Automated Selection of Legacy Systems SOA Modernization Strategies using Decision Theory

Sherif G. Aly and Rafik Amir

Department of Computer Science and Engineering
The American University in Cairo
sgamal@aucegypt.edu

Abstract

Legacy systems modernization is one of the most pressing issues for enterprise organizations. The continuous emergence of new technologies that are proven to be more robust, scalable, and maintainable, such as those using the Service Oriented Architectures (SOA), is currently attracting a lot of legacy systems modernization stakeholders. SOA legacy system modernization projects have induced a lot of researchers to address the full modernization project life cycle starting from architecture reconstruction, code analysis, modernization strategy selection, until eventually the actual strategy to be used in modernization is selected. Such strategy selection could depend on the component/module type of the legacy system (user interface, data layer, business functionality layers) and would also depend upon the platform, programming language used in the legacy system (Mainframe, Windows VC++, etc), as well as the type of strategy used to modernize the system. Since there are a lot of characteristics that control the selection of a modernization strategy, the modernization selection process gets tougher and tougher with the increase of such characteristics. Accordingly, we are presenting a modernization selection process along with a decision making tool that handles much of those characteristics altogether using decision theory to come up with the most optimal strategy to be used in modernizing the legacy systems in question.

Keywords: Legacy, modernization, service orientation, analysis, strategy

1. Introduction

A legacy information system represents a massive long-term business investment. Unfortunately, such systems are mostly brittle in the sense of being resistant to change, are slow, and non extensible when faced with new technologies [1]. Accordingly, such systems need to be evolved to either match the continuously evolving business, or to cope with continuous evolution of the other systems around them.

Legacy systems evolution [2] is actually a broad term that can range from adding a new field in the system screens to complete replacement of the system. This evolution can be categorized into three categories: maintenance, modernization and replacement. These three categories are rather stages such that the system initially does not need any support at all until some business needs started to deviate from its initial intended
usage. Accordingly the organization starts maintenance activities. Repeated maintenance activities are continuously done till the system outdates, which is the case when the maintenance effort does not support the business needs any more and modernization becomes inevitable. The modernization project might involve at the end total replacement of the system to cope the new technologies.

Our main focus in the research is the modernization of the legacy system using the Service Oriented Architecture (SOA). Service Oriented Architecture is a distributed systems architecture where all the components in the system are treated as independent services with minimal coupling between each other. These services are inter-related with a well defined interface and contracts between them [3-5].

The SOA modernization techniques as explained can be categorized into invasive techniques and non invasive techniques [7], which have evolved over the past few years due to significant research efforts.

Legacy non invasive techniques aim to extend the reach and lifetime of legacy systems by exposing intact functionality to additional channels. The non invasive techniques use wrapping of legacy assets to hide unwanted complexity. Legacy invasive techniques on the other hand are long processes that aim to completely replace parts of the system to ease its maintenance and extension.

These modernization techniques should be strongly backed up by a modernization process that completely drives the system towards its new business functionality. Although multiple research work has plotted the modernization process that leads to choosing and applying their modernization strategy, in general the modernization process length depends on several factors, namely, the target business needs, the available documentation of the system and the selected modernization techniques as per [6].

2. Problem Definition

2.1 Overview

We have focused in our research on solving the dilemma of the strategy selection problem (i.e. choosing the best modernization strategy for the legacy system based on multiple criteria that tends to evaluate the strategies). These strategies normally take a long time to evaluate as well as being less accurate since they sometimes depend on personal judgments rather than justified decisions. Also given the multiple components of the system, the decision gets tougher as we try to consider hybrid modernization strategies for each of them. For example considering a simple wrapping strategy for a certain component, and reengineering for another one.

In order to achieve the target modernized system, we have numerous paths from which we will select one. For example, to modernize a legacy system and enable its functionality as web services, we would have many options as for instance a black box modernization that only deals with the interfaces of the system or white box modernization that involves a source code modification such as wrappers or componentization (i.e. Realizing the system into a set of components), or even the most extreme case of a total system reengineering. While the evolution of various modernization strategies has created a wealth of choices for every organization, it also leads to a confusing situation. It was not easy to achieve a good modernization project and even hard to think about the number of things that we should consider while selecting the appropriate modernization technique. Below, we summarized the factors that govern the selection of the modernization techniques:
• **Organizational Factors**: The confusion for the organizations comes from the fact that the following factors should be considered in the strategy selection decision
  - Motivation behind applying this modernization.
  - The cost of applying each strategy on the system.

• **Design Quality of the system**: The design quality of the legacy system is a very important point that governs the amount of work (and hence the cost) needed to modernize the current legacy system towards the target modernized system.

