be applied to the legacy system to migrate toward the target one.

This activity is referenced on page 16.

2.4.3 Assess As-Is



Activity Description

The legacy system is assessed in terms of:

- a quality factor, and
- a business value,

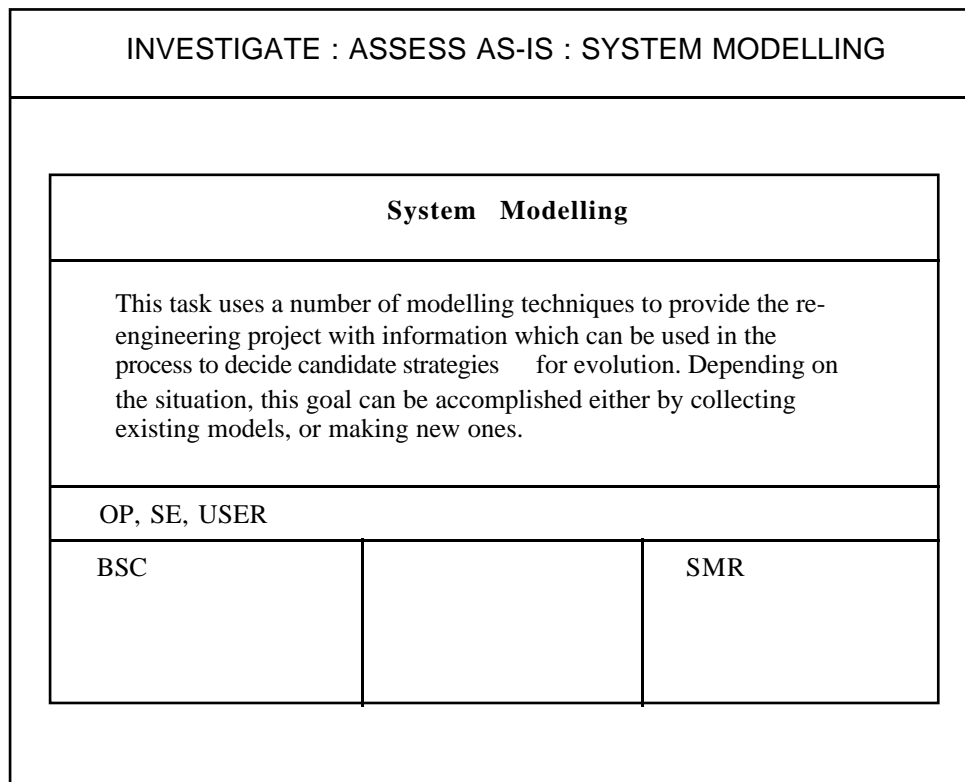
which depend on a lot of characteristics, both company and system specific.

First, the system must be understood on the basis of technical issues, like the hardware used, its interfaces and its subsystems. Then, the combination of such a knowledge on the system and the business goals

must be used to determine a set of characteristics (i.e. the architecture, the lines of code, and so on) to be evaluated in order to compute quality and business values for the system. The identified characteristics are then assessed, and their results are combined and evaluated with the aim of computing a global quality factor and business value of the system.

This activity is referenced on page 18.

2.4.4 System Modelling



System Modelling

Reference: Architectural Models for Evolution - Chapter 2

This task is referenced on page 19.

Task description

When a reengineering project is started, a baseline consisting of a legacy system is established. Usually, the structure of this baseline is not determined, as the documentation of the system has been lost or is outdated, and the expert application developers have been lost to the company.

The *system modelling* process aims at using several techniques for rebuilding the (usually lost) system context. Depending on the situation, this goal can be accomplished either by collecting existing models, if any and if they are sufficient for the project, or making new ones.

Full information about which modelling techniques the RENAISSANCE project has selected for achieving this can be found in the RENAISSANCE consultancy report “Architectural Modelling for Evolution”, Chapter 2.

The models produced during this task do not form a complete model of the whole system. A trade-off must be made to how much should be modelled versus the costs. The modelling should stop when the detail has reached a level sufficient to identify different evolution strategies. More precise

guidelines can be given when the RENAISSANCE method has been applied within the project.

Below follows a summary of the selected modelling techniques:

- *Operational schemas* are used for modelling business processes. Business processes are modelled as a sequence of *activities* which must be completed by *roles*.
- *Use case diagrams* are used to model the communication among actors and functions in the system.
- *Data flow diagrams* are used to model the information flow among processes, data stores and external agents.
- *Extended entity relationships diagrams* are used to document the data structures in the existing application, and form a base for transforming these structures to the re-engineered application.
- *Block diagrams* are used for documenting the relationships among software components in a legacy system.
- Finally, *hardware/network diagrams* are used to document the composition of hardware and network used by the legacy application, and the mapping of software components onto this hardware.

The models produced using the notations provided for context models provide a wealth of information to the re-engineer:

- They provide an explicit representation of the system and its requirements.
- They are a vehicle for communication about the system among analysts, designers and users.
- They form a basis for the design and implementation of new systems, as well as a basis for detailed modelling of existing legacy systems for re-engineering them.
- The context models document the application system at a high level. Since the context model is not so detailed as the design models, the user of the context model can more easily understand the context models before dwelling into the more detailed designs.

Table 2 gives a summary of the modelling dimensions that the different diagramming techniques support.

Diagram Type	Process	Functionality	Data	Software	Communication	Hardware
Operational Schema	√	√			√	
Use Case		√			√	
Data Flow		√	√		√	
Entity Relationships			√			
Block				√	√	
Hardware/Network				√	√	√

Table 2 Comparing context views and diagram.

Pre Conditions

BSC - The main Pre Condition is the availability of the re-engineering baseline, which is an identified configuration of the legacy system which will be subject to re-engineering.

Enabling conditions

The system modelling task is mandatory.

Post Conditions

SMR - The result of the system modelling tasks are the diagrams from the modelling techniques described above.

To be more precise:

- *Components* (COMP): Are supported by the block diagrams and hardware/network diagrams.
- *HW/SW* (HSD): Supported by the hardware/network diagrams.
- *Interfaces* (INTD): Interface documentation is supported on different levels by the operation schema, use case, data flow, block, and hardware network diagrams.
- *Environment* (ENVD): The environment is described in different views supported by the operational schema, use case, and data flow diagrams.

Roles involved

OP - Operational roles provide knowledge on the environment

SE - Service roles provide technical knowledge

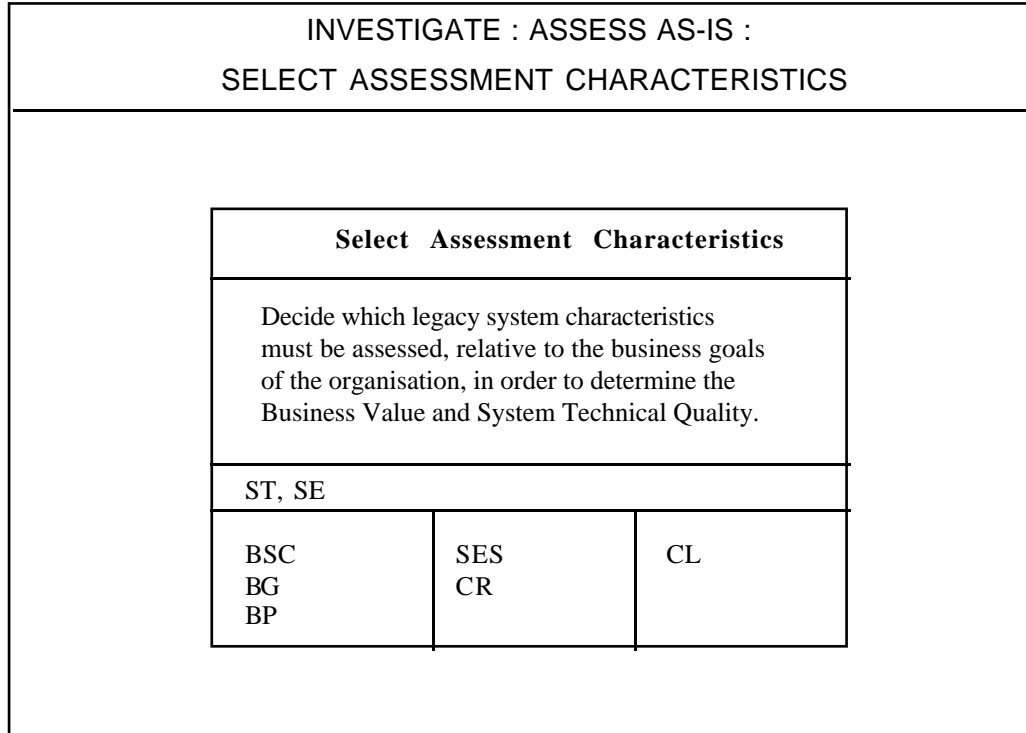
USER - Users need to be involved

In particular, in order to obtain the information needed in the system modelling task, the following roles are involved:

- *Evolution Modeller*: The evolution modeller is a software engineering expert with experience in documenting legacy systems using the modelling techniques described above.
- *Domain Expert*: The domain expert is an expert in the domain where the current application is operating.
- *Application Expert*: The application expert, if available, has detailed knowledge of what the current application does, how it does it, and how the application is organised.
- *User*: The user is the primary source for eliciting knowledge about how the system interacts with the users. Several users from different user categories can be interviewed during the system modelling task.

The evolution modeller co-operates with persons from these role categories to build the initial system model.

2.4.5 Select Assessment Characteristics



Select Assessment Characteristics

Reference: Evolution Management Report - 'Assessment for Evolution'.

This task is referenced on page 19.

Task description

Review the legacy system and its environment to identify characteristics which can be assessed to determine its Business Value (BV) and Technical Quality (TQ). There are five types of characteristics described and listed in the subsection 'Information Gathering and System Understanding'

Pre Conditions

BSC - The Baseline System Configuration for the legacy system.

BG - the business goals of the organisation.

BP - the business processes of the organisation.

Post Conditions

CL - List of characteristics deemed relevant for assessing the BV and TQ of the system

Enabling conditions

SES - There is a requirement to select an evolution strategy

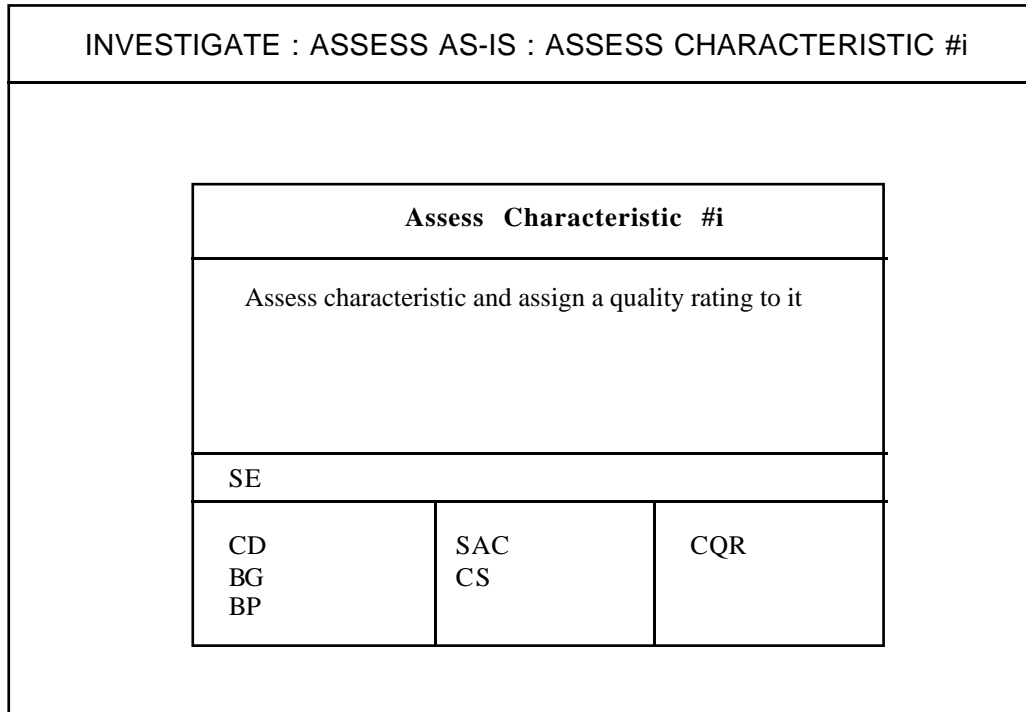
CR - There is a requirement to produce a costing document.

Roles involved

ST - Strategic to define business goals, business processes and system needs.

SE - Service to provide baseline system configuration and select characteristics.

2.4.6 Assess Characteristic #i



Assess Characteristic

Reference: Evolution Management Report - 'Assessment for Evolution'.

This task is referenced on page 19.

Task description

Assign a quality rating to the characteristic. We recommend a simple scoring approach explained in the subsection 'Assessment Characteristics and Rating Procedure'.

Assigning a quality rating is done by identifying quantifiable attributes of the characteristic which are then assessed.

The assessment can be done, using methods ranging from expert opinion to formal metrics, at the system or component level. This is described in the subsection 'Assessment Characteristics and Rating Procedure' and in more detail in the subsections 'External Environment Assessment' and 'Application System Assessment'.

Pre Conditions

CD - Definition of the characteristic

BG - Business Goals

BP - Business Processes

Post Conditions

CQR - Characteristic Quality Rating

Enabling conditions

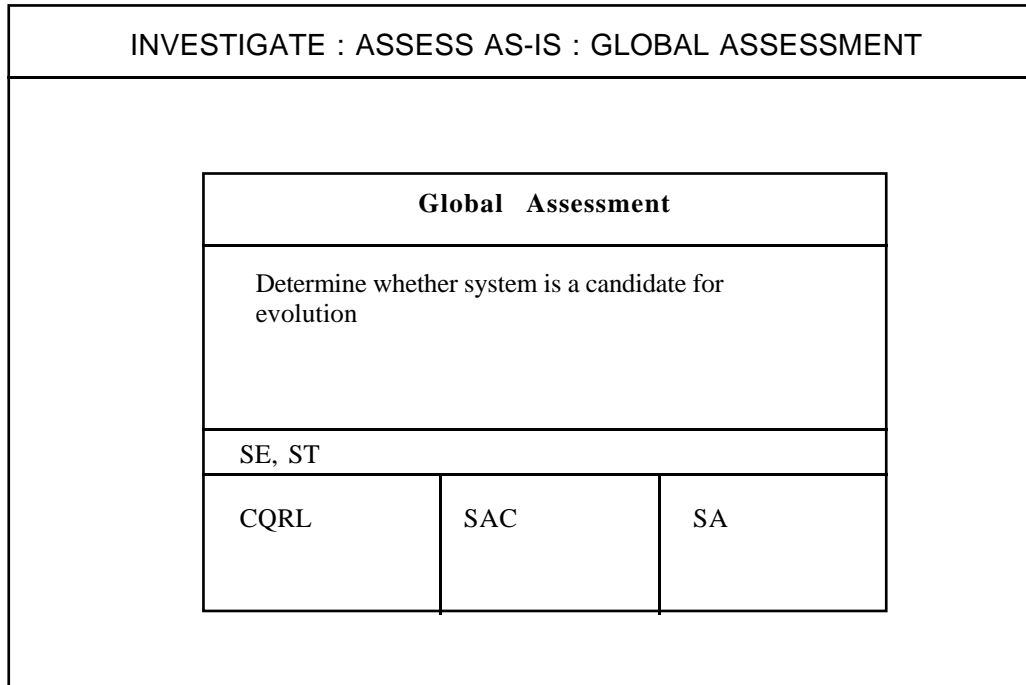
SAC - As for Select Assessment Characteristic.

