Research Journal of Applied Sciences, Engineering and Technology 4(3): 178-185, 2012
ISSN: 2040-7467

Submitted: August 26, 2011 Accepted: October 14 , 2011 Published: February 01, 2012

Deployment Strategies for a Reengineered Information System in Context of Legacy System

[1]Musawwer Khan, [1]Wasif Nisar, [1]Ehsan Ullah Munir, [2]Waqas Anwar and [3]Islam Ali
[1]Department of Computer Science, COMSATS Institute of Information Technology, Wah Cantt (Pakistan)
[2]Department of Computer Science, COMSATS Institute of Information Technology, Abbottabad (Pakistan)
[3]International Islamic University (Pakistan)

Abstract: The Replacement of a legacy system with newly reengineered Information System has always been a challenging task. The end-users and the integrated software environment of legacy system make things harder for the deployment team. The legacy data migration challenge complicates the very important deployment phase. In this paper however, we have discussed and analyzed three different strategies to deploy a newly reengineered Information System and replace the legacy system in a homogeneous environment. This paper mainly addresses the issues concerning minimum disturbance to the legacy environment and smooth deployment of the new Information System. It is pertinent to mention that the issues regarding the development of reengineered information system or the legacy data migration have not been discussed here. At the end we have provided a detailed analysis and finally recommended the most optimal strategy among all. The rationale behind the suggested solutions is an extensive study conducted at our research and development Lab.

Key words: Deployment strategy, information system, legacy system

INTRODUCTION

Replacing a legacy system is very challenging and critical task since there is a big risk to forgo precious organizational data and information owned by the legacy system (Ian Warren *et al*., 1999). Legacy systems carry the full confidence of end-users which must not be shaken at any cost. Replacing the old system with newly reengineered information system greatly increases the new IS ability and evolution to meet future business requirements (Jean *et al*., 2002). To preserve the asset represented by the legacy system, the acquaintance with it achieved by the system maintainers and end-users, and the continuity of execution of current operations during the reengineering process, the system needs to be reengineered gradually(Alessandro *et al.*, 2003). And also the issue of frequently generated change requests by the client during the reengineering must also be dealt effectively. After the system has been reengineered the big challenge of replacement of the legacy system arises. Taking measures to provide very minimum disturbance to the legacy system functionality and to the existing operational and business environment is indeed a great challenge (Jes *et al*., 1999). The (Ahmad, 2010) presents more about the software renovation and explains how legacy software evolutions take place. As the legacy system is proven and tested so the functionality of the system should be used whenever and wherever possible.

The source code transformation of legacy system and generation of target code is the focus in (Correia *et al*., 2006). The author proposed three steps for transformation of legacy system into target system. In the first step the author suggests the reverse engineering of source code, while in the second steps he focused on the preparation of transformation techniques to achieve the target code. In the third and final step the author suggests the generation of target code.

In (Jesu´s *et al*., 1999) Chicken Little Methodology has been introduced by the author, stating an eleven step process with the concept of gateways. According to the strategy both the legacy and target information systems run in parallel but the target system is developed incrementally and hence the legacy system is migrated gradually. When migration is completed the legacy system can then be retired.

Alessandro Bianchi and his co-authors have presented the iterative reengineering of legacy system. This strategy has features that prevents the problem of maintenance request arose during the reengineering process, by freezing only those change requests which are relevant to the part being engineered (Alessandro *et al.*, 2003).

Corresponding Author: Musawwer Khan, Department of Computer Science, COMSATS Institute of Information Technology, Wah Cantt (Pakistan)