• **Components/Modules Heterogeneity**: The current legacy system modernization research addresses the modernization problem with solutions that target the whole system as one unit, while the solutions can be targeted to each and every component/module in the system.

• **Legacy Systems Heterogeneity**: The modernization strategy selection is also tough since there is no such thing as a “one fits all” solution for the legacy systems; this is mainly due to the legacy system heterogeneity.

• **Legacy System Size**: The large Legacy Systems (~1 MLOC) also impose another challenge in Legacy Systems modernization since the bigger the system is, the harder it would be to analyze its components, and the harder would be the measurement of its design quality.

### 2.2 Problem Statement

Accordingly the strategy selection process is a tough process since the organization has to state all the above mentioned factors that may affect the decision making process then assess all possible strategies against the factors mentioned and the future value of the system, and then match all these factors against the target features of the system and come up with the optimum strategy. Most of the organizations currently cannot easily make a good decision regarding their modernization strategy so they go the easy way with black box modernization using the screen scraping technique, which is the most trivial one. So we have defined the problem statement as follows:

“Assisting the selection of a suitable hybrid modernization strategy from the current catalog of modernization techniques while taking into consideration the modernization business drivers, the quo state, the post modernization state of the system, and finally the boundaries of the organizations in terms of cost and duration (effort and resources).”

### 3. Previous Work

The modernization strategy selection problem has been tackled before by many researchers. In this section we will explain two important researches for this problem upon which we will be basing our work.

#### 3.1 The SMART Process

One of the most comprehensive processes of legacy systems modernization is the Service-Oriented Migration and Reuse Technique (SMART) [8] (the word migration is used here to express the idea of modernization) in the Department of Defense (DOD). Below is a summary of this technique which we have also used in our proposed approach as follows:
• **First**- Understand the Target SOA requirements as:
  - Requirements from the target SOA.
  - Characteristics of the target environment.
  - Specify the target quality attributes as performance, security, etc.
  - Estimate the efforts involved in writing the service interface
  - Estimate the Effort involved in data type translation to XML data types.
  - Estimate the effort required to describe the services for their discovery.
  - Estimate the cost required to migrate to SOA.

• **Second**- Analyze the current system
  - Reverse Engineering the system use cases (especially in interactive systems [2]) helps the modernization process focus on maintaining those use cases. If the systems requirement was not originally documented, which is mostly the case, requirement inference would need to be done to provide clear modernization acceptance criteria.
  - Reverse Engineer the system design and recover its architecture through documentation and modeling if the system is not well documented.
  - Identify the classes to be modernized to services.
  - Measure the system design quality:
    - Cohesion.
    - Coupling.
    - Programming standards compliance.
  - Existing system Size and Complexity.
  - Measure Change:
    - Black box vs. white box suitability.
    - Scale of changes required.
    - Commercial mapping software dependency.

• **Third**- Analyze the gap between the target system and the current system
  - Define the migration strategy based on risks of the unknown future state of SOA
  - Identify the dependencies between the services (to be built) and:
    - The user interface.
    - The commercial mapping software.
    - The rest of the code (sometimes you may use the function mining and slicing to isolate the functions with its code dependency as in [9]).

• **Fourth**- Develop a migration strategy
  - Update the current software architecture and documentation.
Define Compliant Services.
Understand the Data model content
Develop the requirements of the service.
Recalculate the cost and effort of the migration based on the code dependencies and the new understandings of the user requirements.

3.2 The Erradi Decision Making Process

Another One of the recent works done in this area is a paper evaluating strategies for the integration of legacy applications as services in SOA [7]. The authors of this paper explore the organizational factors that influence the decision of strategy selection, but they assess the system using a simple paper and pencil matrix as follows:

1. Legacy modernization business context: Identify the key business goals and the key pains in the current system and requirement from the future system in terms of performance and reliability.

2. Legacy portfolio assessment: assess the current system portfolio in terms of size, interfaces with other systems and technologies used. Manual and automated techniques should be used to handle such step.

3. Choosing of legacy modernization options using scoring and pair wise comparison models: using the below matrix as in Figure (1) and above input the legacy system strategy is chosen according the best matching between the factors given through the assessment phase and the context of the modernization.