CS - Characteristic has been selected for assessment.

Roles involved:

SE - Service to perform assessment.

2.4.7 Global Assessment



Global Assessment

Reference: Evolution Management Report - 'Assessment for Evolution'.

This task is referenced on page 19.

Task description

To determine whether a system is a candidate for evolution.

The overall Technical Quality (TQ) of the system is determined by a weighted average of the assessed characteristics as described in the subsection 'Application System Assessment'. This may need to be adjusted by environmental factors assessed in the section 'External Environment Assessment'.

The Business Value (BV) is determined as described in the subsection 'Business Value Assessment'.

Whether or not a system is a candidate for evolution depends on comparing its TQ with its BV. This process is described in the subsection 'Interpretation of Results'.

Pre Conditions

CQRL - A list of assessed system characteristics.

Post Conditions

SA - A System Assessment based on its TQ and BV stating whether or not it should be considered for evolution.

Enabling conditions

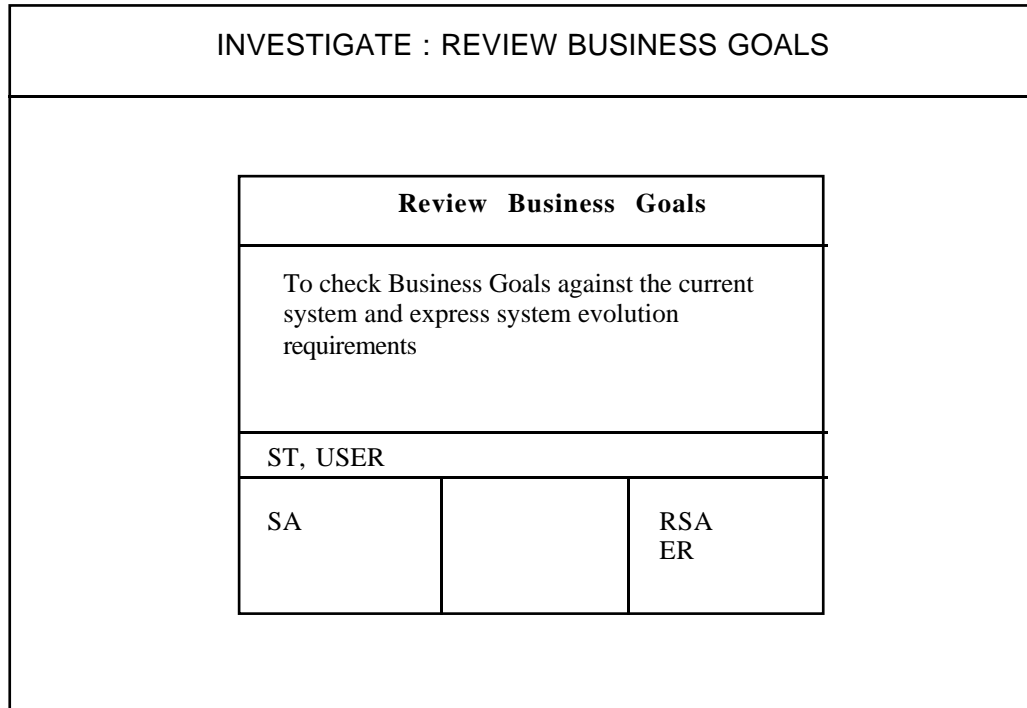
SAC - As for select Assessment Characteristic

Roles involved

SE - Service to provide TQ assessment

ST - Strategic to provide BV assessment

2.4.8 Review Business Goals



Review Business Goals

Reference: None

This task is referenced on page 18.

Task description

To check the Business Goals of the Company against the current system, and elaborate system evolution requirements.

Pre Conditions

SA - The current system assessment.

Post Conditions

RSA - Revised System Assessment.

ER - Evolution Requirements

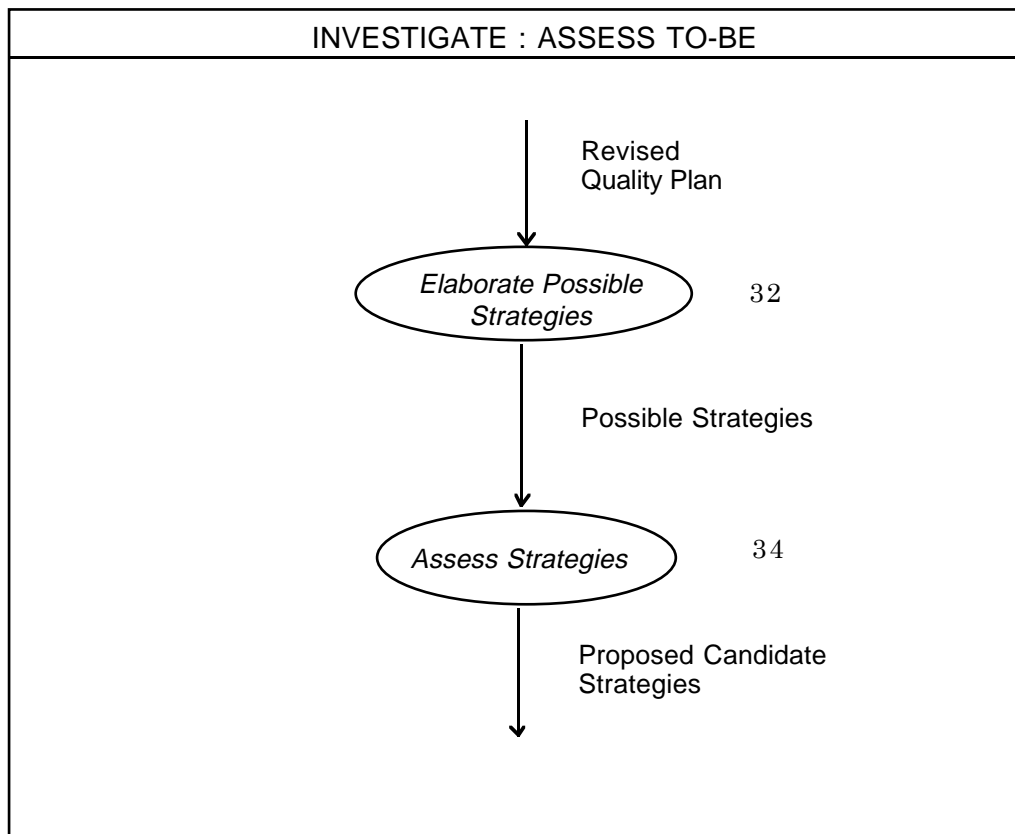
Enabling conditions

Roles involved

USER - The users, to express evolution requirements

ST - Strategic to provide Business Value assessment

2.4.9 Assess To-Be



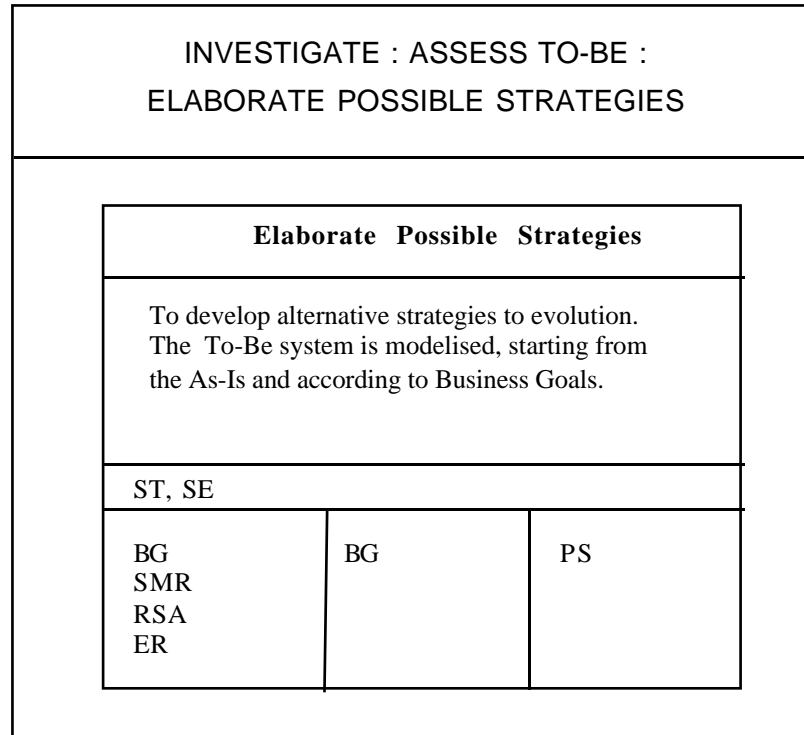
Activity Description

The assessment of the existing system, as detailed in its quality report, is used to determine a set of possible reengineering strategies for transforming the subject system. Such strategies must then be assessed from business goals, user expectations, and feasibility point of view.

The result of the assessment is a set of proposed strategies to be used, which still have to be evaluated and ranked by cost, risk, and benefit in order to select the (best) strategy to implement.

This activity is referenced on page 18.

2.4.10 Elaborate Possible Strategies



Elaborate Possible Strategies

Reference: None

This task is referenced on page 31.

Task description

To develop alternative strategies to evolution.

The To-Be system is modelled, starting from the As-Is model and according to Business Goals and evolution Requirements.

Dependent upon the degree to which the new systems are similar to the current system, models can be reused, adapted or re-developed.

Several possible strategies can be elaborated, depending on the Business Goals.

Pre Conditions

BG - The Business Goals of the company.

SMR - The result of modelling of the current system.

RSA - Revised System Assessment.

ER - Evolution Requirements.

Post Conditions

PS - A set of possible strategies, depending on the As-Is situation and the Business Goals of the company.

Enabling conditions

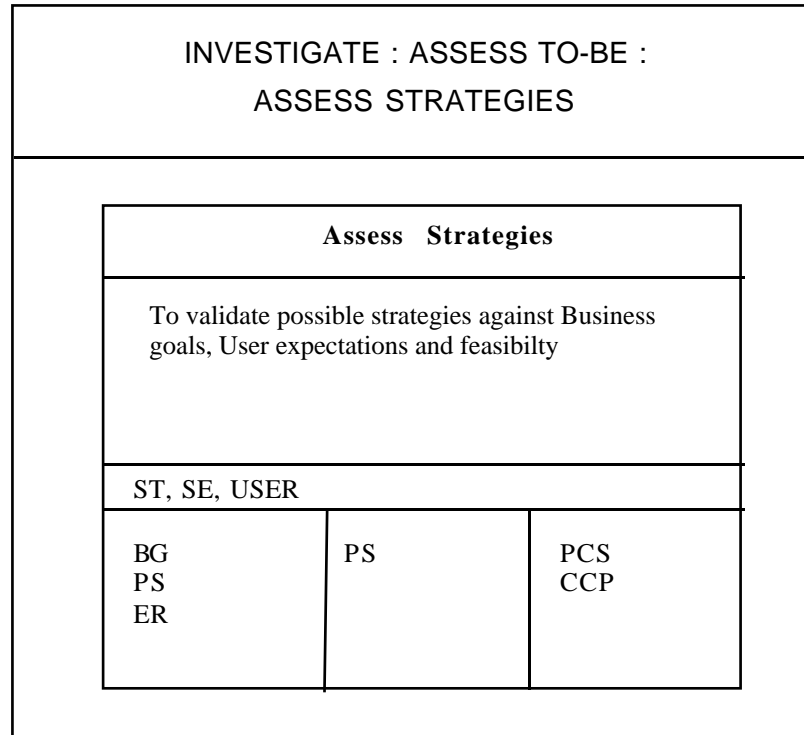
BG - expressed business goals.

Roles involved

SE - Service to provide baseline system description and elaborate solutions

ST - Strategic to provide BG

2.4.11 Assess Strategies



Assess Strategies

Reference: None

This task is referenced on page 31.

Task description

To validate the possible strategies against Business goals, User expectations and feasibility.

Each possible strategy is checked against user expectations (Evolution requirements) and Business goals.

Benefits analysis is prepared.

An operational and technical feasibility statement is prepared.

A Market Survey is performed, identifying candidate commercial packages, and showing strengths and weaknesses.

A set of proposed candidate strategies is decided.

Pre Conditions

PS - Proposed Strategies.

BG - Business goals

ER - Evolution requirements

Post Conditions

PCS - Proposed candidate strategies.

CCP - Candidate commercial packages

Enabling conditions

PS - Possible Strategies

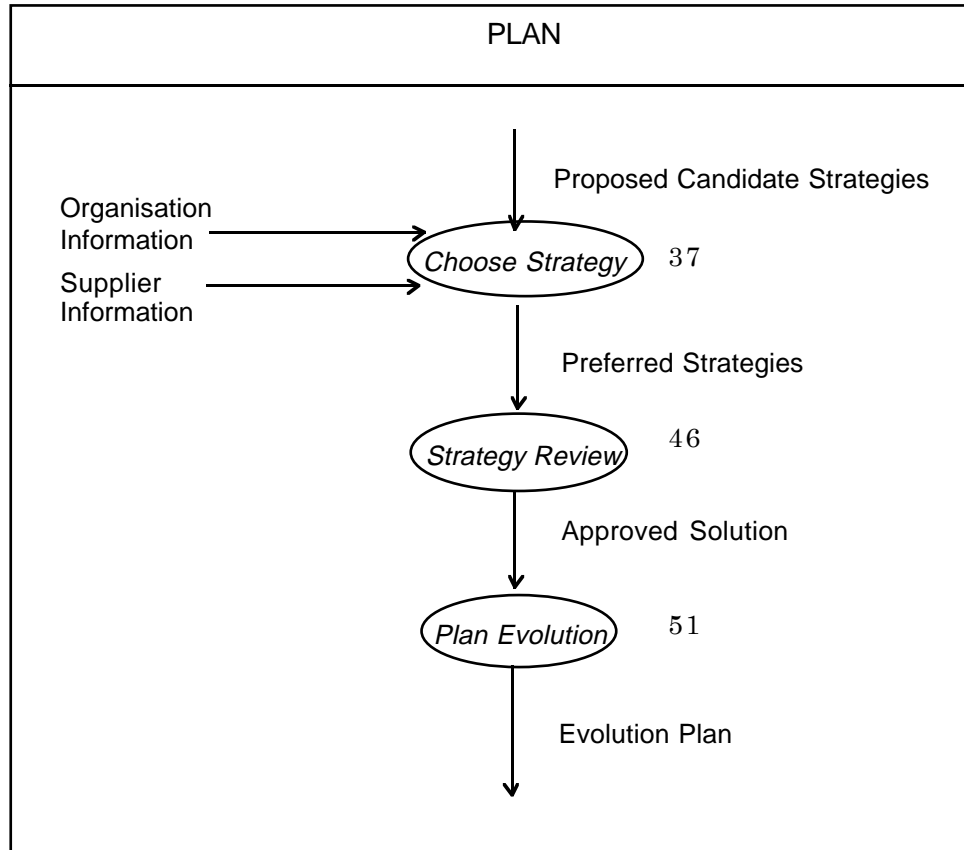
Roles involved

SE - Service to perform Market Study

ST - Strategic to select candidate strategies

USER - User to express and check requirements

2.4.12 Plan



Activity Description

The As-Is vs. Can-Be analysis identifies a set of candidate reengineering strategies for migrating the existing system toward the desired one.