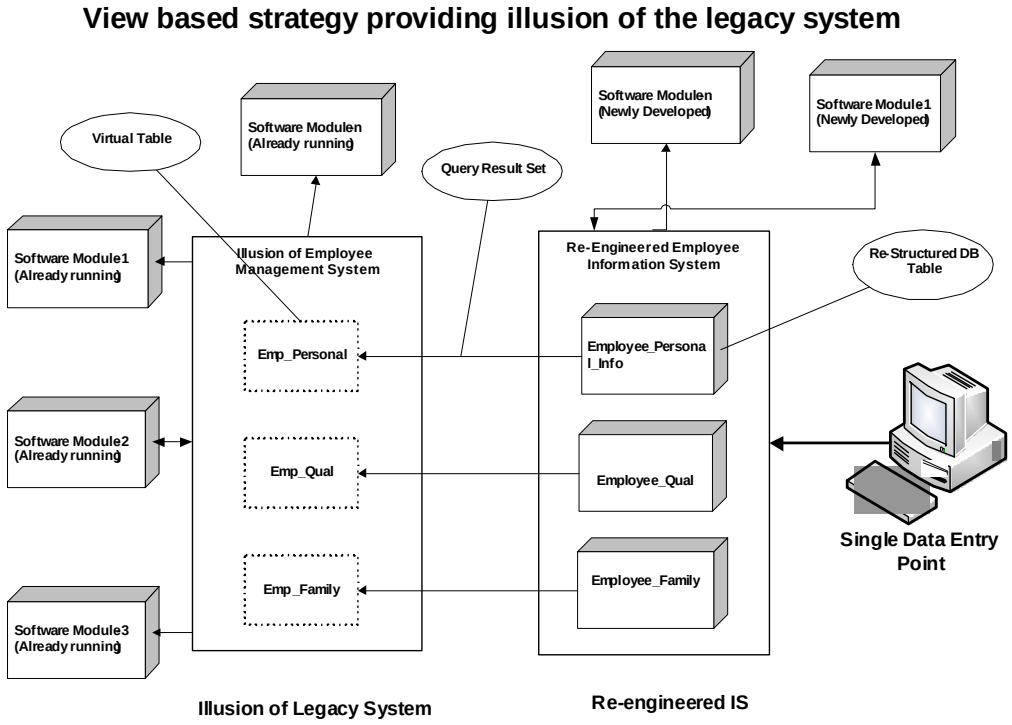

Fig. 1: Illusion of legacy system (Virtual tables)

The aim of the study is neither to explain the problems pertaining legacy system nor discuss various methodologies for reengineering or restructuring the legacy system. This also excludes the discussion on design and development strategies for reengineering or restructuring the information system. The primary and core objective of our work is to explore the pros and cons of three software deployment strategies and evaluate its effectiveness once the information system is reengineered in context of a legacy system. Furthermore we have also given a rationale to choose out the most appropriate and optimal strategy that could cause very minimum disturbance to the legacy environment and also require minimum effort to implement. We have also explained in detail our observations regarding the outputs achieved during the deployment of each strategy. The observations are thoroughly based on the practices we experienced in our research and development lab.

DEPLOYMENT STRATEGIES

Illusion of legacy system Virtual tables: The strategy provides an illusion of the actual database objects being use in the legacy system. Primarily the database tables are more common objects needed to be focused, and the rest after that.

The Fig. 1 shows an abstract layer indicating the illusion of legacy system. All the systems operating in the legacy environment do not get disturbed as shown. The Fig. 1 depicts that the reengineered information system appears in two flavors, one is used to serve the new environment and the other is used for legacy. There is only one entry point as shown. It implies that the end user does not need to maintain both old and new systems; rather he should be interacting with only one system i.e. newly reengineered information system.

Another database object called “view”, is a virtual table based on a query that do not contain the data itself. Table and view can have the same identifier, in-fact they are logically two different database objects. In our PISProject, we took this phenomenon as our first test bed, and replaced all the database tables with views having thesame name providing the illusion of respective tables.

Scenario:
Legacy Table name: Organogram
Illusioned with:
CREATE OR REPLACE VIEW Organogram AS (SELECT dept_id, department
FROM new_organogram_table)

Here we see that a view named *Organogram* has been created, based on the new table new_organogram table, making an illusion of legacy table available with the same name. The Legacy table *Organogram* was replaced with the view having the same name and all the

