



# RENAISSANCE

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

## RENAISSANCE METHOD

©RENAISSANCE Consortium 1997-1998

Version 3.1, October 1998

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 ISM** (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 6 492 011; 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 supports 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.

The report defines the objectives of Renaissance, and the constraints and benefits of applying this technology, together with hints for introducing the technology within an organisation. Then it introduces fundamental concepts necessary to understand the method, together with an informal description of key activities involved in the method. In the remainder of the document, it describes 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 the specific needs of the user of the method, which arise both from company and project constraints and practices. Activities are identified through their interfaces, defined in terms of inputs and outputs. Concerning the implementation of an activity, Renaissance simply describes the actions to be performed, without forcing any implementation approach. Thus, the overall control flow and the activation of a single activity can be customised. Co-operation among the activities and an incremental approach to refining the system under reengineering are achieved through a repository containing all the information needed by the process, which are the inputs and the outputs of the various activities.

The method is structured into two main steps. The first one, or *what to do*, assesses the organisation and the legacy system and identifies both the need and urgency of reengineering and the best strategy to adopt for renewing the system: continue with the current maintenance approach; re-engineer the system from user-interface, structure, architecture, and design perspectives; replace the existing system with a new developed one. The second step, or *how to do it*, 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 shows the traceability from that framework to this method. Readers who are not interested in this traceability can skip this chapter with no impact whatsoever on the understanding and mastering of the Renaissance method.

---

## Table of Contents

1	Introduction.....	1
1.1	Foreword.....	2
1.2	Renaissance Project Objective.....	2
1.3	Renaissance Benefits.....	2
1.4	WHO is this report aimed at.....	3
1.5	WHAT is covered in this report.....	3
1.6	Supporting Renaissance Consultancy Reports.....	4
1.6.1	Renaissance Report “Technology Selection”.....	4
1.6.2	Renaissance Report “Client/Server Migration”.....	4
1.6.3	Renaissance Report “Evolution Planning”.....	4
1.6.4	Renaissance Report “Architectural Modelling for Evolution”.....	5
1.7	Renaissance Adoption Hints.....	5
2	The Renaissance Method.....	7
2.1	Fundamental Concepts.....	8
2.1.1	Diagram of Principal Renaissance Activities.....	10
2.1.2	An informal walkthrough of Renaissance.....	12
2.1.3	Responsibilities in the Method.....	24
2.1.4	Information management and the concept of artifacts.....	26
2.1.5	A journey: how information grows in Renaissance.....	31
2.1.6	Tool support for Renaissance activities.....	34
2.1.7	Activities and tasks: the basic building blocks of Renaissance.....	38
2.2	The Four Phases of the Method.....	42
2.3	PLAN EVOLUTION.....	44
2.3.1	Startup Method.....	46
2.3.2	Describe Business Process.....	52
2.3.3	Assess Current Situation.....	54
2.3.4	Model Context of Current System.....	64
2.3.5	Select Target.....	66
2.3.6	Model Context of Target System.....	82
2.4	IMPLEMENT.....	84
2.4.1	Plan Evolution Project.....	86
2.4.2	Define Test Strategy.....	88
2.4.3	Design Target.....	90
2.4.4	Design Transformation.....	96
2.4.5	Transform.....	98
2.4.6	Test.....	100

---

2.4.7	Prepare Target.....	108
2.5	DELIVER.....	110
2.5.1	Plan Deployment.....	112
2.5.2	Install System.....	114
2.5.3	Migrate Data.....	116
2.5.4	Do Acceptance Test.....	118
2.6	DEPLOY.....	120
2.6.1	Design Changeover.....	122
2.6.2	Train Operators.....	124
2.6.3	Replace Old System.....	126
2.6.4	Document Revised Business Process.....	128
2.6.5	Do In-Use Evaluation.....	130
3	Bibliography.....	132
4	Appendix: Traceability from Framework to Method.....	136

*This page is intentionally blank*

# 1 Introduction

## **Summary**

This chapter introduces the Renaissance method for reengineering legacy systems. The objectives of the method, as well as the benefits of adopting it are presented and discussed. A discussion of adoption issues is provided. A summary of other relevant project consultancy reports concludes the chapter.

*Note to the reader:* at appropriate places throughout the document, blank pages have been intentionally inserted in order to improve legibility. Detailed descriptions of activities and tasks will always appear on facing pages.

## 1.1 Foreword

This is Version 3.1 of the Renaissance method. It incorporates incremental changes caused by the formalisation of the method in a workflow model.

## 1.2 Renaissance Project Objective

The main objective of the **RENAISSANCE** project is:

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

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

## 1.3 Renaissance Benefits

The benefits of reengineering an existing system can be summarised as follows:

- **Lower costs** - As technology progresses, the market will generate a demand for the new technology, and the question arises about how to effect the transition for legacy systems. There is evidence from a number of projects in the United States that reengineering an existing system can cost significantly less than new system development. As an example, figures quoted by Ulrich ("The evolutionary growth of software reengineering" in R.S. Arnold, *Software Engineering*, IEEE Press) showed that reengineering a particular system cost \$12 million compared to an estimated \$50 million cost of redevelopment.
- **Lower risks** - incremental reengineering of a system means that the risks associated with each change are relatively low. It is less likely that the business will be confronted with a system which does not meet its real needs. End-users of the system have time to adapt to system changes and are not faced with a completely new system.
- **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 and other properties** - 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.

The benefits of the Renaissance method include:

- A repeatable re-engineering process;

- More cost effective use of resources;
- Systematic assessment of return on investment;
- Continuous improvement and evolution of the system;
- Coverage of a wide range of re-engineering options.

As an 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 organisation. Using these values and the business goals, the method provides support for identifying the best kind of action to be taken on the existing system.

## 1.4 WHO is this report aimed at

The intended audience includes

- senior managers and project managers who are considering 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, although not mandatory.

Reading the “Renaissance Framework” report is recommended, but not mandatory. People interested in learning more on the background of the method can find details in that report.

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

## 1.5 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;
- the definition of an operational way to use the method;
- adoption hints to successfully introduce Renaissance within an organisation and benefit from its use;
- traceability from the Renaissance framework to the Renaissance method, for interested readers.

Although the Renaissance method is comprehensive for the reengineering of legacy systems, it is not intended to cover specifically certain topics:

- business process reengineering;
- methods for object-oriented development.

## 1.6 Supporting Renaissance Consultancy Reports

The Renaissance method is supported by several other **RENAISSANCE** project consultancy reports, which provide detailed guidelines and suggestions about the execution of the various activities and tasks that compose the method. In this section we provide an overview of these reports, so that you have an opportunity to familiarise yourself with the entire library of Renaissance documentation available to you before entering into a closer study of the method.

The approach used in this document is to emphasise the relationships among the activities and tasks of the method, while referring to the other **RENAISSANCE** project consultancy reports for in-depth treatment of specialised technical topics.

### 1.6.1 Renaissance Report “Technology Selection”

In many projects, the selection of technologies is informal, implicit, and embedded in other activities. The **RENAISSANCE** project took another approach: a key objective was to establish a shared vision of the technologies which should be deployed to support the overall aims of the project. This report describes the results of the *technology selection* activity—a set of technologies and tools which are of direct relevance to the pilot applications that were examined during the course of the project, and which have significant market potential. It provides many insights about the provision of tool support for the Renaissance method.

### 1.6.2 Renaissance Report “Client/Server Migration”

This consultancy report includes a description of target client-server models including legacy system components (two and three level models); an assessment of the applicability of OLE and CORBA and conversion of systems using teleprocessing monitors to these frameworks; and advice on specific problems of re-organising monolithic systems to a client-server model. The utilization of user interface generation systems for client implementation is described, and techniques of data migration from central legacy system databases to distributed systems (components, data, transaction processing monitors, user interfaces, etc.) servers are covered.

### 1.6.3 Renaissance Report “Evolution Planning”

This report provides guidelines for managers on planning and managing the evolution of legacy systems. It describes evolution strategies which may be applied to all or part of a system, explains where these strategies are likely to be cost-effective and suggests how techniques such as algorithmic cost modelling may be used to estimate the cost-effectiveness of these strategies. It also includes advice on other issues such as hardware platform evolution, training, effects on other installed systems, etc.

### 1.6.4 Renaissance Report “Architectural Modelling for Evolution”

This report provides detailed advice on how modern architectural modelling styles may be applied to identify and describe the architecture of legacy systems and especially its reusable building blocks. It recommends a documentation framework for architectural models and explains how systems may be analysed to produce this documentation.

## 1.7 Renaissance Adoption Hints

In this section we present some guidelines for adopting the method in your own organisation, drawn from a wealth of experience in previous evolution projects. By following these adoption guidelines, you can improve your ability to the method into your organisation in a smooth fashion, keeping institutional resistance to a minimum and assuring an enthusiastic and co-operative reception of the ideas of re-engineering projects by your technical and user communities.

As with any new technology, reengineering and reengineering methods should be inserted cautiously within an organisation. Before embarking on a strategic reengineering project, we suggest becoming familiar with the Renaissance method through a set of pilot projects. A pilot project should be small and “isolated,” that is, with little or no impact at all on critical software components and data files.

Such pilot projects should enable an organisation 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 organisation reacts to new methods of working. One of the possible curves, known in literature as Conner and Patterson's Adoption Curve, is detailed in Figure 1.

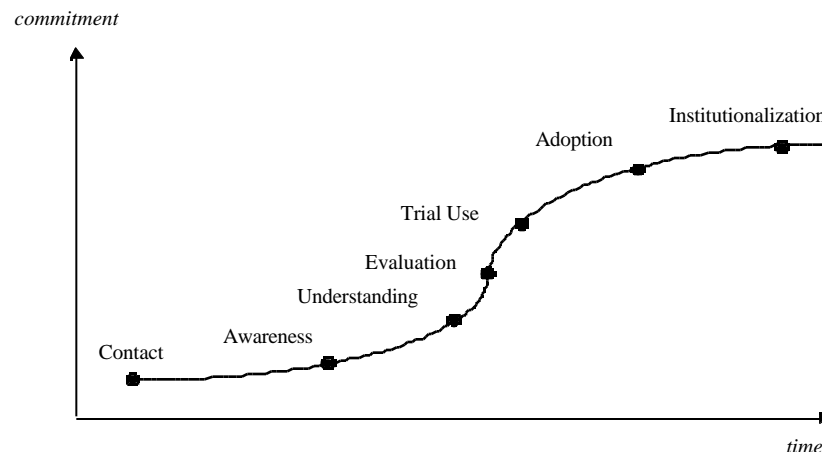


Figure 1: Conner and Patterson's Adoption Curve for a new technology

After encountering a new process or technology, potential users 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 organisation's daily process: we refer to this as *institutionalisation*.

## 2 The Renaissance Method

### **Summary**

This chapter describes the Renaissance method for reengineering legacy software systems. The first section introduces material to help the reader understand the fundamental concepts underlying the method. An informal walkthrough of the principal activities of the method is presented. The notation of the method, both textual and graphical, is introduced. Subsequent sections describe the method using the introduced notation and describe the activities to perform for conducting a reengineering project using the Renaissance approach.

## 2.1 Fundamental Concepts

In this section, we present important information that will help familiarise you with the Renaissance method—its basic principles, its logic, and the way we have chosen to describe the method in subsequent chapters. It will prepare you to be an effective user of the method in your own projects.

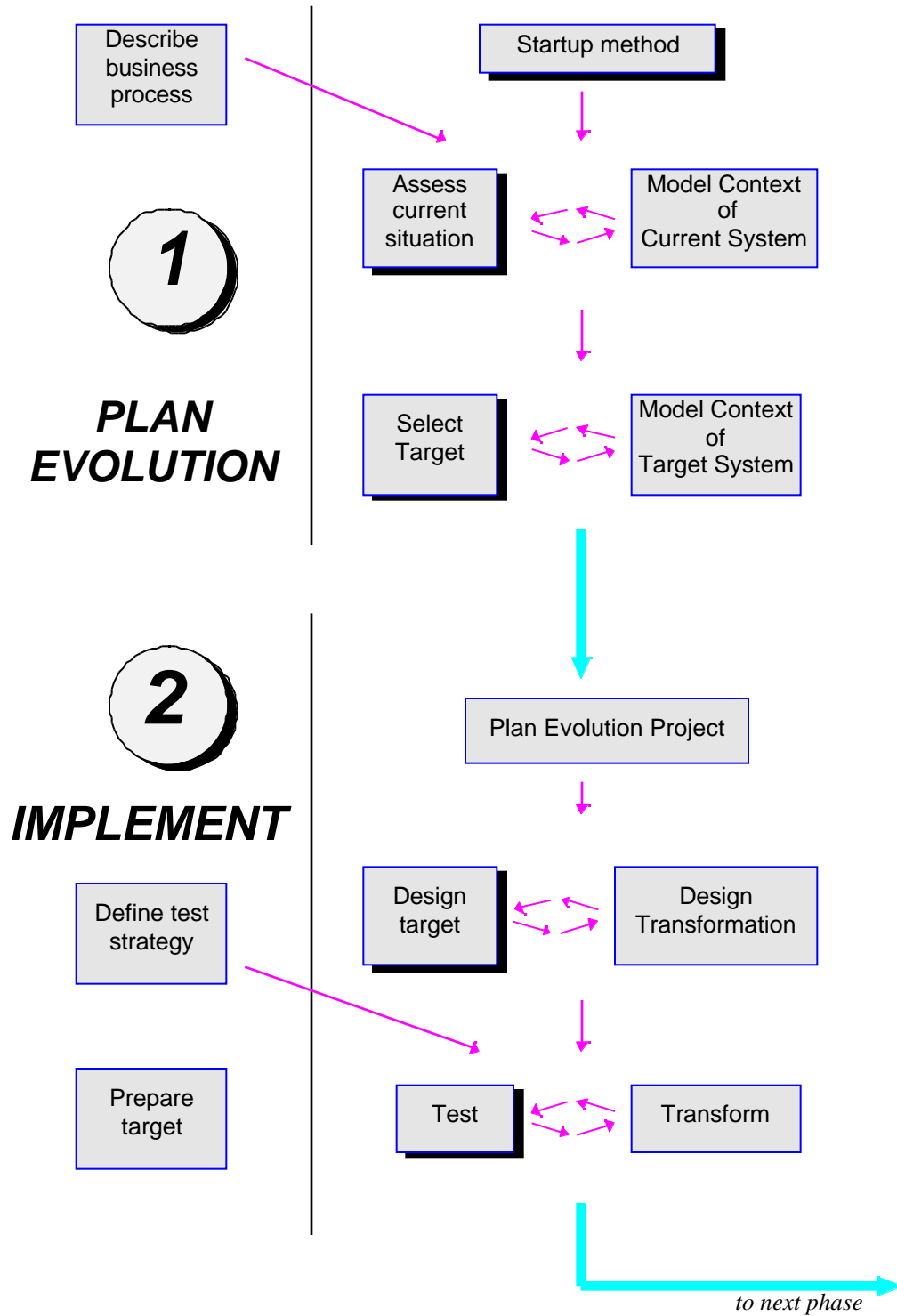
We begin with a “walkthrough” that will guide you in an informal way through the various steps of the method, in preparation for the more formal description presented later. During this informal walkthrough, you will visit all of the major phases of the method, and have a chance to acquaint yourself with the activities and tasks that are carried out during a Renaissance re-engineering project, without having to confront heavy, formal jargon. Above all we try to explain the underlying *approach* and the key underlying concepts—knowledge that will serve you every time you apply the method.

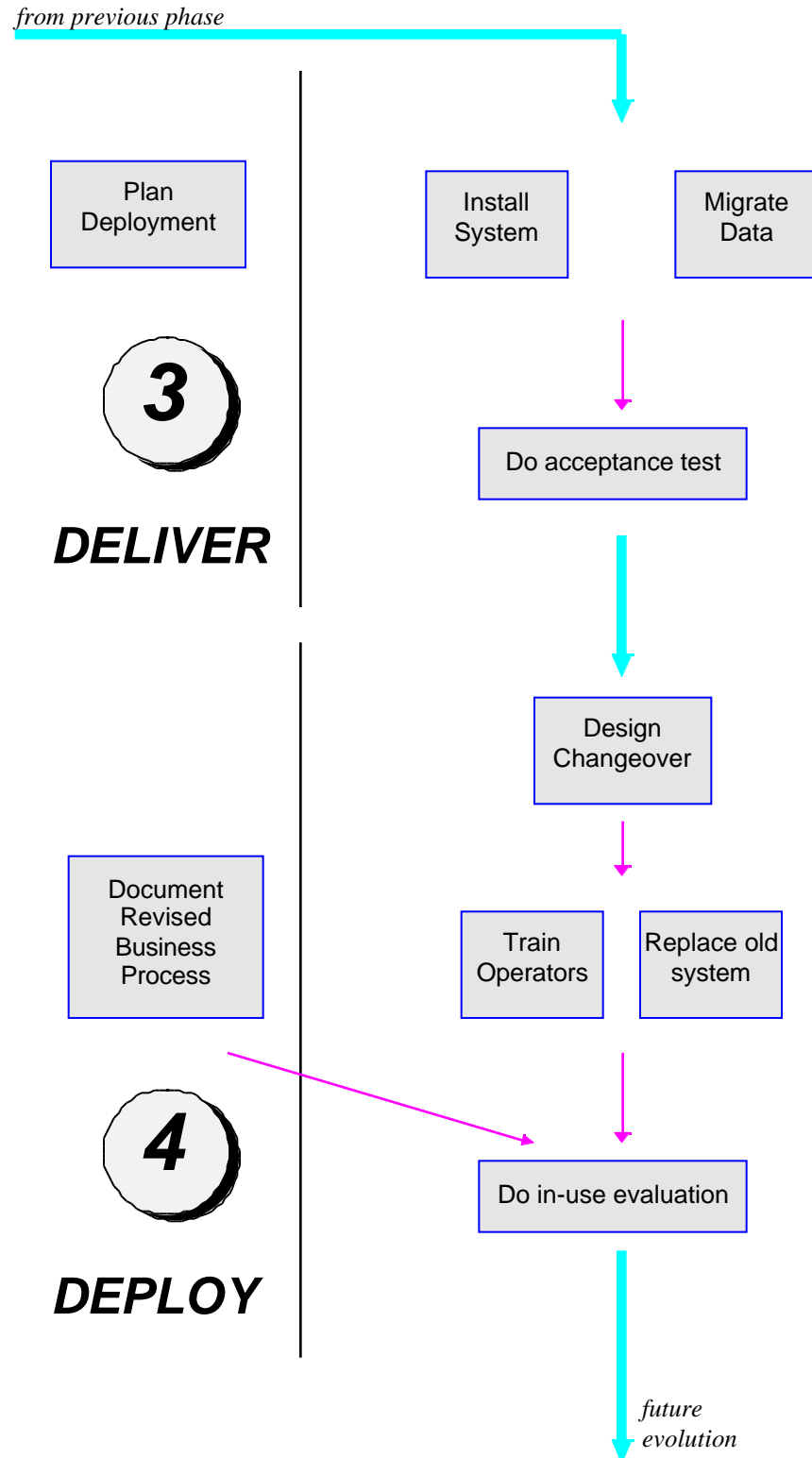
We continue with specialised discussions of a variety of topics that are likewise fundamental to the use of the method:

- We introduce you to of the different kinds of personnel and products that the method deals with. First, we describe the *responsibilities* that you will see involved in Renaissance activities—from users, to project managers, to experts in the workings of your legacy system. Then we provide an overview of the *artifacts*—the workproducts—that you will deal with during application of the method.
- We introduce you to the way in which Renaissance organises its activities around the notion of a project “repository,” which becomes a central information resource for your evolution project and allows your accumulated knowledge to be employed to maximum effect.
- Finally, we prepare the way for the more detailed, formal description of the method in the remaining chapters of this document. We begin with an explanation of the notation that has been selected for presenting the activities and tasks. As you will see, the notation has been selected with a view towards customisation by your organisation to accommodate your own needs and style. By taking a bit of time now to familiarise yourself with the notation, you will have few problems in reading the descriptive sections in the chapters that follow.

*This page is intentionally blank*

### 2.1.1 Diagram of Principal Renaissance Activities





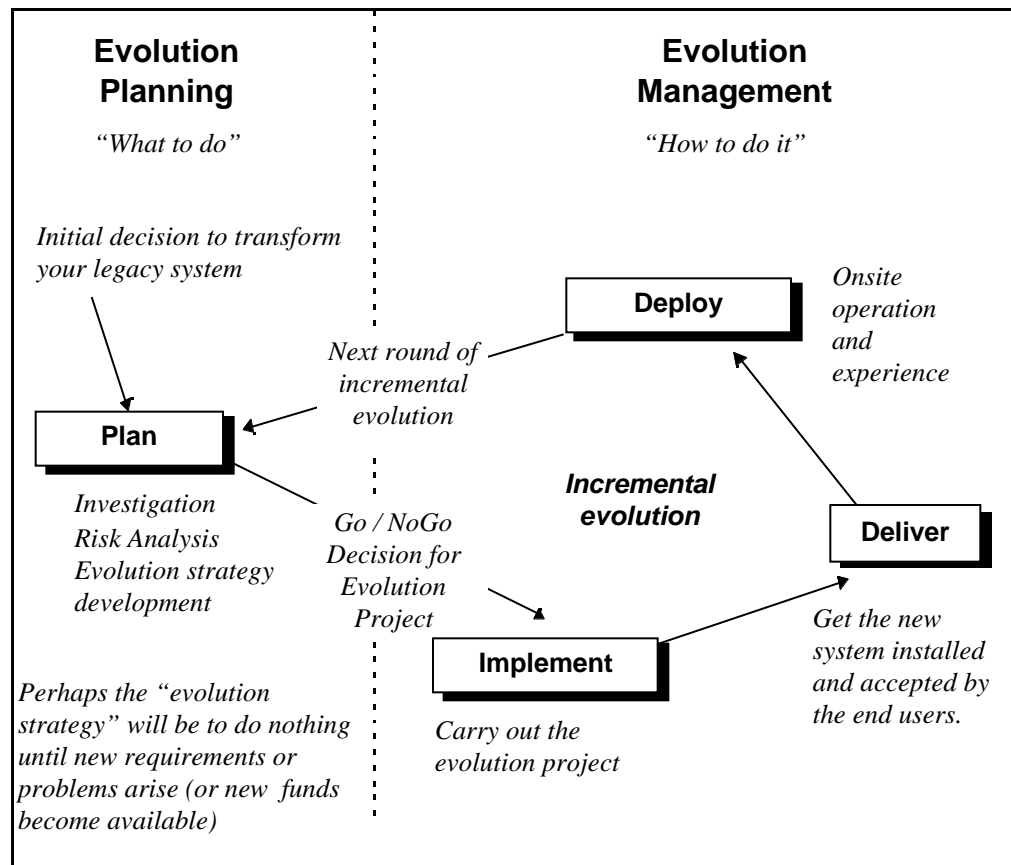
## 2.1.2 An informal walkthrough of Renaissance

The rather formal description of the method in the sections making up the bulk of this document does not do justice to the simplicity and elegance of the Renaissance method. One step follows another in a straightforward, logical fashion, leading you naturally through a sequence of activities that ensure the evolution of your system in a controlled, cost-effective manner.