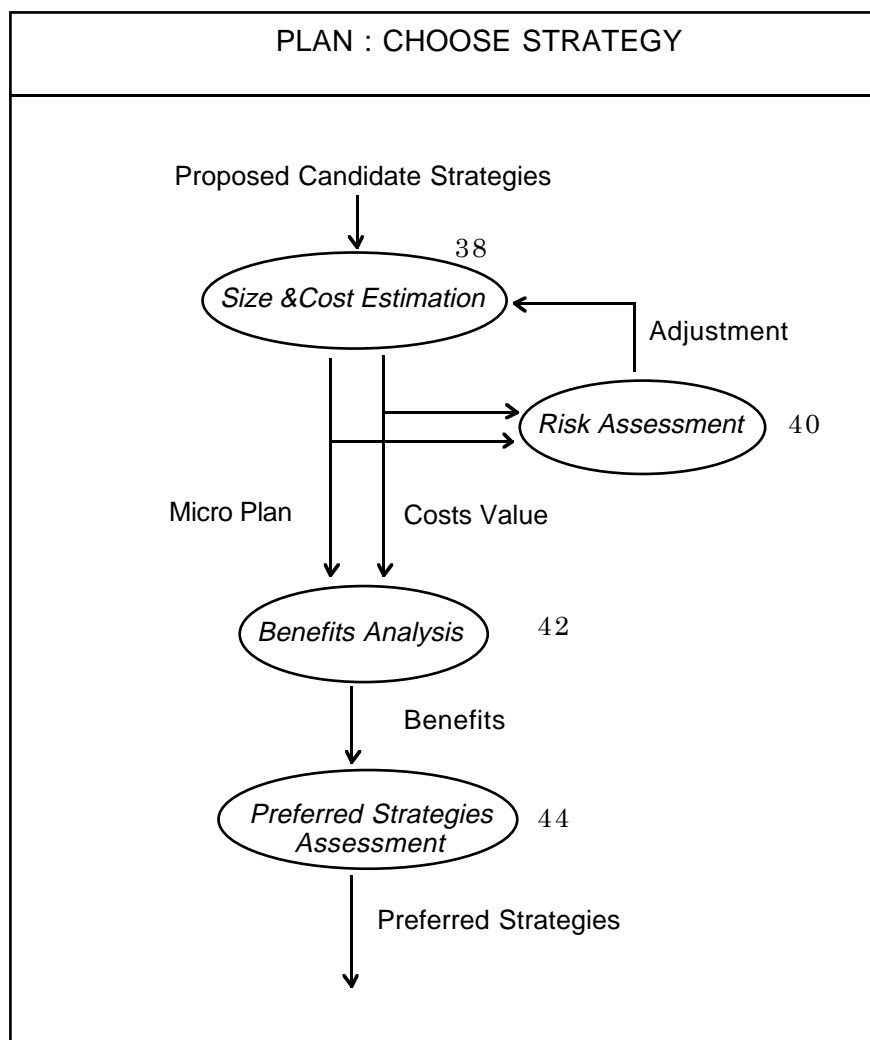
Such strategies must be evaluated on the basis of economic costs and risks, in order to have a set of preferred solutions.

Then, on the basis of other factors, like the availability of expertise for the selected strategies, the solution to be implemented is chosen, and the evolution plan for the reengineering project is built.

This activity represents the borderline for starting the reengineering project, and implements a sort of Go/NoGo decision.

This activity is referenced on page 16.

2.4.13 Choose Strategy

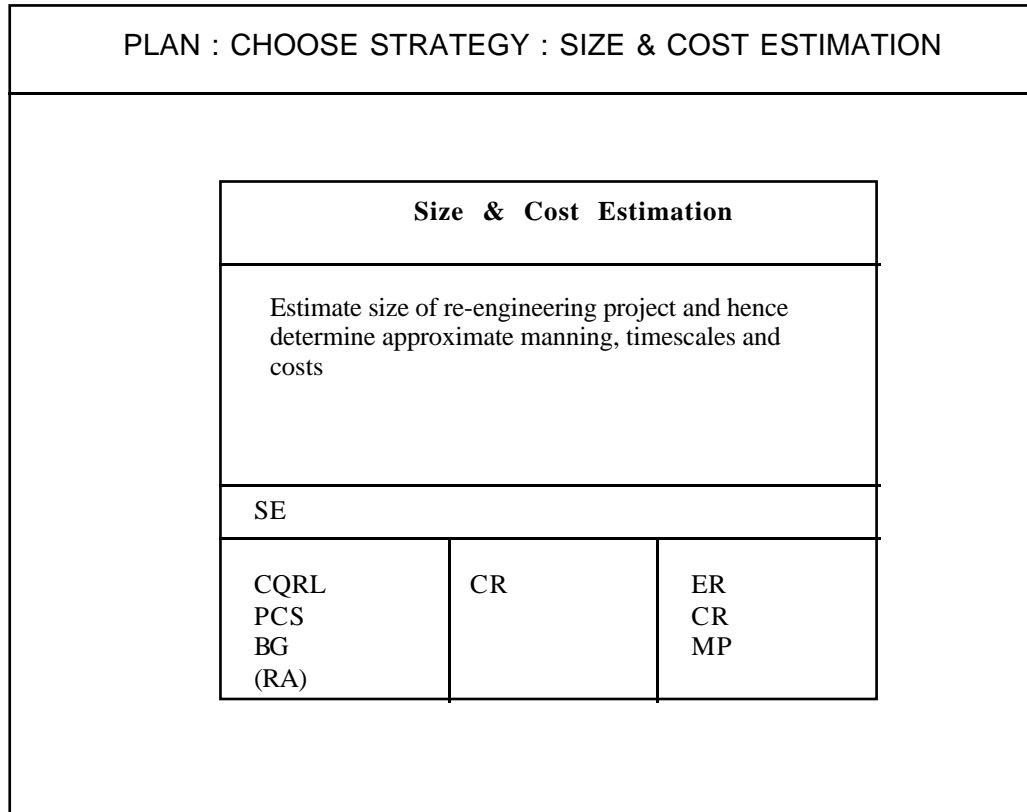


Activity Description

The reengineering strategies proposed as candidate for evolving the system are now evaluated with respect to size, cost and risk of the involved intervention. For each strategy, benefits of applying it are analysed, and, combining all the results, the candidate strategies are ranked by preference.

This activity is referenced on page 36.

2.4.14 Size & Cost Estimation



Size and Cost Estimation

Reference: Evolution Management Report - 'Estimating'

This task is referenced on page 37.

Task description

Using the list of assessed quality ratings, the business goals, the proposed candidate strategy (and the risk assessment if a subsequent iteration), estimate the size and cost of the re-engineering project.

The estimator must quantify the effort (person/hours) and duration (calendar days) for the project's process activities, identify associated costs and state the rationale behind the calculations. The estimator must also state assumptions and areas of suspected risk to highlight any areas of limited understanding related to the requirements, product design or development process.

This is described in the subsections 'Estimating Size' and 'Estimating Costs'

NOTE - this task is performed iteratively with risk assessment

Pre Conditions

CQRL - A list of assessed system characteristics quality ratings

PCS - Proposed candidate evolution strategy

BG - Business goals

(RA - Risk Assessment if a subsequent iteration)

Post Conditions

ER - Estimate Report which should itemise all calculations, assumptions, areas of risk and built in contingency measures.

CR - Costing report for the re-engineering project broken down into logical areas of cost. E.g. hardware, project management overheads

MP - A Micro-Plan for the project giving details for the basis of the cost estimate

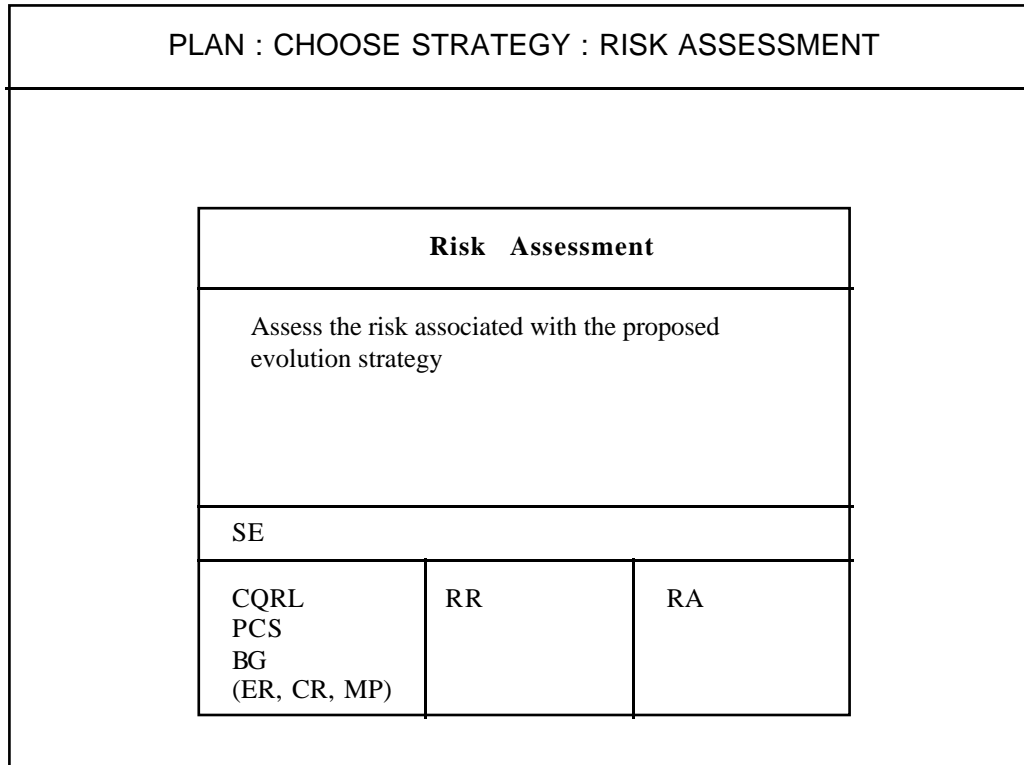
Enabling conditions

CR - There is a requirement to produce costing documents.

Roles involved:

SE - Service to prepare estimate

2.4.15 Risk Assessment



Risk Assessment

Reference: Evolution Management Report - 'Risk Management'

This task is referenced on page 37.

Task description

Assess the risk associated with the proposed evolution strategy and suggest ways of addressing risky project areas.

Risk assessment is a part of the process of risk management which reviews project status and investigates positive action to risks identified as threats.

Risk Assessment is the process which enables qualified statements to be made about the risks pertaining to a project. These will initially have been identified in the estimate.

NOTE - this task is performed iteratively with size and cost estimation

Pre Conditions

CQRL - A list of assessed system characteristics quality ratings

PCS - Proposed candidate evolution strategy

BG - Business goals
(ER - Estimate Report)
(CR - Costing report.)
(MP - A Micro-Plan.)

Post Conditions

RA - The Risk Assessment report

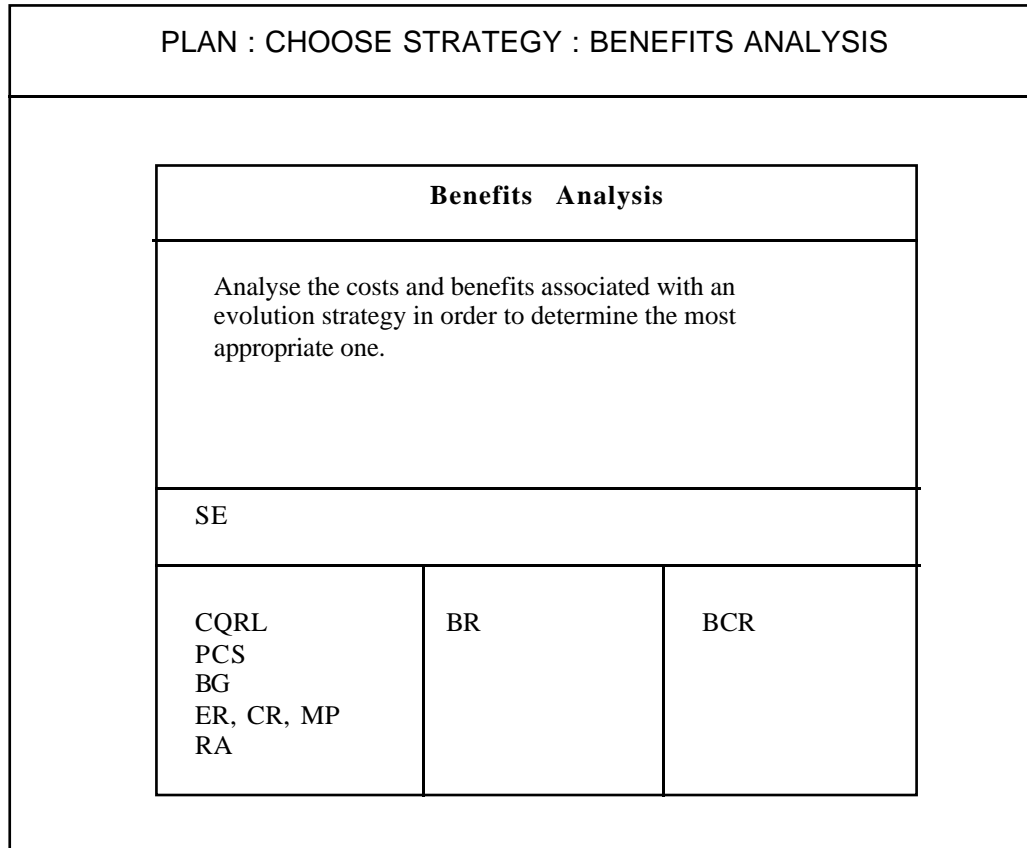
Enabling conditions

RR - There is a requirement for Risk assessment documentation.