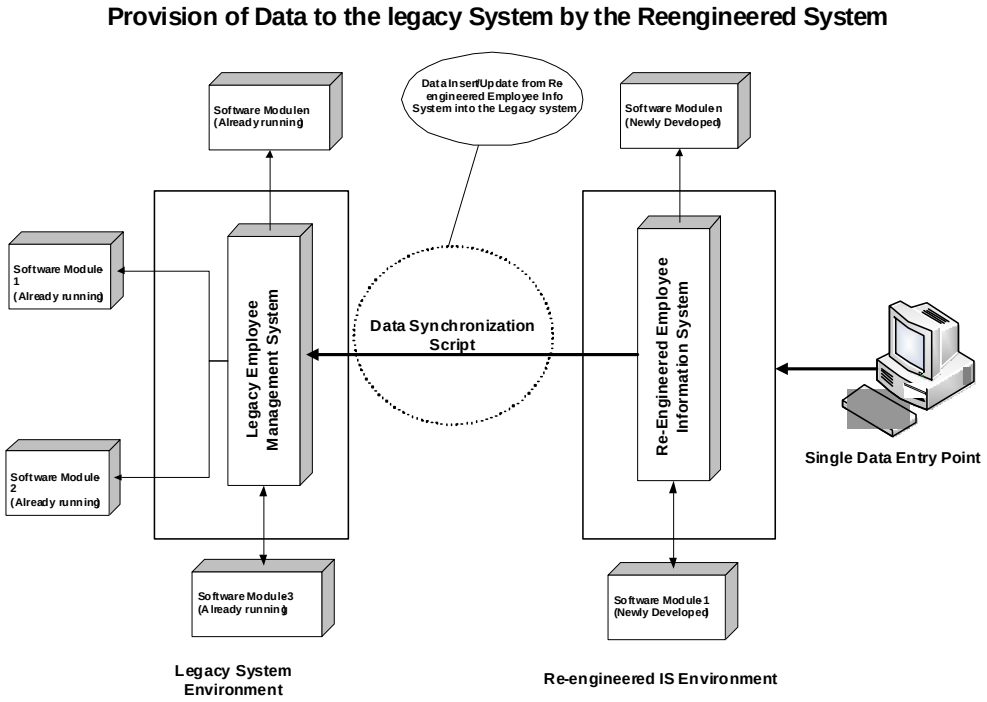

Fig. 2: Feeding to the legacy system

access rights previously been granted to the users, are re-granted on the newly built view. This phenomenon does not affect the end-user layer or the legacy system environment.

To evaluate the effect of this action we shall see the following SQL statements:

Query#1: SELECT* FROM Organogram (Table before replacement)

Query#2: SELECT* FROM Organogram (A view after replacement)

Now evaluating the output of both the queries we get:

R1 = {R| R is a set of records ^ a, b, c, d are the attributes}
P1 = {P| P is a set of records ^ a', b', c', d' are the attributes}

Hence it is always true that R1 = P1. The result achieved from the given scenario was also found more relevant to a programmer's perspective who manipulates the legacy data in his/her own software application. The end-user's view remained unaffected altogether, i.e., the same user-interfaces (Read only in this case) and output reports are unchanged while having only one data entry point.

Feeding the legacy: The Bing Wu and his co-researchers (Bing *et al*., 1997) have suggested a Butterfly Methodology which gives the concept of Data-Access-Allocator (DAA) that redirects the transactions on legacy system to the target system. The results of all these transactions are stored in a series of temporary storage called *TempStores* (TS) and hence the Data Transformer transforms the data to the target system.

The authors has focused primarily on the data migration issue in his work, which involves the temporary storages and transformation of data to the target system. In our study we introduce a new paradigm to keep the legacy alive until the legacy environment is reengineered.

In our research lab we worked out the eeding the legacy strategy that sounds like arallel Deployment but unlike parallel deployment process there is only one data entry point, i.e., via new system. The legacy system remains operational, taking the data from new information system. In PISProject a supplement code was added to make it sure that whenever a transaction is performed in the new system, the same would happen automatically in the legacy system. The end-users will keep using their own interfaces (Read only in this case) and output reports without any hindrance.

In Fig. 2 the bridging line indicates a strong data flow link between the two systems. The reengineered information system constantly provides bread and butter to the legacy system. The entry point is single and that

implies the end user interacts with only one system i.e. reengineered system. This strategy demands that the legacy system shall be switched off gradually means the legacy system can be got rid of via diverting the focus of legacy environment to the newly reengineered information system incrementally at a later stage. The legacy environment shall not be disturbed at any cost doing so may play a havoc.

