Proccedings of the 17th International Database Conference (DATASEM '97), Brno, Czech Republic, October 12 - 14, 1997. pp 129-138, Ed. Czechoslovak Computer Experts.

Legacy System Migration : A Legacy Data Migration Engine

Bing Wu*, Deirdre Lawless*, Jesus Bisbal*, Jane Grimson*
Vincent Wade*, Donie O'Sullivan[#1], Ray Richardson[#2]
*: Computer Science Department, Trinity College, Dublin, Ireland.
Email: {*name.surname*}@cs.tcd.ie
#: Broadcom Éireann Research, Dublin, Ireland.
Email: #1 : dosullivan@ccgate.broadcom.ie; #2 : rr@broadcom.ie

Abstract: The widespread use of computer technology over several decades has resulted in some large, complex systems which have evolved to a state where they significantly resist further modification and evolution. These *Legacy Information Systems* are normally mission-critical : if one of these systems stops working the business may grind to a halt. Thus for many organisations, decommissioning is not an option. An alternative solution is Legacy System Migration which has recently become an important research and practical issue.

This paper presents an approach to mission-critical legacy systems migration: Butterfly methodology. Data migration is the primary focus of Butterfly methodology, however, it is placed in the overall context of a complete legacy system migration.

Key Words - Legacy systems, migration methodologies, data migration.

1. Introduction

Legacy information systems[1] typically form the backbone of the information flow within an organisation and are the main vehicle for consolidating information about the business. If one of these systems stops working the business will generally grind to a halt. These legacy ISs have posed numerous and important problems to their host organisations for years. The worst ones being:

- these systems usually run on obsolete hardware which is expensive to maintain and reduces productivity due to its low speed;
- maintenance to software is generally expensive, tracing failures is costly and time consuming due to the lack of documentation and a general lack of understanding of the internal workings;
- integration efforts are greatly hampered by the absence of clean interfaces;
- legacy systems can hardly evolve to provide new functionality required by the organisation.

Data held in these legacy systems is an important business resource. It represents mission-critical business knowledge which cannot be easily replaced ([3], [6]). Following currently accepted methods for legacy system migration, in particular data migration, few efforts have been successful ([3], [4]). In view of this, many organisations are reluctant to migrate their legacy systems to newer technologies and now find themselves in a catch 22

[1] A legacy information system can be defined as "any information system that significantly resists modification and evolution", [3].

situation: mission critical legacy systems which are the life support system for the organisation are also a road block to progress.

Thus there is an urgent need to provide methodologies, techniques and tools not only for accessing the data which is locked in these closed systems, but also to provide a strategy which will allow the migration of the systems to new platforms and architectures. Up to now, the gateway approach[2] seems to have dominated the thinking in this area. Whilst gateways can provide interoperability between the migrating legacy system and its target system, they also give rise to some very difficult problems, such as maintaining the consistency of the data between the two systems.

This paper proposes that during the migration process it is not absolutely necessary for the legacy system and its target system to interoperate. A gateway-free approach to migrating legacy information systems is presented: the Butterfly Methodology. The discussion of this paper focuses particularly on the issue of legacy data migration. In the following section current approaches to legacy system migration are briefly reviewed. Section 3 outlines the Butterfly methodology. Section 4 presents the data migration engine of the Butterfly methodology in detail. Section 5 further analyses aspects of the methodology. The concluding section presents a summary of findings and discusses a number of future directions.

2. Brief review of migration research

Legacy system migration encompasses many research areas. A single migration project could, quite legitimately, address the areas of reverse engineering, business reengineering, schema mapping and translation, data transformation, application development, human computer-interaction and testing. Due to space limitations, only a brief outline of research directly related to legacy system migration, with particular focus given to data migration, will be presented.

Ganti and Brayman [5] propose general guidelines for migrating legacy systems to a distributed environment. Using these guidelines, the business is first examined and the business processes found are re-engineered as required. Legacy information systems are linked with these processes to determine which systems have data and business logic of value in the new target environment. A set of processes are selected and the associated legacy systems are analysed. Details about the data required for these processes are extracted and transaction databases constructed through which the associated data will be accessed by new applications. Mention is made of retaining logic encoded in applications but it appears that the legacy systems will be discarded and replaced with new applications. [5] recognises that legacy system migration should cause as little disruption to the current business environment as possible. However, it is unclear how the cut-over to the new, separately developed, target system will be handled. Thus, these guidelines are not really suitable for use in migrating a mission-critical legacy system.