<table>
<thead>
<tr>
<th>Evaluation Criteria</th>
<th>Wrapping</th>
<th>Re-hosting</th>
<th>Componentization</th>
<th>Reengineering</th>
<th>COTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business agility - rapid response to changing business requirements</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Systems integration - seamless Interoperability among internal systems and with business partners</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reduce integration cost</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reduce maintenance cost</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reduce total cost of ownership - including costs of operation, maintenance and upgrade of both software and hardware</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Move from batch processing to online processing and a near real time enterprise</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hard-coded business rules</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functional Redundancy</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lack of application knowledge</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Legacy skills shortage- unique skills required but hard to find and retain</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Matrix of SOA Modernization Strategy Evaluations [7].
The organization is expected to do the assessment beforehand, and then correlate their assessment on the given matrix to take a decision. Although for a manual process this should have been sufficient, this is not even the case, since it lacks the following:

1. Considering the organizational factor weights in the decision.
2. Considering the legacy system as a set of components rather than one big system that should be modernized using one strategy fits all.

### 3.3 Difference between Erradi and SMART

The Erradi and SMART processes agree in the main steps, except for the fact that SMART ends by designing the strategy that best fits the application while Erradi ends by decision making for selecting the best strategy out of the existing migration strategies.

After comparing the SMART and Erradi process we have reached the following conclusions:

1. The SMART process adds a very important step to the process which is considering the current capabilities against the requirements (i.e. gap analysis) in the SOA. This step is being neglected by Erradi which actually has its importance to take the right decision. Also Erradi didn’t focus much on the code quality metrics, while SMART focused upon it a lot to give a solid assessment for the legacy system.
2. Although Erradi in his research didn’t explain how he scores the values of each characteristic versus the strategies available, these values didn’t seem to be specific to his use case since they seemed to be generally logical. Accordingly, each organization can simply either use their own scoring or opt for using the one provided by Erradi since it merely depends on the personal judgments.
3. Erradi’s work did not consider the system as a set of components (i.e. to consider the modernization of each component to the target service on a separate context) but considered it as the whole system assessment. On the other hand SMART considered the components of the system and their mapping to the potential services.
4. Erradi on his final decision making step considers some key evaluation characteristics where the strategies get filtered based on them before being evaluated against the benefits or the cost. Although this is an important step to minimize the number of strategies being evaluated, Erradi only considered the goal of the modernization as a key evaluation to filter the strategies. This might not be the case as sometimes it is very important for the organization to consider certain technology pains or even cost as one of their key evaluation characteristics.
5. SMART finished its process with a design of the modernization strategy which can be as hard as the assessment process and will mostly result in a hybrid modernization strategy for the services in consideration. Therefore the strategy selection decision making framework presented by Erradi constitutes an easier initial step for the organization to start from. For example, if the organization runs the process and resulted in an ordered list where the reengineering is on the top of this list, the organization can start from this point to kick a reengineering project for modernization.
4. SOA- Modernization Strategy Selection Framework

In our research we have constructed a unified process out of the previously explored processes for the SOA modernization strategy selection problem. In this section we will present our process, and finally we will present the advantages of using it. The strategy selection framework is built over both the work Erradi and SMART. The framework is composed of:

1. **An assessment process**: The assessment process is mainly following the SMART approach although we have changed some details in the process to match the input for the final decision making.

2. **A decision making process**: The decision making framework was mainly based on the work of Erradi while amending steps in the calculation to get a more rich and accurate decision.

These two processes are presented in the form of a framework allowing the organizations to personalize the details of two processes as much as they can. The two processes flows through a series of steps, each step have a certain output which will be used later in the tool. The final output of the process will mainly come from the tool which is an ordered list of the strategies based on best first strategy.

Following is a brief explanation of the Assessment process and the Decision Making process and the advantages of using it.

4.1 The Assessment Process

**Establish the stakeholders/business context**

- Establishment of the key people that have knowledge about the legacy system.
- Establish an initial list of SOA modernization goals.
- Establish a list of characteristics that need to be identified for every component/module in the legacy system.
- Establish a list of constraints on the key quality attributes as performance, reliability.

This step is a combination of the first step in both works from Erradi and SMART. As explained this step involves also the SOA requirements exploration which logically flows with the identification of the stockholders to identify the business context.

The outputs from this step are the following artifacts:

- Art1: The list of characteristics that will be contributing to the decision
- Art2: Construct a description of the services needed

**Legacy system assessment**

- Discover the current system components.
- Analyze the current components and update the characteristics list
- Update the current issues list and benefits needed from the modernization.

This step is mainly the part of the assessment process where the legacy system gets...
analyzed into both some measurement numbers and modernization issues that contribute to the final decision of the strategy modernization.

The outputs from this step are:

- Art3: The current list of components in the legacy system along with their description.

4.2 Gap Analysis:

This is the new step that we have added to the Erradi process following the SMART approach.

