



We engineer software. Others just craft it

ProClaris
Avenue Louise, 200/132
B-1050 Brussels
Belgium
Telephone: +32 495 24 06 56
Voice Messaging/Fax: +32 2 611 60 46
E-mail: info@ProClaris.com
www.ProClaris.com

Migration vs. Reengineering

Decision Matrix

Migration vs. Reengineering: Decision Matrix



Revision History

Date	Version	Description	Authors
January 11, 2005	1.0	Initial version.	Yannis Manassis
January 12, 2005	1.1	Reviewed and extended with reengineering approach and additional criteria.	Enricos Manassis

Reference & File	Vers.	Date	Page
Migration vs Reengineering.doc	1.1	21/06/2010	2/9

This document is the property of softclarITy SPRL and contains confidential data. It cannot be distributed or duplicated without written authorisation of the softclarITy management.

Migration vs. Reengineering: Decision Matrix



Table of Contents

1	<i>Introduction</i>	4
2	<i>Migration process</i>	4
3	<i>Reengineering</i>	4
4	<i>The decision factors</i>	6
4.1	Critical decision factors	6
4.2	Code Audit vs Code Evaluation	7
4.3	Decision Matrix	8
5	<i>The decision process</i>	9

Tables

<i>Table 1 Decision Matrix: migration vs. reengineering</i>	8
---	----------

Reference & File	Vers.	Date	Page
Migration vs Reengineering.doc	1.1	21/06/2010	3/9

This document is the property of softclarITy SPRL and contains confidential data. It cannot be distributed or duplicated without written authorisation of the softclarITy management.

Migration vs. Reengineering: Decision Matrix



1 Introduction

The objective of this document is to identify the critical factors that will drive the decision whether to proceed to a reengineering or to a migration project of an application developed with Centura into a .NET architecture.

2 Migration process

The migration process typically consists in taking the original code as input and translating it into the destination technology and architecture.

Though the idea is very appealing, practically there are a number of shortcomings that prevent the process of migration from stopping here. These shortcomings translate in a percentage of successful migration, the rest consisting in a kind of reengineering effort (in fact based on manually translating code, see discussion on reengineering bellow).

A company like METEX, announces a 95-99 % of code migration success from Centura to the .NET platform. This translates in a 65-75 % rate of the migration **effort** realized. The main difficulty for the remaining effort consists in analyzing where to manually intervene so that the migration keeps the business logic intact.

In case a joint effort of upgrading the business logic is required, then these ratios of accomplished effort drop further more, as the complexity of upgrading multiplies the difficulty of appropriate manual intervention.

3 Reengineering

Using a reengineering approach for porting implies the use of a rigorous, repeatable, structured and organized process. We shall not consider here a pseudo-reengineering approach where a team of Centura and .NET developers examines the source Centura code and manually translates it into .NET code. While writing the code may show quick progress, this approach is likely to lead to lots of bugs, and therefore the bulk of the effort will concentrate on a testing and debugging phase.

Instead we shall consider a proper reengineering process which follows a proven approach that was developed some time ago in the context of an effort to build a showcase of Microsoft technologies. Suffice is to say that the approach is the basis of the book *“Practical Software Engineering: Analysis and Design for the .NET Platform”* (Enricos Manassis, Addison Wesley 2003), describing a rigorous, structured and controlled process of building software, from analysis

Reference & File	Vers.	Date	Page
Migration vs Reengineering.doc	1.1	21/06/2010	4/9

This document is the property of softclarITy SPRL and contains confidential data. It cannot be distributed or duplicated without written authorisation of the softclarITy management.

Migration vs. Reengineering: Decision Matrix



to code and testing: the *ProClaris Practical Process™*. The guiding principles of this process, as applied to a reengineering effort, are organized in two dimensions:

- 1) Functionally the approach consists in
 - a. Examining the existing application (User Interface, User Documentation, but also some parts of the code in order to find the exact expression of business rules), and extracting anew the product specification in a format suitable for Object Oriented analysis (use cases, business rules and user experience). This in fact amounts to a reverse engineering phase, but only of the functional aspect. In order for this approach to be effective, it is crucial that the current application has a well structured User Documentation that achieves a complete coverage of the functional scope of the product. If this is not the case, any reengineering approach will be contingent on the availability of resources, knowledgeable of the functions of the application, in order to give explanations during the phase of extraction of product specification.
 - b. Producing the object design that will subsequently be implemented on the architecture defined in the second dimension hereunder. This activity shall also take as input the existing code structure in order to define a convergent starting point for the design. Note however that the object design focuses solely on business domain objects (functional) and not any possible existing architecture constructs, as these will be replaced by the architecture defined by the technical stream.
 - c. In parallel, using the use cases as input, produce test cases with measured, complete coverage of the use cases.
 - d. At the end of the development, a last activity can consist of producing anew the user documentation out of the use cases and test cases, and perform a differential evaluation of the new documentation with the original one (used for extracting the product specification). As a measure of the success of the migration, the original and new use documentation should be very similar. Note that the approach of producing the user documentation out of the use cases and test cases is highly effective as it ensures a complete and measurable coverage of the user documentation to the functional scope of the application. Incidentally this approach is also highly cost efficient, as the user documentation is almost automatically generated from the use cases and test cases.

Reference & File	Vers.	Date	Page
Migration vs Reengineering.doc	1.1	21/06/2010	5/9

This document is the property of softclarITy SPRL and contains confidential data. It cannot be distributed or duplicated without written authorisation of the softclarITy management.

Migration vs. Reengineering: Decision Matrix



- 2) Technically, in parallel of the functional stream, define a sound software architecture, expressed in terms of documented architectural mechanisms, that matches the capabilities of the target platform (.NET), and defines a foundation architecture that will facilitate subsequent maintenance and evolution of the product, while permitting to measure the quality and robustness of the code.