In this section we present an informal description of the method for a manager or a user of the method who is seeing it for the first time, and introduce you to the key concepts as you walk through all of the major activities and acquire a feel for their flow and logical connection to each other. You will find that the concepts are simple, intuitive, yet flexible enough to adapt themselves to any evolution project you are contemplating.

### The Phases of the Renaissance Method

Let us begin with a panorama over the major phases in the method. There are only four of them, and each leads naturally to the next.



**Figure 2: The Renaissance Cycle of Continuous Evolution**

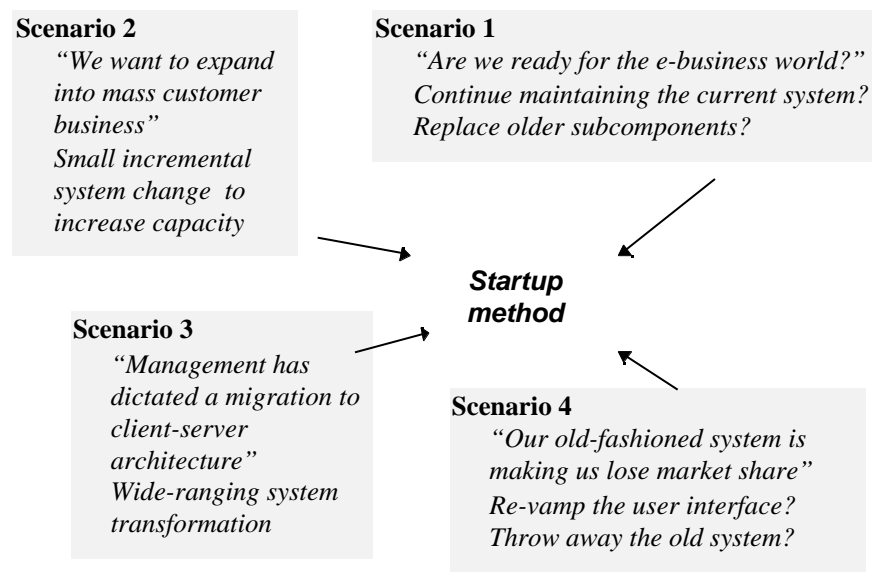
Note first of all that the four phases are linked in a continuous process of *incremental evolution*. Renaissance does not force you to think in terms of one, single "big bang" in which the entire system must be transformed.

Rather, your system can evolve incrementally as new requirements, new organisational and technological developments, and budgetary constraints dictate. For example, in a first step you may prefer only to upgrade the user interface of your system to a modern windows-based interface, but to leave the migration to a client-server architecture for another time—perhaps in the next year when more budget is available. The Renaissance method permits such an incremental approach to the evolution of your system. You will never have to think in terms of “the final version of my system.”

Note secondly that “evolution planning” has a special place in Renaissance. It is during this activity that you will utilise all of the powerful analytical and investigative techniques that Renaissance provides to make an informed decision on whether to proceed with the evolution of your system.

## Evolution Planning

Before trying to go somewhere, you have to understand where you are now. The key to successful evolution planning is a well-organised and executed *assessment* of your current situation. Your assessment will give you a solid understanding of your system’s components and form the basis for developing a good evolution strategy.

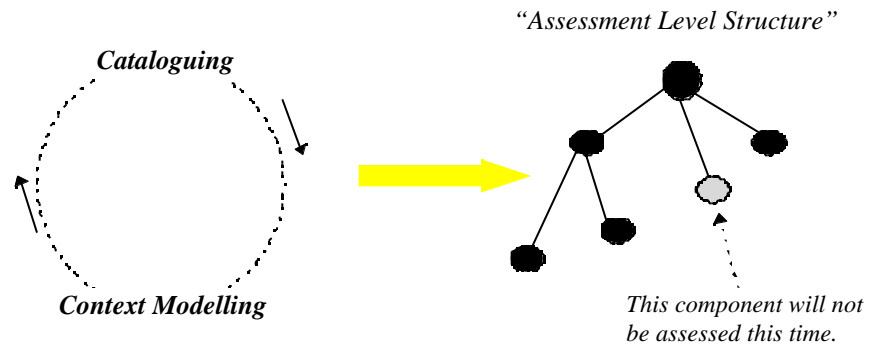


**Figure 3: A variety of possible starting scenarios**

As simple as the concept sounds, it is not always very clear how to organise an assessment. A bewildering variety of initial scenarios is possible, from large scale system overhaul or replacement, all the way down to small incremental changes affecting only parts of your system. Figure 3 shows only a few of the scenarios illustrating the kinds of requirements, problems, and business goals that might lead your organisation to evolve your legacy system with Renaissance.

Renaissance has been designed to guide you through a process of focusing both the method and your resources on your particular scenario as soon as

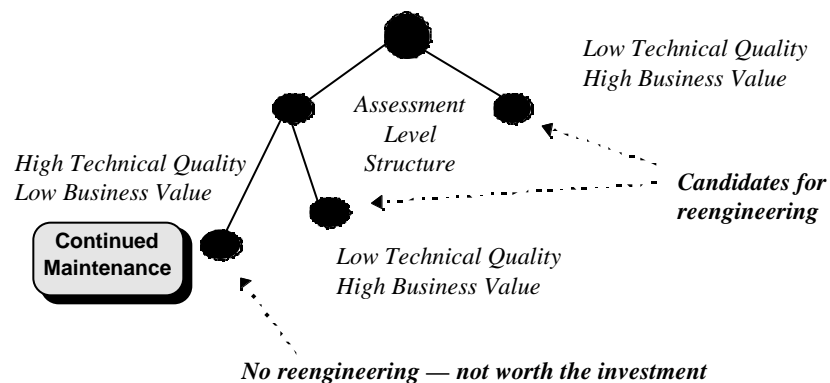




**Figure 5: Determining the level of assessment**

As Figure 5 illustrates, you gradually catalogue the current system's components, doing formal modelling if appropriate, until you have decided which components of the system structure should be assessed for possible evolution this time. You will probably decide not to assess some components, leaving them for another day.

Now that you have decided which parts of the system you want to assess, the Renaissance method provides yet another means to narrow the focus even more, to only those parts of the system that are truly good candidates for evolution. Following the guidelines in the Renaissance *Evolution Planning* consultancy report, you assess each component in your assessment level structure for its *technical quality* and for its value to your *business*.

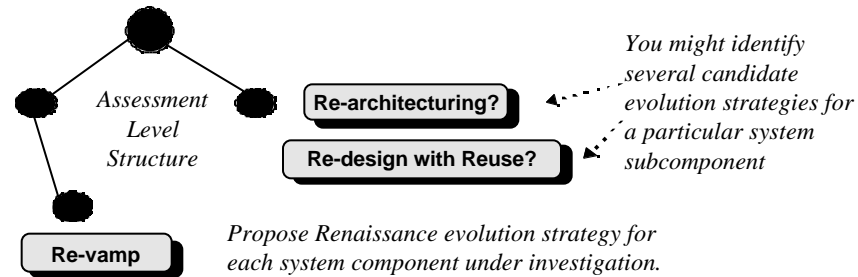


**Figure 6: Finding candidates for evolution**

As seen in Figure 6, particular combinations of business value and technical quality can help you determine which components really should become candidates for evolution, and which are better left alone, or simply subjected to continued maintenance. Components with low technical quality, but with high value for your business, are typically good candidates for evolution. But other combinations are possible, too:

- Components with relatively low business value and *any* technical quality might benefit from replacement with standard COTS software;
- Even components with high business value and high technical quality might benefit from re-engineering (e.g. porting from COBOL mainframe to C/Unix).

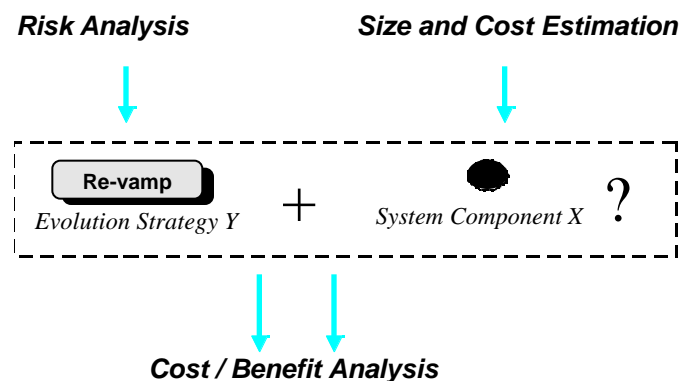
After completing this filtering process of identifying the set of system components you really want to concentrate on, you are ready to start using the techniques provided by Renaissance to select the optimal evolution strategy for each individual component. From the set of ready-made evolution strategies that are documented in the *Evolution Planning* consultancy report, as well as the *Client/Server Migration* report you will be able to identify one or more candidate strategies for each component undergoing assessment, as illustrated in Figure 7. Here, too, the careful groundwork laid while organising for assessment will help you. For example, any *constraints* that might have been imposed by upper management (such as a mandated evolution to client-server architecture) will already be documented, and help you select strategies according to the choices that are realistically available to you.



**Figure 7: Candidate evolution strategies**

Having associated a good set of candidate evolution strategies with each component, you are now ready to utilise the analysis techniques Renaissance provides to reveal the implications of the choices you now have before you, by answering questions such as the following:

- How risky is this evolution strategy? Is there a chance it will fail on this component?
- How much will it cost to re-architecture this component? Would it cost less to replace it?
- Would it be useful to pursue a long-term strategy of re-vamping this component first, with plans to re-architecture it in some future iteration of the Renaissance method?

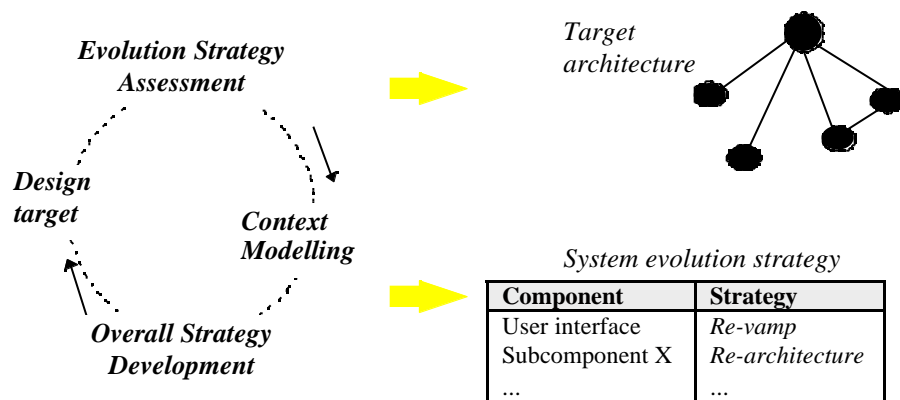


**Figure 8: Costs and benefits of evolution strategies**

The risk analysis and estimation techniques described in the Renaissance *Evolution Planning* consultancy report lead you through the process, with a comprehensive cost/benefit analysis as a result (Figure 8).

Renaissance can help you to look at risks and costs from several points of view. In addition to a *cost/benefit analysis*, you might want to carry out a *cost/risk analysis*, where you examine the cost and risk factors of various alternatives in order to choose the best combination—for example, you may choose a more costly strategy that is less risky than another, cheaper strategy.

With this fund of information available, you are now ready to put all the pieces together and develop your overall system evolution strategy. You will do this at the same time that you design the “target” architecture—that is, the architecture of your evolved system.



**Figure 9: Target architecture and system evolution strategy**

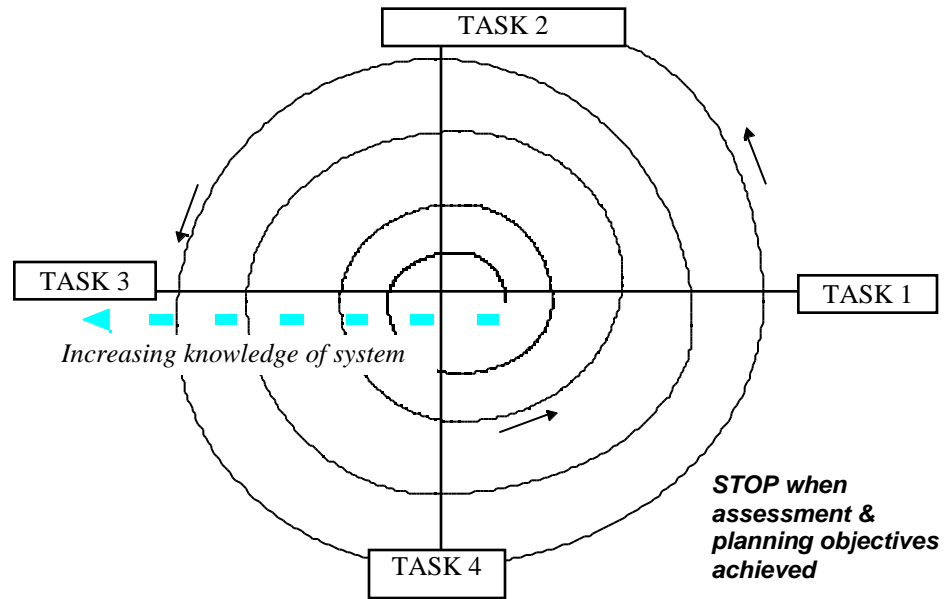
As shown in Figure 9, in a controlled cycle of design, evaluation, and modelling, you will gradually build the architecture of the new system. The Renaissance consultancy report *Architectural Modelling for Evolution* provides a variety of tools to help you model the context of your system to the degree of formality you feel is necessary. The consultancy report *Client/Server Migration* provides you with an especially comprehensive treatment of issues in developing a new distributed architecture. You will make use of the information that the method has helped you elicit for your decision-making process to resolve questions such as the following:

- If I use this evolution strategy for this system component, will it have a major impact on the new architecture?
- Will this target architecture configuration force me to use another evolution strategy for that particular component?

The method also provides assistance in the form of a powerful set of context modelling tools that will help you to describe your envisioned architecture and overall evolution strategy with precision, as illustrated in Figure 9.

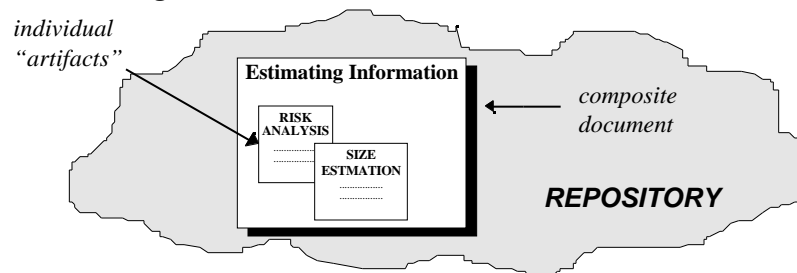
Before leaving our discussion of evolution planning, we should point out one important aspect of *cost-effectiveness*. Renaissance encourages you to carry out evolution planning at successive levels of detail *as appropriate to*

*your specific needs.* For example, your assessment needs may range from a broad spectrum “quick and dirty” (and inexpensive) assessment—that may well suffice for a decision to go ahead—all the way to a thorough, narrowly focused analysis and planning effort utilising all of the tools provided by Renaissance. In this manner, you will always have firm control over the effort expenditure necessary for evolution planning. (See Figure 10)



**Figure 10: Iterative evolution planning**

By the time this process has completed, the method will have guided you through a comprehensive, yet focused and cost-effective evaluation of your current system, yielding sufficient information for you to decide whether to go ahead with the effort and expense of an evolution project in light of your business goals and constraints.



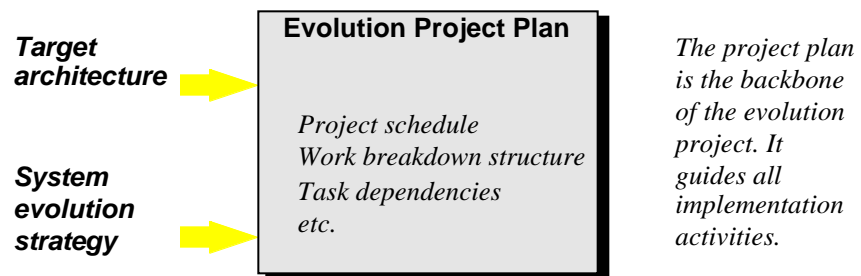
**Figure 11: The Renaissance Repository**

You may think of your work as involving the creation of “artifacts”—items such as cost/benefit reports, risk assessment reports, and the like. In order to manage this valuable information, Renaissance provides a convenient organisation of these “artifacts” into *composite documents*, containing related items (such as the Estimating Information illustrated in Figure 11). In turn, these composite documents may be thought of as

being managed in a central “repository” of information that grows and accompanies your activities throughout the life of your evolution project.

## Implementation — The Evolution Project

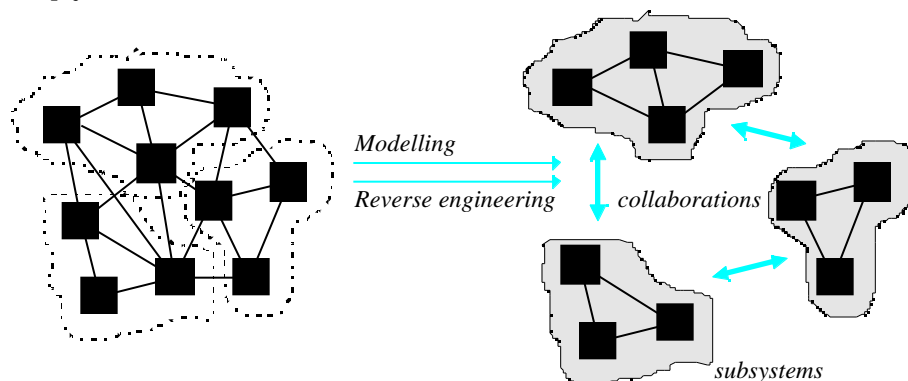
You have completed *evolution planning* and the Go decision has been made. Now you can launch the evolution project that will transform the system according to the architecture and overall system evolution strategy that you have developed. The first step is another kind of “planning,” different from evolution planning: the *evolution project plan*.



**Figure 12: The Evolution Project Plan**

As shown in Figure 12, the evolution project plan is a key document that will follow you all the way through implementation, delivery and deployment of your new system. The Renaissance method offers guidance in developing a solid, workable plan.

With a project plan in place that spells out clearly the evolution steps and their sequencing, you are ready to go ahead with the job of *transforming* the system. An evolution project is obviously different from a traditional implementation project due to the fact that there is already a system in place. Even though you have already *selected* your evolution strategies, you will need to understand the system in more detail in order to *implement* them. In old legacy systems, discovering the detailed structure and behaviour of the system may be a challenging task, since the information may be scattered about the system. Reverse engineering techniques together with sophisticated modern multi-view modelling approaches can help you extract that information.

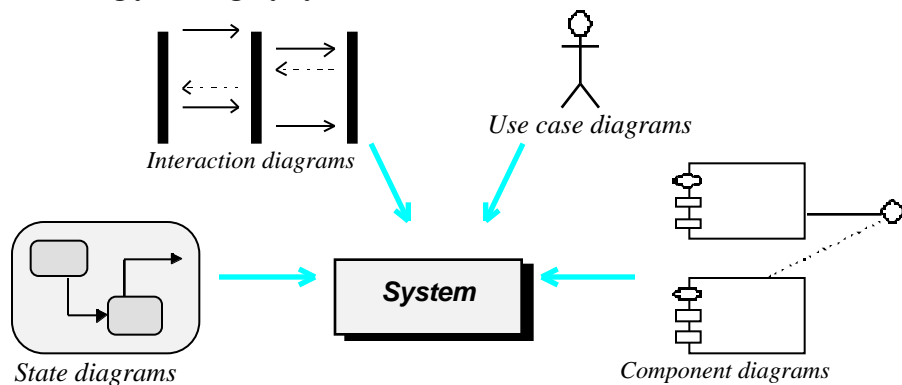


**Figure 13: Technical Modelling**

The Renaissance consultancy report *Architectural Modelling for Evolution* provides a comprehensive set of techniques for *technical modelling*, answering questions such as the following (see Figure 13):

- How does data flow through my current system?
- How should data flow through the new system?
- How do the major components of my system currently interact and collaborate with each other?
- What improved scheme of interaction and collaboration can I develop for my new system?

The consultancy report also shows you how to model several different views (see Figure 14) of your old *and* new system with the Unified Modelling Language, so that you are well prepared for the task of migrating from one system structure to another—and incidentally will have a solid set of documentation for your new system, preventing it from becoming your “legacy system of the future.”



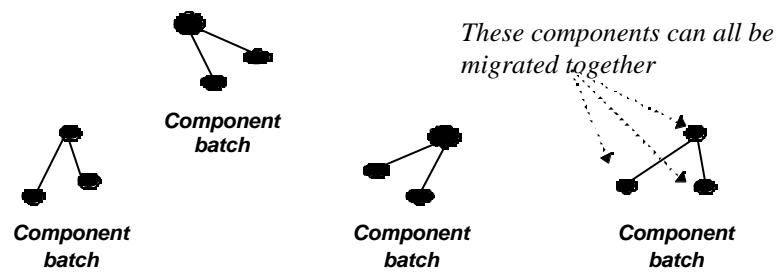
**Figure 14: Modelling from multiple points of view**

An important part of the implementation phase is the strategy that you will develop for the controlled transformation from the old to the new, as illustrated in Figure 15.



**Figure 15: Design of the system transformation**

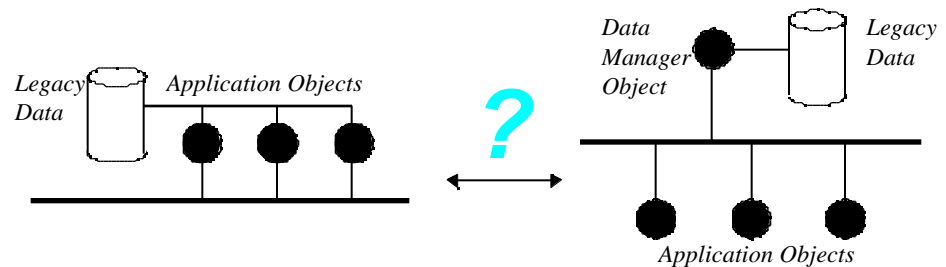
Here, too, the Renaissance method provides special concepts and techniques that have been elaborated for use in evolution projects. The method helps you to identify cohesive “batches” of components that can be migrated together to the new system in an orderly, incremental fashion, as seen in Figure 16.



**Figure 16: Identification of component batches**

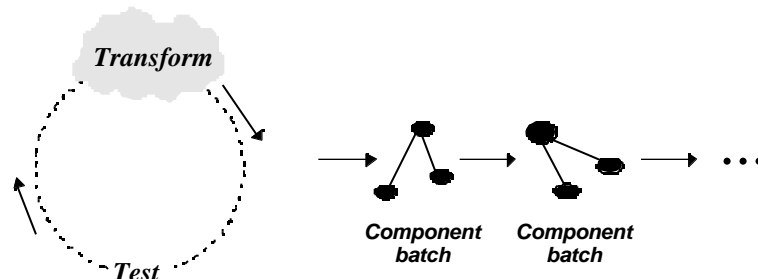
The *Client/Server Migration* report also provides you with considerable guidance on how to go about integrating your migrated components into a new architecture, helping you to answer such questions as:

- Would it be better for me to let my application objects access the legacy data directly, or should I encapsulate the data behind a dedicated data manager object? (Figure 17)
- Should I use CORBA or DCOM for integrating my object services?
- Should I introduce a workflow management system to co-ordinate the collaboration among system components?