Roles involved:

SE -Service to provide risk assessment

2.4.16 Benefits Analysis



Benefits Analysis

Reference: Evolution Management Report - 'Estimating'

This task is referenced on page 37.

Task description

Analyse the costs and benefits associated with an evolution strategy in order to determine the most appropriate one.

This involves computing the Present Value of cost and benefit for each strategy using a variety of economic indicators, projecting these into the future and comparing them with equivalent values for other strategies.

The process is described in the subsection 'Cost Benefits Analysis'.

Pre Conditions

CQRL - A list of assessed system characteristics quality ratings

PCS - Proposed candidate evolution strategy

BG - Business goals

ER - Estimate Report.

CR - Costing report.

MP - A Micro-Plan.

RA - The Risk Assessment report

Post Conditions

BCR - Benefits and Concerns report

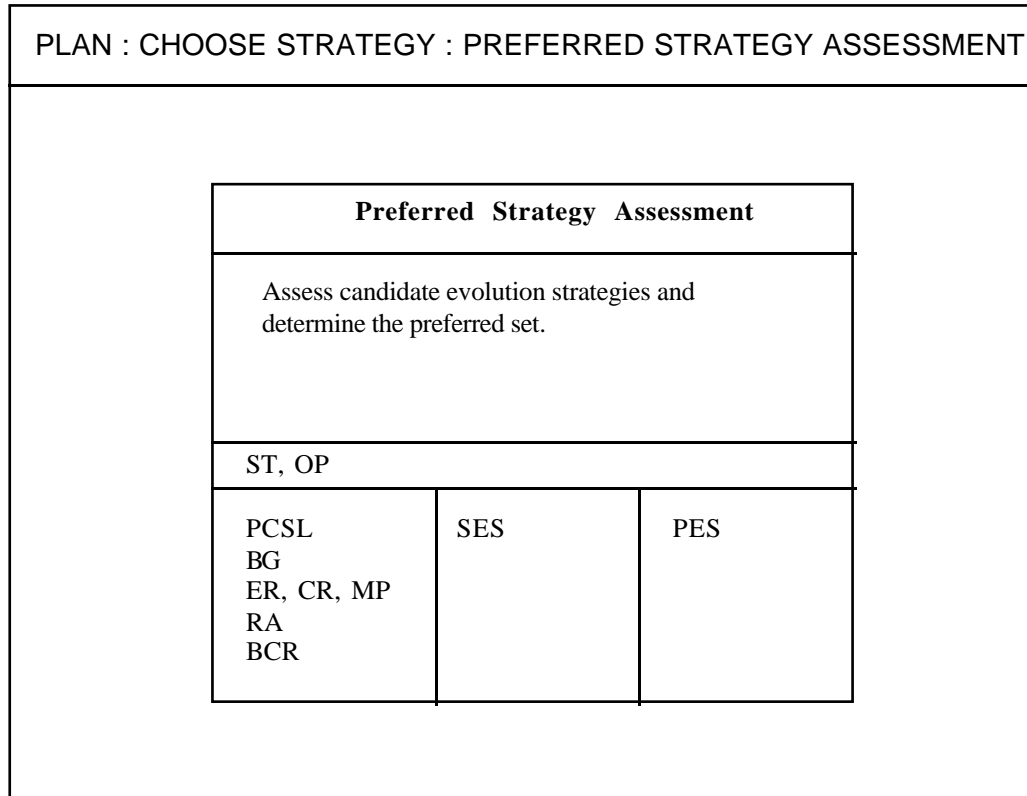
Enabling conditions

BR - Benefits analysis required

Roles involved

SE - Service to provide the computations

2.4.17 Preferred Strategy Assessment



Preferred Strategy Assessment

Reference: None

This task is referenced on page 37.

Task description

Analyse costs, risks and benefits associated with the list of proposed evolution strategies in order to determine the preferred ones.

Pre Conditions

PCSL - Proposed candidate evolution strategy list

BG - Business goals

ER - Estimate Report.

CR - Costing report.

MP - A Micro-Plan.

RA - The Risk Assessment report

BCR - Benefits and Concerns report

Post Conditions

Error! Style not defined.

PES - Preferred Evolution Strategy report

Enabling conditions

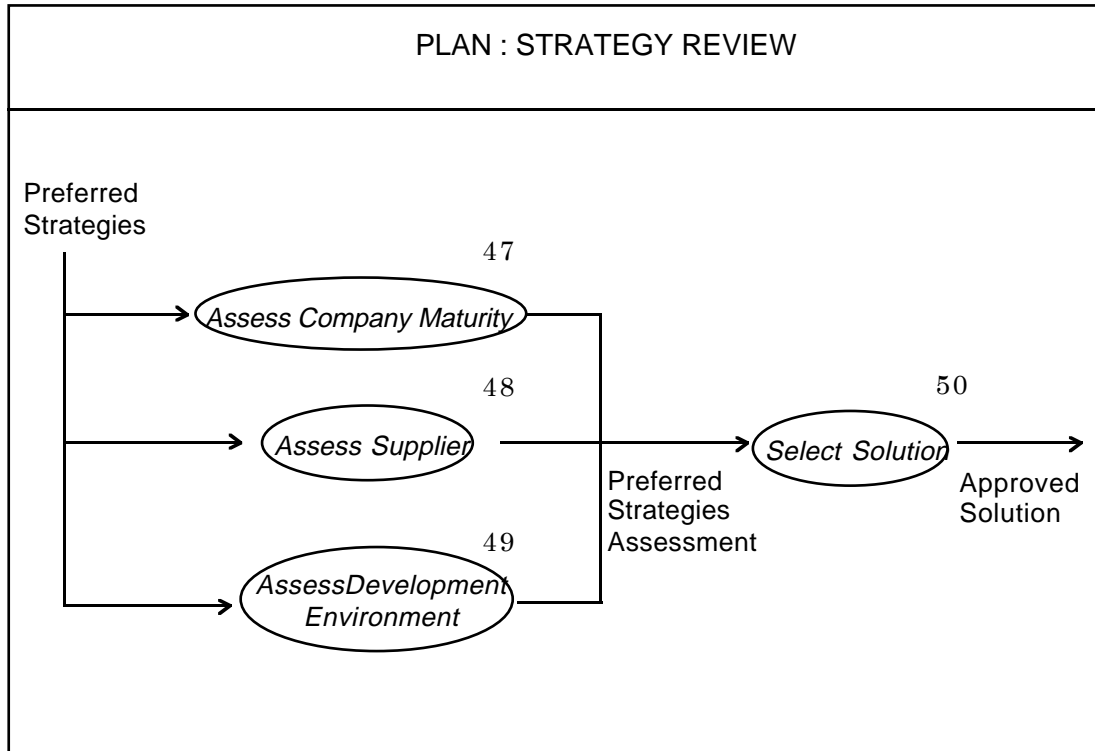
SES - As for "Select Assessment Characteristics"

Roles involved

ST - Strategic to choose on the basis of the business of the organisation

OP - Operational to provide technical support to the choice.

2.4.18 Strategy Review



Activity Description

Preferred strategies must be evaluated taking into account general issues, like:

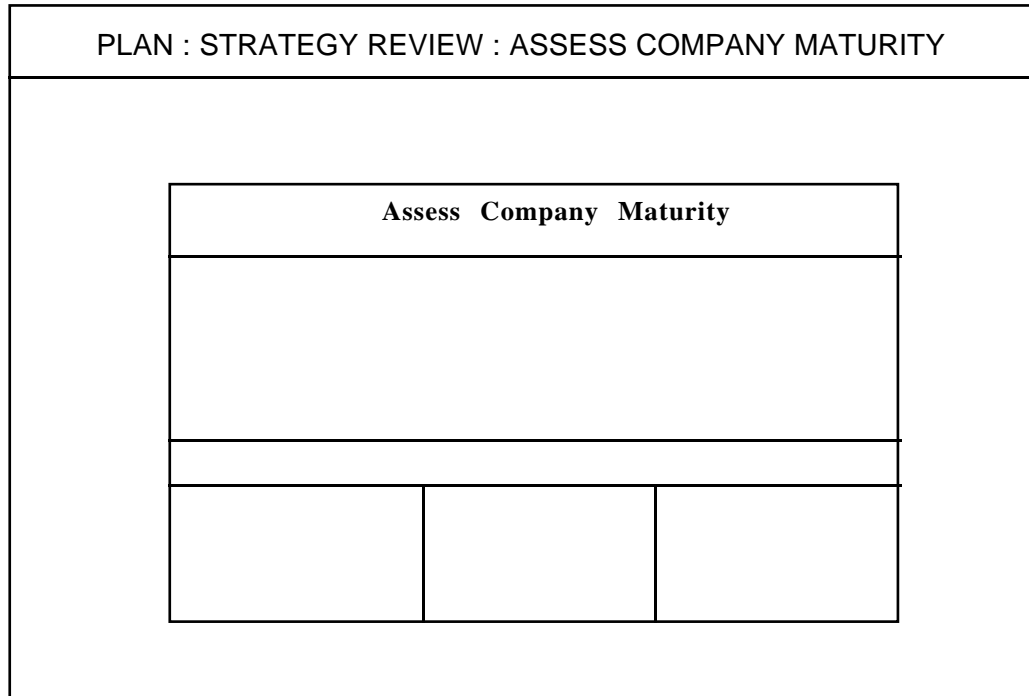
- the level of maturity of the company for applying each reengineering strategy;
- suppliers; and
- the development environment.

The result is the approved solution to be implemented.

All these issues will be detailed in the next version of the method, which will provide support to these assessment and selection activities.

This activity is referenced on page 36.

2.4.19 Assess Company Maturity



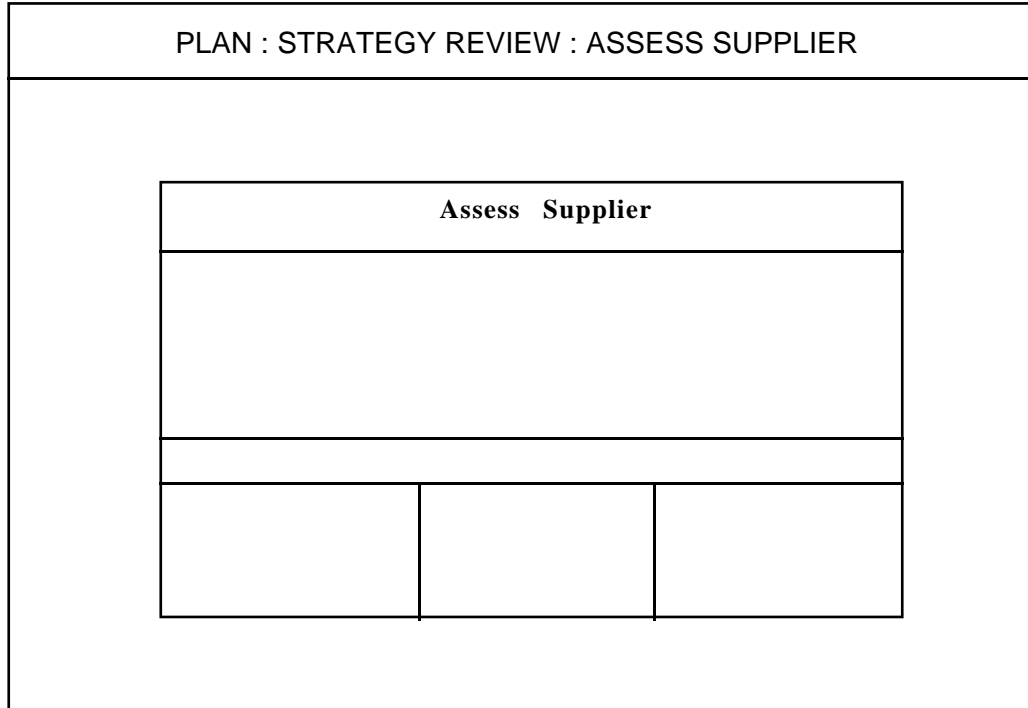
Assess Company Maturity

Reference: Evolution Management Report

This task is referenced on page 46.

Task Description

This task will be detailed in the next version of the method.

2.4.20 Assess Supplier**Assess Supplier**

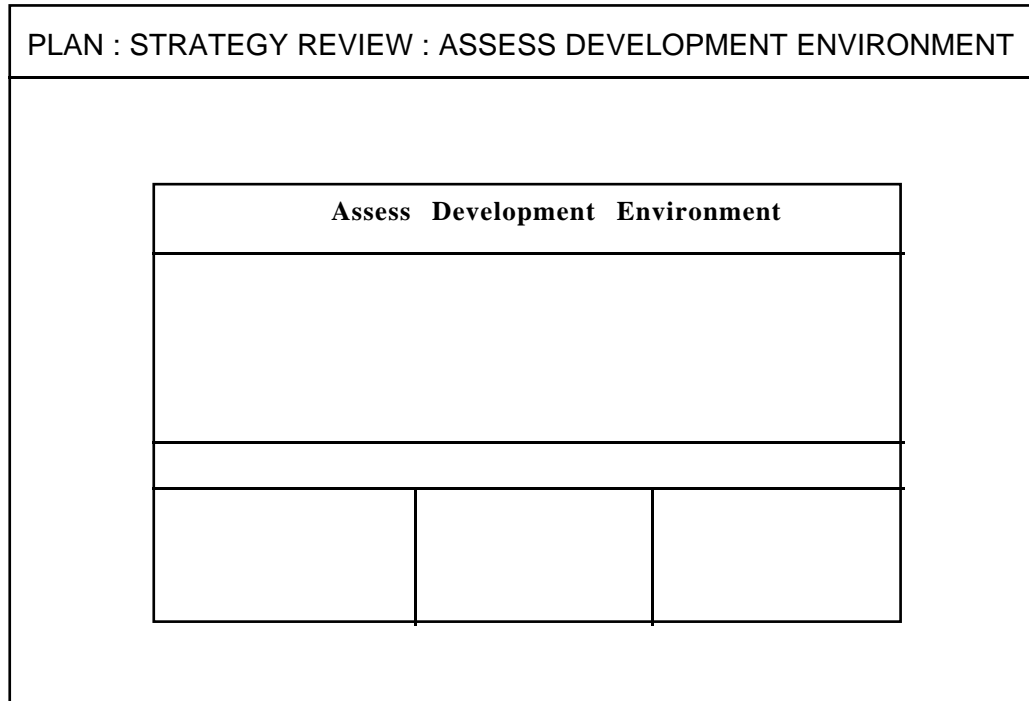
Reference: Evolution Management Report

This task is referenced on page 46.

Task Description

This task will be detailed in the next version of the method.