The *Database First* (Forward Migration) method [1] involves the initial migration of legacy data to a modern, probably relational, Database Management System and then incrementally migrating the legacy applications and interfaces. While legacy applications and interfaces are being redeveloped, the legacy system interoperates with its target system through a Forward Gateway. This enables the legacy applications to access the database

[2] A Gateway is "A software module introduced between operation software components to mediate between them", [3].

environment in the target side of the migration process. This gateway translates and redirects these calls forward to the new database service. Results returned by the new database service are similarly translated for use by legacy applications.

Using the *Database Last* (Reverse Migration) Method [1] legacy applications are gradually migrated to the target platform while the legacy database remains on the original platform. The legacy database migration is the final step of the migration process. A Reverse Gateway enables target applications to access the legacy data management environment. It is employed to convert calls from the newly created applications and redirect them to the legacy database service.

The Reverse Gateway will be responsible for mapping the target database schema to the legacy database. This mapping can be complex and slow which will affect the new applications. Also many of the complex features found in relational databases (integrity, consistency constraints, triggers etc.), may not be found in the archaic legacy database, and hence cannot be exploited by the new application. For both the Forward and Reverse migration methods, the migration of the legacy data may take a significant amount of time during which the legacy system will be inaccessible. When dealing with mission critical information systems this may be unacceptable.

In their *Chicken Little* methodology Brodie and Stonebraker ([2], [3]) propose an 11 step generic migration strategy employing complex gateways, shown in Figure 1. In this method, the legacy and target information systems operate in parallel throughout the migration. Initially the target information system is very small, perhaps only one application with a very small target database. However as the migration progresses the target system will grow in size. Eventually the target system performs all the functionality of the legacy system which can then be retired. During migration, the operational mission-critical information system will be some composite of the target and legacy information systems using gateways to provide the necessary interoperability.

Step 1 : Incrementally analyse the legacy information system
Step 2 : Incrementally decompose the legacy information system structure
Step 3 : Incrementally design the target interfaces
Step 4 : Incrementally design the target applications
Step 5 : Incrementally design the target database
Step 6 : Incrementally install the target environment
Step 7 : Incrementally create and install the necessary gateways
Step 8 : Incrementally migrate the legacy databases
Step 9 : Incrementally migrate the legacy applications
Step 10 : Incrementally migrate the legacy interfaces
Step 11 : Incrementally cut over to the target information

Figure 1 Chicken Little Migration Approach

Using *Chicken Little*, data is stored in both the migrating legacy and the growing target systems. In most cases, gateway co-ordinators [3] have to be introduced to maintain data consistency. However, "Update consistency across heterogeneous information systems is a much more complex technical problem with no general solution yet advised, and it is still an open research challenge" [3]. Thus it seems that to apply *Chicken Little* approach would be a big challenge to any migration engineer. Besides, *Chicken Little* does not provide any guidelines on the testing aspects on migration, a necessity for any migration approach.

In summary, the few migration methodologies available are either so general that they omit many of the specifics or are too complex to be applied in practice. Brodie and Stonebraker's *Chicken Little* methodology offers the most mature approach. However, the need for the legacy and target systems to interoperate via gateways during the migration

process adds greatly to the complexity of an already complex process and is also a considerable technical challenge. Thus a need exists for a simple, safe, gateway-free approach to legacy system migration.

3. The Butterfly methodology

The Butterfly methodology is being developed as part of the MILESTONE project, a collaborative project involving Trinity College Dublin, Broadcom Éireann Research, Telecom Éireann, and Ericssons. MILESTONE aims to provide a migration methodology and a generic supporting tool-kit for the methodology to aid migration engineers in the process of migrating legacy information systems to target systems. The project began in July, 1996 and will finish in June, 1998. A trial legacy system migration following Butterfly methodology is currently being planed and results will be available in the future.

The objective of Butterfly methodology is to migrate a mission-critical legacy system to a target system in a simple, fast and safe way. The methodology eliminates, during the migration, the need to simultaneously access both the legacy and target systems, and therefore, avoids the complexity of maintaining the consistency between these two (heterogeneous) information systems.

It is very important to bear in mind that, using Butterfly methodology, the target system will not be in production while the legacy system is being migrated. The legacy system will remain in full production during the whole migration process. There will never be a case where live data is stored, at the same time, in both the legacy and target systems.