Scenario#01-Insertion:

INSERT INTO new_employee_table
(employee_id, employee_name, Marital_Status)
VALUES (5, 'Ali', 'Single')

Consider the following SQL statement being executed in reengineered information system.

This statement will add a new record to the table ew_employee_table at the same time the supplement code executes as below:

INSERT INTO old_employee_table
(emp_id, emp_name)
VALUES (5, Ali

Similarly the updation of record must also take place in both the systems.

Scenario#02-Updation:

UPDATE new_customer_table
SET city_name = arachi
WHERE customer_id = 18

The same statement was executed via supplement code for the legacy system as:

UPDATE old_customer_table
SET city_name = arachi S
WHERE customer_id = 18

Similarly the deletion case can also be implemented using this strategy. Hence the legacy system can be degraded gradually when all other software modules operating in the legacy environment are reengineered or integrated with the newly reengineered information system.

Clean sweep: In this strategy the legacy system is simply vanished, and the new system is made operational. The author of (Jes *et al*., 1999) has also suggested a cut-and-run strategy little bit similar to clean sweep strategy with a little difference of approach towards implementation. Unlike cut-and-run, the clean sweep strategy ensures that the legacy environment has been redirected to the new

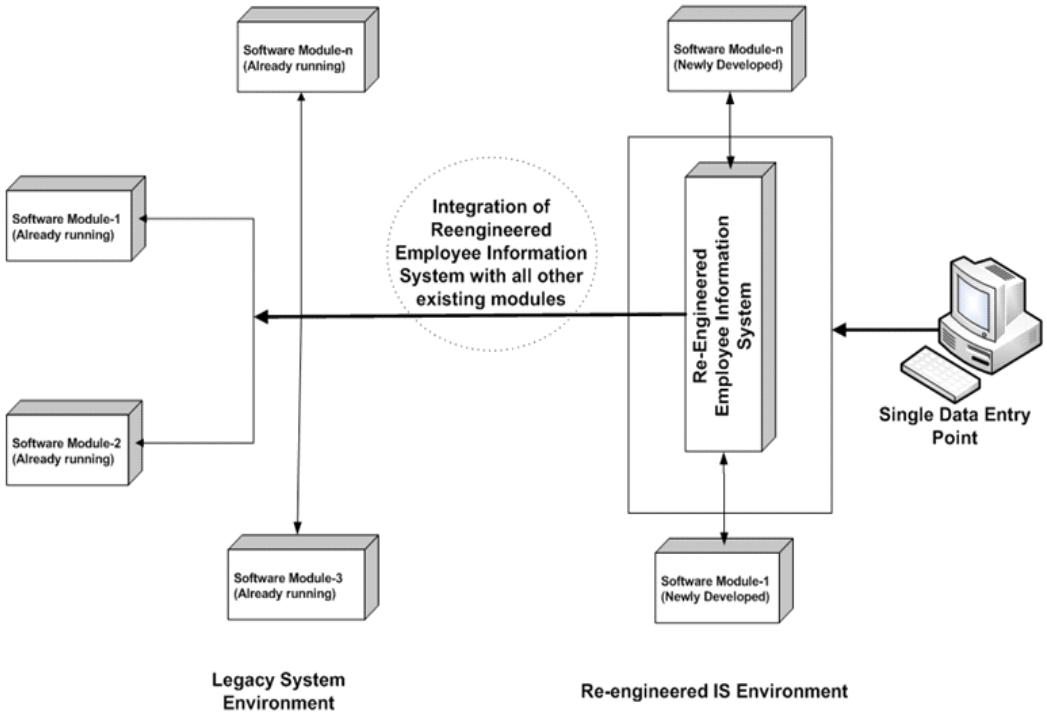

Fig. 3: Reengineered system vanishes the legacy system

system before brushing away the legacy system, so this approach minimizes the risk of turbulence in the legacy environment. This strategy is less suitable in the situation where the legacy system is more dependable and hard to retire. Nevertheless this strategy is simple and easy to implement as compared to the other two strategies.