- Map the list of services from Art2 to the current components list from Art3.
- Personalize Art1 to list the modernization issues (technology pains) that follow for each component to service modernization (if the service will be a result of legacy component/module modernization). The cost and effort will typically flow from the significance of these issues list.

This step mainly aims to introduce a new control level for the organization to control the significance of a certain characteristic. Erradi’s work was mainly done under the assumption that the significance of a certain characteristic will follow directly from its value such that, for example if the coupling between the objects is high, this should implicitly indicate a high significance. This might not be the case on modernization since the organization might choose to neglect this characteristic for the time being this specific service and produce it with the high coupling between objects.

The outputs from this step are:

- Art4: The significance (rates on the scale of 0 – 2) of the modernization issues for each component mapped to a service

**Example:** If the gap analysis showed a need for considering the technology pains (as coupling, obsolescence, etc) of the legacy components as the legacy component will be migrated to the service rather than making a new service then for example the lack of application knowledge should be highly significant. Its significance will flow from the fact that such legacy component/module will be migrated to a service and we need to be acquainted as much as possible to be able to fix the technology pains. On the other hand the significance tends to cancel the effects of certain factors when it is set to “Neglected” value such that it doesn’t affect any more in the strategy selection process.

4.3 Choosing potential modernization strategies:

This step aims at choosing the potential modernization strategies that the organization wants to choose one from them.

- Choose the potential modernization strategies, or you can just use the matrix of Erradi.
- Construct a matrix mapping the strategies to the characteristics. This matrix shows the importance of each characteristic versus each strategy, such that. For example, if characteristic A is important for strategy X, then it should have a high value.

The outputs from this step are:

- Art5: Mapping each modernization strategy to Art1 (characteristics artifact).
4.4 The Decision Making Process:

This is the second process in our framework which represents our final step for the selection of strategies suitable for each service. This step is the main theory behind our tool. This process is composed of multiple steps using the decision theory (Multi Criteria Decision Making) process to evaluate the strategies. This process can be scoped over either a certain service or a group of services or even the whole legacy system modernization project.

The MCDM [10, 11] is mainly used to take a decision under multiple conflicting criteria using many methods as the Analytical Hierarchical process (AHP) which involves pair-wise comparison and the Weighted Sum method. The MCDM mainly relies on analyzing the criteria’s contribution to the decision and rating those criteria’s using any of the methodologies.

The decision making process therefore uses this simple mapping between the criteria and the characteristics of the legacy system as decision criteria, such that each characteristic is treated as a criteria contributing to the final decision of the strategy selection. So we will be using the following artifacts:

- Art1: As the source of the criteria’s for the decision making process.
- Art5: As the importance provided between the criteria’s (Legacy system characteristics) and the multiple options (strategies) available for the decision.

**Initial Strategy Filtering:**

This is the initial step for filtering the strategies under consideration before the real assessment.

The input to this step is Art5.

Choose any three key evaluation characteristics. Rate the relative importance of these characteristics versus each other. Using Analytical Hierarchical Process (AHP) [11] to get the priority vector for the initial list of strategies according to these key evaluation characteristics. The evaluation process goes as follows:

1. Start with the relative importance of each characteristic versus each other.
2. Pair wise comparison between the characteristics using the input of the relative weights to get the weight of each priority vector these characteristics.
3. Using the matrix (Artifact 5) values of these characteristics vs. the strategies
4. Pair wise comparison between the strategies for each strategy to compute the priority vector of all the strategies for these all the characteristics.
5. Construct a matrix between the strategies and characteristics putting the values of the priority vectors of these strategies (as the row) versus each characteristic (as the column).
6. Multiply the first priority vector corresponding to each characteristic by the column of this characteristic (in the characteristic vs. strategy matrix).
7. Sum for each strategy and divide by the total to get a percentage for each strategy.
8. Compute the average percentage for the strategies and filter all the ones falling below the average.
4.5 Strategy Evaluation:

This is the step of the strategy evaluation versus the characteristics of the legacy system. The inputs to this step are legacy assessment result (Art1), the significance of each of the characteristics (Art4).

This step is done over a certain service/group of services/the legacy system as whole using, the characteristics values of the legacy component (between certain component and service) as output of the assessment phase and the significance of each of the characteristics according to there technology pains for gap analysis this service and finally Art5 after filtration, we can evaluate the most beneficial strategy for this service using the **weighted sum** methodology as follows:

- Multiply the values from Significance ($SI_x$) and the Characteristic ($C_x$) importance for every characteristic to get the Weight of each characteristic ($W_x$).

$$W_x = SI_x \times C_x$$

- Using the characteristics rate versus strategies matrix for each characteristic ($R_x$) and the output from the previous step ($I_x$), multiply both values and sum over each strategy ($ST_x$).

$$ST_x = \sum (W_x \times R_x) \quad x=1...y$$

Where $y$ is the number of characteristics

- Divide the sum of each strategy over the total sum, and order the strategies

The output from this step is the evaluated set of strategies ordered in a list based on the benefits from each of them with respect to the characteristics.