Butterfly methodology divides legacy system migration into six major phases, Figure 2. Due to the space limitation, this paper only discuss the data migration phase : Phase 4. For the discussion of Butterfly methodology as whole, please refer to [7].

Phase 0 : Prepare for migration.
Phase 1: Understand the semantics of the legacy system and develop the target data schema(s).
Phase 2: Build up a Sample Datastore, based upon the Target SampleData, in the target system
Phase 3: Migrate all the components (except for data) of the legacy system to the target architecture.
Phase 4: Gradually migrate the legacy data into the target system and train users in target system
Phase 5: Cut-over to the completed target system

Figure 2 Six phases of the Butterfly methodology

Before explaining the detail of Butterfly data migration, it is worth emphasising two points. First, Butterfly methodology deliberately stores live data at the legacy system side during migration, and the target system will not be in production before the full migration process finishes. This is different from gateway-based migration approaches where live data is distributed at both legacy and target systems during migration. This presents a great technical challenge to maintain data consistency, for which no general solution is available currently. Second, Butterfly methodology proposes a legacy data migration engine, suitable for mission-critical system migration, so that the legacy system need only be shut down for a minimal amount of time. This differs from the so called *Big-Bang, Forward and Reverse Migration* [1] approaches where the legacy system must be shut down for a considerable time to facilitate data migration before the target system is made available.

Butterfly methodology deals with legacy system understanding and target system development in phases 2 and 3. Target system development is supported by a sample-

datastore, derived from legacy data and mapped to target side. Once phases 0, 1, 2 and 3 have finished, Phase 4 : data migration can then start. *Only after all data in the legacy datastore and TempStores has been transformed to the target system, will the target system be in production.*

4. The data migration engine

4.1 Principles for data migration using the Butterfly methodology

Brodie and Stonebraker point out that "the fundamental value of a legacy IS is buried in its data, not in its functions or code" [3]. On one hand, MILESTONE agrees that the data of a legacy system is logically the most important part of the system. However, on the other hand, MILESTONE also realises that from the viewpoint of the target system development it is not the ever-changing legacy data that is crucial, but rather its semantics or schema.

Once data migration commences, the legacy data store is "frozen" to be read-only. Manipulations of legacy data are redirected by the *Data-Access-Allocator* (DAA), and the results stored to a series of auxiliary datastores named TempStore(s) (TS). When legacy applications access data, DAA retrieves data from the correct source, e.g. the legacy data or the correct TempStore.

To be more specific, Butterfly methodology does not use gateways and introduces several new concepts for the data migration:

- Sample DataStore, Legacy SampleData and Target SampleData;
- TempStore;
- Data-Access-Allocator;
- Data-Transformer;
- Termination-Condition and Threshold Value.

A *Sample DataStore* stores the *Target SampleData* based upon the target system data model. The *Target SampleData* is transformed from the *Legacy SampleData*, which is a representative subset of the data in the legacy data store[3]. The *Sample DataStore* is employed to support the initial development and testing of all components (except for data) of the target system.

In order to migrate the legacy data without the need for synchronised manipulations of the target and legacy systems, Butterfly methodology employs a series of auxiliary datastores: *TempStores* (*TS*). These TempStores record the results of manipulations on the legacy data during the course of the migration. Once the migration of legacy data commences, all manipulations on the legacy data are redirected by the *Data-Access-Allocator* (*DAA*). The *DAA* effectively stores the results of manipulations in the latest TempStore and retrieves required data from the correct TempStore (or TempStores in case of some derived data).

Butterfly methodology employs a *Data-Transformer*, named *Chrysaliser*, to migrate, in turn, the data in the legacy system as well as in the TempStores to the target system. The *Chrysaliser* is responsible for transforming the data from its legacy format to the data model format of the target system. This transformation will take the form of a set of rules to be applied to the data. The rules will be derived from the earlier process of determining a target schema which is conceptually equivalent to that of the legacy data store.

---

[3] System experts are responsible for ensuring the representativeness of the sample.

Butterfly methodology also introduces a *Termination-Condition*, and a *Threshold Value* (represented by ε), to determine if the migration has reached the final stage for cutting-over to the target system. ε is a pre-determined[4] threshold value representing the allowable amount of data in the final TS. If size(TS) $\leq \varepsilon$, it indicates that the amount of time necessary to migrate the data is sufficiently small to allow the legacy system to be brought down without causing any serious inconvenience to the core business. The Termination-Condition is met when TS_n has been fully transformed and, at the same time, there exists size(TS_{n+1}) $\leq \varepsilon$ ($n \geq 0$).