**Figure 17: Integration of legacy data in new architecture**

Under continuous control of testing, you now integrate the component batches one at a time to the new system according to your integration scenario, as illustrated in Figure 18.



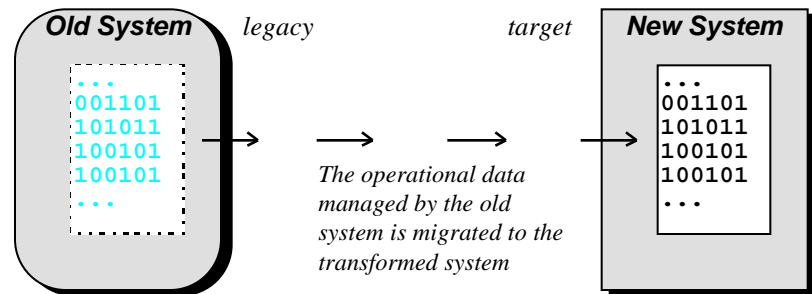
**Figure 18: Controlled integration of component batches**

At the end of this controlled implementation process, you will have a transformed and fully tested system, ready for final installation and acceptance testing.

## Delivery

During implementation, you carried out transformations and tests with your own, possibly specially constructed test data. Now it is time for the users to take final delivery of the system, while operating under realistic conditions with their own actual enterprise data.

Under the supervision of the users, you now migrate the actual enterprise data managed by the legacy system to the new system, as shown in Figure 19. This may be a simple task—for example, perhaps the legacy system managed little data; or perhaps you only transformed a part of the system that was not concerned with data management. If the core data management facilities were transformed, though, then data migration will have to be planned and executed with some care.

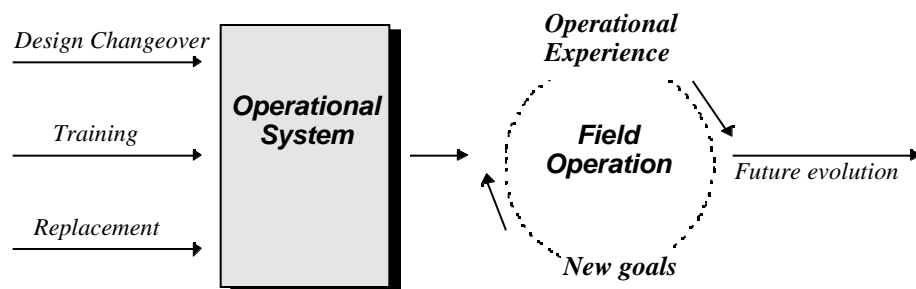


**Figure 19: Migration of the data from old to new system**

With the completely installed system in place, and fully operational with the migrated enterprise data, final acceptance can take place under the responsibility of the users. Once they have given the new system the green light, it can finally be deployed in its operational environment.

## Deployment

The first task of deployment is to install the new system on site—on the actual hardware upon which it will operate, with the actual software packages, etc. This can be done together with the onsite operational personnel.



**Figure 20: Deployment and operation of the new system**

The operational system now goes into use again at your organisation. As Figure 20 illustrates, the conscientious application of the Renaissance method goes beyond a single evolution step in the life of your system: even during field operation, you will be gathering experience and considering

the influence of your own business goals as they affect both the performance and the future requirements of your system. Based upon this new experience, you will be ready at some future moment to evolve your system once again in a controlled, cost-effective manner.

### 2.1.3 Responsibilities in the Method

The Renaissance method foresees that its activities will be carried out by persons fulfilling a number of different “responsibilities.” The term “responsibility” should not be taken to mean a one-to-one correspondence with human beings—rather, it describes responsibilities and capabilities that are involved in the application of the method to an evolution project. The actual correspondence between these responsibilities and members of your evolution project team will depend upon your customisation of the method (Section 2.1.7) for your specific organisation and project.

For example, there will almost always be several persons who have the responsibility of “Software Engineer” on your project team. There is usually only one “Software Project Manager,” but that very same person may *also* have the responsibility of “Software Engineer.”

The purpose of this set of responsibilities is to give you a clear understanding of the distribution of duties and capabilities that is encouraged by Renaissance. It is up to you to find the most natural correspondence with your own situation.

**ABE (Application Business Expert)** — This is someone who knows deeply the business sector supported by the legacy system (for example, financial services or transportation logistics). He or she is consulted for elicitation of the business processes supported by the legacy system, as well as the new requirements for the evolved system.

**CL (Client Representative)** — Someone who has the authority to represent the client for the new system, and assist in establishing business priorities for the new system.

**DM (Data Modeller)** — This person is responsible for creating the new representation of the data in the evolved system. The role is especially critical during the implementation phase.

**EM (Evolution Modeller)** — This person is an expert in strategies and techniques for system evolution (although not necessarily in the specific domain in which the legacy system operates). He is likely to have been specially trained in the Renaissance method.

**EST (Estimator)**—This person is responsible for size and cost estimation activities. He has a robust understanding of legacy system software, re-engineering, estimating, modelling and economic indicators. The estimator will require access to system analysts, business analysts, modellers, hardware experts and software experts for his activities.

**LFE (Legacy Functional Expert)** — An expert in the functionality offered by the legacy system to its users, though not necessarily in its implementation. This person is able to answer questions about the services, performance, operational requirements (e.g. response time), and constraints of the legacy system.

**LIE (Legacy Implementation Expert)** — An expert in the internals of the legacy system. (May not be directly available for the evolution project.) Ideally, he will understand the structure of the legacy system, as well as the source code and the hardware requirements; and he will know the technical history of the legacy system. He is consulted by the evolution modeller during evolution planning.

**OPMG (Legacy System Operation Manager)**—The operators of legacy systems possess a wealth of valuable information about the functionality, performance, and other characteristics of systems. They play an important role in assessment of the legacy system.

**OTS (Target System Operational Technical Service)**—The operators of the evolved target system will be responsible for many aspects of delivery and deployment.

**QE (Quality Engineer)** — This is a separate role intended to provide independent verification and validation of the transformation of the legacy system to the evolved target system. He is involved in all aspects of implementation and system delivery.

**SAE (Software Architecture Expert)** — This is an expert in the rapidly expanding field of modern software architecture, who is responsible for ensuring that the target architecture for the evolved system adheres to sound, accepted principles.

**SWE (Software Engineer)** — The technical software development personnel involved in the transformation of the legacy to the target system.

**SWPM (Software Project Manager)** — The single focus of responsibility for execution of the evolution project. Renaissance does not specify any more precisely the technical management structure of the evolution project, leaving it to customisation for your own particular organisation.

**TE (Test Engineer)** — A separate and independent responsibility for carrying out the technical validation of the transformed legacy system, in co-ordination with the software development team.

**USER (User)** — Broadly defined responsibility to indicate the clients of the legacy system and the target system. Although we use this particular term in Renaissance, another, more accurate term which is commonly used for this group is *stakeholder*. The term encompasses several types of users (stakeholders), from operators of the system itself to the end users who benefit only indirectly from the services of the system.

## 2.1.4 Information management and the concept of artifacts

During application of the method, you will work with many “things,” which may range from reports—such as risk assessment reports and test plans—to pieces of software—such as commercial packages, or reusable code modules.

In order to have a disciplined approach to the management of all of these different kinds of entities, we have collected all of them under the single concept of *artifacts* that are “created and used” during method application.

Although in some cases the exact physical nature of an artifact will be clear (for example, a commercial package that you might want to include in your re-engineered system), in other cases it will depend on your circumstances and how you intend to apply the method. A test plan may be a paper document; or it may be a script in an automated testing tool. A conceptual model of your legacy system may be a simple summary of a discussion with your local system expert; or it may be a formal, multi-view set of analyses complete with use case, interaction, and state transition diagrams.

The purpose of the artifact concept is to give you a clear idea of the different kinds of entities that you will deal with when using the method. Their concrete realisation in the context of a real evolution project will vary according to your needs.

### The concept of *composite documents*

Artifacts don’t simply come out of nowhere, at random—rather, they are created and used during the course of carrying out the activities of the method, and they tend to be related to others that are created and used in a similar context. In other words, they tend to fall into natural *groupings*. Since many of the Renaissance artifacts are “documents” of various kinds—assessment reports, costing reports, test plans—we have found it useful to capture the idea of artifact groupings in the concept of *composite documents*. Each of the following sections presents a “composite document” which gathers together a set of related artifacts into a single, coherent set of information. As you will see, there is only a small core set of composite documents, each of which manages the information associated with a key aspect of the Renaissance re-engineering process.

#### 2.1.4.1 Business Information

*Composite document acronym: BUSI*

While much of the information in Renaissance is technical in nature, there is a significant amount of information in a re-engineering project that concerns organisational and business-related issues.

**BG (Business Goals)** — A clear description of the goals of your organisation and their impact on the planned evolution of the legacy system. *Plan phase.*

---

**BPD (Business Process Description)** — Description of the business process that is supported by the current (legacy) system; for aiding the assessment of its business value. *Plan phase.*

**PS (Problem Statement)** — A description of the situation which has led to consideration of evolution of the current legacy system. This description will normally include user-level requirements. *Plan phase.*

#### 2.1.4.2 Current System Information

*Composite document acronym: CSI*

During the application of Renaissance you will deal with many kinds of information about your legacy system. Current System Information contains a catalogue of legacy system components (stating properties, whereabouts, responsibilities, etc.) and a set of reports containing assessment information.

**CSA (Current system assessment report)** — The overall report of the results of assessment on your current system, which serves as the basis for the Go/NoGo decision on evolving the system. *Plan phase.*

**ALS (Assessment Level Structure)** — Contains a description of which system components will be assessed, and at which level of detail assessment will be carried out. *Plan phase*

**CD (Characteristic Definition)** — The definition of a characteristic according to which a component of your legacy system will be assessed; for example, cohesion or modularity. *Plan phase.*

**CDL (Characteristic Definition List)** — The collection of all characteristics for which assessment will be carried out on a component of your legacy system. *Plan phase.*

**CQR (Characteristic Quality Rating)** — A rating assigned by you to the quality of a system component according to a particular characteristic. *Plan phase.*

**CQRL (Characteristic Quality Rating List)** — The overall list of all quality ratings according to each individual quality characteristic, that may be used to assign an overall quality rating to a particular system component under assessment. *Plan phase.*

**CMCS (Context model of current system)** — A conceptual model of your current legacy system, which may describe its major components and their relationships to each other. *Plan phase.*

**TMCS (Technical model of current system)** — Detailed model of the architecture of the current legacy system. *Implement phase.*

#### 2.1.4.3 Estimate Information

*Composite document acronym: ESTI*

The Renaissance method features a particularly thorough guide to making good estimations of size, cost, and risk in the context of evolution planning. A Renaissance estimate report will be a composite document containing many individual kinds of estimating information.

**ASM (Estimate Assumptions)** — This report establishes the rules under which the estimate is prepared. It describes any assumptions made by the estimator which may have an impact on eventual project planning (for example, “assume that  $n$  people with skill  $x$  will be available at time  $t$ ”). *Plan phase.*

**CBA (Cost Benefits Analysis)** — Consolidated report on results of risk assessment, costing, and estimating schedule reports. It uses a computation for the projected future costs and benefits of employing a candidate system evolution strategy. *Plan phase.*

**CES (Cost Elements)** — This report lists the cost elements (staffing and resources) of employing a particular strategy along with an indicator for each describing the level of confidence in the figure (best/worst values). *Plan phase.*

**SCHEM (Estimating Schedule)** — An “anticipated” schedule for the evolution project. Such a plan is necessary because the fixed costs over time must be calculated. The time span may be imposed on an estimator if there are time constraints, in which case the estimator must “tailor” the estimate. Otherwise, the estimator must devise a time span for the evolution project. Such a time span will be that which most realistically fits the requirements and will be used as a basis for actual evolution project planning. *Plan phase.*

**RA (Risk Assessment Report)** — This report lists areas of risk concerning the evolution project as mapped out in the estimated schedule and Costs Breakdown for an evolution strategy. The estimating assumptions have an important role in risk assessment because the more assumptions the estimator must make, the more uncertain is the estimate and hence the more risky. **Note:** The risk assessment and costs breakdown will often be prepared together by the same set of people. *Plan phase.*

**CRA (Cost Risk Analysis)** — When several alternatives exist, this report can be produced. It adds (or subtracts) an element of cost, to each of the alternatives, depending on the corresponding risk analysis, in order to adjust each to the same level of confidence for comparison purposes. *Plan phase.*

#### 2.1.4.4 Target System Information

*Composite document acronym: TSI*

As the activities surrounding the conception of the new system are carried out, several kinds of reports and workproducts are created and managed, which are collected together in this composite document.

**TS (Target System)** — The evolved system; the result of transforming the legacy system. *Plan, implement phases.*

**CCP (Candidate Commercial Packages)** — Identified commercial software that could become incorporated into the target system as part of the evolution of the legacy system. *Plan phase.*

**CMTS (Context model of target system)** — A conceptual model of the architecture of the system towards which you intend to involve your current system. *Plan phase.*

---

**TMTS (Technical model of target system)** — Detailed model of the architecture of the target system for evolution. *Implement phase.*

**TSA (Target System Assessment Report)** — The consolidated results of assessment to determine the architecture of the target system. *Plan phase.*

**PENV (Prepared Environment)** — The target environment, prepared for operation, with the operational software and hardware installed. *Implement phase.*

#### 2.1.4.5 Evolution Strategy Information

*Composite document acronym: ESI*

One of the central activities of Renaissance is the development of a complete and coherent set of evolution strategies to be applied to all or parts of your legacy system.

**ES (Evolution Strategy)** — One of the six individual evolution strategies provided by the Renaissance method (continued maintenance, re-vamping, re-structuring, re-architecting, re-design with reuse, replacing). Each such strategy may be associated with a particular component of the system under assessment. *Plan phase.*

**PCS (Proposed candidate strategies)** — The set of individual evolution strategies associated with individual system components that has been assessed and determined to be acceptable. *Plan phase.*

**PES (Possible Evolution Strategies)** — The set of individual Renaissance evolution strategies that have been identified for possible application during evolution. This set will undergo subsequent assessment for feasibility. *Plan phase.*

**SES (System Evolution Strategy)** — The overall combined evolution strategy for evolving the current system. This comprises the set of individual Renaissance evolution strategies that will be applied to different individual system components during evolution. *Plan phase.*

#### 2.1.4.6 Transformation Information

*Composite document acronym: TFMI*

This section groups together the information that is managed as the transformation of the legacy system to the desired target system is carried out.

**PP (Project Plan)** — The implementation plan (e.g. schedule, resource allocation, etc.) for the evolution project after the Go decision has been made. *Implement phase.*

**CB (Component Batches)** — Groups of legacy system components that can be migrated together from the legacy system to the target system. *Implement phase.*

**DRC (Description of reusable components)** — An identification of those components of your legacy system that might be reused during evolution to the desired target system. *Implement phase.*

#### 2.1.4.7 Test Information

*Composite document acronym: TSTI*

The testing activity in a Renaissance evolution project is relatively separate from other activities, for the same motivations as seen in many traditional kinds of software development projects: to preserve independence and objectivity. The information managed during testing-related activities is grouped together in this composite document.

**TSP (Test strategy plan)** — Documents the approach that will be used to test, as the current system is gradually transformed to the target system. *Implement phase.*

**TD (Test Data)** — Data used for testing the transformed system during the implementation phase. *Implement phase.*

**TENV (Test Environment)** — The environment that is prepared for testing the system during implementation. *Implement phase.*

**TP (Test Plan)** — The plan for testing activities on the target system. More precisely, it concerns the test specifications. *Implement phase.*

**TR (Test Report)** — Results of target system testing. *Implement phase.*

**TSYS (Tested Target System)** — The target system after transformation and testing have been completed. *Implement phase.*

#### 2.1.4.8 Developed Target System Information

*Composite document acronym: DTSI*

After the legacy system has been successfully evolved to the desired target system, its delivery and deployment in its real operating environment is managed with a small set of artifacts representing significant testing, installation, and deployment milestones.

**ATS (Accepted Target System)** — The target system after successful acceptance testing. *Deliver phase.*

**DP (Deployment Plan)** — The deployment plan (schedule, resources allocated, etc.) for the target system. *Deliver phase.*

**ITS (Installed target system)** — The tested target system, delivered and installed, ready for acceptance testing. *Deliver phase.*

**MD (Migrated Data)** — The migrated data from the old to the new system. *Deliver phase.*

**OTS (Operational Target System)** — The completed (tested, accepted, installed) target system, operational onsite. *Deploy phase.*

**TM (Training Material)** — All artifacts related to training, from plans and schedules to training manuals. *Deploy phase.*

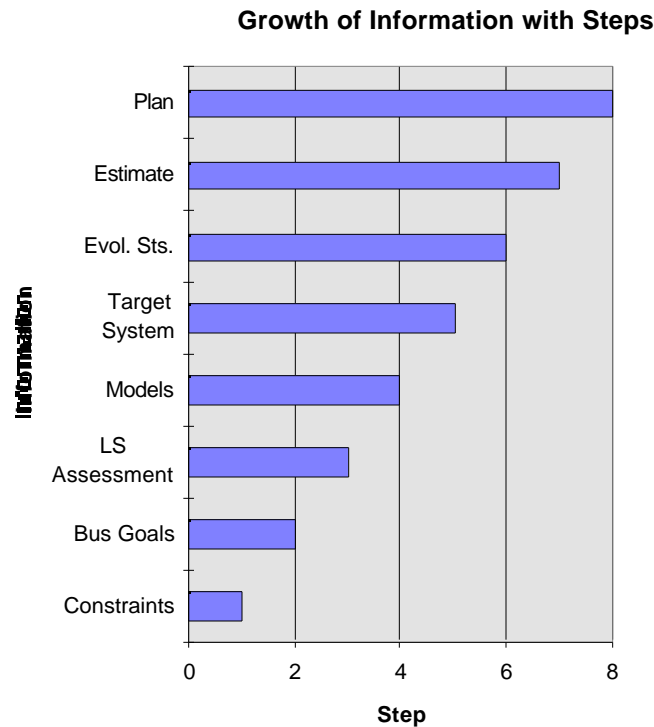
### 2.1.5 A journey: how information grows in Renaissance

In order to acquire an intuitive feeling for how the artifacts described in the previous section are created and used in Renaissance, it is useful to view an evolution project as a kind of *journey*. In a journey by automobile, the *plan* (who will drive which parts, how much fuel will be needed and so on) for a *route* from the *point of departure* to the *destination* can be formulated only when the *reason* for the journey (for example, the shortest route, the most scenic route) and the locations of the point of departure and destination on a *map* are known. Bearing this in mind, after unfolding the map, the steps in planning a journey are to determine the location of the point of departure, followed by the location of the destination, the identification of the possible routes between the two and the assessment of these, bearing in mind the reasons for the journey. The final step is to choose the best route.

Planning a Renaissance evolution project is analogous to planning a journey. In this case the first step is to define any constraints on the project (*reasons*) and the business goals and processes (*map*) of the organisation. The next step is to obtain an understanding of the legacy system (*point of departure*) and how it supports the business goals and processes, followed by the formulation of a (or several alternatives) target system (*destination*) and corresponding evolution strategy(ies) (*route*) which can then be assessed taking into account any constraints imposed. These activities create the pool of information necessary to formulate an evolution project plan (*plan*) which will be realistic and workable.

Journey Planning	Evolution Planning	Composite Documents
Determine reason for journey	Define constraints on evolution project	BUSI
Unfold map	Define business goals and processes	BUSI
Locate point of departure	Obtain understanding of legacy system	CSI
Locate destination	Formulate possible target systems	TSI
Identify possible routes	Formulate possible evolution strategies	ESI
Assess possible routes	Assess target system/strategy alternatives	ESTI
Choose route	Choose best alternative	ESTI
Plan journey	Plan evolution project	TFMI

Now consider how information is gathered into “composite documents” during this process, building on the information that was gathered in each of the previous steps. The following diagram illustrates how the amount of information grows as you proceed through the method.



Note how the amount of information managed grows with each succeeding step—this occurs partly because the information obtained in each step is necessary not only for carrying out the next step, *but for all subsequent steps*. To see this, let us walk through the evolution planning phase informally once again—this time from an information-gathering perspective:

1. Constraints on any target system, timescale and budget must be specified *before* an evolution assessment can be started in order to avoid unnecessary, and possible costly, assessment work.
2. Business goals and processes are then described.
3. The understanding of a legacy system comes from exercises which assess it. This assessment involves collecting, documenting, and modelling information concerning it and its various components and interpreting that information against the business goals and processes *previously described*.
4. A target system (and corresponding architecture, or several alternatives) and corresponding evolution strategy (or several alternatives) are formulated using, as guidelines, the legacy system assessment and business goals and processes defined in the previous steps, and any imposed constraints.
5. The activities of Step 4 continue through this step.
6. To decide whether it is worthwhile implementing a target system using a strategy, further assessment must be performed. This assessment involves modelling the target system and estimating what would be involved in its implementation in terms of time scales, resources and

costs and performing a cost benefits analysis on the results. *All the information collected during the previous steps* may be used to inform this estimating exercise.

7. Where several alternative target architectures and evolution strategies exist, a cost risk analysis can be performed to decide on the best strategy. Again, *all information collected in the previous steps* is used to inform this process.
8. If implementation is deemed worthwhile, an evolution project plan will be formulated. A good project planner will take into account information documented in all the previous steps. For example, the initial legacy system assessment will contain information about current staffing, the size of the legacy system and models showing where it supports business processes; the target system definition will contain information concerning any hardware and software changes which will need to be made to the organisation and the assessment of the target will map projected development activities and resources to a timescale and budget to form the foundation for a project plan.

### **The concept of the *repository***

From the above, it can be seen that each stage of evolution planning supplies information used by all subsequent stages. In Renaissance we think of planning information as being organised into a series of “placeholders” or “pigeon holes,” which we call the ***repository***. These are systematically filled as the various stages complete. Only when the requisite placeholders (“pigeon holes”) are filled by a previous step can any step commence.

It is important to understand that this repository is more conceptual than physical in nature. You do not necessarily need a special-purpose database to implement the repository—the term is only intended to indicate *some* kind of central resource in which you can gather, structure, and store information as you apply the method. Therefore, depending on how you customise Renaissance for your own organisation, the repository may take the form of:

- a database management system
- a collection of files in a directory structure
- a collection of papers stored in a file cabinet
- a mixture of paper-based and electronic data management

The exact nature of the repository mechanism is not important—only its ability to maintain information and make it accessible to you during the application of the method.

### 2.1.6 Tool support for Renaissance activities

Renaissance was conceived in such a way that it would not be necessary to purchase tool support that was specifically designed for the method. Not only would such an approach be very expensive and cumbersome, but it would also be unrealistic: the method is intended to be *customisable* for your projects and organisation, and no single support tool could handle all of the many possible scenarios.