4.6 Cost Benefit Analysis:

Using the strategy rates computed in the previous step along with the cost, we can compute the cost benefit analysis as follows:

- Use the predefined relative cost of every strategy provided in the matrix
- Compute the cost of each strategy as its cost divided by the total cost
- Divide the Cost based rating of the strategies over the benefits based rating and this should result in how many cost unites will you pay for each benefit units.

The output from this step is an ordered list of the strategies based on the cost benefit analysis.

Advantages of SOA-MSSF

The main advantage of SOA-MSSF is that it brings the best of both worlds from Erradi and SMART and amends it with an automated tool to support the framework to make the organization job easier. The advantages of our framework are presented as a solution to the problems explained before in Erradi-SMART comparison.
Assessment Process

- **Significance of the legacy characteristics:** In the assessment process we have added a new dimension for the characteristics of the legacy system which flows logically from the gap analysis step in the SMART process. This significance which was assumed by Erradi to flow directly from the value of each characteristic. We have argued this fact since the organization may want to keep the value as it is in the new service which flows directly from the gap analysis result.

Decision making process

- **Tool automating the process:** The decision making process of the MSSF is automated using a software wizard that gathers the input from the user and evaluates the options on his behalf. This powers our process as it minimizes the effort needed from the organization and gives the organization a defined set of steps with defined outputs to use in the tool.

- **Evaluate the service-components rather than a whole legacy system evaluation:** The evaluation of the service components is already introduced by the SMART process but was not considered by the Erradi process. In MSSF we have considered the evaluation using the same concepts of SMART but rather at the component level.

- **Strategy selection rather than strategy design:** The SMART process ends by designing a strategy for the migration of the legacy system. This migration approach however induces more effort for the organization. We, in this work however, utilize the various modernization strategies presented by Erradi’s, and eventually select one of those strategies for modernization based on the earlier indicated factors.

5. Automatic Decision Making Tool

Our second contribution is the decision making tool, as described by the authors of this article in [14]. The tool is mainly based on the enhanced decision framework we presented in this article. The tool helps the organization through the whole process either using the artifacts created from our process, or helping them to input the values of some artifacts on the tool itself, so the tool starts with the initial step of framework and ends with automation of the final evaluation step.

The real contributions of the tool are:

- **Guides the SOA modernization strategy selection** framework through the mapping between the needed artifacts by the tool and output artifacts from the process.

- **Automates the source code analysis** for code quality metrics while forming a questionnaire for the rest of legacy system characteristics (the current version allows only for C++ code, although one could simply feed in the code quality characteristics by themselves. The source code analysis is currently been done using the CCCC API [12].

- **Automates the decision making process** of the framework over a component-service scope through an easy to follow wizard.

The tool is a wizard application walking the organization through the MSSF using the
artifacts and executing the decision making process at the end. The wizard mainly ends by a report explaining the output of each step in the decision making process. The wizard screens is a one to one mapping to the decision making process although using the output of the assessment process, therefore we will be explaining the tool in the form of a mapping to the process.

![Figure 2: Screen Flow of the Tool](image)

The screen flow of the tool as shown in Figure (2) is a simple mapping to the MSSF-Decision Making process. The tool is associated with the decision making process, and accordingly it starts after the assessment process. The screen flow is organized to start with the decision making process and then uses the results of the assessment process. The assessment process also involves automatic collection of the code quality metrics. The decision making process on the other hand involves the user in only two input steps and the rest of the evaluation is done automatically by the tool while the user is waiting for the final evaluation report.

**Screens Description**

**Welcome Screen**

This the first screen of the application which mainly aims at introducing the wizard to the user to expect what is next and explain the requirements of each screen. This screen also is the point where the application loads the artifact files and validates them.

**Choose Key Evaluation Factors**

This is the screen corresponding to the decision making process second step which is mainly concerned with selecting key evaluation characteristics which will affect two
steps of the process.

- **Filtration**: The filtration step is an important step where the user gets the opportunity to filter the modernization strategies based on important key characteristics. For example, the organization can choose a certain modernization driver to be used as filtering criteria where all the strategies that fall in this criterion are ignored.

- **Significance**: By default, all the key characteristics get a double significance to favor the strategy that has higher rates for these characteristics.