The Fig. 3 shows a direct link between the legacy environment and reengineered information system, which implies that there is no existence of legacy system any more. All the systems in legacy environment are directly accessing the newly reengineered information system. It indicates that all the systems have to change their focus at once, towards the new system which is although irrational. As shown there is only one entry point but the end user input/output environment will change completely. Despite the simplicity and easy implementation, the clean sweep is a bit impractical strategy because turning down the legacy environment abruptly is not that easy.

Study outcomes: All the three strategies were implemented, analyzed and deeply observed. A detailed comparative analysis has been presented here which depicts our observations based on the outcomes regarding each strategy.

After implementing the first strategy in PISProject, we came up with the following conclusions:

Illusion of legacy system (Virtual tables):

- The illusion of Table independence factor is very impressive but it carries the complexity of adopting alltheriggers based on the legacy tables. This practice is followed though in the other two strategies as well.
- The referential Integrity constraints on the table need to be adopted by the relevant views; otherwise no insertion/ updation could be possible in the referenced tables.
- The strategy is tantamount to urn all the boats That is after deploying the new system, rolling back to the legacy system in case of failure, shall be a big bang. As all the transactions took place in the new system only, so it could be extremely difficult to identify all the changes and post these back to the legacy system.
- An overhead as compared to the second strategy is that all the table level access rights already been granted to the existing users need to be re-granted on the views.
- View is nothing but a set of one or more tables joined together via SQL Statements. Querying a view is just to multiply the cost of table joins in the underlying queries.
- Substantial effort and care is required to extract the relevant data of the underlying table into its corresponding columns of the view.

The above mentioned observations lead us to a new strategy “Feeding the legacy system”. Following are the key points being observed after implementing the strategy:

Feeding the legacy system:

- Adopting the underlying riggers of the legacy tables is not required, and also the issue of user access rights does not arise.
- Besides the above mentioned advantage, a huge effort is required to keep both the system synchronized in terms of design structure and data, and even sometimes it becomes unmanageable to cope with this issue.
- New requirements generated by the end-user causing a major/minor change in the database structure should be handled tactfully, as it would cost double effort to maintain the changes at both ends.
- Feeding the legacy system sometimes causes a compromise on provision of handy feature in the new system because it can adversely affect the effort to keep both the system synchronized.
- The big plus in adopting this strategy is the minimum risk of failure. Rolling back to the legacy system at any point-in-time is not a big deal as the same set of data exists at both ends.
- No need to worry about the interfacing issues with the systems running in the legacy environment. The integrated environment is uninterrupted and end-users are undisturbed as they use the unchanged system.
- The output of the legacy system is unaffected and the end-users feel no difference in terms of productivity.
- Using this strategy the data being migrated into the new system can best be validated by the end-user, because the user shall be able to compare the data at both ends
- The greatest advantage of adopting this strategy is that the legacy system is not being replaced abruptly and the re-engineered system gets matured before it becomes indispensable.
- Before adopting this strategy the re-engineered system must be thoroughly tested and validated by the customer, otherwise every change will cost a big price.
- If the databases at both ends are heterogeneous, more effort shall be required to keep both the databases aligned.
- After adopting this strategy the Database administrator may get busy while managing the two databases simultaneously.

The outcomes of the third strategy are:

Table 1: Summarized evaluation results

Strategy-A	Strategy-B
The strategy has embedded data inconsistency, data authenticity and environment independence etc issues, which tends to the issue of discrepancy in the previous and existing data. So only a medium level adjustment can be achieved with the legacy environment	The data feeding methodology brings no change to thedesign, code or implemented business logic of legacy system andalso high data authenticity, output consistency etc causeahighlevel adjustment factor i.e low or minimal disturbance to the legacy environment

Table 2: Comparisons of adjustment factors

	Strategy-A	Strategy-B
Adjustment Factor	(Low, Medium, High)	(Low, Medium, High)
Output consistency	2	3
User's independence	2	2
Data authenticity	1	3
Environment independence	2	2

Clean sweep strategy:

- The abrupt change to the legacy system causes a high risk of unavailability of critical information already being used by the end-users and the legacy environment.
- It is completely urning all the boats strategy, because un-acceptance or system failure greatly affects the data integrity aspect.
- Besides all its inadequacies it a plug and play strategy and require little effort to implement.