2.4.21 Assess Development Environment



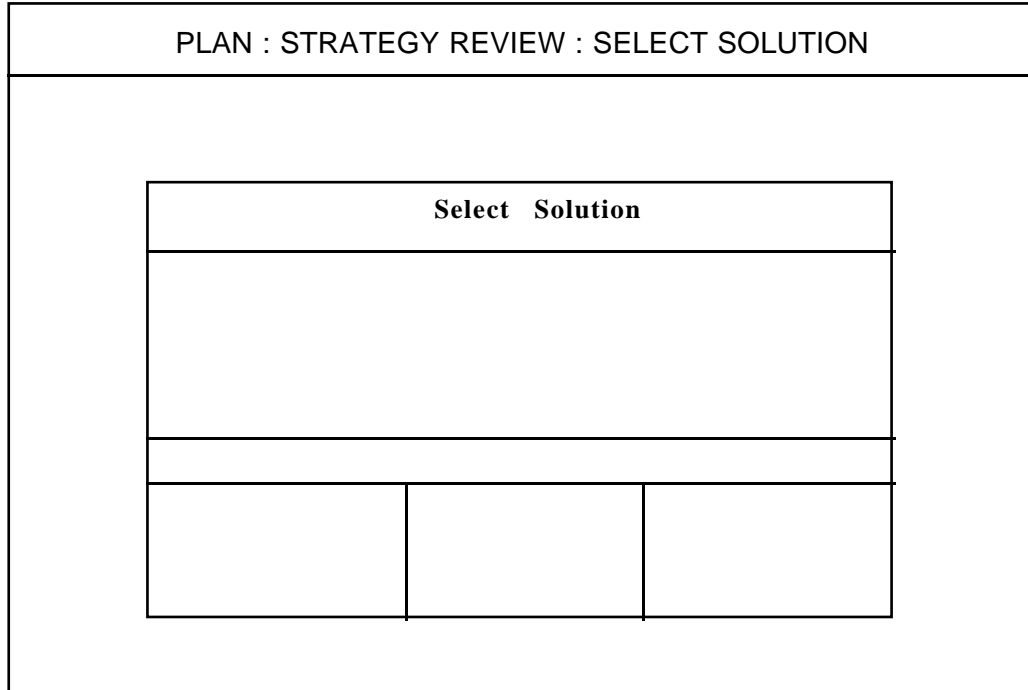
Assess Development Environment

Reference: Evolution Management Report

This task is referenced on page 46.

Task Description

This task will be detailed in the next version of the method.

2.4.22 Select Solution**Select Solution**

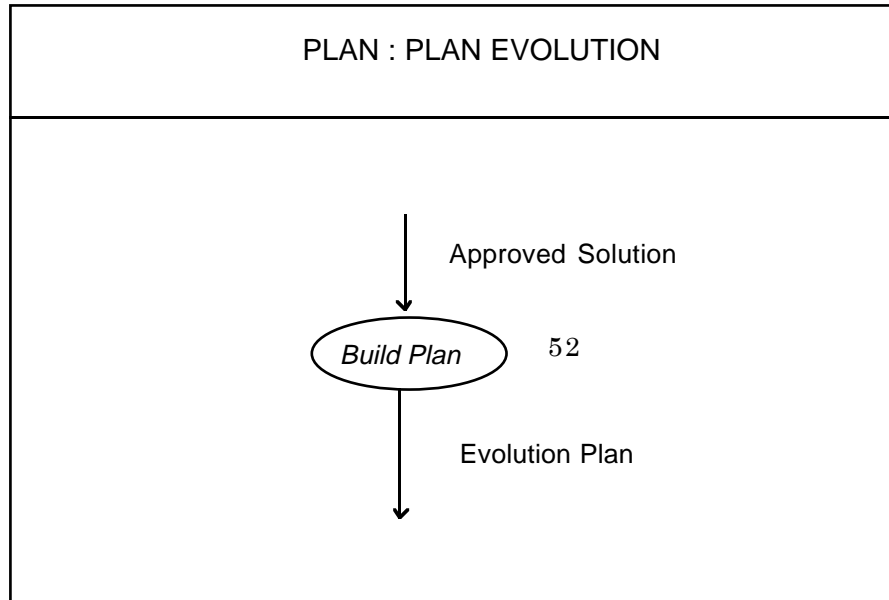
Reference: Evolution Management Report

This task is referenced on page 46.

Task Description

This task will be detailed in the next version of the method.

2.4.23 Plan Evolution

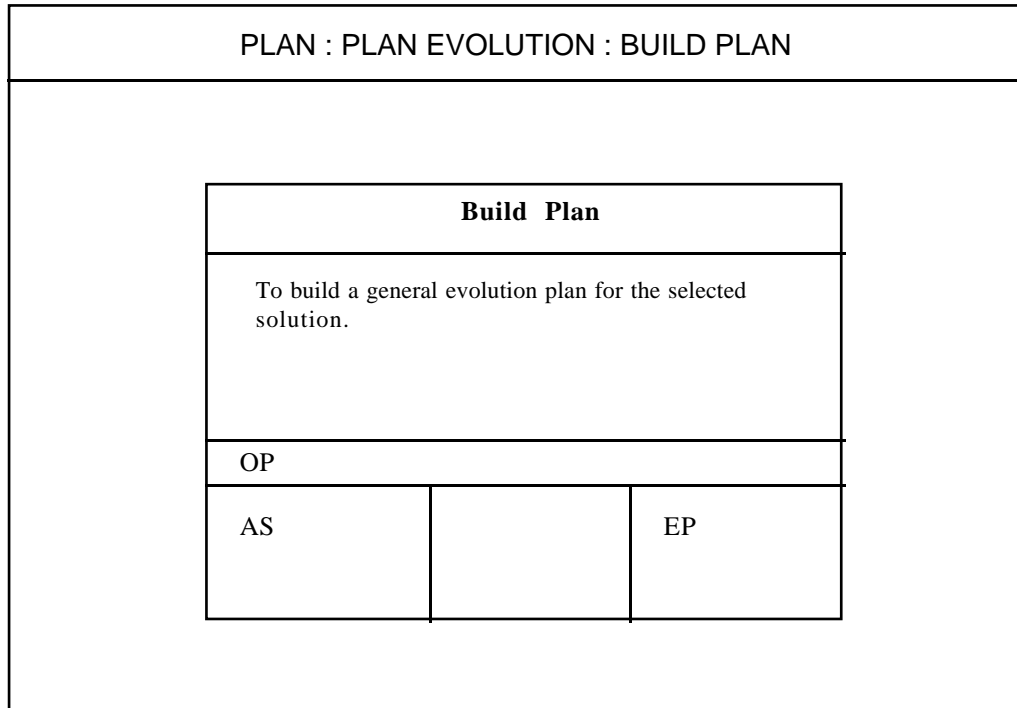


Activity Description

Planning the evolution of the current system means to build a detailed plan for leading the implementation of the approved solution. The objective of this activity is to provide service level with such an evolution plan.

This activity is referenced on page 36.

2.4.24 Build Plan



Build Plan

Reference: None

This task is referenced on page 51.

Task description

To build a general evolution plan for the selected strategy.

Identify main phases and tasks for the proposed evolution strategy.

Create a first schedule, showing the dependencies between the tasks.

Determine the needed resources (Persons, skills, commercial tools...).

Prepare training plan.

Calculate an initial budget.

Pre Conditions

AS - Approved Solution.

Post Conditions

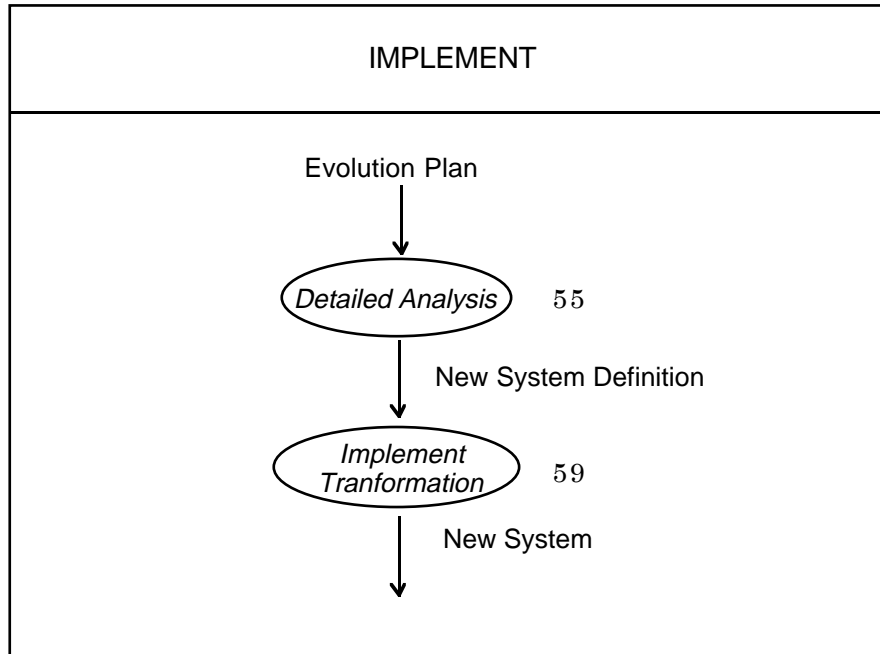
EP - Evolution Plan.

Enabling conditions

Roles involved

OP - Operational, to provide Project Management

2.4.25 Implement

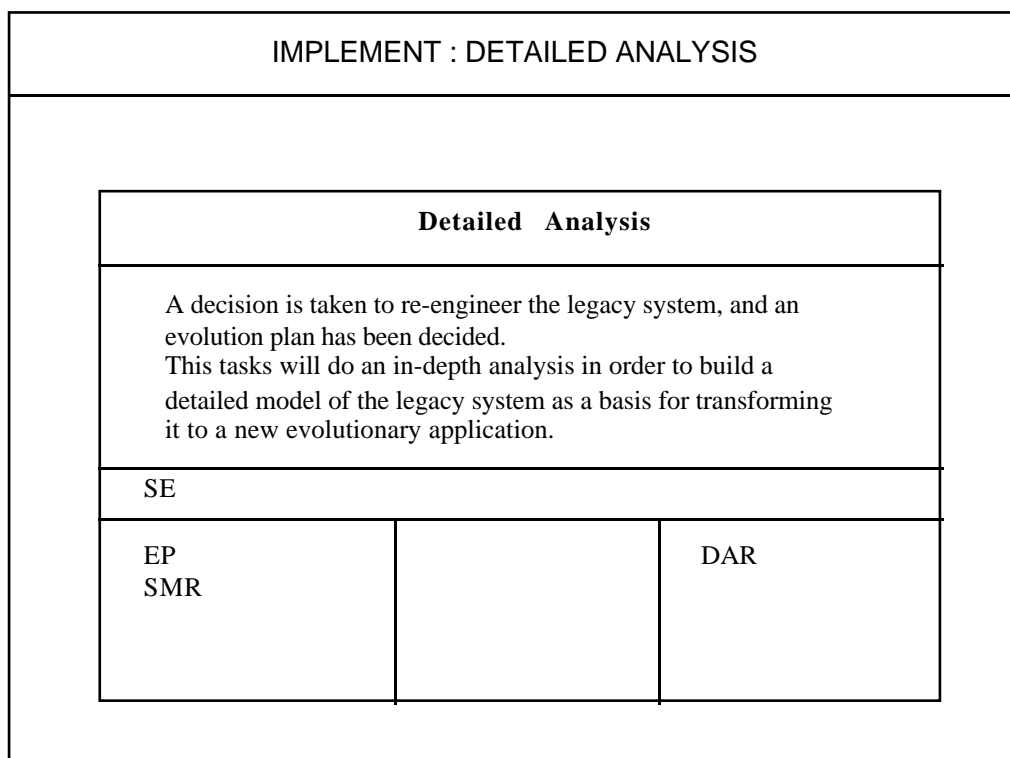


Activity Description

In order to implement the planned transformation, it is necessary to perform a detailed analysis of the existing system.

This activity is referenced on page 16.

2.4.26 Detailed Analysis



Detailed Analysis

Reference: Evolution Strategies report

Reference: Architectural Models for Evolution - Chapter 3

This task is referenced on page 54.

Task description

Based, among other things, on the output of the task “System Modelling”, the re-engineering project has decided on a strategy for the evolution of the legacy system. The evolution plan determines what evolution strategy that is selected. Different evolution strategies can be selected for different parts of the legacy system. These strategies include:

- Continued maintenance
- Re-engineering
 - Re-vamping
 - Re-structuring
 - Re-architecturing
 - Re-design with reuse
- Replacing

The different strategies are described in the RENAISSANCE consultancy report “Evolution planning”.

The *Detailed Analysis* task will focus on describing the different components of the current system with respect to what strategies that are selected. E.g. if the evolution plan specifies that parts of the existing system should only be re-vamped (i.e. replace the user interface), only the user-system interaction needs to be analysed in detail. However, if some subsystem is selected to undergo a re-architecting exercise, a more detailed model of the system internal parts must be made.

Chapter 3 in the RENAISSANCE consultancy report “Architectural Modelling for Evolution” describes in detail the properties that should be modelled for legacy 3GL and 4GL systems.

The models produced during this task will typically be extensions of the models produced during the “System Modelling” task, with additional models using other techniques where appropriate.

The modelling techniques selected for the detailed analysis are based on the Unified Modelling Language (UML):

- Extended entity relationship diagrams
- Block diagrams
- Class and object diagrams
- Hardware and network diagrams
- Implementation diagrams
- Use case diagrams
- Data flow diagrams
- Sequence diagrams
- Collaboration diagrams
- State diagrams
- Activity diagrams

In the above-mentioned consultancy report, we describe how these techniques should be used to model the properties of 3GL and 4GL applications. These properties are listed in Table 3 and Table 4:

	COBOL	FORTRAN	C/C++	Script
Source File	√	√	√	√
Executable Program File	√	√	√	
Object File			√	
Routine/Function	√	√	√	√
Variable	√	√	√	√
Data Type Declaration			√	
Class			√	
Method			√	
Attribute			√	
Object			√	

Logical File	√			
Physical File	√	√		

Table 3 Summary of 3GL properties

Property	SQL Windows	unifAce	Power Builder	RDBMS
Subsystems	√	√	√	√
Files	√	√	√	√
File dependencies	√	√	√	√
Format descriptions	√	√	√	
External functions	√	√	√	
Constants	√	√	√	
Resource definitions	√	√	√	
Variable definitions	√	√	√	
Functions	√	√	√	
Messages	√	√	√	
Named menus	√	√	√	
Classes	√	√	√	
Objects	√	√	√	
User Interface	√	√	√	
Database interface	√		√	
Internal data model		√		
Extended Attributes		√	√	
Tables				√
Views				√
Indexes				√
Triggers				√
Stored procedures				√
Domains				√

Table 4 Summary of 4GL properties

Pre Conditions

The Pre Conditions of the detailed analysis task are

EP - The evolution plan which dictates the selected strategies for evolving the legacy system.

SMR - The output of the system modelling task which will be used as a starting point for the detailed analysis.

Enabling conditions

The task is mandatory.