We have instead chosen to arrange the method around a set of carefully chosen activities that are already well-known and accepted in the software development and management communities; and therefore already have a broad installed base of support tools of many kinds—many of which are probably already in use in your organisations. In this section we give an overview of the applicable tool support for the various activities of the method, providing references to other **RENAISSANCE** project reports where detailed information about individual products can be obtained.

Following the recommendations and references provided in this section, you should have few problems creating a robust and cost-effective tool support environment that is customised for your needs.

#### Evolution Planning

As noted in the informal walkthrough of the method, evolution planning can range from a high-level, paper-based assessment all the way to a detailed assessment with a full tool-support environment. In the following we highlight the areas in which you may find tool support.

*Business Process Description.* One of the first activities in evolution planning is the capture of the business processes implemented by your legacy system. This can be done at several possible levels, up to and including formal language descriptions. The Renaissance Report “Technology Selection” includes a comprehensive overview of related tool support, should you wish to deal with business processes in a more formal manner. In addition, you will find references to process support tools (e.g. InConcert, FlowMark, Staffware, Teamware) in Appendix A of the Renaissance Report “Client/Server Migration”. The techniques discussed below for system modelling (Unified Modelling Language diagrams, dataflow diagrams, etc.) can also be useful for modelling business processes. See the Renaissance Report “Architectural Modelling for Evolution” for more details.

*Context Modelling.* If you undertake detailed context modelling during evolution planning, there is tool support on the market for a broad range of modelling activities, including operational schemas, use case diagrams, data flow diagrams, and entity-relationship diagrams. The Renaissance Report “Architectural Modelling for Evolution” includes a full chapter on tool support for each of these activities.

*Metrics.* The detailed assessment of the technical quality of software in evolution planning can involve the application of *software metrics*. The field of software metrics has been mature for several years, and a large market of tools exists. The Renaissance Report “Evolution Planning”

contains a detailed discussion of tools that can support various kinds of metrics-related activities, from static analysis to process measurements.

*Risk Management.* Standard office automation tools are now sufficiently sophisticated to lend considerable support to risk management activities. Good results can be obtained with spreadsheet templates, and associated statistical functions provided by the spreadsheet programs.

*Size and Cost Estimation.* During assessment, you will be estimating the size and cost associated with candidate evolution strategies. Due to its importance in the field of software development management, there are now many tools of varying quality on the market to support estimation. Renaissance Report “Evolution Planning” includes a chapter that discusses many support tools, as well as trade-offs and pitfalls in using them.

## **Implementation**

The implementation phase much more closely resembles a traditional software development phase than evolution planning, and therefore you can expect to find adequate tool support for most activities in the traditional market.

*Project Planning and Management.* There are a large number of project planning tools available on the market (e.g. Microsoft Project). They are all perfectly suitable for use in evolution project planning.

*Technical Modelling.* During the implementation phase, you may undertake several technical modelling activities. Technical modelling in Renaissance is based upon techniques associated with the Unified Modelling Language (UML), including a variety of diagrams. Although the market for UML tool support is still young, there are already sophisticated tools available (e.g. Rational ROSE) and the market is expected to grow quickly. A discussion of tool support for technical modelling may be found in the Renaissance Report “Architectural Modelling for Evolution”.

*Testing.* A Renaissance evolution project is subject to the same requirements for testing at various levels (unit, integration, system level) as any traditional development project. A full range of tools is available on the market.

## **Delivery**

The delivery phase of Renaissance does not exhibit demanding requirements for specific tool support, except in one particular area.

*Data Migration.* The migration of legacy data to a new system can be a demanding and complex task, and an emerging market for tool support exists. These tools can assist you in transforming database schemas in a semi-automated fashion, transforming data formats, etc.

## Deployment

In the deployment phase, several types of tool support may be useful. If the system is to be deployed on many sites or servers—with multiple configurations—then version or configuration management tools will be critical. Furthermore, the distribution of the software could also be at least partially automated through some tools that are specialised for the distribution of software to multiple sites. Many such tools now exist on the market.

## Enterprise CASE tools—another approach

Although we have favoured the approach of encouraging the selection of individual tools to support the different phases, as outlined above, the class of tools known as *Enterprise CASE Tools* offers another approach that you may wish to consider, depending on your own circumstances. Usually these tools cover several phases, often starting from design and going through to delivery and deployment. (In fact, these tools are likely to have a run-time component.)

As a specific example of this approach, RENAISSANCE project partner Telesoft has customised the Renaissance method for the Telesoft software development lifecycle using the HPS (High Productivity System) enterprise CASE toolset from Seer.

## The RENAISSANCE project web site as a source of tool information

In order to provide users of the Renaissance method with consistently up-to-date information on tool support, the members of the project consortium have adopted a policy of posting this information on the project WWW site. Users of the method who are interested in tool support are strongly encouraged to consult this site for further information.

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

*This page is intentionally blank*

### 2.1.7 **Activities and tasks: the basic building blocks of Renaissance**

In the following sections a complete description of the method is provided, with the use of two fundamental concepts:

- *activities* identified by Renaissance are generic activities, which can be customised 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 interfaces, which are defined in terms of inputs and outputs. For 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;

Thus, you have a considerable amount of freedom in making use of activities and tasks to implement the Renaissance method for your own organisation. In the following, we discuss how to read the descriptive sections of this document and customise them for your own needs.

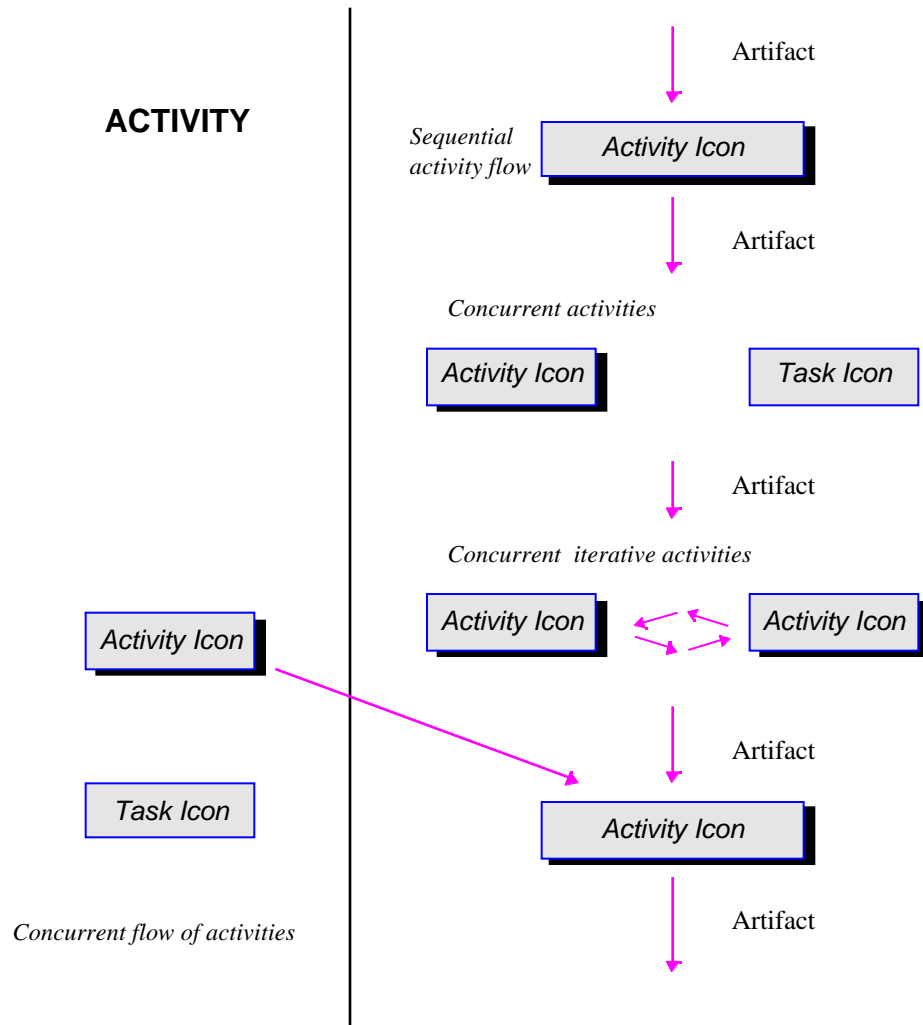
#### **Activities: Representing the Renaissance process**

Beginning with Section 2.2, the flow of work associated with the Renaissance method is described using a special notation, which we introduce and discuss now. In order to facilitate reading, we use a very simple notation, both graphical and textual, which comprises the intuitive concepts of

- activity,
- activity decomposition,
- inputs and outputs

However, for readers who are already familiar with a different process modelling language, the chosen representation can be easily mapped to that language.

The adopted graphic notation is summarised in Figure 21.



**Figure 21: The Renaissance notation for activities**

As illustrated, the following mapping rules hold:

- the intuitive notion of *activity* is mapped into a rectangle labelled with the activity name;
- the *control and data flow* among activities is represented by arrows from sources to targets of the flow;
- parallel or concurrent activities are represented in a simple side-by-side fashion;
- activities that are carried out iteratively with each other are represented with a circular set of arrows between them;
- major parallel activity flows are separated by a line (much as in structure diagram notation);
- an activity can be decomposed into *sub-activities*; in that case, the sub-activity description will include the same form of graphical decomposition—whereby the shadowed border indicates that the activity is further decomposable;

- the word *task* denotes an activity which is not further decomposed, and its associated icon does not have a shadowed border, in order to distinguish it from a (decomposable) activity icon;
- *artifacts* (or *products*) are usually associated with the inputs and outputs of activities, and are denoted with a diamond label.

### **Tasks: customising Renaissance for your organisation**

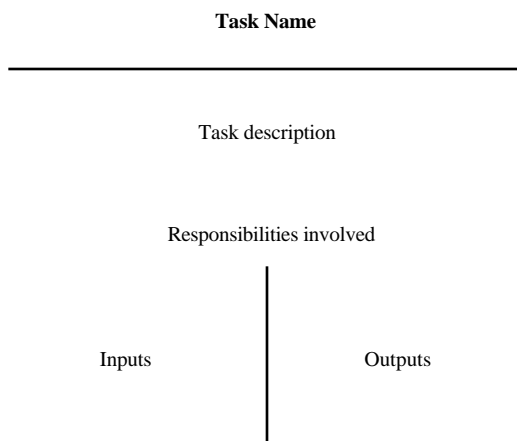
While a graph defines the activity decomposition and control flows among activities, *tasks* describe the actions which have to be done in order to accomplish the intended objective, and also the inputs and the output products. This means that Renaissance defines a general reengineering method, which can be *customised* 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 describe actions to perform, they need a specialised notation. The 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 a 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 customisation of the method for specific companies or projects. The graphical representation of a task is shown in Figure 22.



**Figure 22: The Renaissance notation for tasks**

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

---

<b>Previous task:</b>	an indication of which task immediately preceded this task.
<b>Next task:</b>	an indication of which task immediately succeeds this task.
<b>References:</b>	References to <b>RENAISSANCE</b> consultancy reports and other sources of information.
<b>Task description:</b>	a description of the actions performed within the task.
<b>Inputs:</b>	the list of elements that must be available in order to allow to start task execution.
<b>Outputs:</b>	the list of elements that must be available at the end of the task execution.
<b>Responsibilities involved:</b>	the list of responsibilities involved to perform the task.

The inputs and outputs listed for each individual task are intended to indicate the *most important* artifacts that are concerned with the execution of that particular task. As noted in Section 0, however, *all* previously produced artifacts are generally relevant for the execution of many tasks (especially during the evolution planning phase). Therefore, the description of the specific inputs and outputs of any particular task should not be interpreted in a strict fashion, but only be considered as an aid to focus on particular artifacts at that particular time.

#### **Notation for artifact membership in composite documents**

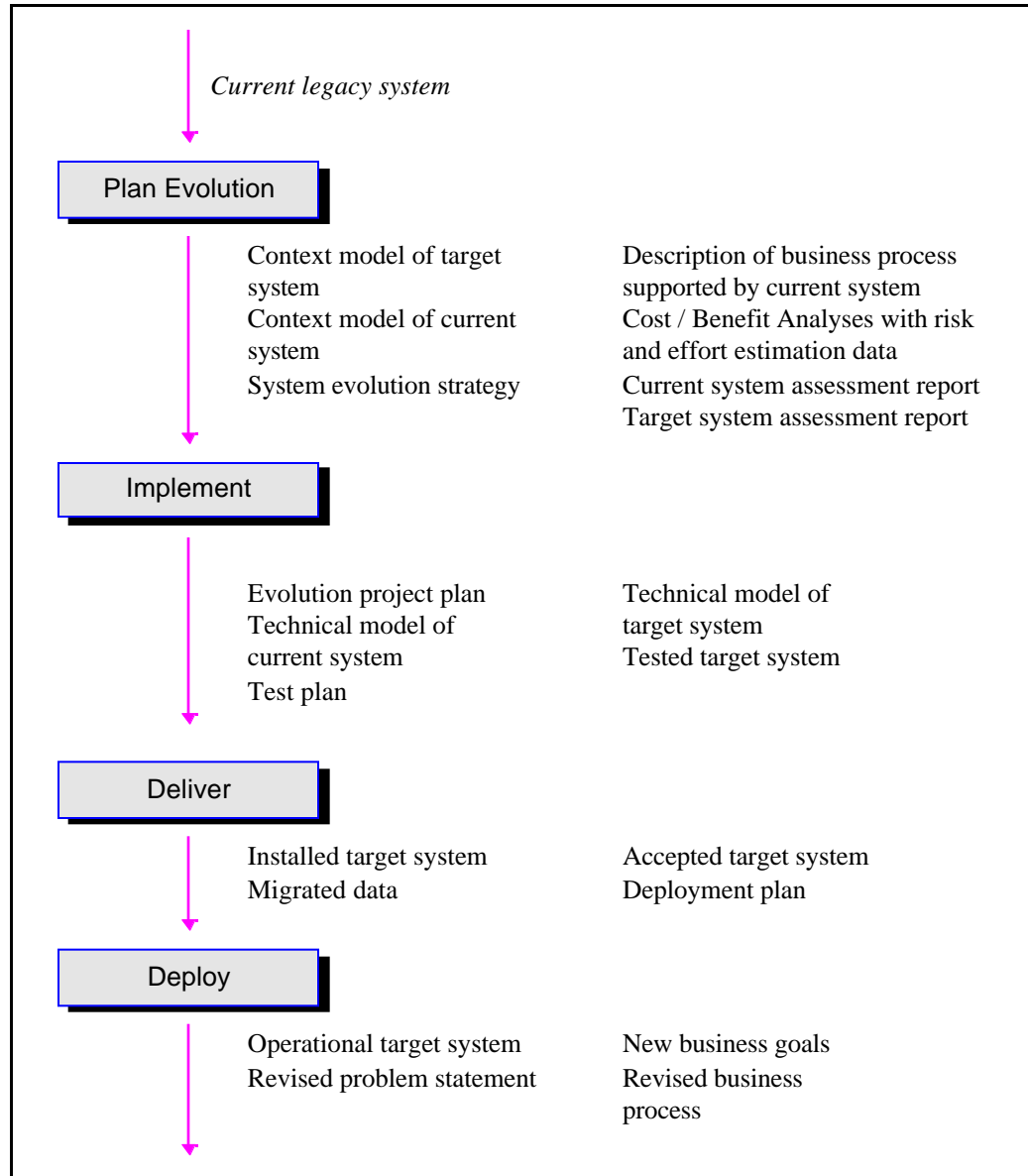
In order to highlight how the artifacts take part in composite documents (see Section 2.1.4), a special notation is used in the textual description of those artifacts in the task descriptions.

<Compound Document Acronym>.<Artifact Acronym>

For example, the artifact “Business Goals” is part of the “Business Information” composite document, and is denoted

BUSI.BG

## 2.2 The Four Phases of the Method



### Top-Level Description of Renaissance Phases

With the business goals in mind, the legacy system is investigated with the aim of assessing its technical quality and business value, as well as 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 or a migration strategy that has been imposed by higher management. Note that when a system includes several programs, or *components*, it is possible that

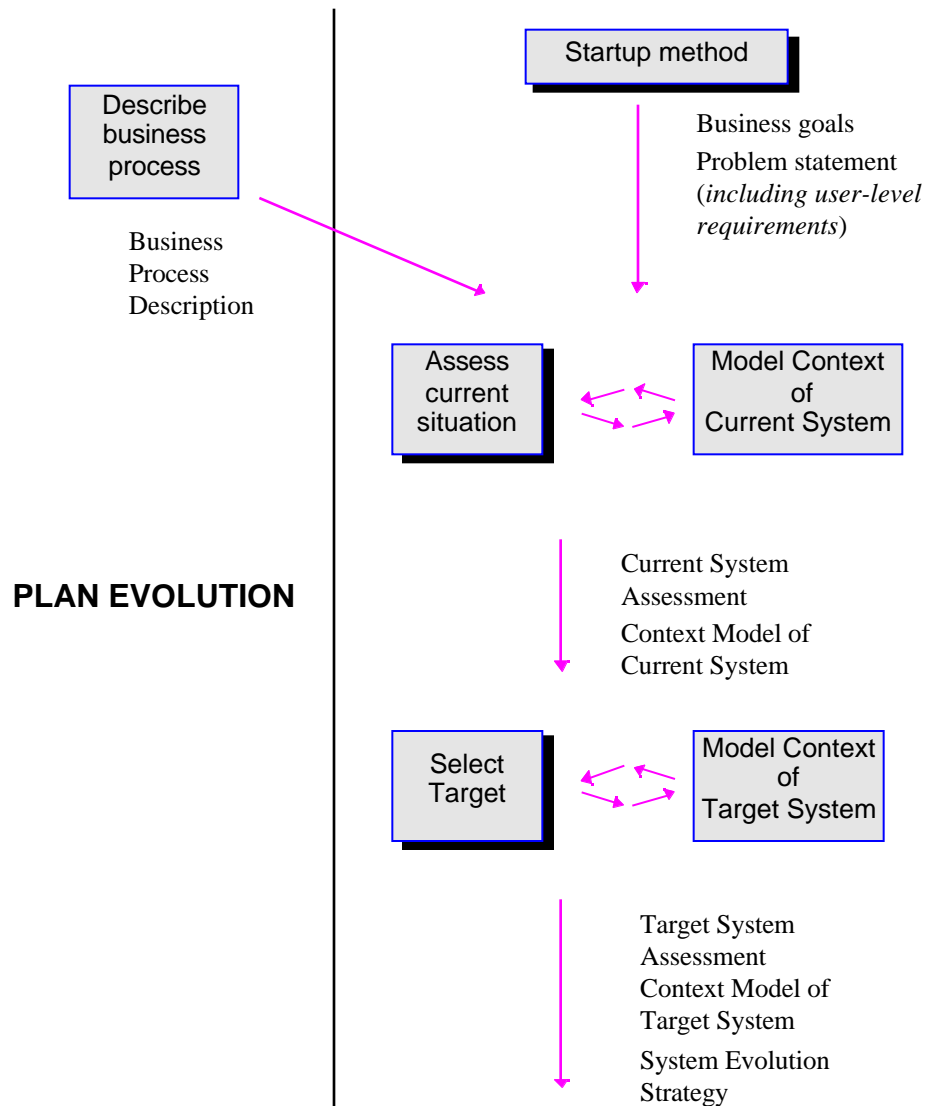
several different evolution strategies might be applied to the different components comprising the system.

The final result of this technical and economical assessment is the selection of a strategy for evolving the system (or perhaps a decision *not* to evolve the system).

After this Go/NoGo decision is taken, a detailed plan for driving the evolution project is created, in order to implement the desired 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 application of the method.

## 2.3 PLAN EVOLUTION



### Description of the Plan Evolution Phase

The first important milestone in the Renaissance method is a viable, cost effective *system evolution strategy* which can be presented to higher management. The goal of this phase is to determine a set of candidate Renaissance reengineering strategies needed to address the business goals of the company and to develop an overall approach—supported by various kinds of assessment artifacts such as risk analysis and size/cost estimation—that applies one or more of these individual evolution strategies to different components of the system to achieve the desired effect in the most efficient manner.

The process of determining candidate strategies consists of:

- assessing the technical quality and business value of the current system;
- refining this value according to the business goals; and
- assessing the desired system and providing a transformation mapping between the existing system and the desired one.

The activity of *assessing* the current system is supported by an activity of *modelling* the current system. This modelling activity is carried out in order to increase your level of knowledge about the system on a conceptual level (e.g. its logical structure, functionality, performance characteristics, etc.). As seen in the diagram, assessment and modelling are carried out *iteratively* with each other—each iteration of assessment and modelling therefore allows for progressively more detail and accuracy, until you have reached a sufficient level of understanding and evaluation of the current system for your purposes.

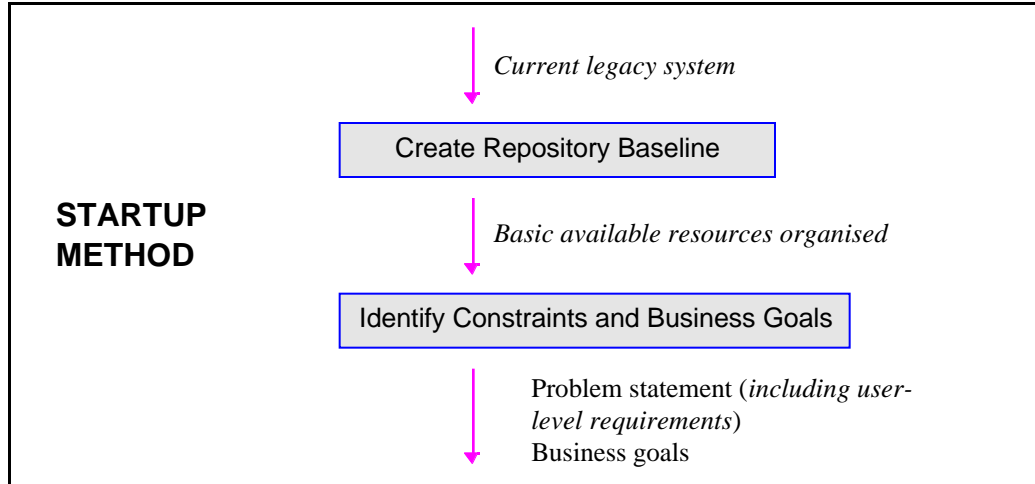
The next step identifies a set of possible strategies to be applied to the current system to migrate toward the target system. Once again, this process of strategy selection and evaluation is supported by a process of modelling the conceptual context of the target system, in order to gain a deeper understanding of the envisioned logic and architecture of the future system. Likewise, strategy selection and modelling are carried out *iteratively*, so that the level of detail and accuracy can grow gradually and be stopped when a sufficient level has been reached. In this fashion, you do not waste valuable effort and resources unnecessarily.

Which will be finalised first? Strategy selection or target architecture? In the absence of a directive from higher management, the target architecture is likely to be finalised before the strategy. After all, the only way you can ensure that your strategy is workable is by knowing that the architecture is available in the required time frame, supportable (e.g. with staffing), and at the right price—otherwise, the risk and cost estimation cannot be completed. At the beginning, there will be many high-level alternative strategies and architectures. After several iterations, there will be few or just one detailed candidate (e.g. “Client-Server Migration to CORBA on this vendor’s platform”).