4.2 Data migration

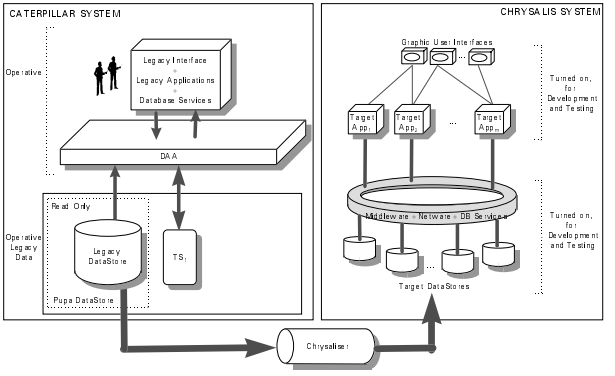

Figure 3 Migrating Data in the Legacy DataStore (TS_0)

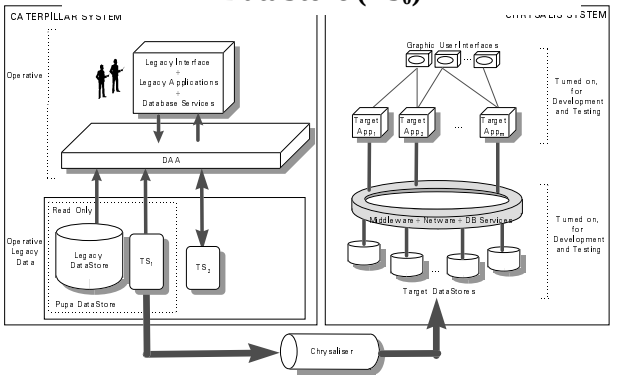

Figure 4 Migrating Data in TempStore TS_1

Gradually migrate the legacy data into the target system by employing the services of a series of TempStores, a fast Data Transformer (termed Chrysaliser), and a Data-Access-Allocator (DAA) as follows, (see Figures 3, 4 5 and 6).

Step 4.1 Create TempStore TS_1; then set legacy datastore (TS_0) to read-only, (Figure 3).

Step 4.2 Migrate all the data in TS_0 into the target datastores through the *Chrysaliser*. While this migration is taking place all access to the legacy data store is redirected by the *DAA* and all the results of manipulations are stored into the new TempStore, TS_1. Continue until TS_0 has been fully migrated, (Figure 3).

Step 4.3 Create TempStore TS_2; then set TS_1 to read-only, (Figure 4).

[4] Most likely by the Administrator of the Legacy Systems.