### Rate Relative Importance

This screen is intended to mainly rate the selected key characteristics relative to each other. The idea to simply say which one of the key evaluation characteristics is more important than the other, the reason behind that as explained before is giving the organization a control to order the priority of these keys or to even give them equal priority. This will allow the focus to be more on the higher priority characteristics rather than the lower ones during the filtration step.

### Input Significance of Each Factor

This is the step added to the process of Erradi which gives the organization more control over the variables using the result of the gap analysis. As explained before, the significance of the variable does not necessarily flow from how bad is it, because some times it is bad, but the organization chooses to neglect its effect when modernizing this component to the service. The catch here is that the scale is from 0 to 1 rather than from 0 to 5. This variable is mainly multiplied by the rate variable, such that if the rate indicates for example a very high value like (5) while its significance indicates a very insignificant value like (0.2), it will be basically considered as very low (1), so it will not affect the decision much. We have also added the value highly significant that is basically used to double the effect of a certain characteristic, which will give it more emphasis over the other characteristics.

### Choose Source Code

This is the step where you get to select the source code of the component in the legacy system that is currently being evaluated. The source code selected is measured using the CCCC API. In our application, we selected three sample variables (Depth of Inheritance Tree, Complexity, Coupling Between Objects) to use in our use case, although the tool collects a lot of measuring criteria. For the complete list of parameters that can be used, please refer to the artifacts section.

This is a recurrent screen. The idea is that this screen is being entered each time you need to consider a new component evaluation. As mentioned before, the organization may want to consider a hybrid modernization approach, so it should evaluate each component vs. service in a separate thread to focus on its criteria’s values alone (in this screen we focus on the code quality metrics).

### Input Legacy System Characteristics

This is the screen where the organization gets to input the result of their legacy system assessment for the characteristics given in the Artifact Art1. These are the values that are not involved in the code quality metrics or that are involved but the tool does not provide them. For more information on how to specify that a certain characteristic’s value will be come from the code metrics and other will come from the user directly as in this step, please refer to the artifacts section.

### View Evaluation Report

This is the final screen of the tool where the organization gets to see a report of the process steps output as follows:
First, the organization sees the output of the filtration process indicating the rates of the strategies before the filtration and the resulting filtered strategies.

Second, the organization sees the output of the assessment process in the form of an ordered list of the strategies giving the organization more benefits based on the following:

- Rates of the characteristics
- Code Quality Metrics
- Significance of the characteristics

Finally, the cost benefit analysis, figure (3) shows the result of the automatic cost benefit analysis using the benefits computed and the cost provided in Artifact Art5. The output is also an ordered list of the strategies based on the cost benefit analysis.

Figure 3: Final Report and Decision Made by the Tool

6. Case Study

We finally show a case study applying our process to select the best modernization option. The case study aims to:

- Proof the applicability of the process and the tool.
- Verify the results of the decision making process.

The case study is a simple open source financial application which does simple mathematics calculations. The application is developed in C++ with a command line interface. The application was a good candidate for SOA modernization since a simple requirement can be to transform these mathematical functions into web services to be able to
access them over the internet (integration driver).

Therefore, we adopted this project as a case study of our process with a formal set of requirements and targets to meet. The process is mainly applied over this software using the set of requirements that was provided with the Erradi matrix. The tool recommended a certain strategy which we executed to modernize the application.

The resulted system was tested versus the higher significant characteristics and was found to verify them.

Legacy System

The legacy system to be modernized is selected from the open source community. So the legacy system is an open source financial application called terreneuve [13] which is “A lightweight C++ library for quantitative finance applications. The purpose of this project is to design a framework to price a broad range of financial products, with an aim to be able to reuse the built objects if need be later. The project uses yield curves, assets, volatility surfaces, credit curve and Monte Carlo engine as its infrastructure for financial mathematics” [13].

The project is split into components according to the financial functions to be performed. Accordingly the components are named after their function, every component has a main method which is called with the required inputs to execute the function; this main function calls the other functions in the component.

The functions to execute need some product information which all the financial functions manipulate during their actions. This data is either imported before running the functions, or the user can use the sample products provided with the application which is for demo purposes only.

The user interface of the system is block mode GUI system based over the windows shell as shown on figure (4).

![Figure 4: Terreneuve Main Menu Interface](image-url)
Modernization Project

The modernization of this legacy system is a requirement imposed by us to test our process and tool. The modernization aims at mainly enabling the application using the Service Oriented Architecture (SOA) to be able to access the financial functions from the internet using web services as an infrastructure. So our main driver from this modernization project is integration. For the sake of just testing the modernization project is to mainly select one of the functions and enable it using web services.