Evaluating the strategies: We categorized a successful deployment strategy in three metrics given below:

- The magnitude of Adjustment Factor (AF) for Re-Engineered Information System with the Legacy environment whose possible values are Low, Medium and High.

Results description: A summarized evaluation has been mentioned in the above Table 1. The overall assessment about the two strategies is given. The strategy A, has medium level adjustment factor as shown, indicates substantial disturbance to the legacy system and its environment while on the other hand the strategy B, has a high level of adjustment factor which indicates that it shall cause minimum disturbance to the legacy system and its environment.

Following are the adjustment factors being considered for the study with their observed output values.

Results description: In the given Table 2 the evaluation results have been elaborated in more detail. The adjustment factors are listed, with their observed or judged ratings. Output consistency and data authenticity have high AFs in strategy B, implies very minimum disturbance to the system while data authenticity has very low adjustment factor for strategy A, indicates very high disturbance in terms of data authenticity because of a new

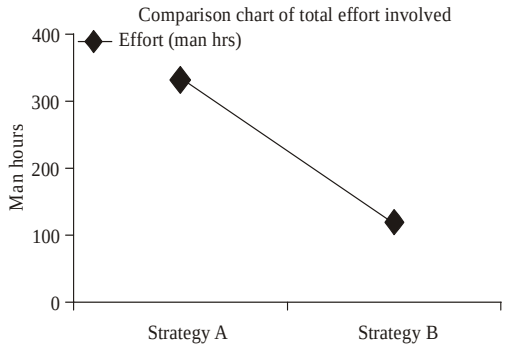

Fig. 4: A comparison chart of total effort involved for strategy A and B

layer of views, which picks the data from various corresponding tables. User's and environment independence has same ratings in both the strategies because the input aspect of all users' using the legacy system is affected while the output remains unchanged. Likewise the legacy systems which are operational in the legacy environment feel no change whatsoever hence a medium level adjustment factor for both the strategies.

- From the given chart it is clear that the strategy B is comparatively a better strategy.
- The amount of effort required to deploy the strategy: After implementing the two strategies we got the following results:

After plotting the total effort involved in each strategy we get the following line graph:

Results description: The Fig. 4 shows the difference in the overall effort involved in the implementation of both strategies. When adopting the strategy-A, we had to undergo three main tasks as shown in the Table 3. The preparation of views is a very complex task as it needs to contemplate on each corresponding table and attributes that is why it took the maximum time i.e 176 man hours.

Table 3: Comparison table of deployment effort

Strategy-A		Strategy-B	
Task	* Man (h)	Task	Man (h)
Prepare views	176	Develop data synchronization Triggers	120
Adopt the legacy triggers	120		
Rehashing the referential integrity constraint and granting legacy object access rights	40		
Total	336		120

*: The man hours were calculated for the same persons who worked on the implementation of both the strategies

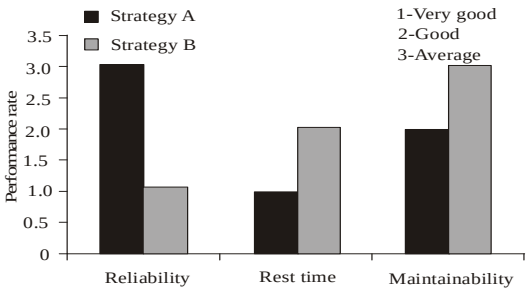

Fig. 5: Performance rating chart of the two strategies

Adopting the legacy triggers is a common task which involves the implementation of same business logic in the new system as it was there in the legacy system. Rehashing or changing the definition of referential integrity constraints is an additional task triggered by the preparation of views and took 40 man hours. So when all these tasks were performed it took a total of 336 man hours to complete. While on the other side, we had to complete only one task which took 120 man hours. The difference is obvious, not only with respect to the no of tasks but also in the total effort involved. From the table it is evident that strategy B leads with the minimum effort.