Post Conditions

DAR - The activity is finished when the legacy system to be evolved is sufficiently described in the *new system definition*.

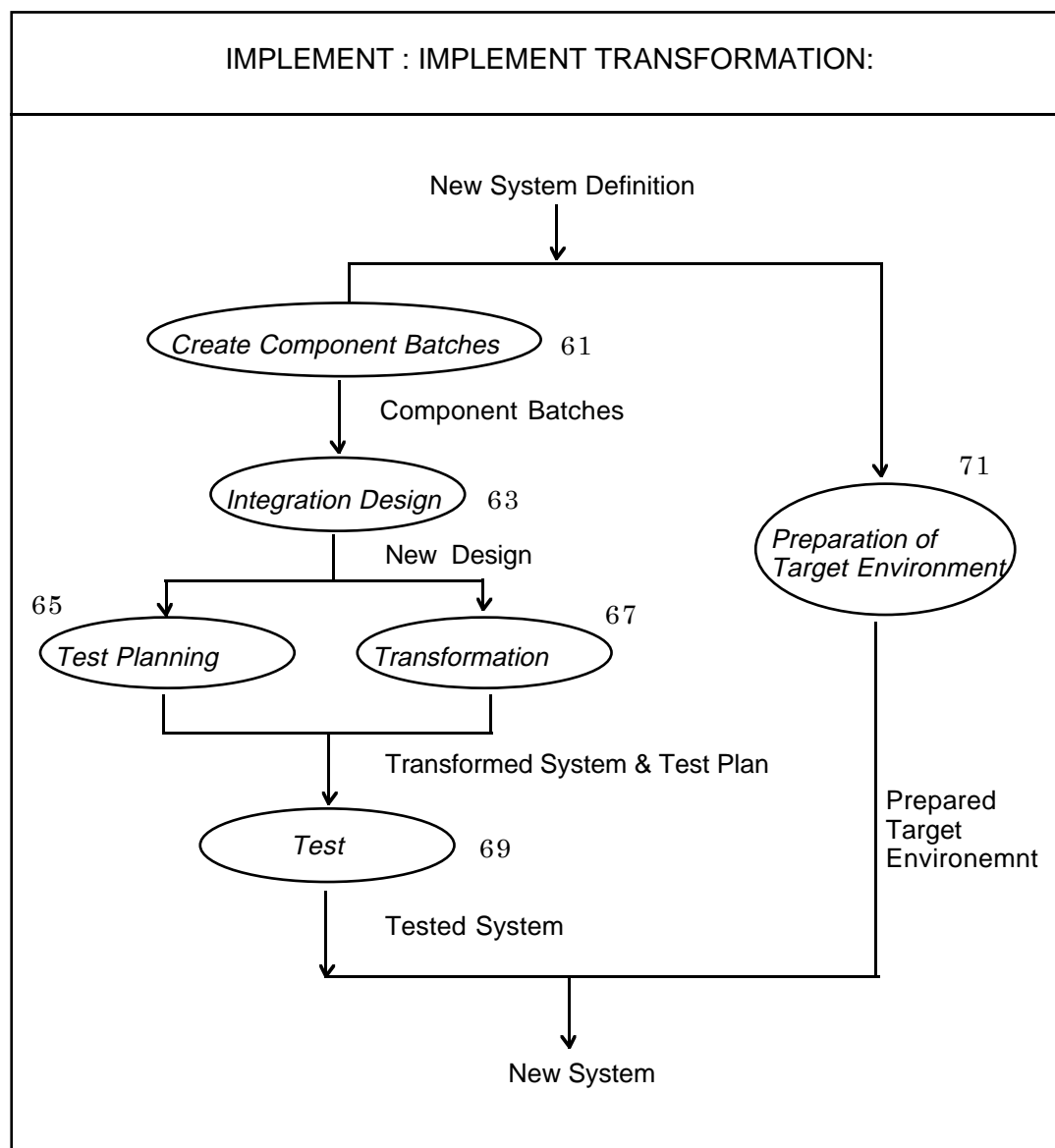
Roles involved

SE - Service roles have the necessary skill on the system.

In particular, main roles involved are:

- *Designers* which are familiar with the modelling techniques used for modelling the legacy system.
- *Application Experts* (if available), which will assist designers in making the detailed analysis of the legacy system.

2.4.27 Implement Transformation



Activity Description

The approach suggested by RENAISSANCE to implement the planned transformation is based on the notion of *interface*, and enforces an iterative migration toward the desired system.

The result of the Detailed Analysis phase is used to identify, via their interfaces, sub-components of the existing system, whose combined transformation allows the legacy system to migrate toward the target system.

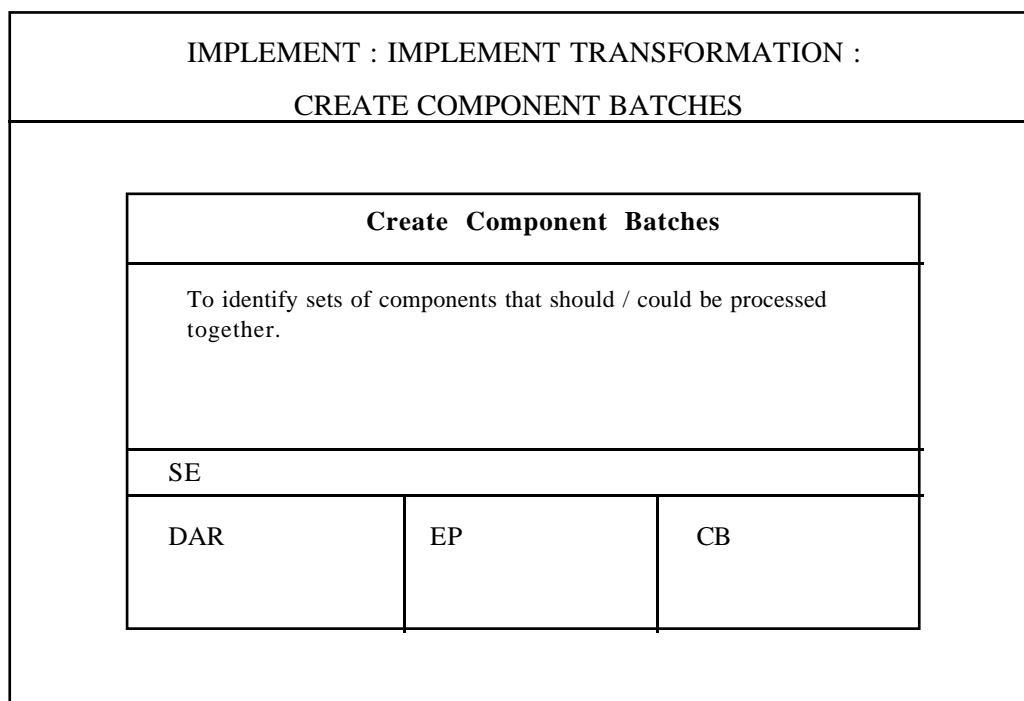
For each component, it is planned both the integration within the global system and the tests which are necessary to validate the new system. The migration is then implemented and the result is tested.

Error! Style not defined.

In parallel, it is necessary to prepare the new operational environment on which the new, tested system needs to run.

This activity is referenced on page 54.

2.4.28 Create Component Batches



Create Components Batches

Reference: None

This task is referenced on page 59.

Task description

To identify sets of components that should / could be migrated together.

Component batches allow to perform system evolution in an incremental way; components are grouped together, to facilitate the development, depending on i.e. their functionality, their level of coupling and so on.

The component batches will be the unit of development for the evolution project.

Pre Conditions

DAR - New system definition, as resulted from the Detailed Analysis activity.

Post Conditions

CB - Components Batches.

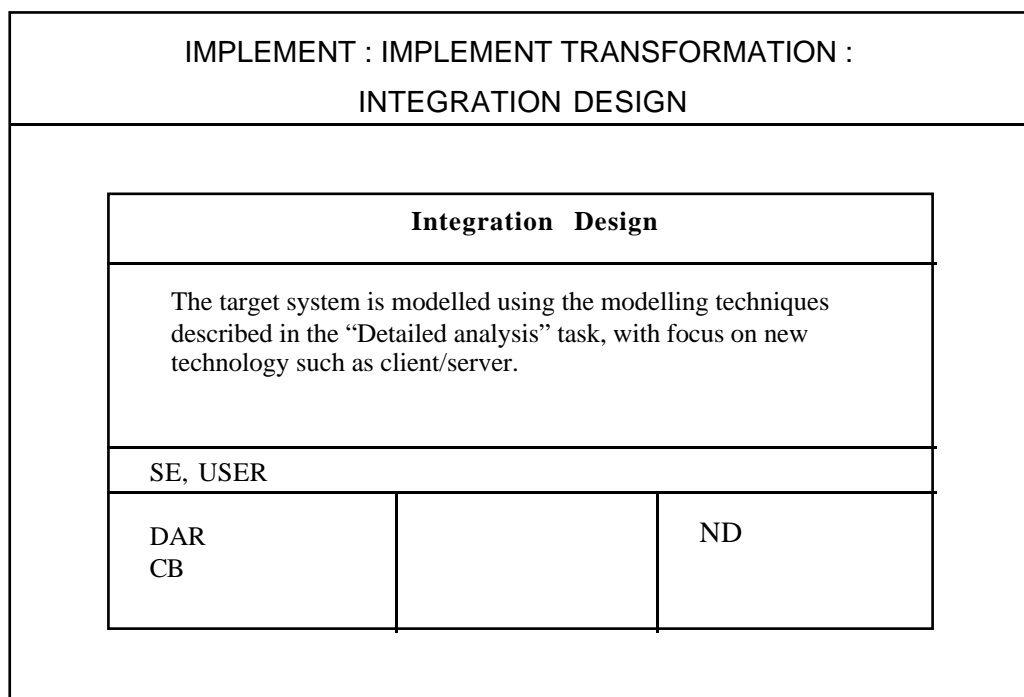
Enabling conditions

EP - The evolution plan, with information about the level of complexity of the evolution project.

Roles involved

SE - Service level, to identify batches.

2.4.29 Integration Design



Integration Design

Reference: Architecture Models for Evolution

Reference: Client/Server Migration

This task is referenced on page 59.

Task description

While the focus of the earlier modelling activities in the RENAISSANCE method (“System modelling”, “Detailed analysis”) has been on modelling and analysing the old legacy system, the activities undertaken in this task are committed to designing the new target system.

The modelling techniques used for designing the new system are those that were described in the “Detailed analysis” task. The technology chosen for the new system is determined by the evolution plan. The focus is however on using new architectures, moving towards a client/server solution.

The modelling guidelines are described in the RENAISSANCE consultancy report “Architectural Modelling for Evolution”, while new technology is described in the RENAISSANCE consultancy report “Client/Server Migration”.

Pre Conditions

The Pre Conditions of the task is the description of component batches (CB) and the new system definition (DAR), which comes from the Detailed Analysis activity:

- The component batches dictates which components in the legacy systems that should be re-engineered together, and therefore dictates the constraints of the new design.
- The new system definition provides the designer with information for understanding the current functionality of the system, and how to reuse parts of the existing system. As described earlier, it is the evolution plan which dictates which evolution strategy should be used on the different parts of the legacy system.

Enabling conditions

If decisions are made for re-engineering an existing legacy system, indeed this task is mandatory. The task documents the design of the new system, and hence ensures that knowledge is made explicit, ensuring that the new system will not become a “future legacy system”.

Post Conditions

The new design (ND) is documented by the diagrams outlined in the description of the “detailed analysis” task contained in the DAR, and additional reports.

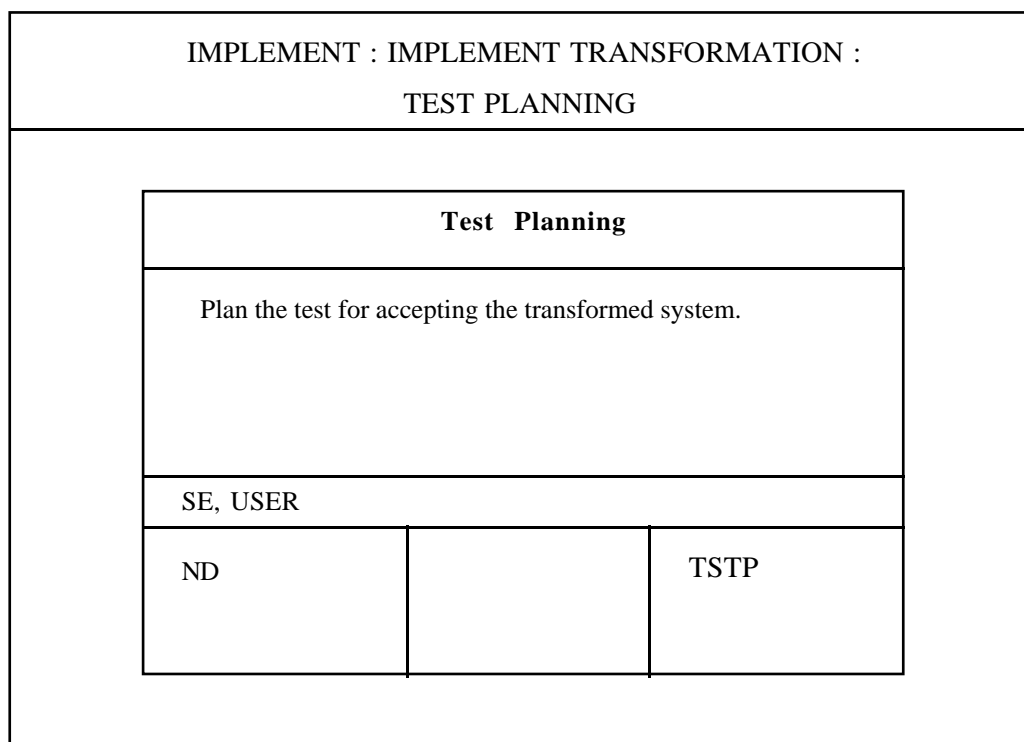
Roles involved

SE - Service roles provide technology knowledge

USER - Users should be strongly involved in the definition process

In particular, a *designer* is assisted by *domain experts* and *users* in order to make correct design decisions for the application target system.

2.4.30 Test Planning



Test Planning

Reference: None

This task is referenced on page 59.

Task description

On the basis of the detailed analysis result and new system information of the ND input, a test plan for evaluating the new implementation is prepared.

Pre Conditions

ND - The new design, which is the output of the *Integration Design* activity.