When you are satisfied that a sufficient level of evaluation has been attained, an overall, *system-level* evolution strategy is developed to be followed in the subsequent activities.

It should be noted that one possible outcome of this activity is a NoGo decision, by which it is concluded that the best course of action is to forego evolution and remain with the current system.

### 2.3.1 Startup Method



#### Activity Description

The initial steps are taken to make the method fit the requirements of the evolution project to be managed. The following tasks are involved in method startup:

1. Organise system baseline repository
2. Identify constraints and business goals

The first task consists of making available as much information as possible concerning the current system and its environment to the evolution planning team. This will include organisation of sufficient pre-existing documentation (wherever possible), representatives from the user community, domain experts—in general, any information necessary to understanding the departure point of the system under consideration. Naturally, the documentation gathering (and producing) process will continue throughout the re-engineering project—here we are merely concerned with creating the starting point.

The second task consists of utilising the resources obtained above to elicit a clear statement of the evolution problem associated with the current system—the *problem statement*. In particular, the *a priori* constraints on the evolution project (perhaps imposed by management directive) that must be elicited so that they can be taken into account during assessment. Therefore, this task also involves a critical review of the *business goals* of the organisation that is preparing to embark on the evolution project, to determine how they will affect the decision-making process during assessment, and conversely, what impact the evolution project could have on the business goals.

*This page is intentionally blank*

### 2.3.1.1 Create Repository Baseline

<b>PLAN EVOLUTION : STARTUP METHOD : CREATE REPOSITORY BASELINE</b>	
Organise the basic set of information on the system under consideration and customise the method and repository for application of the method.	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• Evolution Modeller</li> <li>• Software Project Manager</li> <li>• Software Engineer</li> </ul>	
<p><b><i>Inputs</i></b> (none)</p>	<p><b><i>Outputs</i></b> initial repository organisation</p>

*Previous task:* This task initiates method startup.

*Next task:* Identify Constraints and Business Goals, page 50

*References:* Section 0 of this document, page 30

#### Task description

The process of evolution planning begins with the organisation of a baseline set of information on the current system. As noted in Section 0, the Renaissance method is intended to be customisable to be company and project specific. Thus, the first step is to set up the tasks and activities, and resulting artifacts, that will exist for this particular evolution project and your particular organisation. As noted in Section 0, the execution of the activities and their associated tasks, together the production of the various associated artifacts (reports, documentation, plans, even the system itself), is organised around the idea of a “repository” that contains the results. The repository is gradually filled as the method is progressed.

Thus, in this task the ground is prepared by creating high level “placeholders” or “pigeon holes” for the artifacts to be contained in the repository—a kind of baseline that establishes the current state of the system, the point of departure for the method. The exact composition of the repository will depend on your organisation’s nature and needs. This is part of the customisation of the method.

Note that it will already be possible to place important information in the repository, such as existing system documentation, the identification of a user community, domain experts, etc.—any information that is useful in providing a snapshot of the current situation.

**Inputs**

No specific artifact, since this task is carried out at method startup.

**Outputs**

There is no specific artifact as an output of this task—rather, the initial organisation of the repository is created.

**Responsibilities involved**

EM - Evolution modeller understands which types of characteristics are needed to determine the evolution strategy.

SWPM - Software project manager brings in managerial responsibility.

SWE - Software engineer brings technical knowledge.

In addition to these formal responsibilities, it is expected that more clerical tasks, such as looking for information whose location is not known, will be carried out by junior personnel, such as a System Librarian/Archivist with a knowledge of where legacy system artifacts might be archived.

### 2.3.1.2 Identify Constraints and Business Goals

<b>PLAN EVOLUTION : STARTUP METHOD : IDENTIFY CONSTRAINTS AND BUSINESS GOALS</b>	
Elicit all explicit and implicit business goals and constraints on the solution for evolution	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• Evolution Modeller</li> <li>• Software Project Manager</li> <li>• Operations Manager</li> <li>• User</li> </ul>	
<p><b><i>Inputs</i></b> (none)</p>	<p><b><i>Outputs</i></b> Problem Statement Business Goals</p>

*Previous task:* Create Repository Baseline, page 48

*Next task:* This is the final task in method startup. Following are the two concurrent activities Assess Current Situation (page 54) and Model Context of Current System (page 64).

*References:* none..

#### Task description

The current evolution problem to be solved is described in technical and business terms—in particular, *any pre-existing constraints* on the form of the solution are elicited and put into explicit form for use during assessment for evolution. This requires the preparation of a “problem statement” that not only states what *should* be done, but what *cannot* be done—that is, any constraints, either technical or business-related. The elicitation of business-related constraints involves a review of the business goals of the organisation.

It is important to keep technical constraints separate from business-related constraints as much as possible. Often, what seems to be a technical issue is really a business goal in disguise, and can lead to unnecessary constraints being identified. For example, a technical constraint that “the system must be client-server” may in fact only be an analyst’s interpretation of the business goal that “we want to grow market share”—which may permit different technical solutions.

As a rule, the minimum possible set of constraints—both technical and business-related—should be identified, giving the maximum leeway for the subsequent assessment activities. Only when constraints (mostly technical) are imposed by user-level requirements (such as the need to support given standards or have certain response times), or when

constraints (mostly business-related) are imposed by upper management (e.g. to run on the new company-wide platform) should they be included.

The Problem Statement should document mostly the technical issues, including those related to user requirements. The Business Goals should contain a clear statement of those issues related to company policy. This will help inform the assessment activity.

### **Inputs**

(none)

### **Outputs**

BUSI . PS - Problem Statement

BUSI . BG - Business Goals

### **Responsibilities involved**

EM - Evolution modeller understands which types of characteristics are needed to determine the evolution strategy.

SWPM - Software project manager will have knowledge of technical problems associated with current system.

USER - Users will input to user-level requirements and constraints

OPMG - Operations manager. Operators will often have ideas about how a system could be improved from the operations side

Note that there will also be involvement at the high management level in the definition of goals and constraints—clearly, they must be accepted by the company at this level.

### 2.3.2 Describe Business Process

PLAN EVOLUTION : DESCRIBE BUSINESS PROCESS	
Elicit an explicit definition of the business process(es) supported by the current system and to be supported by the target system.	
<p><b>Responsibilities Involved</b></p> <ul style="list-style-type: none"> <li>• Application Business Expert</li> <li>• Software Engineer</li> </ul>	
<p><b>Inputs</b></p> <p>(none)</p>	<p><b>Outputs</b></p> <p>Business Process Description</p>

*Previous task:* This task is concurrent with method startup

*Next task:* Assess Current Situation (page 54), which is concurrent with Model Context of Current System (page 64).

*References:* Renaissance Report “Technology Selection”, Renaissance Report “Client/Server Migration”

#### Task description

Reengineering is closely related to the business processes of the organisation, because the current system was built to support those processes in some way. It is important to render these processes as *explicit* as possible, by describing them in a systematic way; and by revealing business rules that are embedded in the current system itself as reengineering progresses. The explicit revelation of the business processes supported by information systems creates opportunities within a larger context of reengineering the overall business.

At this stage in the method, the task is to make a first attempt to provide an explicit description of the business processes supported by the current system. You may wish to use some descriptive formalisms that have proven successful in workflow management technology (e.g. Lotus Notes). A discussion of tool support issues may also be found in Section 2.1.6.

#### Inputs

(none)

#### Outputs

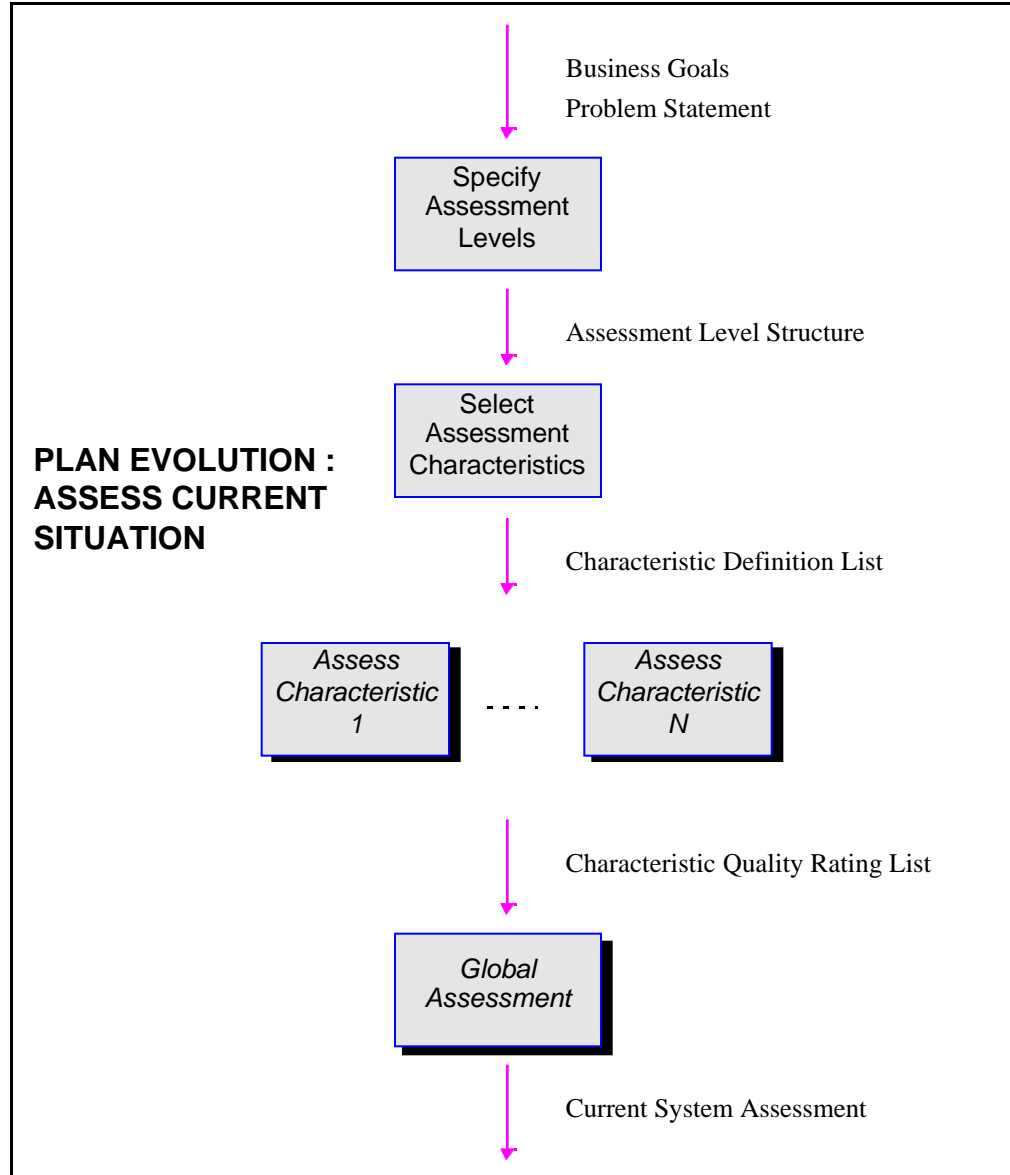
BUSI.BPD - description of the business processes supported by the current system and to be supported by the target system.

**Responsibilities involved**

ABE - Application Business Expert to provide business rules

SWE - Software engineer to determine their implementation in the system

### 2.3.3 Assess Current Situation



#### Activity Description

This activity is concurrent with the activity Model Context of Current System (page 64).

The current system is assessed in terms of:

- a quality factor, and
- a business value,

which depend on many 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 the level of assessment to be carry out—the Assessment Level Structure. This will involve a decision concerning which components of the system to assess, and at which level of detail to make the assessment.

Then, a set of characteristics is identified (i.e. the architecture, the number of 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.

### 2.3.3.1 Specify Assessment Levels

PLAN EVOLUTION : ASSESS CURRENT SITUATION : SPECIFY ASSESSMENT LEVELS	
Analyse the structure of the system, in order to determine the components that will be assessed and at which level of detail.	
<p><b>Responsibilities Involved</b></p> <ul style="list-style-type: none"> <li>• Evolution Modeller</li> <li>• Software Engineer</li> </ul>	
<p><b>Inputs</b></p> <p>Business Goals Problem Statement</p>	<p><b>Outputs</b></p> <p>Assessment Level Structure</p>

*Previous task:* This is the first task of Assess Current Situation. Previous tasks are Describe Business Processes and Startup Method, executed concurrently with each other.

*Next task:* Select Assessment Characteristics, page 58.

*References:* Renaissance Report “Evolution Planning” - ‘Assessment for Evolution’ subsection ‘Information Gathering and System Understanding’.

#### Task description

Renaissance encourages assessment to be carried out in the most cost-effective manner, and therefore provides for several possible *levels*. At one extreme, a top-level assessment of the system as a whole may be carried out, using the technique of expert opinion; at the other extreme, assessment may be carried out over the entire structural hierarchy of system components—each assessed individually—using a formal, detailed metrics-driven assessment procedure. In this task, this Assessment Level Structure is determined and documented, in which you select the components upon which you wish to carry out assessment. The Assessment Level Structure will drive the assessment procedure in subsequent activities, and will also drive the selection of individual Renaissance evolution strategies to be applied to the individual components of the system.

In summary, this task allows the assessment team to exert control over the balance of effort and detail in the assessment procedure.

#### Inputs

BUSI . BG - the business goals of the organisation.

BUSI . PS - the problem statement

**Outputs**

CSI . ALS - Assessment Level Structure

**Responsibilities involved**

EM - Evolution modeller brings in the judgement needed to determine what levels of assessment (and on which components) are necessary.

SWE - Software engineer brings technical knowledge.

### 2.3.3.2 Select Assessment Characteristics

<b>PLAN EVOLUTION : ASSESS CURRENT SITUATION : SELECT ASSESSMENT CHARACTERISTICS</b>	
Decide which current system characteristics must be assessed, relative to the business goals of the organisation, in order to determine the Business Value and Technical Quality	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• Evolution Modeller</li> <li>• Software Engineer</li> <li>• User</li> <li>• Operations manager</li> </ul>	
<b><i>Inputs</i></b> All business information	<b><i>Outputs</i></b> Characteristic Description List

*Previous task:* Specify Assessment Levels, page 56

*Next task:* Assess Characteristic #i, page 60

*References:* Renaissance Report "Evolution Planning" - 'Assessment for Evolution'

#### **Task description**

A system can be thought of as a hierarchy of interacting elements. For the purpose of system assessment within Renaissance, these elements can be hardware, support software, organisation and application software. Each element type consists of components (which, in some cases, may be broken down further into constituents) possessing characteristics which can be measured. This task involves a review the elements of the current system to identify components and characteristics, considered important the organisation performing the assessment, which can be assessed to determine its Business Value (BV) and Technical Quality (TQ).

#### **Inputs**

BUSI - *all business information*

#### **Outputs**

CSI.CDL - List of characteristics deemed relevant for assessing the Business Value and Technical Quality of the system

**Responsibilities involved**

EM - Evolution modeller understands which types of characteristics are needed to determine the evolution strategy.

SWE - Software engineer brings technical knowledge from the software point of view.

USER - To provide information on system quality from a user point of view

OPMG - The operations manager technical knowledge from the operations and hardware point of view

### 2.3.3.3 Assess Characteristic #i

<b>PLAN EVOLUTION : ASSESS CURRENT SITUATION : ASSESS CHARACTERISTIC #I</b>	
Assess characteristic and assign a quality rating to it.	
<b>Responsibilities Involved</b>	
<ul style="list-style-type: none"> <li>• Evolution Modeller</li> </ul>	
<b>Inputs</b> Characteristic Definition All business information	<b>Outputs</b> Characteristic Quality Rating

*Previous task:* Select Assessment Characteristics, page 58.

*Next task:* Global Assessment, page 62

*References:* Renaissance Report "Evolution Planning" - 'Assessment for Evolution'

#### Task description

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

Assigning a quality rating is done by identifying quantifiable attributes of the characteristic, which are then assessed and given a score between 1 and 4 (where 1 is a low score).

The assessment can be carried out using methods ranging from expert opinion to formal metrics, at the system, component or component constituent 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'.

For example, the complexity of an application system can be assessed by comparing the number of components it contains to the number of relationships between them.

#### Inputs

CSI . CD - Definition of the characteristic. Note here that we will often rely on intuitive knowledge of the characteristic rather than explicit knowledge.

BUSI - *All business information*

**Outputs**

CSI . CQR - Characteristic Quality Rating

**Responsibilities Involved**

EM - the evolution modeller performs the assessment.

### 2.3.3.4 Global Assessment

PLAN EVOLUTION : ASSESS CURRENT SITUATION : GLOBAL ASSESSMENT	
Determine whether the system is a candidate for evolution.	
<p><b>Responsibilities Involved</b></p> <ul style="list-style-type: none"> <li>• Software Engineer</li> <li>• Application Business Expert</li> </ul>	
<p><b>Inputs</b></p> <p>Characteristic Quality Rating List</p>	<p><b>Outputs</b></p> <p>Current System Assessment</p>

*Previous task:* Assess Characteristic #i, page 60

*Next task:* This is the final task in Assess Current Situation, which is concurrent with Model Context of Current System. The next tasks are Select Target and Model Context of Target System, which run concurrently with each other.

*References:* Renaissance Report "Evolution Planning" - 'Assessment for Evolution'

#### Task description

According to the Assessment Level Structure, selected elements of the system will have been assessed at various levels, for various components. Each assessed component will have an assessment rating.

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 called *Portfolio Analysis* and involves plotting the TQ and BV on a graph. The values for many systems may be plotted using the same axes. As a general rule, those systems with a high BV but low TQ are good candidates for evolution, that is, those in the bottom right quadrant of the graph. This is described in the subsection 'Interpretation of Results'.

**Inputs**

CSI . CQRL - A list of assessed system characteristics.

**Outputs**

CSI . CSA - A Current System Assessment based on its Technical Quality and Business Value, stating whether or not it should be considered for evolution.

**Responsibilities involved**

SWE - Software engineer to provide Technical Quality assessment

ABE - Application Business Expert to provide Business Value assessment

### 2.3.4 Model Context of Current System

PLAN EVOLUTION : MODEL CONTEXT OF CURRENT SYSTEM	
<p>A context model is made to document the current system from business, functional, structural, and physical viewpoints. This information is used in the process of assessing the state of the legacy system and evaluating candidate strategies for evolution.</p>	
<p><b>Responsibilities Involved</b></p> <ul style="list-style-type: none"> <li>• Software Engineer</li> <li>• Application Business Expert</li> <li>• Legacy Functional Expert</li> <li>• Legacy Implementation Expert</li> <li>• User</li> </ul>	
<p><b>Inputs</b> (the legacy system itself)</p>	<p><b>Outputs</b> Context Model of Current System</p>

*Previous task:* This task begins when Assess Current Situation and Model Context of Current System finish, and is carried out *concurrently* with Assess Current Situation. See page 44 for an overview of these relationships.

*Next task:* The next tasks are Select Target and Model Context of Target System, which run concurrently with each other. See page 44.

*References:* Renaissance Report “Architectural Modelling for Evolution”, Chapter 2.

#### Task description

The *system context modelling* process uses 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. To assess a legacy system and then make an informed decision about an evolution strategy, it is important to have basic knowledge of how the system is organised. This task consists of applying a range of modelling techniques to document the current system. This documentation elicits information about business, functional, structural and physical aspects of the system.

- The business viewpoint has its focus on modelling the business processes that are supported by the system. Techniques such as operational schemas, use cases, and sequence diagrams are used.

- The functional viewpoint is used to discover and/or model the functionality and behaviour of the system. Techniques such as use cases, sequence diagrams, and data flow diagrams are used.
- The structural viewpoint is particularly important for identifying structural components that must undergo separate assessment exercises. The structural models form the base for translating the context model into a technical model described in the Implement phase of the method. Techniques such as block/component, and extended entity relationship diagrams are used to model the application and data structures of the system.
- The physical viewpoint documents the system as seen from its environment. This covers the communication and hardware dimension of the system. Hardware/network diagrams are used to model this viewpoint.

The models produced during this task need not model the whole system. A trade-off must be made regarding how much should be modelled versus the costs. The modelling should stop when the detail has reached a level sufficient to perform an informed assessment and identify different evolution strategies. To sum up, the context models of the current system provide high level documentation of the current system; have a baseline for assessment and evolution planning; and identify reusable parts and areas for reverse-engineering activities

The task should be carried out when business, technical, and economic conditions warrant. If only a quick high-level assessment is performed, the task may be omitted.

### **Inputs**

(The input is the legacy system itself.)

### **Outputs**

CSI . CMCS - Context Model of Current System, or to be more precise: a model divided into business, functional, structural and environmental views of the current system.

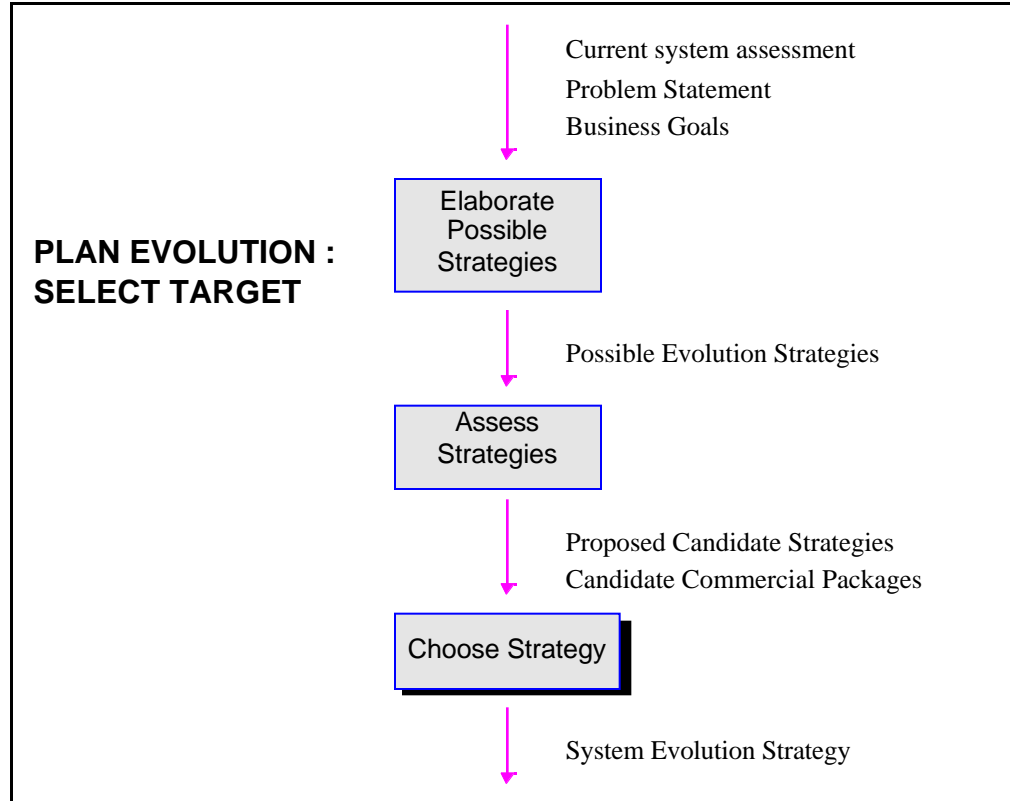
### **Responsibilities involved**

ABE, USER - Application Business Expert is a key resource for naming all business processes supported by the legacy system. The User elaborates with knowledge about actual use of the system.