The question that our process and tool answers is what strategy we should follow to modernize this financial function. To answer such question we will follow the MSSF process and use the tool to assess the options available.

1. **MSSF:** The MSSF process was applied over the legacy system using the same steps explained before and filling in the checklist of every step.
   - **Assessment Process:** The assessment process as explained before runs through a series of steps analyzing the application and its SOA requirements and ends by analyzing the gap between the system and the requirements. In the case the study the following were the output of the assessment process:

     - **Establish the stakeholders/business context:**
       - The List characteristics used was the same ones provided by Erradi work beside the legacy code quality metrics
       - The modernization goals were defined as:
         - Integration: Integrate this application with other applications easily through web services.
         - Modernize one service which is the European Options and the Black Scholes Model.

     - **Legacy system assessment**
       - The system components were discovered and analyzed using the following Characteristics:
         - **Code Quality Metrics:** This part will be done automatically by the tool.
         - **Others:** This part was evaluated manually and provided to the tool in the form of Q/A.

     - **Gap Analysis**
       - This step was reflected in the significance of the factors in the tool.

     - **Choosing potential modernization strategies**
       - The potential list was used from the Erradi work along with the list of legacy system characteristics.

   - **Decision Making Process (Tool)**
     The Decision making process was executed directly using the tool where the
above inputs were provided and the evaluation report were presented at the end of
the wizard.

The Decision making tool steps were processed on the same order of the screens
provided above as follows:

- **Key Evaluation Characteristics:**
  - The key evaluation characteristics chosen were Systems Integration, Modernization Duration, and Reduce Integration Cost.

- **Key Evaluation Characteristics Importance:**
  - **Systems Integration vs. Migration Duration:**
    - The integration as a driver was much more important than the duration, so it was given “Very High” versus the duration.
  - **Reduce Integration Cost vs. Duration:**
    - The integration cost was much more important than the duration but not as high as the integration characteristic, so it was given “Medium” versus the duration.
  - **Systems Integration vs. Reduce Integration Cost:**
    - The integration was given “Low” versus the reduce integration cost, since it represents higher importance than the cost.

- **The significance of each Characteristic:** The following characteristics were given higher significance while the others were neglected.
  - **Lack of application knowledge:**
    - Was given a “Very Significant (1.2)” significance since we do not know anything about the application.
  - **Modifications tend to have heavy ripple-effects:**
    - Was given a “Significant (1)” since it considers the effort to be done which we need to minimize as much as possible
  - **Difficult to integrate with new breed of technologies:**
    - Was given a “Highly Significant (1.5)” significance since it servers our main modernization goal.

- **Input legacy system characteristics:** The following inputs were given to the legacy system characteristics:
  - **How do you rate your acquaintance of the application?**
    - Since we do not have much any knowledge about the code skills, then this parameter is “Very Low”.
  - **Modifications tend to have heavy ripple-effects.**
    - This factor according to our configuration depends on Coupling between the objects, complexity of the code,
business agility as a driver (which is not applicable in our case), and batch processing as a driver (which is also not applicable in our case).

○ How much difficult is it to integrate with the new breed of technologies?
  ▪ The current system cannot integrate with new technologies, so it was given a “High”.

○ How do you rate systems integration as your modernization driver?
  ▪ This is the main aim of the modernization so it was given “Very High”.

○ How do you rate reduction of integration cost as your driver?
  ▪ The cost reduction is “High”, since we need to finish this project in minimum cost.

○ How much duration have you allocated for the modernization project?
  ▪ The duration allocated in our case to modernize the application is “Very Low”.

▪ Evaluation Report: As we can see from the report below, figure (5), that the best choice in benefits is “Wrapping” and the best choice after the cost benefit analysis is also wrapping. So obviously the recommended solution is to wrap the component as a web service using any of the new technology wrappers.
2. **Legacy Components Wrapping:** The next step after using our tool and evaluating the case study was to simply wrap these components using web service technology. Such a project was executed in order for us to verify that the modernization goals aimed through the assessment process were met successfully. So the financial application was successfully wrapped using the .NET Web Services technology and the project progress and results were analyzed in the below table (1).