Together the two dimensions are related by the means of the process that continues to support the evolution of the product. Note that, as the process itself is documented in the form of a book, it can easily be evaluated.

4 The decision factors

4.1 Critical decision factors

In order to decide which way to follow in order to execute the project of porting the existing Centura application into the .NET architecture, the following elements appear to be of critical importance:

- a) How good is the **structure** and how **clean** is the **code** of the existing application in Centura?

The less structured and the less clean is the present code base, the higher the cost of migration, and the more effective the reengineering effort would be, because unstructured code will seriously increase the need of manual intervention for the migration (and in fact this will equate to a covert pseudo-reengineering effort). Moreover, unstructured code is unlikely to easily translate into the application architecture generated by the migration tool. This in turn may impact product stability, performance and maintainability.

- b) How **satisfied** or how **close** is the existing solution to **present functional** and other **wishes** of the users/customer (business logic)?

The further from it, the higher will the migration effort be and the more effective the reengineering effort would prove. Two main categories are to be considered: functional corrections (satisfaction of the tool regarding its functionality) and functional improvements (closing the gap between present functional situation and the wished one).

- c) Another way of looking at the above criteria is the need for subsequent **evolution** and **maintenance**. The more important this need, the more advantageous will be an initial reengineering effort. Therefore this evolution and maintenance aspect is transcendent to all the previous criteria of: existing code structure and cleanliness (technical aspect), AND closeness of existing application to present and wished functional requests of users (functional aspect).

Reference & File	Vers.	Date	Page
Migration vs Reengineering.doc	1.1	21/06/2010	6/9

Migration vs. Reengineering: Decision Matrix



- d) How complete is the existing **user documentation** in relation to the functional scope of the application. As noted in the discussion of the reengineering approach, if the current documentation is not complete enough, implementing a reengineering approach to porting will be contingent on the availability of functionally knowledgeable resources, to contribute to the reengineering effort during the initial phase of extracting anew the product specification in an appropriate format for analysis (see above the point “a” of the functional aspect of reengineering).
- e) What is the **code size** of the application?
The larger the application the more it may become interesting to use a migration tool. Conversely the smaller the size the faster it can be reengineered, and the fixed high cost of the migration tool would need an equivalent amount of work to be realized in order to cover it. Yet this has to be weighted against the previous questions, particularly the cleanliness of the code and its structure.
- f) What is the strategy of **skill development** for the existing workforce? A reengineering effort is often the opportunity to involve the people that are currently in charge of maintenance and evolution of the Centura code base, henceforth seamlessly but effectively cross train them to .NET technologies, and permitting them to directly achieve advanced mastery of their new platform.

These simple questions imply the benefit of a previous audit of the source code and of the functional deviation from the wished business logic, which will help assess correctly the investment effort of migration.

4.2 Code Audit vs Code Evaluation

The code audit presents an extra cost. This extra cost shall only be compensated if it can provide a measurable indication as to which direction to follow and guarantee the success of this direction.

This is rather difficult to provide, and therefore an alternative method should be considered, a method based on the code evaluation. A valid method could consist in using the knowledge of specialists who maintain presently the application, providing them and guiding them with a code estimation framework (in order to avoid the definite bias of their code knowledge) and thus, deriving the answers as to the structure, cleanliness, functional satisfaction, functional deviation and size of the existing code.

Reference & File	Vers.	Date	Page
Migration vs Reengineering.doc	1.1	21/06/2010	7/9
This document is the property of softclarITy SPRL and contains confidential data. It cannot be distributed or duplicated without written authorisation of the softclarITy management.			

Migration vs. Reengineering: Decision Matrix



4.3 Decision Matrix

As a summary from previous section, the following decision matrix will facilitate a rational decision between the two choices of Reengineering vs. Migration. The corresponding criterion in favor of each solution is noted in the corresponding box. Each criterion indicates the situation that commends the choice of the corresponding approach e.g. if the **Structure of existing code** is *high*, then the choice of migration is most likely to bring the best value, or if the **Need for Evolution and Maintenance** is *important*, then the choice of reengineering is most likely to bring the best value.

		Reengineering	Migration
1	Functional satisfaction of existing application	Medium to lower	Very high
2	Functional deviation (Business Logic deviation) of existing application	Medium to important	Very small
3	Need for Evolution and Maintenance	Medium to high	Low
4	Existing documentation: coverage of functional scope	Complete	Partial
5	Structure of existing code	Medium to lower	High
6	Cleanliness of existing code	Medium to lower	Very clean
7	Source code size	Medium to small	High
8	Skill development	Important	Unimportant

Table 1 Decision Matrix: migration vs. reengineering

Reference & File	Vers.	Date	Page
Migration vs Reengineering.doc	1.1	21/06/2010	8/9
This document is the property of softclarITy SPRL and contains confidential data. It cannot be distributed or duplicated without written authorisation of the softclarITy management.			



5 The decision process

As a summary of the above discussion, we suggest the following action plan in order to determine which approach would be the most efficient for porting the existing Centura application into the .NET architecture:

1. Define the methodology to evaluate the code situation.
2. Proceed to the evaluation according to the critical factors mentioned.
3. In case the Reengineering method is selected, proceed to the corresponding Reengineering project.
4. In case the Migration method is selected:
 - a. Evaluate and select the migration tool.
 - b. Proceed to the corresponding Migration project.

Reference & File	Vers.	Date	Page
Migration vs Reengineering.doc	1.1	21/06/2010	9/9

This document is the property of softclarITy SPRL and contains confidential data. It cannot be distributed or duplicated without written authorisation of the softclarITy management.