LFE, LIE - Legacy Functional/Implementation Experts are key resources for modelling the functional, structural and operational aspects of the legacy system.

SWE - Software Engineering roles provide knowledge of context modelling techniques.

### 2.3.5 Select Target



#### 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.

This activity is carried out currently with Model Context of Target System (See page 80).

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.

A strategy is selected in the next step according to the results of these rankings. This becomes the system evolution strategy which drives the development of a plan for the evolution project in the next step of the method.

*This page is intentionally blank*

### 2.3.5.1 Elaborate Possible Strategies

<b>PLAN EVOLUTION : SELECT TARGET : ELABORATE POSSIBLE STRATEGIES</b>	
Alternative strategies for evolution are developed in this task, corresponding to the potential target architectures that are being elaborated in the (concurrent) target context modelling activity.	
<b><i>Responsibilities Involved</i></b>	
<ul style="list-style-type: none"> <li>• Evolution Modeller</li> </ul>	
<b><i>Inputs</i></b> Business Goals Context Model of Current System Target System Assessment (as it emerges) Context Model of Target System (as it emerges)	<b><i>Outputs</i></b> Possible Evolution Strategies

*Previous task:* This is the first task of the Select Target activity. The previous activities were Assess Current Situation and Model Context of Current System (see page 44).

*Next task:* Assess Strategies, page 70

*References:* Renaissance Report “Client/Server Migration”; Renaissance Report “Evolution Planning”; Renaissance Report “Architectural Modelling for Evolution”

#### **Task description**

Recall that there is another task running concurrently with this task: Model Context of Target System (page 82). The goal of that task is to elaborate possible *architectures* for the new, evolved system—for example, a distributed client-server architecture, or a centralised but completely redesigned architecture. As each such “candidate target architecture” emerges in that other task, a companion activity is carried out in this task consisting of elaborating a set of possible strategies for evolving toward that candidate architecture. There may be several possible strategies for evolving toward even a single architecture, depending on your business goals and any pre-existing constraints. (For example, you may elaborate several strategies which differ in their costs and associated risks, in order to give upper management several alternatives to choose from.)

The work in this task is clearly of an iterative nature, since it will be carried out as architectural information emerges from the context modelling task. There may be several iterations before a complete set of possible evolution strategies is settled upon.

### **Inputs**

BUSI . BG - The business goals of the organisation.

CSI . CMCS - The result of modelling of the current system.

TSI . TSA - Target System Assessment.

TSI . CMTS - Context Model of Target System. (Recall that all target system information will not necessarily be available at the start of this task. It will emerge gradually from the concurrent context modelling activities, and be elaborated as it becomes available.

### **Outputs**

ESI . PES - A set of possible evolution strategies, depending on the current situation, any pre-existing constraints, and the business goals of the company.

### **Responsibilities involved**

EM - evolution modeller has the knowledge to select strategies

### 2.3.5.2 Assess Strategies

<b>PLAN EVOLUTION : SELECT TARGET : ASSESS STRATEGIES</b>	
Possible evolution strategies are validated against business goals, user expectations, pre-existing constraints on the solution, and feasibility considerations.	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• Evolution Modeller</li> <li>• Software Engineer</li> <li>• Operations Manager</li> <li>• User</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>Business Goals Possible Evolution Strategies Problem Statement</p>	<p><b><i>Outputs</i></b></p> <p>Proposed Candidate Strategies Candidate Commercial Packages</p>

*Previous task:* Elaborate Possible Strategies, page 68

*Next task:* Choose Strategy, page 72

*References:* Renaissance Report “Evolution Planning”

#### **Task description**

Each possible strategy is checked against user expectations (evolution requirements) and business goals.

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 elaborated.

#### **Inputs**

ESI . PES - Possible Evolution Strategies.

BUSI . BG - Business goals

BUSI . PS - Problem Statement

#### **Outputs**

ESI . PCS - Proposed candidate strategies.

TSI . CCP - Candidate commercial packages

**Responsibilities involved**

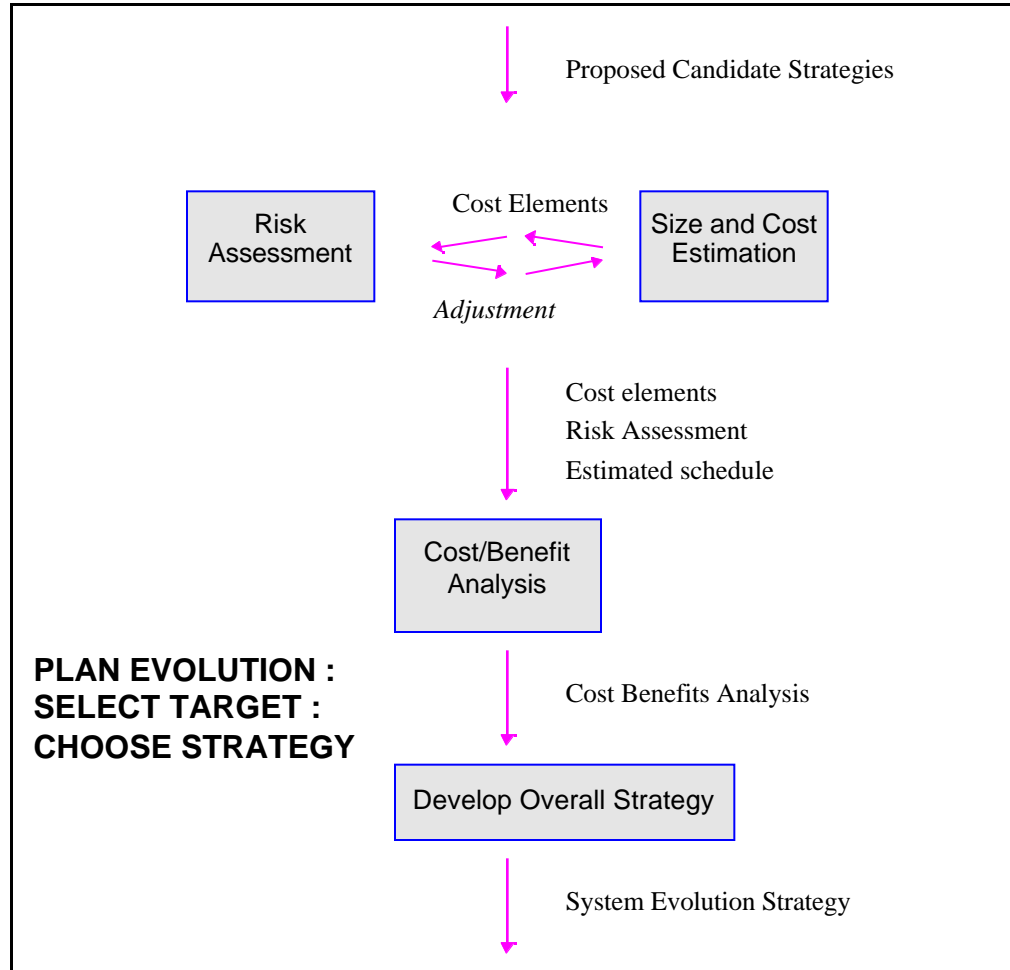
EM - The evolution modeller always has the lead at this point

SWE - The software engineer has knowledge about technical constraints and feasibility of solutions

USER - The users provide critical inputs on the requirements that the evolved system must fulfil.

OPMG - Operations Manager of the legacy system.

### 2.3.5.3 Choose Strategy



#### Activity Description

The current versus target system analysis has identified a set of candidate reengineering strategies and, consequently, an embryonic target architecture for evolving the existing system toward the desired one. The assessment may have been carried out at several structural component levels, and it is possible that individual candidate Renaissance evolution strategies may be associated with different components of the system.

These candidate strategies must now be evaluated on the basis of economic costs and risks, in order to have a set of candidate solutions. The benefits of each strategy are analysed, and, after combining all the results, the candidate strategies are ranked by preference.

Then, on the basis of other factors, such as the availability of expertise for the selected strategies, an overall, *system-level evolution strategy* is developed, that may specify the simultaneous application of several individual Renaissance evolution strategies—e.g. re-vamping one component of the system, replacing another component, etc. This System Evolution Strategy, along with the models, and estimation/risk documents, will subsequently serve as the roadmap for the

implementation phase, where the first activity will be the creation of a concrete project plan (complete with schedules, resource allocation, etc.) to carry out the evolution specified in this System Evolution Strategy.

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

### 2.3.5.3.1 Size and Cost Estimation

<b>PLAN EVOLUTION : SELECT TARGET : CHOOSE STRATEGY SIZE AND COST ESTIMATION</b>	
Estimate the size of the evolution project, and hence determine approximate manpower, time scales, and costs.	
<b><i>Responsibilities Involved</i></b>	
<ul style="list-style-type: none"> <li>• Evolution Modeller</li> <li>• Estimator</li> </ul>	
<b><i>Inputs</i></b> Business Information Proposed Candidate Strategy Current System Information Risk Assessment information (as it becomes available)	<b><i>Outputs</i></b> Cost Elements Estimating Schedule Estimating Assumptions

*Previous task:* This is the first task of Choose Strategy, and runs concurrently with the Risk Assessment activity (page 76). The previous activity was Assess Strategies (page 70).

*Next task:* Cost/Benefit Analysis, page 78

*References:* Renaissance Report "Evolution Planning", Chapter 'Estimating for Evolution'

#### **Task description**

Using the business and current system information and the proposed candidate strategy (and the risk assessment if a subsequent iteration), estimate the size and cost of the evolution project.

The estimator must quantify the effort (person/hours) and duration (calendar days) for the project's activities, suggest a schedule for the project, identify, breakdown and assess associated cost elements 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. Initially the estimator will, therefore, prepare Estimating Schedule, Cost Elements and Estimating Assumptions documents

This is described in the 'Estimating Process' subsection of the report referenced above.

**Note** - this task is performed *iteratively* with risk assessment to produce an estimate which can be used with the maximum confidence.

**Inputs**

CSI - *All available current system information*

BUSI - *All business information*

ESI . PCS - Proposed candidate evolution strategy

ESTI . RA - Risk assessment information, as it becomes available.

**Note:** This simplistic listing of inputs does not adequately reflect the fact that *all* documentation which has been produced up to this point is necessary and useful—the better the information, the more reliable the estimate.

**Outputs**

ESTI . CES - Cost elements report for the re-engineering project broken down into logical areas of cost. E.g. hardware, project management overheads

ESTI . SCHED - an estimated schedule for the project giving details for the basis of the cost estimate

ESTI . ASM - The assumptions on which the estimate is based

**Responsibilities involved**

**Note:** Estimating and Risk Assessment activities must be performed by personnel having a robust understanding of legacy system software, re-engineering, estimating, modelling and economic indicators. The estimator will require access to system analysts, business analysts, modellers, hardware experts and software experts. A reliable estimate could, therefore, require involvement of many people.

EST - Estimator

### 2.3.5.3.2 Risk Assessment

<b>PLAN EVOLUTION : SELECT TARGET : CHOOSE STRATEGY RISK ASSESSMENT</b>	
Assess the risk associated with the proposed candidate evolution strategy.	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• Quality Engineer</li> <li>• Estimator</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>Business Information Proposed Candidate Strategy Current System Information Estimating Information</p>	<p><b><i>Outputs</i></b></p> <p>Risk Assessment report</p>

*Previous task:* This is the first task of Choose Strategy, and runs concurrently with the Size and Cost Estimation task (page 76). The previous activity was Assess Strategies (page 70).

*Next task:* Cost/Benefit Analysis, page 78

*References:* Renaissance Report “Evolution Planning”, Chapter on “Risk Management”

#### **Task description**

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

Risks will be inherent in the initial Estimating Schedule, Costs Breakdown and Estimating Assumptions prepared by the estimator. The risk Assessor must identify and document the areas of risk within the estimating documents associated with a candidate strategy so that risk containment measures can be built into the eventual project plan

**Note** - this task is performed *iteratively* with size and cost estimation until containment measures have been proposed for all areas of risk identified in the estimating documents. Risks are inherent in our world and can't be eliminated, and so risk analysis should be carried out until deemed “acceptable.”

#### **Inputs**

All the documents which were required by estimating as well as the actual estimating documents are effectively needed, in addition to the inputs listed here.

ESI . PCS - Proposed candidate evolution strategy

CSI - *All available current system information*

BUSI - *All business information*

ESTI - *All available estimating information*

### **Outputs**

ESTI . RA - The risk assessment report

### **Responsibilities involved**

QE - Quality Engineer to provide risk assessment

EST - Estimator

**Note:** the same person who did the estimating is likely to do the risk assessment - either that or a specialised risk assessor - or will, at least *co-ordinate* the estimating and risk activities

### 2.3.5.3.3 Cost/Benefit Analysis

<b>PLAN EVOLUTION : SELECT TARGET : CHOOSE STRATEGY COST/BENEFIT ANALYSIS</b>	
Analyse the costs and benefits associated with each evolution strategy	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• Quality Engineer</li> <li>• Estimator</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>Business Information Proposed Candidate Strategy Current System Information Estimating Information</p>	<p><b><i>Outputs</i></b></p> <p>Cost/Benefits Analysis report Cost/Risk Analysis</p>

*Previous task:* The previous tasks are Risk Assessment (page 76) running concurrently with the Size and Cost Estimation task (page 76).

*Next task:* Develop Overall Strategy, page 80

*References:* Renaissance Report “Evolution Planning”, ‘Estimating for Evolution’

#### **Task description**

This task involves deciding whether system evolution is worthwhile and, if so, which is the best strategy to employ.

A Cost/Benefit analysis is used to decide the former and a Cost/Risk Analysis the latter.

Cost/Benefit Analysis compares the relative future costs and benefits for a strategy. It involves computing the Present Value of all future costs and benefits using a variety of economic indicators. These can be compared with equivalent values for other strategies.

The process is described in the subsection ‘Cost Benefit Analysis’.

Cost/Risk Analysis adds (or subtracts) an element of cost to an estimate for a strategy, depending on the associated risk, to adjust it to the same level of confidence as estimates for other strategies. It may be that a more costly strategy which is less risky than another will be chosen.

The process is described in the subsection ‘Cost Risk Analysis’.

The intangible benefits (e.g. improved customer satisfaction) should also be kept in mind during the analysis. Although these benefits are more

difficult to quantify, a good deal of progress has been made in recent years in strategies for capturing the value of these types of benefits.

**Inputs**

ESI . PCS - Proposed candidate evolution strategy

CSI - *All available current system information*

BUSI - *All business information*

ESTI - *All estimating information documents up to this point*

**Outputs**

ESTI . CBA - Cost benefits analysis

ESTI . CRA - Cost/risk analysis

**Responsibilities involved**

QE - Quality Engineer carries out the analysis

EST - Estimator

**Note:** The same observations that were made for estimating and risk activities apply here. (See page 76)

### 2.3.5.3.4 Develop Overall Strategy

<b>PLAN EVOLUTION : SELECT TARGET : CHOOSE STRATEGY DEVELOP OVERALL STRATEGY</b>	
Assess candidate evolution strategies and develop an overall system-level evolution strategy	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• Evolution Modeller</li> <li>• Software Engineer</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>Business Information Estimating Information Evolution Strategy Information Current System Information</p>	<p><b><i>Outputs</i></b></p> <p>System Evolution Strategy</p>

*Previous task:* Cost/Benefit Analysis, page 78

*Next task:* This is the final task of the Choose Strategy Activity—in fact, it is the final task of the Plan Evolution Phase. The next phase is the Implementation Phase (page 84).

*References:* Renaissance Report “Evolution Planning”

#### **Task description**

Based upon the results of the cost/benefit analysis, an overall system evolution strategy is developed. The current system will have been assessed at various structural levels (according to the Assessment Level Structure) and the results will have been documented in the Current System Assessment. Candidate evolution strategies for each part of the system that was assessed will have been identified and their respective costs and benefits documented.

In this task, the individual evolution strategies, together with the individual components of the current system on which they are to be applied, are organised and documented in the overall system-level evolution strategy (SES), which will serve as a roadmap for the developers of the evolution project plan in the next phase of the method. The actual process of how the individual strategies are implemented in different parts of the system will take place in the Implement phase, where the transformations corresponding to the implementations of the individual strategies will be planned and executed.

Alternatively, you might document and justify a NoGo decision not to evolve the current system. In that case, the method stops at this point.

**Inputs**

ESI - *All evolution strategy information that has been produced so far*

BUSI - *All business information*

CSI - *All current system information*

ESTI - *All estimating information available*

**Outputs**

ESI . SES - System evolution strategy.

**Responsibilities involved**

EM - Evolution modeller handles strategic considerations

SWE - Software engineer handles technical considerations

### 2.3.6 Model Context of Target System

PLAN EVOLUTION : MODEL CONTEXT OF TARGET SYSTEM	
<p>This task uses a number of modelling techniques to model the context of the target system. The models produced will be high level descriptions of the target system and will help decide how to get there and how costly and difficult the evolution path will be.</p>	
<p><b>Responsibilities Involved</b></p> <ul style="list-style-type: none"> <li>• Application Business Expert</li> <li>• Client Representative</li> <li>• Software Engineer</li> <li>• User</li> </ul>	
<p><b>Inputs</b></p> <p>Context Model of Current System Business Goals</p>	<p><b>Outputs</b></p> <p>Context Model of Target System</p>

*Previous task:* The previous tasks are the concurrently executed Assess Current Situation and Model Context of Current System. This activity runs concurrently with Select Target. See page 44 for an overview of these task relationships.

*Next task:* Together with the tasks of the Select Target activity, this is the final task in the Plan Evolution phase. The subsequent tasks form part of the Implement phase (page 84).

*References:* Renaissance Report "Architectural Modelling for Evolution" - Chapter 2

#### Task description

This task is carried out in an iterative manner with the "Select Target" activity. The context model of the target system can start off with the context model of the current system and refine this model, based on decisions made in the Select Target activity. These decisions are a combination of strategic business choices, technical opportunities, and economic trade-offs. If the steps in the refinement process are logged, there is an opportunity to reuse this refinement procedure in the later system transformation tasks of the re-engineering project.

For further information about the suggested techniques and models, see the reference mentioned above.

**Inputs**

CSI . CMCS - The Context Model of Current System is needed to make the roles involved aware of the current situation.

BUSI . BG - The Business Goals are needed to make them clear to the modeller.

**Outputs**

TSI . CMTS - Context Model of Target System, or to be more precise - a model divided into business, functional, structural and environmental views of the target system.

**Responsibilities involved**

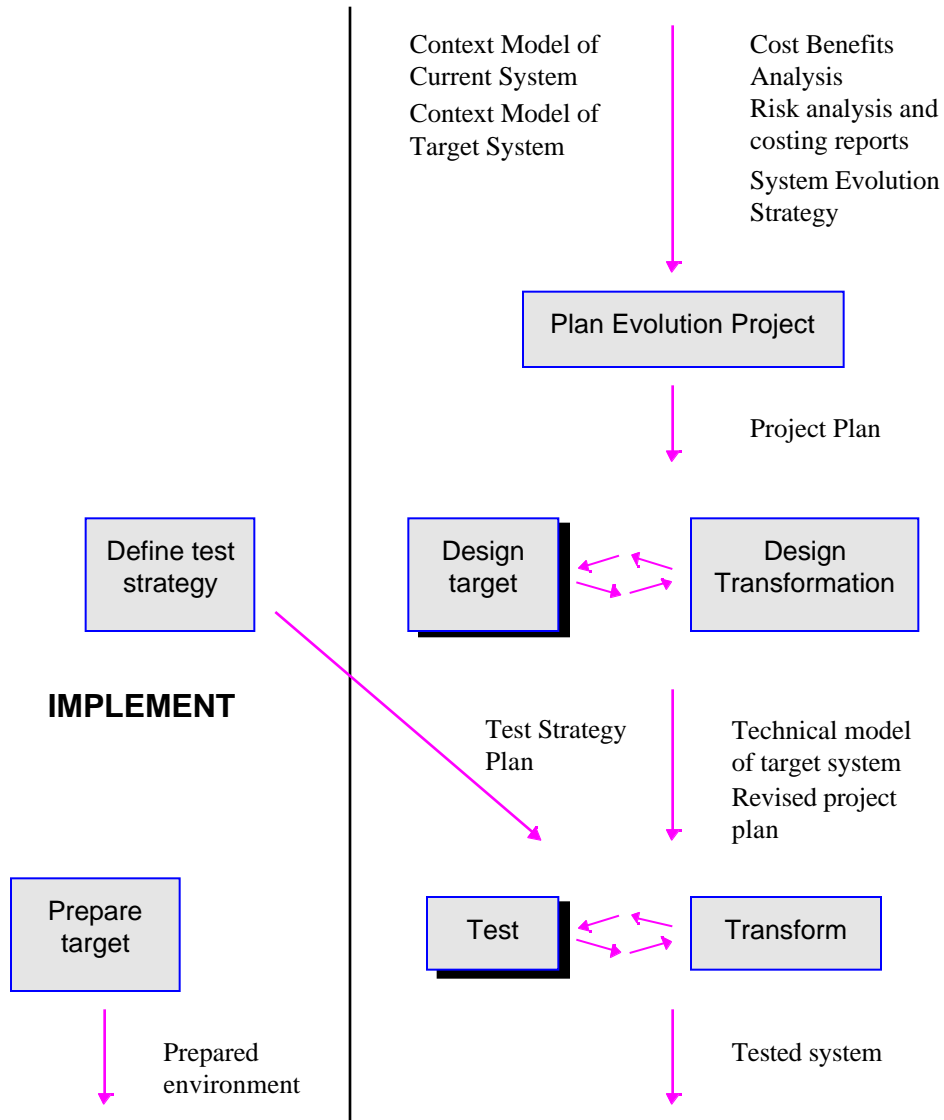
ABE - Application Business Expert provides input to which business processes that are needed in the target system.

CL - Client Representative provides input about prioritising business goals.

SWE - Software Engineering roles provide context modelling knowledge and architecture expertise.

USER - User has a QA role to ensure that the target system will add benefit to him/her.

## 2.4 IMPLEMENT



### Description of the Implement Phase

Upon exit from the Plan Evolution phase, a Go decision has been taken, and a System Evolution Strategy has been created. In this phase, an *evolution project* is planned, created, and executed.

The first step is the development of a detailed project plan. One purpose of this plan is to define the sequence in which evolutionary activities are to be carried out. For example, it may be possible to re-vamp the user interface at the same time that the migration to client-server architecture is taking place. Or, perhaps a re-architecting of the overall system must be carried out before any other evolutionary activities can take place. This

sequencing is spelled out, together with timetables, in the evolution project plan.

The next step is to prepare for the actual transformation of the legacy system. Two activities are carried out at the same time: the design of the architecture of the new system, and the design of the transformations that will evolve the legacy system to the new system. These two activities are so closely inter-related that they are best carried out *iteratively*, progressively refining the new architecture and the incremental transformations of the old architecture towards the new one.

The final implementation step is the execution of the incremental transformations. The Renaissance method takes advantage of the incremental nature of the transformations to encourage incremental *testing* of the transformations in an *iterative* fashion, so that the evolution of the old to the new architecture takes place in a controlled manner, and problems and errors emerge as early as possible.