Table 1: Case study analysis of the implementation vs. requirements

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Significance</th>
<th>Value</th>
<th>Achievement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration</td>
<td>Highly Significant</td>
<td>Very High</td>
<td>Yes</td>
<td>The integration goal is definitely achieved using this wrapping</td>
</tr>
<tr>
<td>Duration</td>
<td>Highly Significant</td>
<td>Very Low</td>
<td>Yes</td>
<td>The duration was minimal which achieves our goal of minimum duration</td>
</tr>
<tr>
<td>Reduce Integration Cost</td>
<td>Highly Significant</td>
<td>Very High</td>
<td>Yes</td>
<td>The cost was minimal since we didn’t need time or resources to modernize the service and we used the available windows and IIS service.</td>
</tr>
<tr>
<td>Lack of application Knowledge</td>
<td>Very Significant</td>
<td>High</td>
<td>Yes</td>
<td>We didn’t need much application knowledge while using the wrapping strategy.</td>
</tr>
<tr>
<td>Modifications tend to have heavy ripple-effects</td>
<td>Significant</td>
<td>Low</td>
<td>Yes</td>
<td>Although this characteristic were significant but its value was low which means low effort, and as we can see the project was finished using minimal effort</td>
</tr>
<tr>
<td>Difficult to integrate with new breed of technologies</td>
<td>Highly Significant</td>
<td>High</td>
<td>Yes</td>
<td>Using the wrapping of the WS, we can easily now integrate with the new technologies</td>
</tr>
</tbody>
</table>

The above table (1) has shown the analysis of the modernization project achievement versus the goals and characteristics placed for this project. Generally as we can see that all the modernization goals are achieved and the modernization characteristics are considered with great success.

7. **Conclusion**

The research done in this area of SOA modernization strategy selection is still at its beginning. Two of the major strategy selection processes presented in the research was explored with the aim of automating the selection process. The two processes explored were Erradi strategy selection process and the SMART process. The two processes were
compared to show the similarity and difference between them where it was found that both processes agrees in the main legacy assessment part except that the SMART process has an extra step of the gap analysis. The gap analysis step mainly aims at analyzing the gap between the current state of the system and the future state of the system. Both processes on the other hand differ in the final modernization strategy output such that SMART designs a new strategy while Erradi selects one of the existing.

Accordingly our thesis has contributed in two main parts:

- A new framework “SOA Modernization Strategy Selection Framework” SOA-MSSF was designed that combines the best of both processes in a highly customizable framework where the details of the process can be easily personalized by the organization using the guidelines that we have presented in the framework. The framework is mainly divided into two parts: First an assessment process of the legacy system which assesses the legacy system according to a predefined set of characteristics considering the requirements of the modernization project, system assessment and the future state of the system with a gap analysis between the Quo state and the future state. Second, a decision framework which follows Erradi’s work while extending it with a gap analysis of the assessment process. This decision making framework helps the organization to take the decision of the modernization strategy using the results of the assessment process.

- Our second contribution is automating the decision making part of the framework thus making the organization’s job easier. The strategy selection decision making was previously presented in the research of Erradi through a manual process with the aid of some analysis tools to feed in the parameters of the decision a tool. We automated the entire process.

Finally, we evaluated the above two contributions using a real case study of an open source C++ legacy system called Terreneuve. The Terreneuve legacy system – Black Scholes component was analyzed for modernization using the MSSF and the decision making tool. The selected strategy was Web Service wrapping which was executed in a separate project to test the goals and characteristics upon which the decision was made. According to the analysis of results, the modernization project was found to meet all the goals of the modernization and to consider all the characteristics of the legacy system with the optimal strategy selected.

References

Authors

Sherif G. Aly received his B.S. degree in Computer Science from the American University in Cairo, Egypt, in 1996. He then received his M.S. and Doctor of Science degrees in Computer Science from the George Washington University in 1998 and 2000 respectively. He worked for IBM during 1996, and later taught at the George Washington University from 1997 to 2000 where he was nominated for the Trachtenberg prize-teaching award for his current scholarship and scholarly debate. He spent two years as a guest researcher for the National Institute of Standards and Technology at Gaithersburg, Maryland from 1998 to 2000. Dr. Aly also worked as a research scientist at Telcordia Technologies in Morristown, New Jersey, in the field of Internet Service Management Research, and as a Senior Member of Technical Staff at General Dynamics Network Systems. He also consulted for Mentor Graphics and taught at the German University in Cairo. He is currently a faculty member at the Department of Computer Science at the American University in Cairo. Dr. Aly published numerous papers in the area of distributed systems, multimedia, digital design, and programming languages. His current research interests include pervasive systems, programming languages, multimedia, directory enabled networks, and image processing. Dr. Aly is a member of IEEE.

Rafik Amir received his M.Sc. degree in Computer Science from the American University in Cairo. He is currently a doctoral student at the University of Birmingham, and has many interests including Software Engineering as well as Systems Biology. He is currently working on the evolution of Genetic Regulatory Networks (GRNs) in bacteria. Rafik has numerous publications in the domain.