Post Conditions

TSTP - Test Plan

Enabling conditions

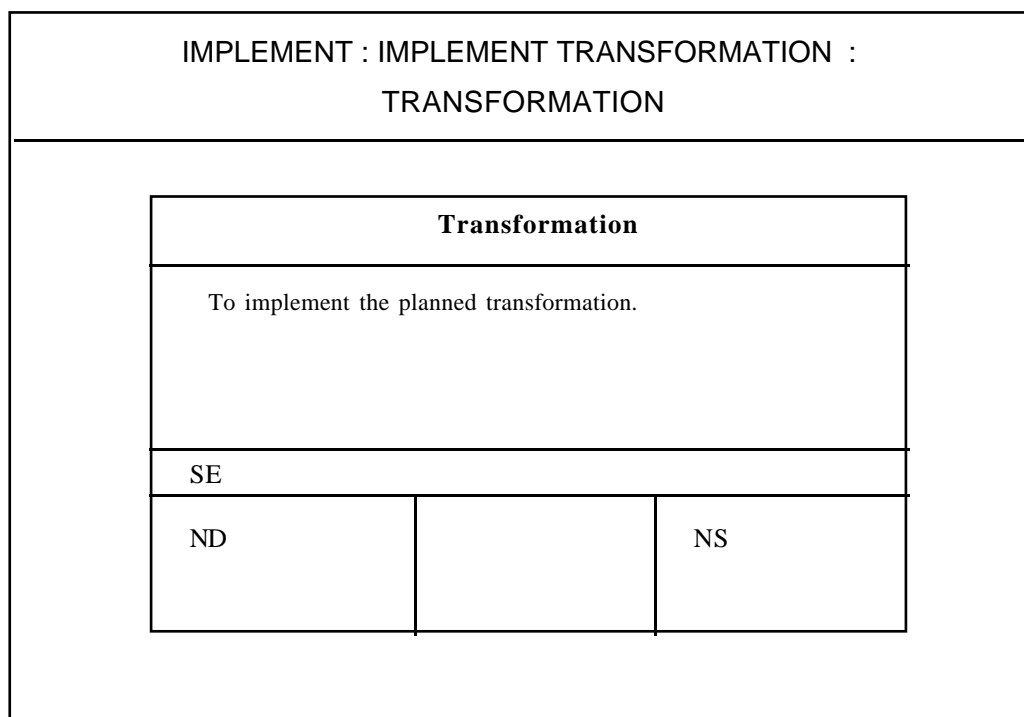
This task is mandatory.

Roles involved

SE - Service roles are necessary to plan the test

USER - User could be involved in planning tests

2.4.31 Transformation



Transformation

Reference: Client/Server Migration

This task is referenced on page 59.

Task description

- To implement the planned transformation.
- To establish development environment.
- To produce code.
- To build the modification to existing system, to ensure correct system interfacing.
- To conduct unit testing.
- To conduct integration testing.

Pre Conditions

ND - New Design.

Post Conditions

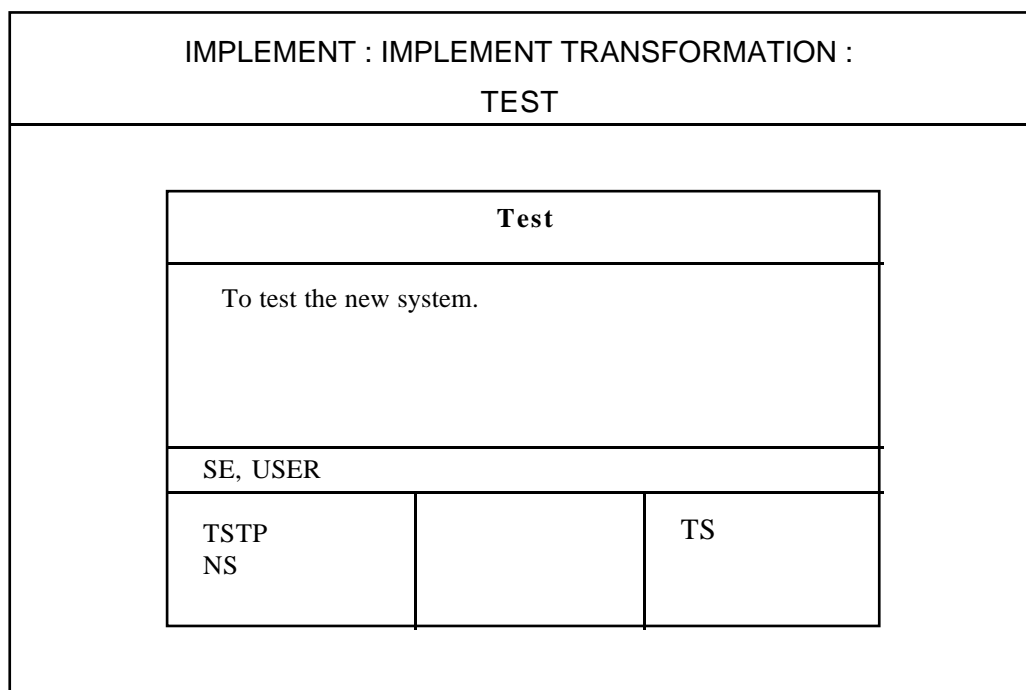
NS - Transformed, or new, System.

Enabling conditions

Roles involved

SE - Service level, to develop the transformation

2.4.32 Test



Test Planning

Reference: None

This task is referenced on page 59.

Task description

The transformed system must be tested on the defined test plan.

Pre Conditions

TSTP - The test plan

NS - The new system

Post Conditions

TS - The tested system

We call NS (New System) the couple TS (Tested System) and ENVP (Prepared Target Environment).

Enabling conditions

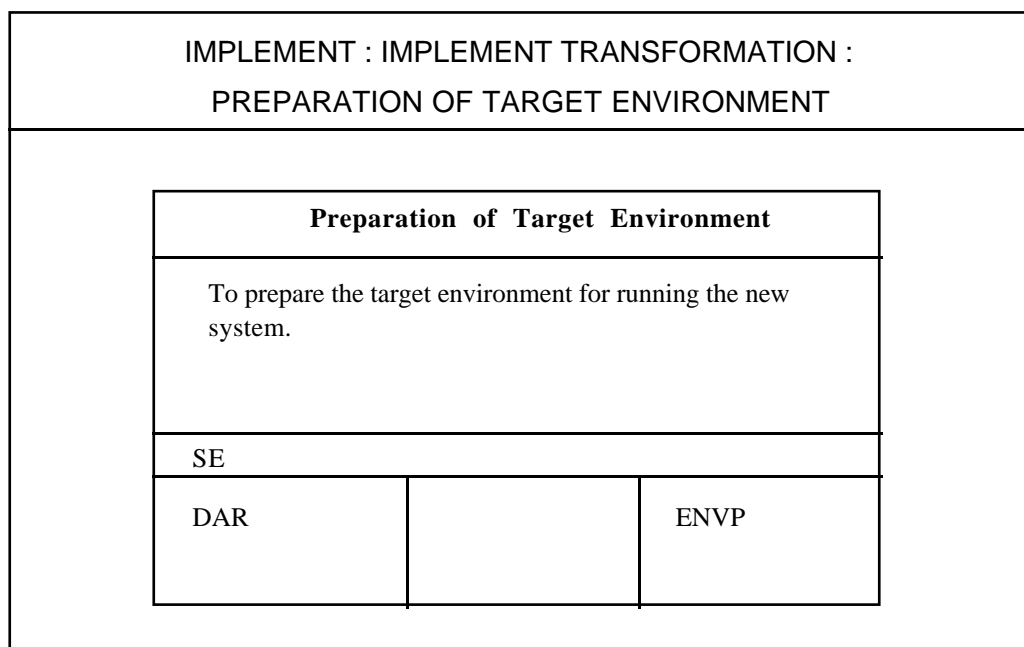
This task is mandatory.

Roles involved

SE - Service roles are necessary to perform the test

USER - User could be involved in performing tests

2.4.33 Preparation of Target Environment



Target Environment Preparation

Reference: None

This task is referenced on page 59.

Task description

Depending on the kind of transformation done, the target environment has to be prepared, in order to allow the system to run. For example, COTSs or new operating systems or database servers, and so on, have to be installed.

Pre Conditions

DAR - The Detailed Analysis Result

Post Conditions

ENVP - Target environment, prepared for running the transformed system.

We call NS (New System) the couple TS (Tested System) and ENVP (Prepared Target Environment).

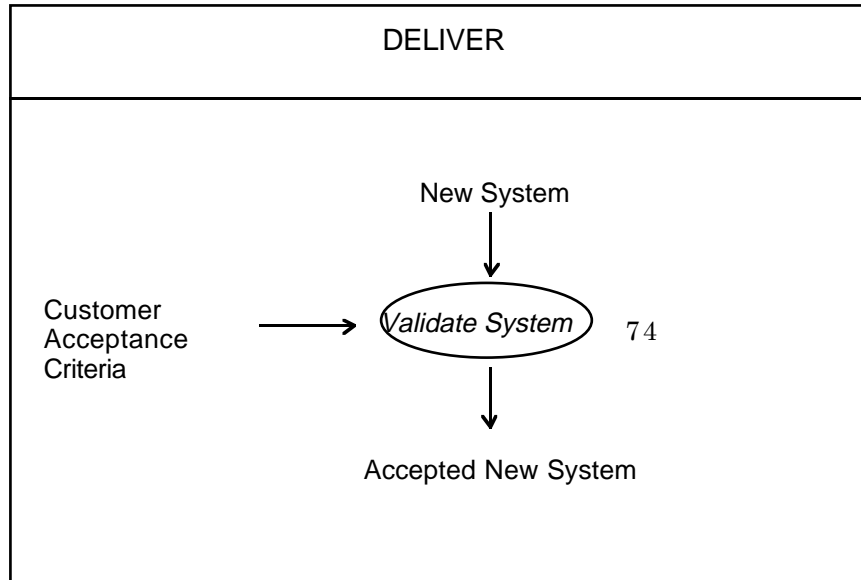
Enabling conditions

This task is mandatory.

Roles involved

SE - Service roles are necessary to prepare the environment

2.4.34 Deliver

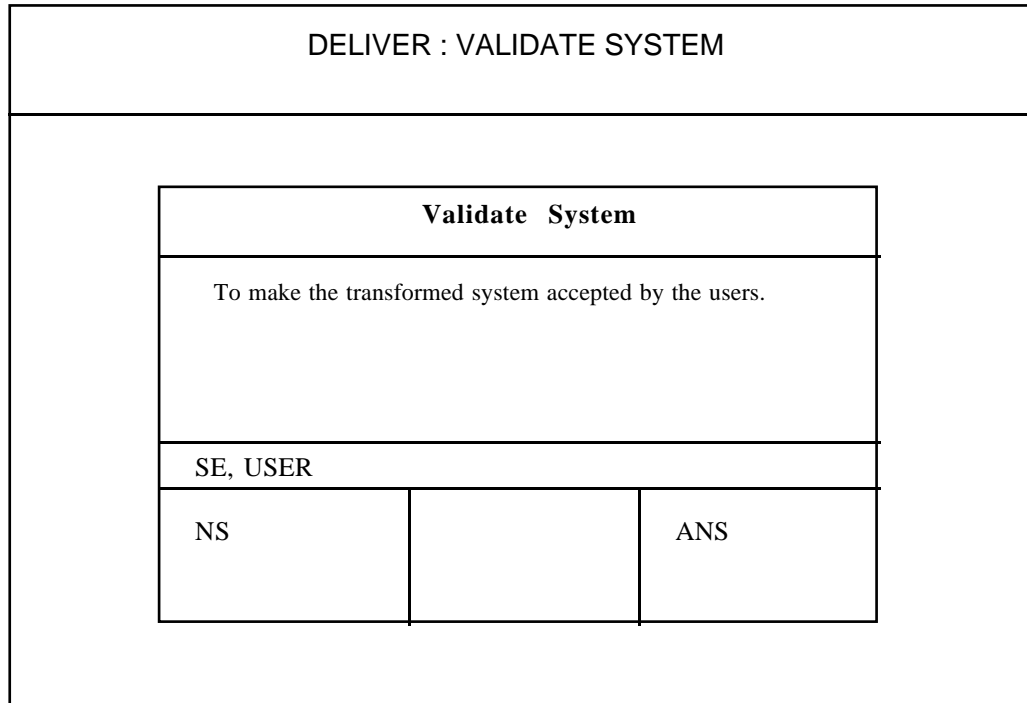


Activity Description

Acceptance criteria must be identified in order to validate the new system, and make it accepted by final users. Users should be involved in this activity both to plan the criteria and to validate the system.

This activity is referenced on page 16.

2.4.35 Validate System



Validate System

Reference: None

This task is referenced on page 73.

Task description

- To make the transformed system accepted by the users.
- To install the system
- To run internal validation.
- To conduct acceptance test.

Pre Conditions

NS - The New System.

Post Conditions

ANS - Accepted New System.

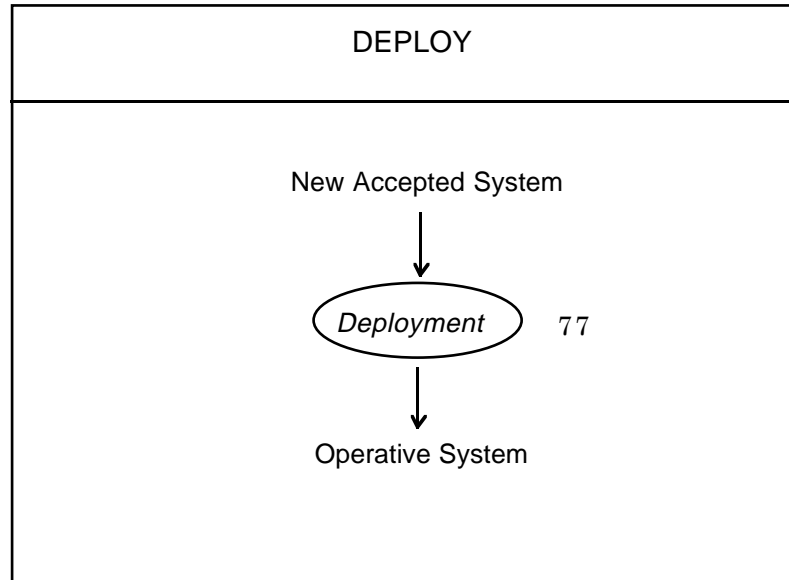
Enabling conditions

Roles involved

SE - Service level

USER - To contribute to the validation activity

2.4.36 Deploy

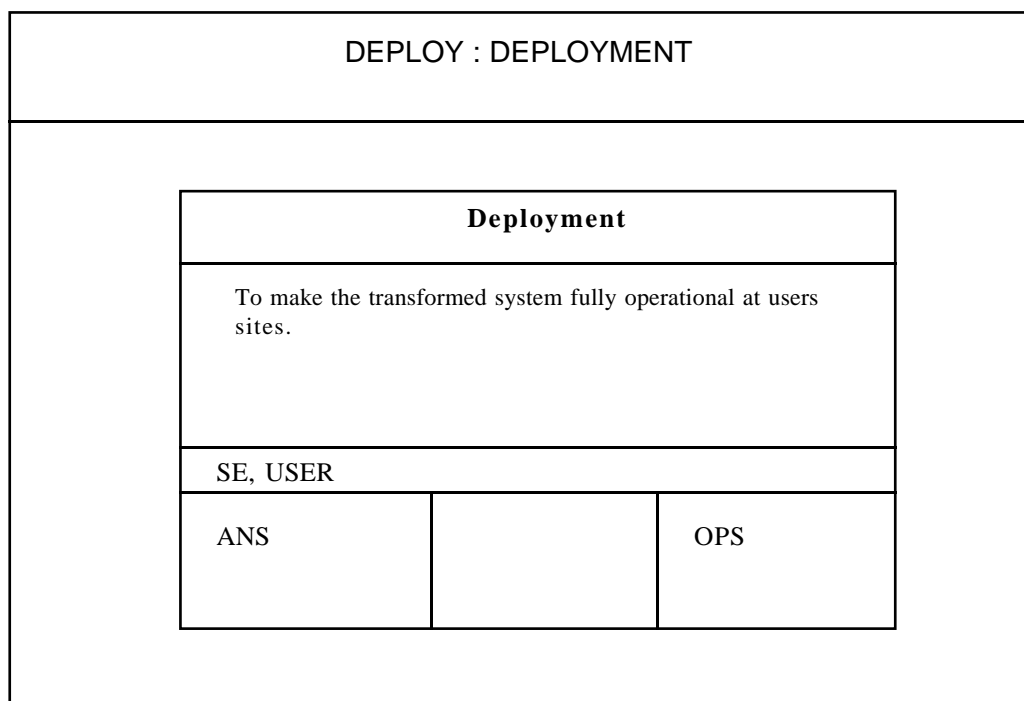


Activity Description

The new system, accepted by final users, must be deployed to the target operational environment. The deployment usually needs to install new COTSs and hardware, to train users in using the new system, and so on.