During the course of the actual implementation activities, the operational environment in which the new system will actually run (that is, with the actual licensed commercial packages, etc.) is prepared, anticipating entry into the next phase of system delivery.

### 2.4.1 Plan Evolution Project

<b>IMPLEMENT : PLAN EVOLUTION PROJECT</b>	
Create a project plan to guide the process of evolving the current system to the target system.	
<b><i>Responsibilities Involved</i></b>	
<ul style="list-style-type: none"> <li>• Software Project Manager</li> </ul>	
<b><i>Inputs</i></b> Context Model of Current System Context Model of Target System System Evolution Strategy Business Goals Business Process Description Estimate Information	<b><i>Outputs</i></b> Project Plan

*Previous task:* The previous activities are those of the Plan Evolution phase.. This activity runs concurrently with Select Target. See page 44 for an overview.

*Next task:* This task leads into all subsequent activities of the Implement phase. The next tasks are the concurrently running tasks Design Target (page 90) and Design Transformation (page 96).

*References:* Renaissance Report “Evolution Planning” - “Plan Project”

#### **Task description**

This activity is the classical project planning activity. It is a realistic mapping of estimated activities to a formal project schedule. This enables maximum project organisation and control by project managers. The objective is to produce a project plan which contains:

- A description of what is to be produced
- A hierarchical breakdown of the work into work-packages and tasks
- A description of the task activities
- A description of the task results
- A description of tasks required to assure product quality
- A description of task dependencies
- A list of anticipated resources (persons, skills, tools, etc.)
- A time schedule

- A list of milestones and deliverables
- A risk containment plan

Effectively *all* of the information, reports, and models developed in the Plan Evolution phase serve as roadmaps for developing the project plan. They contain a complete description of the individual Renaissance evolution strategies that will be applied to various components of the system, often concurrently. (This will be reflected in the dependencies among the task activities.)

### **Inputs**

TSI . CMTS - Context Model of Target System

CSI . CMCS - Context Model Current System

ESI . SES - the overall System Evolution Strategy

BUSI . BG - Business Goals

BUSI . BPD - Business Process Definition

ESTI - *All estimating information*

### **Outputs**

TFMI . PP - Project Plan

### **Responsibilities involved**

SWPM - Software Project Manager

## 2.4.2 Define Test Strategy

<b>IMPLEMENT : DEFINE TEST STRATEGY</b>	
The testing strategy for the evolution project is defined, describing the specific tests that are planned, the test objectives, and the personnel involved.	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• Quality Engineer</li> <li>• Test Engineer</li> <li>• User</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>Technical Model of Target System Business Process Description</p>	<p><b><i>Outputs</i></b></p> <p>Test Strategy Plan</p>

*Previous task:* This task is carried out after Plan Evolution Project (page 86), and concurrently with the Design Target and Design Transformation activities. See page 84 for an overview of the task relationships.

*Next task:* The outputs of this task become the input into the Test and Transform activities. See page 84 for an overview of these relationships.

*References:* -

### **Task description**

The objective of this activity is to define and document the overall test strategy. The test strategy plan describes which tests are planned, the test objectives, and people involved. In addition it will be defined if and how current test data and test environments can be used for the tests of the target system. If the test strategy has a larger effect on required resources then it must interact with the project planning activity. The user can be involved to ensure that the aspects which are important for the user are really tested.

### **Inputs**

TSI . TMTS - the technical model of the target system.

BUSI . BPD - Business Process Description

### **Outputs**

TSTI . TSP - Test Strategy Plan

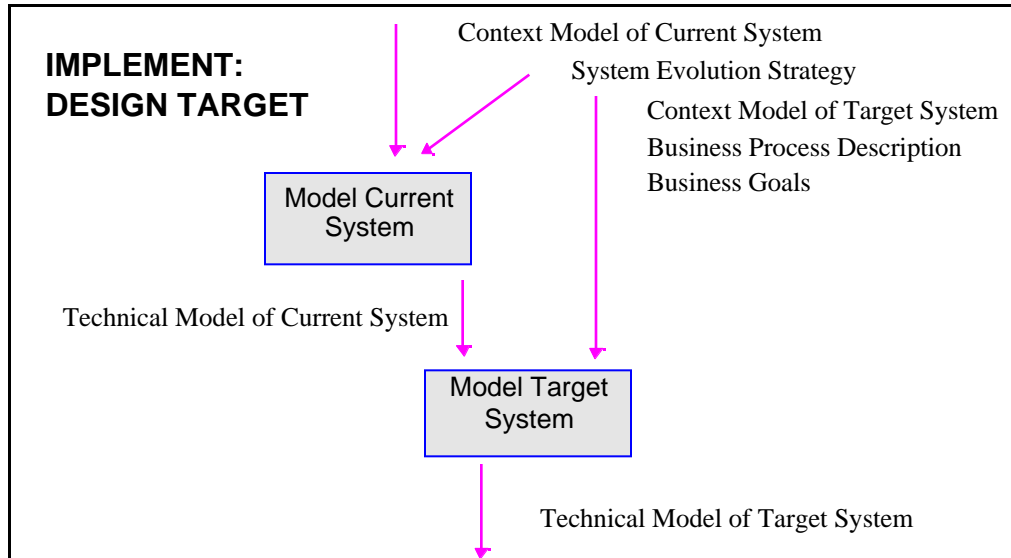
**Responsibilities involved**

QE - the quality engineer is in charge of this activity

TE - the test engineer

USER - User could be involved in planning tests

### 2.4.3 Design Target



#### Activity Description

The purpose of the 'Design Target' activity is to develop a detailed technical model of the target system. This is accomplished by first understanding the current system in detail, before proceeding with designing the target system based on the system evolution strategy and the business constraints and processes. This activity is carried out concurrently with the activity Design Transformation (page 96).

The *Model Current System* task is based on the existing *Context Model of Current System* artifact and aims to rediscover the current system at the level of detail needed to proceed with the re-engineering project. In particular, the functional and structural viewpoints are elaborated compared to the context modelling phase. It is in this task that reusable components are identified and described, based on the evolution strategy.

The *Model Target System* task is based on the *Context Model of Target System* and the *Technical Model of Current System* artifacts, as well as several other sources of information from the 'Plan Evolution' activity. This task produces the technical level design model of the target system which is used further in the implementation phase of the project.

UML is the notation used to describe visually both the technical model of the current and target system.

*This page is intentionally blank*

### 2.4.3.1 Model Current System

<b>IMPLEMENT : DESIGN TARGET : MODEL CURRENT SYSTEM</b>	
<p>The purpose of this task is to model the current system at a level of detail sufficient that a technical model of the target can be constructed. The focus is on reverse engineering the current system to be able to discover subsystems that may be reused in the target system, and to define the interfaces to these subsystems.</p>	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• Legacy Functional Expert</li> <li>• Legacy Implementation Expert</li> <li>• Software Engineer</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>System Evolution Strategy Context Model of Current System</p>	<p><b><i>Outputs</i></b></p> <p>Technical Model of Current System</p>

*Previous task:* Plan Evolution Project, page 86

*Next task:* Model Target System, page 94. It is concurrent with the task composing the Design Transformation (page 96) activity.

*References:* Renaissance Report “Architectural Modelling for Evolution” - Chapter 3/Appendix A

#### **Task description**

This task is a continuation of the top-down reverse engineering started in the context modelling of the current system earlier and is an important task for recovering the technical characteristics of the current system.

Modelling the current system is essentially reverse engineering. This is done to identify reusable components and obtain ideas on how to use them in the new system. The degree of reverse engineering needed and the weight put on identifying reusable components depends on the evolution strategy chosen for the project. The evolution strategy report documents which components are most likely to be reused in the re-engineering project.

Note that the level of detail of the Technical Model of the Current System is based on a number of factors. The two major ones are related to how good the existing model of the current system is, and time constraints. Remember that 80% of the time is spent on the last 20% of details; so a

consideration must be made regarding the degree of detail of the Technical Model of the Current System.

The consultancy report 'Architectural Modelling for Evolution' documents properties that should be looked for in various 3GL and 4GL applications, and how these should be modelled with the Unified Modelling Language (UML).

### **Inputs**

CSI . CMCS – The *Context Model of Current System* is the starting point for the technical modelling (reverse engineering) of the current system.

ESI . SES - The System Evolution Strategy describes how the system should evolve, and is a guideline for deciding how detailed the description of different parts of the system should be.

### **Outputs**

CSI . TMCS – The Technical Model of the Current System is a detailed description of the current system.

### **Responsibilities involved**

LFE – The Legacy Functional Expert provides functional expertise of the system.

LIE – The Legacy Implementation Expert provides expertise about languages used to implement the current system. This is important expertise, given that reverse-engineering is a key activity in the modelling of the current system.

SWE – The Software Engineer provides technical modelling expertise to the re-engineering project.

### 2.4.3.2 Model Target System

<b>IMPLEMENT : DESIGN TARGET : MODEL TARGET SYSTEM</b>	
This task involves designing the target system, based upon the evolution strategy chosen and the technical model of the current system.	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• Application Business Expert</li> <li>• Software Architecture Expert</li> <li>• Software Engineer</li> <li>• User</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>System Evolution Strategy          Technical Model of Current System          Context Model of Target System          Business Goals          Business Process Description</p>	<p><b><i>Outputs</i></b></p> <p>Technical Model of Target System</p>

*Previous task:* Model Current System, page 92. This task runs concurrently with the task Design Transformation (page 96).

*Next task:* This is the final task in the Design Target activity. The next tasks are those of the Test (page 100) and Transform (page 98) activities, which run concurrently.

*References:* Renaissance Report “Architectural Modelling for Evolution” - Chapter 3; Renaissance Report “Evolution Planning”, Chapter 3; Renaissance Report “Client/Server Migration”

#### **Task description**

An overall system evolution strategy has been selected for the legacy system, as described in the Renaissance Report “Evolution Planning”:

- Continued maintenance
- Re-engineer (*re-vamp, re-structure, re-architecture, re-design with reuse*)
- Replace

The *Model Target System* task focuses on describing the different components of the current system with respect to the strategies that are selected. E.g. if the evolution strategy 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; the technical application logic is reused from the technical model of the current system.

However, if some subsystem is selected to undergo a re-architecturing exercise, a more detailed model of the system internal parts must be made, based on the context model for the target and the technical model for the current system.

Chapter 3 in the consultancy report “Architectural Modelling for Evolution” describes 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 “Model Context of Target System” task, mainly elaborating the structural view. Additional models using other techniques may be developed where appropriate. The modelling techniques selected for the target design are based on the Unified Modelling Language (UML).

### **Inputs**

ESI . SES - The elaborated System Evolution Strategy for the evolution project.

CSI . TMCS - Technical Model of Current System

TSI . CMTS - The Context Model of Target System

BUSI . BPD - The Business Process Description describes the business processes to be supported by the target system.

BUSI . BG - The Business Goals express some general, high-level requirements for the target system.

### **Outputs**

TSI . TMTS - The activity is finished when the legacy system to be evolved is sufficiently described in the *technical model of the target system*.

### **Responsibilities involved**

ABE - The Application Business Expert has the quality assurance role ensuring that the target system meets the functional properties expected.

SAE - The Software Architecture Expert is an expert in modern software architecture who is responsible for ensuring that the Technical Model of the Target System adheres to given standards.

USER - The User is involved in the design group to make sure that the target system supports the user needs.

SWE - The Software Engineer provides competence in modelling systems with the Unified Modelling Language, and works closely with the Application Business Expert and Test Engineer to produce a detailed technical model of the target.

## 2.4.4 Design Transformation

<b>IMPLEMENT : DESIGN TRANSFORMATION</b>	
<p>The transformation from the current to target system is designed through the identification of <i>component batches</i>. These are sets of components that should be transformed together, so that the transformation can be carried out in an incremental way either before or after the target system has been completely designed.</p>	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• Evolution Modeller</li> <li>• Software Engineer</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>System Evolution Strategy            Technical Model of Current System            Technical Model of Target System            Project Plan</p>	<p><b><i>Outputs</i></b></p> <p>Component Batches            Project Plan (revised)</p>

*Previous task:* The previous task is Plan Evolution Project, page 86. This task runs concurrently with the Design Target activity (page 90).

*Next task:* The next tasks are those of the Test (page 100) and Transform (page 98) activities, which run concurrently.

*References:* Renaissance Report “Architectural Modelling for Evolution” - Chapter 3; Renaissance Report “Evolution Planning”; Renaissance Report “Client/Server Migration”

### Task description

The approach suggested by Renaissance for the transformation from legacy system to target system is based on the notion of *interface*, and enforces an incremental migration toward the desired system.

The design transformation task separates the technical target model into *component batches* that can be transformed together. Based on the duration of the Design Target activity, the design transformation task is carried out either after that activity, or concurrently with it, to identify component batches so that transformation can start at the same time as the design activities. The result of this task is used to identify sub-components of the existing system, whose combined transformation allows the current system to migrate toward the target system.

The component batches will be the unit of development for the evolution project. If the legacy system is a large one, the tasks of modelling the current and target systems may stretch out in time. A component batch includes components that have been understood and documented both in the current and target system modelling tasks, and the transformation of the component batch can be carried out concurrently with the continued modelling in the these tasks. While the focus of the earlier modelling activities in the Renaissance method has been on modelling and analysing the old legacy system, the activities undertaken in this task are primarily concerned with the migration to the new system.

The technology chosen for the new system is determined by the system evolution strategy. The focus is however on using *new* architectures, moving towards a client/server solution.

As it becomes clearer what will be done in each transformation increment, the evolution project plan should be updated with new tasks and resource estimates.

### **Inputs**

ESI . SES - The System Evolution Strategy.

CSI . TMCS - The technical model of the current system

TFMI . PP - The evolution project plan

TSI . TMTS - The technical model of the target system must be available so that the designer can uniquely locate the component batches for the individual increments. The new system definition provides the designer with information for understanding the current functionality of the system, and for reusing parts of the existing system.

### **Outputs**

TFMI . CB - The Component Batches dictate which components in the legacy systems should be re-engineered together, and therefore dictate the constraints of the new design. A component batch lists the files that should be removed from the current system, which should be included in the target system, and any other information necessary to perform an efficient transformation.

TFMI . PP - The project plan will be revised as it becomes more clear which activities that will be carried out in each increment. The estimates may be recalculated as more information has become available.

### **Responsibilities involved**

EM - The evolution modeller will aid the software engineer with decisions about which legacy system components that should be grouped into one component batch.

SWE - The software engineer involved in the MTS task is also involved in this task to walk the evolution modeller through the TMTS.

## 2.4.5 Transform

<b>IMPLEMENT : TRANSFORM</b>	
<p>To implement the planned transformation, the necessary steps are carried out, including many of the traditional implementation activities associated with a software development project.</p>	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• Software Project Manager</li> <li>• Software Engineer</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>Technical Model of Target System Project Plan</p>	<p><b><i>Outputs</i></b></p> <p>Target System</p>

*Previous task:* The previous tasks are the concurrently running Design Target (page 90) and Design Transformation (page 96) activities; and the Define Test Strategy (page 88). This task runs concurrently / iteratively with the Test (page 100) activity.

*Next task:* Together with the concurrently running Test activity, and the independently executed Prepare Target (page 108) task, this is the final task of the Implement phase.

*References:* Renaissance Report “Evolution Planning”; Renaissance Report “Client/Server Migration”

### Task description

The essential work of this task is straightforward to describe. It is the traditional work of implementation:

- To implement the planned transformation.
- To establish the development environment.
- To produce code.
- To ensure correct system interfacing.

### Inputs

TSI . TMTS - technical model of target system.

TFMI . PP - the project plan.

**Outputs**

TSI . TS - target system

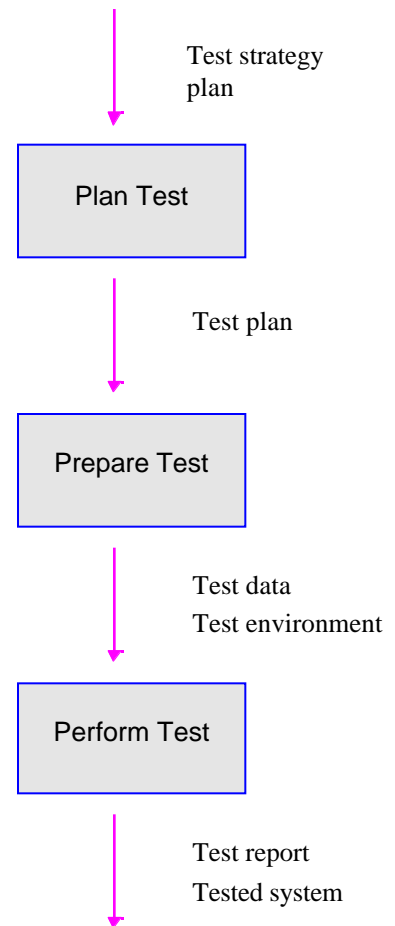
**Responsibilities involved**

SWE - Software Engineer

SWPM - Software Project Manager

## 2.4.6 Test

### IMPLEMENT : TEST



#### Activity Description

During system transformation, several test activities are performed on different levels:

- In a **unit or module test** single transformation units are tested.
- In an **integration test** the interfaces of the single implementation units and their interaction are tested.
- In the **system test** the final target system is tested.

The different tests mentioned above have all the same structure: first the concrete test must be planned, then the test is prepared and finally the test is performed. These steps are described in more detail in the following subsections. As the different test levels are the same for evolution projects as for new developments we find some differences in the single activities, especially the test planning and test preparation.

There is a high degree of interaction between test and transformation activities on all implementation levels. The concrete order of activities must be individually defined in the project plan and in the test strategy.

In addition, integration strategies play a role in the development of test plans.

### 2.4.6.1 Plan Test

<b>IMPLEMENT : TEST : PLAN TEST</b>	
Plan the concrete tests to be carried out during transformation from current to target system.	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• Quality Engineer</li> <li>• Software Engineer</li> <li>• User</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>Test Strategy Plan            Technical Model of Target System            Business Process Description</p>	<p><b><i>Outputs</i></b></p> <p>Test Plan</p>

*Previous task:* The previous tasks are the concurrently running Design Target (page 90) and Design Transformation (page 96) activities; and the Define Test Strategy (page 88). This task runs concurrently / iteratively with the Transform (page 98) task — as part of the overall Test activity.

*Next task:* Prepare Test, page 104

*References:* Renaissance Report “Evolution Planning”-

#### **Task description**

The principal result of this task is a *test plan*—which can be a separate document or a refinement of the test strategy plan. The plan contains the following information:

- A definition of the functionality to be tested;
- A definition of used test methods, tools and procedures;
- An identification of areas associated with particularly high risk, and therefore to be tested with special care;
- A definition of existing test data and environments which can be reused;
- A list of required resources for this test including persons, skills, hardware, software, organisational needs, etc.

If test documentation for the current system exists, it can become a starting point for creating the new test plan. The existing test documentation and existing test data, environments, etc. must be critically reviewed to check if they are appropriate for the new target system.

**Inputs**

TSTI . TSP - The test strategy plan

TSI . TMTS- the technical model of the target system.

BUSI . BPD - the business process description

**Outputs**

TSTI . TP - the test plan

**Responsibilities involved**

QE - the quality personnel must oversee the testing activities

TE - the test engineer

SWE - software engineering personnel

USER - the user ensures that the most critical functions are well tested, that test data is correct, etc.

### 2.4.6.2 Prepare Test

<b>IMPLEMENT : TEST : PREPARE TEST</b>	
Define test procedures, test data, and expected results, documenting them at appropriate levels.	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• Quality Engineer</li> <li>• Test Engineer</li> <li>• User</li> <li>• Software Engineer</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>Test Plan</p>	<p><b><i>Outputs</i></b></p> <p>Test Data Test Environment</p>

*Previous task:* Plan Test, page 102

*Next task:* Perform Test, page 106

*References:* Renaissance Report “Evolution Planning”

#### **Task description**

In this activity the test procedures are defined; and test data and expected results are prepared and documented. If existing test data can be reused they must be checked and if necessary brought up to date. If the current and the new system should produce the same results, the current system can be used to produce comparison data. If the tests are intended to be supported by tools and utilities they must be created or existing ones must be adapted.

Many kinds of testing techniques may be brought into consideration—for example:

- regression tests
- use of automatic comparisons
- back-to-back testing

#### **Inputs**

TSTI . TP - The test plan

#### **Outputs**

TSTI . TD - The test data

TSTI . TENV - The test environment

**Responsibilities involved**

QE - the quality personnel must oversee the testing activities

TE - the test engineer

USER - as in the previous task, the user can help ensure that the critical functionality is being tested, that the test data is correct, etc.

SWE - software engineering personnel

### 2.4.6.3 Perform Test

<b>IMPLEMENT : TEST : PERFORM TEST</b>	
Carry out the actual testing activities.	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• Quality Engineer</li> <li>• Test Engineer</li> <li>• Software Engineer</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>Test Plan Test Data Test Environment</p>	<p><b><i>Outputs</i></b></p> <p>Tested System Test Report</p>

*Previous task:* Prepare Test, page 104

*Next task:* Together with the Transform (page 98) task, running concurrently; and the independent Prepare Target (page 108) activity, this is the last task in the Implement phase. The next tasks form part of the Deliver phase.

*References:* Renaissance Report “Evolution Planning”-

#### Task description

In this activity the transformed system or components of it are tested following the test plan and using the prepared test data and environments. The test outputs are compared with the expected results. The test results are documented in a test report. If a test is not successful, then the transformation steps are repeated and the detected errors are fixed. If we are at the system test level, and the test was successful, then we have a tested target system for delivery.

#### Inputs

TSTI . TP - The test plan

TSTI . TD - The test data

TSTI . TENV - The test environment

#### Outputs

TSTI . TSYS - The tested target system

TSTI . TR - The test report

**Responsibilities involved**

QE - the quality personnel must oversee the testing activities

TE - the test engineer

SWE - software engineering personnel

## 2.4.7 Prepare Target

<b>IMPLEMENT : PREPARE TARGET</b>	
Prepare the target environment for running the new system	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• Software Engineer</li> <li>• Operational Technical Service</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>Technical Model of Target System</p>	<p><b><i>Outputs</i></b></p> <p>Prepared Environment</p>

*Previous task:* There is no “previous” task, in the sense that this task is carried out independently of other, concurrently running tasks. See page 84 for an overview of this task’s relationship to the others in Implement.

*Next task:* Together with the Transform (page 98) and Test (page 100) activities, this is the last task in the Implement phase. The next tasks form part of the Deliver phase.

*References:* Renaissance Report “Evolution Planning”

### 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, COTS or new operating systems or database servers, and so on, have to be installed. At this point, the operational technical service may enter in order to initiate preliminary activities on training and operations.

### Inputs

TSI . TMTS - technical model of the target system.

### Outputs

TSI . PENV - Target environment, prepared for running the transformed system.