Performance of the strategies:

Reliability: Strategy-B is more reliable than strategy-A, because unlike Strategy-A, it does not break the link with the legacy system and its environment in terms of data flow. It has an embedded advantage of data authenticity as the legacy system is always available to help in data authentication.While incase of Strategy-A, the data is being gathered from various normalized tables into single view mimicking the actual legacy table. There might be a chance to pick the inappropriate data into a corresponding attribute.

Response time: Strategy-A has relatively a good response time (availability) as the current and up-to date data is available for use all the time. While in case of strategy B, the updating mechanism may not work properly causing a delay in providing the current and fresh data. But sometimes the strategy-A may get worse when the definition of views contain complex queries.

Maintainability: The maintenance of system in case of strategy-B is relatively difficult, because making even a tiny change or enhancement require more effort if handled carelessly. And also wrong data provision can cause valuable information loss or can adversely affect system reliability.

The Fig. 5 shows the rating of various performance parameters i.e. 1-very good, 2-good, 3-average.

Keeping the legacy system alive requires highly reliable approach; and here we see that the strategy-B is better with the reliability performance parameter. Although strategy-A is good in response time and maintainability nevertheless these performance parameters are not that crucial in this context. So after evaluating the results of the three performance parameters, it is evident that the strategy-B i.e feeding the legacy system provides optimum solution as compared to the strategy A, so we recommend the strategy-B as best deployment strategy for a reengineered information system to replace the legacy system.

CONCLUSION

In this study three deployment strategies of reengineered information system in the context of legacy systems have been discussed in detail. The outcomes of all the strategies were explained in sufficient details stating the pros and cons of each deployment strategy. At the end, two applicable strategies were evaluated in with respect three common core criteria parameters. We also tried to present a rationale that could help in selecting and adopting the more adequate and suitable strategy between the two. After a detailed evaluation work we recommend the strategy B as the most optimum and stable strategy, because it not only causes minimum disturbance to the legacy environment but also requires relatively little effort to implement. Here we also propose that adopting a hybrid strategy means the amalgamation of the two strategies being discussed could be an area of potential research. A new strategy could be introduced in which both the strategies reinforce each other by using the advantages of one another. To explore the benefits and issues associated with the proposed strategy could be the area of interest for future researchers.

REFERENCES

Ahmad, D.S., 2010. An Integrated Approach for Legacy Information SystemEvolution Department of Computer Science College of Arts & Science inWad Al-Dawasir, King Saud University Wad Al-Dawasir-11991 Kingdom of Saudi Arabia.

Alessandro, B., C. Danilo, V. Marengo and V. Giuseppe, 2003. Iterative Reengineering of Legacy Systems, Dipartimento di Informatica-Universit di Bari-Via Orabona, 4, 70126 Bari-Italy, bianchi, caivano, Dipartimento di Statistica-Universit di Bari-Via Rosalba, 53: 70124, Bari-Italy.

Bing, W., L. Deirdre, B. Jesus, R. Ray, G. Jane, W. Vincent and Donie, O.S., 1997. The Butterfly Methodology:AGateway-free Approach for Migrating Legacy Information Systems, Engineering of Complex ComputerSystems, Proceedings. Third IEEE International Conference on Digital Object Identifier:10.1109/ICECCS.1997.622311Publication, pp: 200-205.

Correia, R., C. Matos, M. El-Ramly, R. Heckel, G. Koutsoukos, L. Andrade, 2006. Software Reengineering at the Architectural Level: Transformation of Legacy System, Department of Computer Science, University of Leicester, U.K, ATX Software, Lisboa, Portugal.

Ian Warren (University of Bradford) with contributions from Jane Ransom (Lancaster University, UK) Markus Breuer (GEI debis, Germany) Claude Villermain (CAP Gemini Innovation, France) John Favaro (Intecs, Italy) Eirik Tryggeseth (NTNU, Norway), 1999, The Renaissance of Legacy Systems, ISBN: 1-85233-060-0.

Jean, H., H. Jean-Marc, T. Philippe and H. Jean-Luc, 2002. Strategies for Data Reengineering, Institut d Informatique, University of Namur Rue Grandgagnage, 21-B-5000 Namur-Belgium.

Jesu's , B., L. Deirdre, W. Bing and G. Jane, 1999. Legacy Information Systems: Issues and Directions, Trinity College Dublin, Country.