This activity is referenced on page 16.

2.4.37 Deployment



Deployment

Reference: None

This task is referenced on page 76.

Task description

To make the transformed system fully operational at user's sites:

- Prepare system operation.
- Prepare user support.
- Run Users training.
- Transfer and install the system at users sites.
- Run internal validation.

Pre Conditions

ANS - The Accepted New System.

Post Conditions

OPS - The Operative System.

Enabling conditions

Roles involved

SE - Service level, in particular the so-called *User Support Team*

USER - Users are involved in the deployment activity

2.5 References to RENAISSANCE Reports

Task descriptions contain references to other RENAISSANCE reports, which provide detailed guidelines and suggestions about the way such tasks should be executed.

The following table summarizes references from method activities to RENAISSANCE reports, while references to specific sections of those reports can be found in the corresponding task description.

Method Activity	Page	RENAISSANCE Report
Assess As-Is	19	Architecture Models for Evolution Evolution Strategies
Assess To-Be	31	Client - Server Migration Architecture Models for Evolution Evolution Strategies
System Modelling	21	Architecture Models for Evolution
Select Assessment Characteristics	23	Evolution Strategies
Assess Characteristic #i	26	Evolution Strategies
Global Assessment	28	Evolution Strategies
Size and Cost Estimation	38	Evolution Strategies
Risk Assessments	40	Evolution Strategies
Benefits Analysis	42	Evolution Strategies
Assess Company Maturity	47	Evolution Strategies
Assess Supplier	48	Evolution Strategies
Assess Development Environment	49	Evolution Strategies
Select Solution	50	Evolution Strategies
Detailed Analysis	55	Architecture Models for Evolution
Integration Design	63	Client-Server Migration Architecture Models for Evolution
Transformation	67	Client-Server Migration

3 Framework - Method Traceability

Contents

- 2.1 Introduction
- 2.2 Traceability Description

Summary

This chapter describes the traceability from the RENAISSANCE Method described in this report and the RENAISSANCE Framework as described in the corresponding report. Familiarity with the framework is mandatory for understanding the content. Readers who are not interested in pointing out such a traceability can skip this chapter, with no impact at all on the understanding and mastering of the RENAISSANCE method.

3.1 Introduction

This chapter provides traceability from the RENAISSANCE framework, as defined in the corresponding report, and the RENAISSANCE method, as defined in the previous chapter.

The RENAISSANCE method must be seen as an instance of the framework. This means that a traceability from the framework to the method can be shown. Next section describes such a traceability.

For a proper understanding of the rationale underlying the definition of the RENAISSANCE method, the reading of the RENAISSANCE framework report is strongly recommended, even if the reading of such a report is not necessary neither for understanding nor for applying the method. The knowledge of the framework is indeed mandatory in order to understand this chapter. In fact, concepts and naming from the framework will be used throughout the remaining of this chapter, without re-introducing them: we assume that readers are already familiar with the framework itself.

Readers who are not interested in pointing out the traceability from the framework to the method can skip this chapter, with no impact at all on the understanding and mastering of the RENAISSANCE method.

3.2 Traceability Description

The RENAISSANCE method is an instance of the RENAISSANCE framework. This means that the activities of the framework can be mapped in activities of the method. Such method activities specialise the general activities described in the framework. Naming can be different, owing to the specialisation of the activities.

The traceability from the framework to the method is represented in the following three tables and detailed in the remainder of this section: the first table concerns the top level of the framework; the second one refers to the definition of such top-level activities; the third one shows the logical traceability.

Method Activity	Framework Activity
Investigate	Trade-Off Analysis
	Issue Assessment
Plan	Decision Analysis
Implement	Solution Implementation
Deliver	Solution Deployment
Deploy	

Table 5 : Top-Level Traceability

Method Activity	Framework Activity
Assess As-Is	As-Is Analysis
Review Business Goals	Environment Assessment Application Assessment
Assess To-Be	Can-Be Analysis Transition Analysis Strategies Assessment
Choose Strategy	Cost/Benefit/Risk Analysis
Strategy Review	Decision Making
Plan Evolution	Planning
Detailed Analysis	Detailed Analysis
Implement Transformation	Metrics and Testbed Definition System Re-Implementation
Preparation of Target Environment	Environment Preparation
Validate System	Evaluation and Acceptance
Deployment	System Deployment

Table 6 : Activities Traceability

Method Activity	Framework Activity
Investigate Plan	What To Do
Implement Deliver Deploy	How To Do

Table 7 : Logical Traceability

The mapping from framework activities to method activities is a one-to-one mapping, with only three exceptions, which are explained and justified in the following:

- the *Investigate* activity implements both the *Trade-Off Analysis* and the *Issue Assessment* phases of the framework;
- *Deliver* and *Deploy* are both covered by *Solution Deployment* in the Framework;

-
- the *Implement Transformation* activity implements both *Metrics and Testbed Definition* and *System Re-Implementation*;

All the exceptions depends on choices made in developing the method.

Concerning the *Investigate* exception, it is true that a technology trade-off is largely independent from the quality of the system, but it is true as well that the analysis needed to evaluate possible trade-offs involves also technical evaluation of the system itself. Hence, even if from the abstract point of view of the framework it is right to see *Trade-Off Analysis* and *Issue Assessment*, which involves the technical assessment of the existing legacy system, as distinct phases, it is more practical and comprising, from the operative point of view, to perform only one assessment, one of whose results is the suggested technology trade-off. This is the reason for fusing the two framework activities into one activity of the operational method.

A similar reason holds for splitting *Solution Deployment* into two distinct activities: *Deliver* and *Deploy*. From an abstract point of view, there is no real distinction between a new system and its delivery, but from an operative point of view, there is a big difference: the delivery is not integrated in the development environment. This means that the acceptance test must be done on the delivered version of the system, and not on the development one. This is the reason for having two different activities in the method, the first concerning delivering the system and the second one concerning deploying the delivered system.

The third exception concerns the re-implementation of the system. The framework does not suggest any re-implementation strategy, and simply states that the system must be re-implemented. Indeed the method specialises this generic activity, and suggests to use a specific construction approach for re-implementing the system, based on the concept of subsystem interface. The proposed approach, which could be changed by the user when instantiating the method for its company and project needs, suggests to perform testbed construction in parallel with the implementation, and with the preparation of the target environment as well. This is why the method groups these activities.

Concerning *role categories*, there is a direct mapping between the categories identified in the framework and the categories used within the method.

Other two issues need to be traced:

- the *System Repository*, and
- the implementation of the *Kaizen*, or continuous improvement, activity.

Concerning the *System Repository* concept, it is maintained and implemented by the System Repository described in the method as a

repository for all the artefacts needed for applying the method, that is both produced and consumed by activities.

Concerning *Kaizen*, there are no explicit activities which supports it within the method, because all activities should contain elements for implementing it. It is supported as described below:

- the RENAISSANCE method supports continuous customization (with reuse of the System Repository), which can be used to best tune the method for a given project within a given organization; this implements the Kaizen of the method itself;
- the continuous improvement of the reengineered system is realised applying the method (or only the implementation phase in case of forward engineering) more than once; this is the Kaizen of the reengineered system.

4 Bibliography

- [ALAN 95] Alan E. Giles and Dennis Barney, "Metrics Tools: Software Cost Estimation", STSC Crosstalk, June 1995.
- [AMES 94] AMES Consortium, AMES ESPRIT project #8156, "AMES Methodology and Process Model", 1994
- [ARN 93] Arnold, R. S., "Software Reengineering", IEEE Computer Society Press, CA, 1993
- [BENN 95] Bennet, K., "Legacy Systems: coping with success", IEEE Software, January 1995
- [BERG 96] Bergey, J. K., S. R. Tilley, et al. (1996). An Enterprise Perspective of Reengineering, Software Engineering Institute, Carnegie Mellon University.
- [BOEH 81] Boehm Barry, "Software Engineering Economics", Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981.
- [BUSH 90] Bush, E., "A CASE for existing system", Language Technology White Paper, Salem, MA, 1990
- [CAPE 96] Capers Jones, "Activity-based software costing", Software Challenges Computer, Vol. 29, No. 5, May 1996.
- [COU 94] Coulouris, George, et. al., "Distributed Systems - Concepts and Design 2nd ed.", Addison-Wesley, 1994.
- [CSC 96] Computer Science Corporation, "Legacy Asset Management: Setting a Course from the Past to the Future", Technical Report, May 1996
- [DOD 97] "Software Reengineering Handbook: Version 3", US Department of Defense, March 1997

- [FENT 91] Fenton, N. E. (1991). Software Metrics, A Rigorous Approach, Chapman & Hall.
- [GAL 95] Gall, Harald C., et. al., "Architectural Transformation of Legacy Systems", 17th International Conference on Software Engineering, Seattle, Washington, U.S.A, April 1995.
- [GANTI 95] Ganti, N., Brayman, W., "The transition of Legacy Systems to a Distributed Architecture", John Wiley & Sons, 1995
- [GARM 96] Garmus, D. and D. Herron (1996). Measuring the Software Process - A Practical Guide to Function Points, Yourdon Press.
- [GRE 85] Green, M., "Report on Dialogue-Specification Tools", Springer-Verlag New York, 1985, pp. 9-20.
- [HAMM 86] Hammer, M., "Reengineering Work: Don't Automate, Obliterate", Harvard Business Review, 1990
- [HAMM 93] Hammer, M., Champy J., "Reengineering the Corporation", Harper Collins, New York, 1993
- [IMAI 86] Imai, M., "Kaizen: The Key to Japan's Competitive Success", McGraw-Hill Publishing Company, New York, 1986
- [KAR 95] Karlsson, Even Andrè, et. al., "Software Reuse – A Holistic Approach", Wiley, 1995.
- [LEHM 80] Lehman, M. M., "Programs, Life-Cycles, and the Laws of Program Evolution", Proc. IEEE, 1980
- [LOND 87] Londeix, Bernard, "Cost Estimation for Software Development", Addison Wesley, 1987.
- [MCCL 90] McClure, C. L., "The Three Rs of Software Automation: Re-engineering, Repositories, Reusability", Extended Intelligence, Inc., Chicago, IL, 1990
- [MIL 95] MIL-HDBK-171, Work Breakdown Structure for Software Element, July 1995.
- [MOAD 90] Moad, J., "Maintaining the Competitive Edge", Datamation, February 1990
- [MUL 95] Müller, Hausi A., Tutorial "Understanding Software Systems using Reverse Engineering Technologies", 17th International Conference on Software Engineering, Seattle, Washington, U.S.A, April 1995.
- [REL 90] Reliability, C. f. S. (1990). Software Reliability Handbook, Elsevier.
- [RENF 97] RENAISSANCE Consortium, RENAISSANCE ESPRIT Project #22010, "RENAISSANCE Framework", 1997
- [RENC 97] RENAISSANCE Consortium, RENAISSANCE ESPRIT Project #22010, "Client/Server Migration", 1997
- [RENA 97] RENAISSANCE Consortium, RENAISSANCE ESPRIT Project #22010, "Architecture Models for Evolution", 1997

-
- [RENE 97] RENAISSANCE Consortium, RENAISSANCE ESPRIT Project #22010, "Evolution Strategies", 1997
- [ROCH 91] Rochester, J. B. and D. P. Douglass. (1991.). "Reengineering Existing Systems." *I/S Analyzer*, Vol. 29,(No. 10.): pp. 1-12.
- [SEI 96] SEI (1996). *Assessing the Evolvability of a Legacy System*, Software Engineering Institute, Carnegie Mellon University.
- [SNEE 94] Sneed, H. and E. Nyary (1994). "Downsizing Large Application Programs." *Software Maintenance: Research and Practice* 6(6): 235-247.
- [SNEE 95] Sneed, H. M. (1995 January). "Planning the Reengineering of Legacy Systems." *IEEE Software* 12(1): pp.24-34.
- [SNEE 91] Sneed, H. M. (September 1991). "Economics of Software Reengineering." *Journal of Software Maintenance: Research Practice* Vol. 3: pp. 163-182
- [SRHA 95] American DoD Technical Report, JLC-HDBK-SRAH, "Software Reengineering Assessment Handbook", Volume I, II, Version 2.0, 1995
- [STO 97] Storey, Margaret-Anne D., et. al., "Manipulating and Documenting Software Structures", World Scientific Publishing Co., in press
- [STSC 93] Software Technology Support Center, "Reengineering Technology Report", STSC Hill AFB, UT, 1993
- [TILL 95] Tilley, S., "Perspectives on Legacy Systems Reengineering", Software Engineering Institute, Reengineering Center, Draft Version 0.3, 1995
- [TRY 97] Tryggeseth, Eirik, "Support for Understanding in Software Maintenance", PhD Thesis, NTNU, Trondheim, Norway, March 1997.
- [WON 96] Wong, K., "Rigi Blurb", <http://www.rigi.csc.uvic.ca/>, February 1996.
- [WOOD 92] Wood Michael, "A Reengineering Economics Model" STSC Crosstalk, June 1992.
- [YOU 89] Yourdon, Ed, "RE-3 - Part 1", *American Programmer*, Vol. 2, No. 4, April 1989, pp. 3-10.