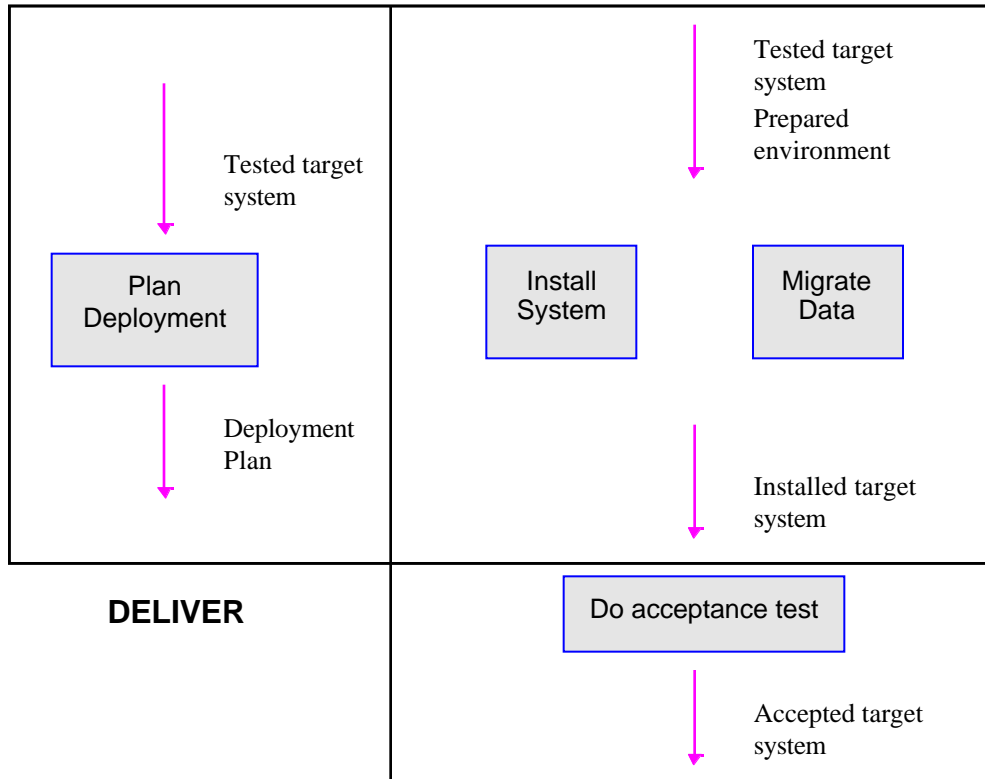
### Responsibilities involved

SWE- software engineering personnel are in charge of this activity.

OTS - operational technical services begin preliminary activities.

*This page is intentionally blank*

## 2.5 DELIVER



### Description of the Deliver Phase

The system is delivered, including installation, data migration from current to target system, and acceptance testing. Acceptance criteria must be identified in order to validate the target 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.

In this phase, data migration from current to target system is likely to be the activity of most concern. It can be a major task all on its own, and must not be underestimated.

*This page is intentionally blank*

## 2.5.1 Plan Deployment

<b>DELIVER : PLAN DEPLOYMENT</b>	
This is a forward-looking activity, in preparation for field-level deployment after delivery and acceptance testing are completed.	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• Software Engineer</li> <li>• Software Project Manager</li> <li>• Operational Technical Service</li> <li>• User</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>Tested Target System</p>	<p><b><i>Outputs</i></b></p> <p>Deployment Plan</p>

*Previous task:* There is no “previous” task in the Deliver phase. This task is carried out currently with the other tasks of this phase.

*Next task:* There is no task directly following this one in the Deliver phase. It is an independent task.

*References:* Renaissance Report “Evolution Planning”-

### **Task description**

Even as the system is being installed and the data migrated, planning for the future deployment, after validation, begins in parallel.

Activities to be carried out are:

- Identify the different sites on which to deploy the system, defining the factors that characterise these sites (specific characteristics, number of users, storage and processing capacity, etc.)
- Analyse the types of risks involved in operating the system in these environments,
- Identify data to be migrated
- Identify necessary steps for the installation at user sites,
- Identify needed manpower

The results of the activities will be described in a Deployment Plan, which is under the responsibility of the operational technical service. Documents will be created on the topic of “how to deploy” for each kind of hardware/software pair, data servers, processing servers, network servers, user stations, etc.

**Inputs**

TSTI . TSYS - the tested target system.

**Outputs**

DTSI . DP - the Deployment Plan.

**Responsibilities involved**

SWE - software engineers assist the operational technical service

OTS - the operational technical service has direct responsibility for deployment

SWPM - software project manager, validating the planning

USER - Users are directly affected by deployment

## 2.5.2 Install System

<b>DELIVER : INSTALL SYSTEM</b>	
Several activities around system installation (e.g. preparation of system operations and training manuals) are carried out.	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• Software Engineer</li> <li>• User</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>Tested Target System Prepared Environment</p>	<p><b><i>Outputs</i></b></p> <p>Installed Target System</p>

*Previous task:* There is no “previous” task in the Deliver phase. This task is carried out concurrently with the Migrate Data task (page 116).

*Next task:* Recalling that this task is carried out concurrently with the Migrate Data task, the next task is Do Acceptance Test, page 118.

*References:* Renaissance Report “Evolution Planning”-

### Task description

After the characteristics of the target system are determined, installing the target system will involve the following activities:

- Adapt the new system to the target environment depending on the characteristic factors (volume of data handled, users specific factors, etc.)
- Produce a System Installation Guide and an Operating Manual
- Generate the new system following the procedure described in the Installation Guide,
- Turn the system on and install the technical operation resources

### Inputs

TSTI . TSYS - the tested target system

TSI . PENV - the prepared environment

### Outputs

DTSI . ITS - the installed target system.

**Responsibilities involved**

SWE - software engineering personnel carry out the installation.

### 2.5.3 Migrate Data

<b>DELIVER : MIGRATE DATA</b>	
The data managed by the current system is now migrated to the evolved system.	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• Software Engineer</li> <li>• User</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>Tested Target System Prepared Environment</p>	<p><b><i>Outputs</i></b></p> <p>Migrated Data</p>

*Previous task:* There is no “previous” task in the Deliver phase. This task is carried out concurrently with the Install System task (page 114).

*Next task:* Recalling that this task is carried out concurrently with the Install System task, the next task is Do Acceptance Test, page 118.

*References:* Renaissance Report “Client/Server Migration”

#### **Task description**

In parallel with system installation, the data must be migrated from the old system. There a number of techniques for data migration available, most often provided by the data management system provider. The technique you use will depend on the exact nature of the data and system in use in the current system. In case of particular situations (i.e. different data structures for source and target systems) the data migration activity can impose the development of specific software for data migration.

#### **Inputs**

TSTI . TSYS - the tested target system.

TSI . PENV - the prepared environment.

#### **Outputs**

DTSI . MD - the migrated data.

#### **Responsibilities involved**

SWE- software engineering personnel carry out the data migration.

USER - To oversee the data migration activity

*This page is intentionally blank*

## 2.5.4 Do Acceptance Test

<b>DELIVER : DO ACCEPTANCE TEST</b>	
Acceptance testing is carried out on site at the user installation—the main characteristic which differentiates it from normal system testing during the Implement phase.	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• User</li> <li>• Quality Engineer</li> <li>• Test Engineer</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>Installed Target System All testing information</p>	<p><b><i>Outputs</i></b></p> <p>Accepted Target System</p>

*Previous task:* The two previous tasks, carried out concurrently with each other, are the Migrate Data task (page 116) and the Install System task (page 114).

*Next task:* This is the final task in the Deliver phase. Successive tasks are part of the Deploy phase.

*References:* Renaissance Report “Evolution Planning”

### **Task description**

The acceptance test activity has the same substructure as the other test levels with test planning, test preparation and test execution. The main difference is that the acceptance test is done by the user of the system at the customer site. In some situations the customer can be supported by the provider but he has the final responsibility for accepting the new system.

Acceptance tests should be developed in parallel with other re-engineering activities. During test planning and test preparation, existing documents can be reused. In evolution projects special attention should be given to avoid an error inheritance from the old system to the target system. In addition the introduction of new bugs must be avoided.

### **Inputs**

DTSI . ITS - the installed target system.

TSTI - *All* test information that has been developed (including the Test Strategy Plan, Test Data, etc.)

**Outputs**

DTSI . ATS - the accepted target system.

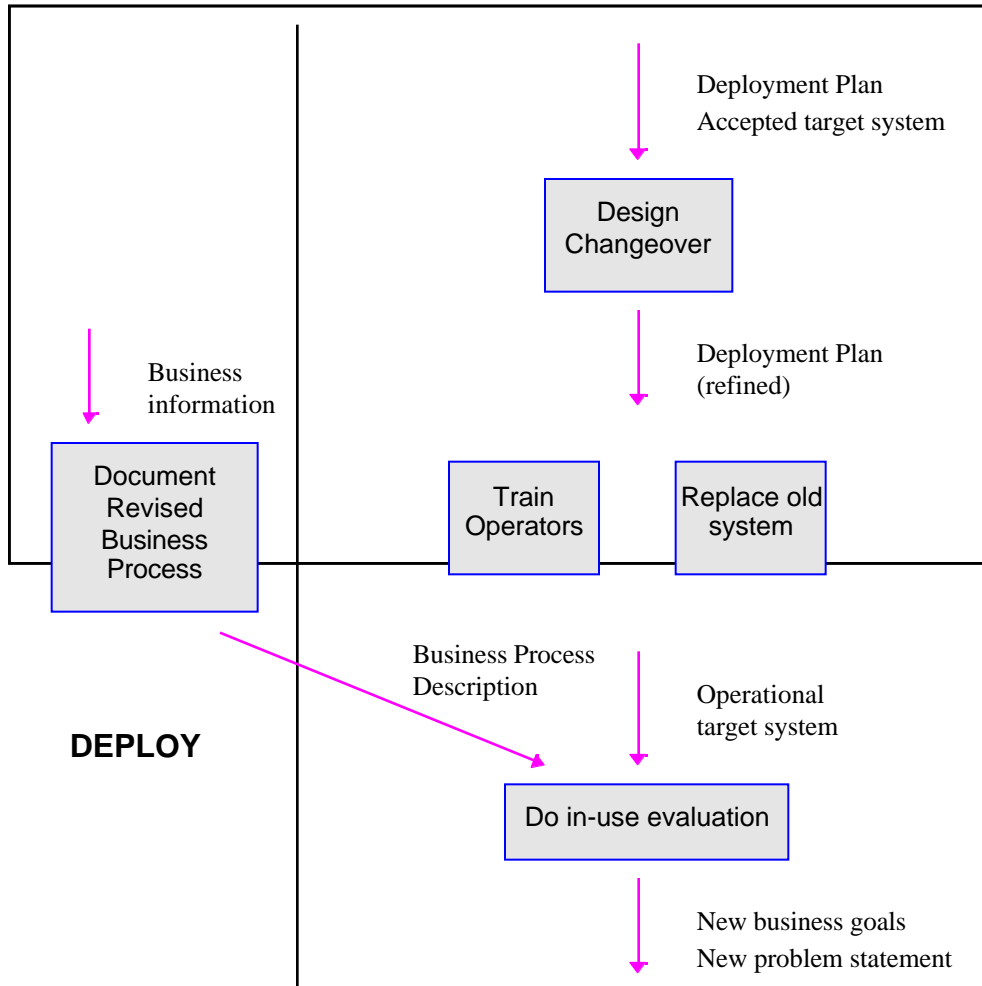
**Responsibilities involved**

USER - To contribute to the validation activity

QE - quality assurance engineers can support the user during the acceptance testing

TE - test engineers can support the user during the acceptance testing

## 2.6 DEPLOY



### Description of the Deploy Phase

The new system, accepted by final users, must be deployed to the target operational environment. The deployment may involve installing new COTS and hardware, the re-training of users in using the new system, and so on.

At the same time, the continual evolution of the system is assured by in-use evaluation activities which lead to revision of the business goals and business processes.

It should be pointed out that if the system is simply a re-engineered version of the legacy system, then activities such as business process revision and user training may well not be needed. Thus, the activities that are actually carried out by you in the Deploy phase will vary considerably according to the nature of your particular re-engineering project.

*This page is intentionally blank*

## 2.6.1 Design Changeover

<b>DEPLOY : DESIGN CHANGEOVER</b>	
A number of decisions are made concerning the way in which the organisation will change over from use of the legacy system to the new target system.	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• User</li> <li>• Software Engineer</li> <li>• Operational Technical Service</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>Accepted Target System Deployment Plan</p>	<p><b><i>Outputs</i></b></p> <p>Deployment Plan (refined)</p>

*Previous task:* There is no “previous” task in this phase. See page 120 for an overview of task relationships.

*Next task:* Two tasks running concurrently—Train Operators (page 124) and Replace Old System (page 126).

*References:* Renaissance Report “Evolution Planning”-

### Task description

Changeover is a critical path within the deployment process. The choice between global and progressive changeover must meet the real evolution project objectives. Where both possibilities are acceptable, it is recommended to choose progressive changeover. This makes it possible to

- Exert more accurate control over the changeover process;
- Reduce risks, as well as the impact of non-identified risks;
- Respect the changeover schedule more easily;
- Transfer operating personnel from the source system to the target system progressively.

For each step identified for the deployment, specific characteristics of the target environment have to be clearly identified.

### Inputs

DTSI . ATS - the accepted target system.

DTSI . DP - the deployment plan.

**Outputs**

DTSI.DP - the deployment plan, refined according to the changeover design

**Responsibilities involved**

OTS - the operational technical service has primary responsibility for deployment

SWE - software engineering personnel

USER - Users are involved in all deployment activity

## 2.6.2 Train Operators

<b>DEPLOY : TRAIN OPERATORS</b>	
<p>A set of training activities appropriate to the precise nature of the re-engineering project is prepared and carried out. The training activities could vary widely from project to project.</p>	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• User</li> <li>• Software Engineer</li> <li>• Operational Technical Service</li> </ul>	
<p><b><i>Inputs</i></b> Accepted Target System</p>	<p><b><i>Outputs</i></b> Training Material</p>

*Previous task:* Design Changeover, page 122. This task runs concurrently with Replace Old System, page 126

*Next task:* Do In-Use Evaluation, page 130

*References:* Renaissance Report "Evolution Planning"-

### Task description

A training plan is to be provided, describing:

- The list of functions that must be presented and explained;
- The list of hardware and software components that will be used for training;
- A survey of the participants involved, their qualifications and the operations that they will have to carry out;
- A training schedule.

Training material has to be produced (training scenarios, course notes, handouts, slides, training aids, etc.

Training sessions are delivered, carrying out the defined teaching scenarios, and assuring that the trainees have really acquired the needed expertise to run the target system.

### Inputs

DTSI . ATS - the accepted target system.

**Outputs**

DTSI . TM – Training Material.

**Note:** there will be various types of training material produced, many of them described in the above discussion; but their nature varies to such an extent that these artifacts are not explicitly listed here.

**Responsibilities involved**

OTS - the operational technical service has primary responsibility for deployment

SWE - software engineering personnel is needed for this activity, in order to give the operational personnel the necessary technical background

USER - Users are involved in all deployment activity

### 2.6.3 Replace Old System

<b>DEPLOY : REPLACE OLD SYSTEM</b>	
Several activities are carried out involving the replacement (either global or progressive) of the legacy system with the target system, according to the strategy outlined in the changeover design.	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• User</li> <li>• Operational Technical Service</li> <li>• Software Engineer</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>Accepted Target System Deployment plan All testing information</p>	<p><b><i>Outputs</i></b></p> <p>Operational Target System</p>

*Previous task:* Design Changeover, page 122. This task runs concurrently with Train Operators, page 124.

*Next task:* Do In-Use Evaluation, page 130

*References:* Renaissance Report "Evolution Planning"-

#### **Task description**

The old system is replaced, according to the refined deployment plan that was elaborated in the Design Changeover task.

The target system is tuned to the specific characteristics of the operational environment, and operational data migrated. The process is identical to the one used in the Install System and Migrate Data tasks in the Deliver phase

For each step defined in the plan, the acceptance tests have to be re-done, verifying that the target system runs properly in the specific target environment.

#### **Inputs**

DTSI . ATS - the accepted target system.

DTSI . DP - the deployment plan, refined with changeover activities.

TSTI - All testing information, for using in re-doing acceptance tests as necessary during replacement of the old system.

**Outputs**

DTSI .OTS - the operational target system.

**Responsibilities involved**

OTS - the operational technical service has primary responsibility for deployment

SWE - software engineering personnel are likely to be needed for technical assistance as the old system is replaced

USER - Users are involved in all deployment activity

## 2.6.4 Document Revised Business Process

<b>DEPLOY : DOCUMENT REVISED BUSINESS PROCESS</b>	
<p>The target system will almost certainly support at least an incrementally modified business process, as a re-engineered system. Before putting the system into field operation, it is important to document this new process, which will also serve as input to future evolution of the system.</p>	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• User</li> <li>• Operational Technical Service</li> <li>• Application Business Expert</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>Operational Target System</p>	<p><b><i>Outputs</i></b></p> <p>Business Process Description</p>

*Previous task:* There is no “previous” task in this phase. See page 120 for an overview of task relationships.

*Next task:* Do In-Use Evaluation, page 130

*References:* Renaissance Report “Evolution Planning”-

### **Task description**

The evolved system is likely to support a business process that is different (even if only incrementally) from the original business process supported before the current cycle of evolution. This new business process is documented in this task, *at the latest*—that is, it is equally likely that the new business process can be documented over the course of all re-engineering activities in the evolution project.

### **Inputs**

DTSI .OTS - the operational target system.

### **Outputs**

BUSI .BPD - a revised business process description.

### **Responsibilities involved**

ABE - application business experts, who have a major stake in how the business processes are defined and evolved.

**USER** - Clearly the users always have a stake in the future evolution of the system.

**OTS** - the operational technical service will, at the very least, be able to assist the application business experts and users in gathering information about the system and documenting the business process.

## 2.6.5 Do In-Use Evaluation

<b>DEPLOY : DO IN-USE EVALUATION</b>	
<p>This task is preparatory for future evolution of the system. Rather than using the transformed system with no thought to future evolution—dooming it to become another “legacy system” at some point in time—information is gathered that will assist decision makers in following the incremental approach to system evolution embraced by Renaissance.</p>	
<p><b><i>Responsibilities Involved</i></b></p> <ul style="list-style-type: none"> <li>• User</li> <li>• Application Business Expert</li> <li>• Operational Technical Service</li> <li>• Software Engineer</li> </ul>	
<p><b><i>Inputs</i></b></p> <p>Operational Target System All business information</p>	<p><b><i>Outputs</i></b></p> <p>(new) Business Process Description (new) Business Goals (new) Problem Statement</p>

*Previous task:* There are four different tasks preceding this task. See page 120 for an overview of the different task relationships.

*Next task:* This is the final task in the Renaissance method.

*References:* -

### **Task description**

Through extensive evaluation activities while the evolved target system is running, the way is prepared for future applications of the Renaissance method. As experience is gathered with the system, desired incremental changes in the business process (and therefore the system itself) are identified, which allow future iterations to have minimal negative impact on the user community of the system.

Several kinds of information about the system can be gathered during this activity:

- User satisfaction with the functionality of the system, ranging from facilities offered to ease of use;
- Ability of the system to fulfil its functional operating requirements;

- Performance of the system, including response times, storage capacity, number of users, etc.;
- Taking note of new requirements as they arrive both from the user community and the business community, and evaluating the capability of the system to handle these requirements;
- Considering the business goals of the organisation as they evolve in time, and continually evaluating the ability of the system to contribute to the pursuit of these goals;
- Evaluating the adequacy of the business process supported by the system as the needs and goals of the organisation evolve.

### **Inputs**

DTSI .OTS - operational target system.

BUSI - *All business information.*

### **Outputs**

BUSI .BG - incrementally revised business goals

BUSI .BPD - new (desired) business process descriptions corresponding to new functionalities for future implementation

BUSI .PS - new problem statement, corresponding to the experience gathered during in-use evaluation

### **Responsibilities involved**

SWE- software engineers will report on their experience with the operational system.

ABE - application business experts will gather experience and ideas for new functionality in the next iteration in the evolution of the system.

USER - Users will feed back their experiences with the system.

OTS - the operational technical service will not only assist the other individuals in this task in their responsibility of gathering and documenting experience—but will also be able to make its own contribution to the experience reporting process (since operational personnel will become intimately familiar with the performance of the new system).

## 3 Bibliography

This bibliography is intended to be indicative only of the large and mature re-engineering community. It represents a starting point for the user of the method to familiarise himself with the many aspects of re-engineering that are treated in the literature. Other references may be found in the RENAISSANCE consultancy reports.

- [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 2<sup>nd</sup> 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", 17<sup>th</sup> 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 André, 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", 17<sup>th</sup> 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.

*This page is intentionally blank*

## 4 Appendix: Traceability from Framework to Method

This appendix provides traceability from the Renaissance framework, as defined in the corresponding report, and the Renaissance method, as defined in this document. For a proper understanding of the rationale underlying the definition of the Renaissance method, the reading of the Renaissance framework report is strongly recommended, although not required. The knowledge of the framework is, however, mandatory in order to understand this appendix. In fact, concepts and terminology from the framework will be used throughout the remaining of this text without re-introducing them.

The Renaissance method is an instance of the Renaissance framework. This means that the activities of the framework can be mapped into activities of the method. These method activities specialise the general activities described in the framework. However, naming may 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 reminder 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
Plan Evolution	Trade-Off Analysis
	Issue Assessment
	Decision Analysis
Implement	Solution Implementation
Deliver	Solution Deployment
Deploy	

**Table 1: Top-Level Traceability**

Method Activity	Framework Activity
Assess Current Situation	As-Is Analysis
Identify Constraints and Business Goals	Environment Assessment
	Application Assessment
Assess Target	Can-Be Analysis
	Transition Analysis
	Strategies Assessment

Choose Strategy	Cost/Benefit/Risk Analysis
Plan Project	Planning
Design Target	Detailed Analysis
Design Transformation	
Implement Transformation	Metrics and Testbed Definition System Re-Implementation
Preparation of Target Environment	Environment Preparation
Do Acceptance Testing	Evaluation and Acceptance
Deliver	Evaluation and Acceptance
Deployment	System Deployment

**Table 2: Activities Traceability**

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

**Table 3: 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 *Plan Evolution* activity implements the *Trade-Off Analysis*, *Issue Assessment*, and *Decision Analysis* 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 are due to choices made in developing the method.

Concerning the *Plan Evolution* exception, it is true that a technology trade-off is largely independent from the quality of the system, but it is equally true 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 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. It was reasonable to include Decision Analysis in the Plan Evolution activity because all of the elements for taking the decision are already available by the end of that activity, and a clean Go / NoGo decision can be made at that point, *before* moving into the costly implementation phases.

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 performing acceptance testing on that system) and the second one concerning deploying the delivered (and accepted) 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 the user of a specific construction approach for re-implementing the system, based on the concept of a *subsystem interface*. The proposed approach, which could be changed by the user when instantiating the method for his 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 artifacts needed for applying the method, and which are both produced and used by activities.

Concerning *Kaizen*, the deployment activity includes specific tasks for the revision of business processes and goals, and preparation of a new problem statement for a subsequent iteration of the method. These tasks support the application of *Kaizen* in the following way:

- the Renaissance method supports continuous customisation (with reuse of the System Repository and the updating of business goals,

processes and problem statements as described above), which can be used to best tune the method for a given project within a given organisation; 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.