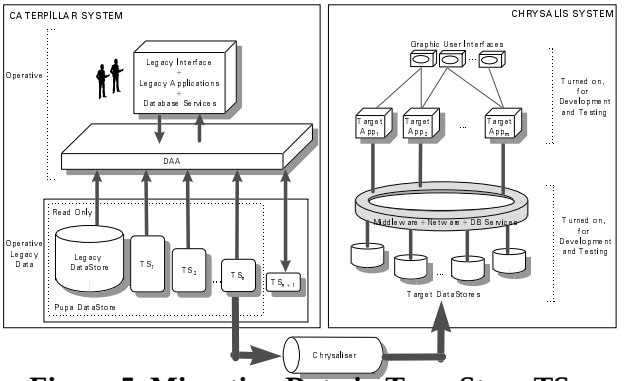

Figure 5 Migrating Data in TempStore TS_n

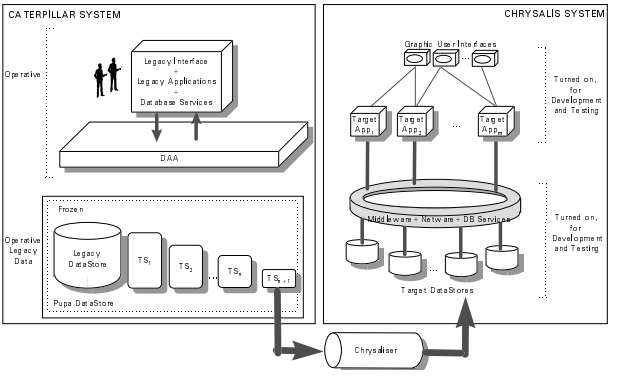

Figure 6 Migrating the last TempStore TS_{n+1}

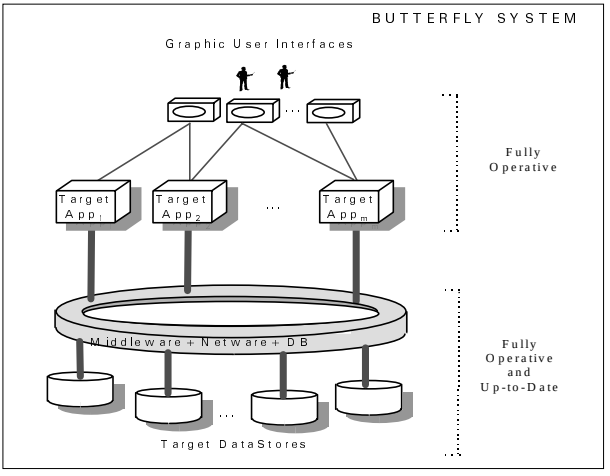

Figure 7 The Fully Functional Target System

Step 4.4 Migrate all the data in TS_1 into the target datastore(s) through *Chrysaliser*. As before all access to the legacy data is redirected by the *DAA* and all manipulation results are stored in the new TempStore TS_2. Continue until TS_1 has been fully migrated, (Figure 4).

Step 4.5 Repeat step 4.3 and 4.4 for TS_{n+1} and TS_n until the *Termination-Condition* is met, i.e. TS_n has been fully transformed and, at the same time, there exists size(TS_{n+1}) $\leq \varepsilon$. (Figure 5)

Step 4.6 Freeze the entire legacy system and migrate all the data in TS_{n+1} into target datastore(s) through the *Chrysaliser*, (see Figure 6).

Once step 4.6 has been finished, it is the time to cut over to the completed target system. (see Figure 7)

5. Analysis

It is possible to make a number of observations with regard to the Butterfly methodology. This section will illustrate the termination and completeness of Butterfly methodology, and also give some performance estimate when applying this method.

5.1 Termination of data migration

Given a reasonably big Threshold Value ε, the migration process (on data) of Butterfly methodology will terminate if the speed of transforming TempStore TS_i using Chrysaliser is greater than the speed of creating TempStore TS_{i+1} using Data-Access-Allocator ($i \geq 0$).

Suppose that the initial size of legacy data, size(TS_0), is X_0; the speed of transforming legacy data into target datastore(s) is **u**; the speed of creating TempStore TS using *Data-Access-Allocator* is **v**. Then, by the given precondition, it holds:

$$\frac{v}{u} < 1 \qquad (1)$$

At any given moment, for any TempStore TS_i, it can easily be seen that:

$$\text{size}(TS_i) = X_0 \left(\frac{v}{u}\right)^i \qquad i > 0 \qquad (2)$$

As $\frac{v}{u} < 1$, therefore:

$$\lim_{i \to \infty} \text{size}(TS_i) = 0 \qquad (3)$$

This obviously implies that there must exist a $n \geq 0$ such that for every $j \geq n$

$$\text{size}(TS_j) \leq \varepsilon \qquad n \leq j \qquad (4)$$

According to the steps 4.5, 4.6 and Phase 5 of Butterfly methodology and equation (4) it can be deduced that the migration process (on data) of Butterfly methodology will terminate.

Of course the speeds of **u** and **v** are not constant but an averaged value will suffice. It should be noted that the speed of **v** is a function of the transaction throughput which itself will go through periods of high and low activity, (office hours vs. night and weekend periods).

For any given migration it is possible to predict how many TempStores will be needed. From (2) it can be seen that for the final TempStore TS_n :

$$X_0 \left(\frac{v}{u}\right)^n \leq \varepsilon$$

So, *n* will be the first integer which satisfies:

$$n \geq \frac{\log X_o - \log \varepsilon}{\log u - \log v} \qquad (5)$$

By graphing the relationship between *n* and $\frac{u}{v}$ (see Figure 8) we can readily predict the number of TempStores needed for a given value of **ε** and X_0. In Figure 8, we have assumed the Threshold Value is 1 million (10^6) bytes and have graphed three scenarios, where the legacy data store (TS_0) is 1 billion (10^{12}) bytes, 10 billion ($10*10^{12}$) bytes, and 100 billion ($100*10^{12}$) bytes respectively.

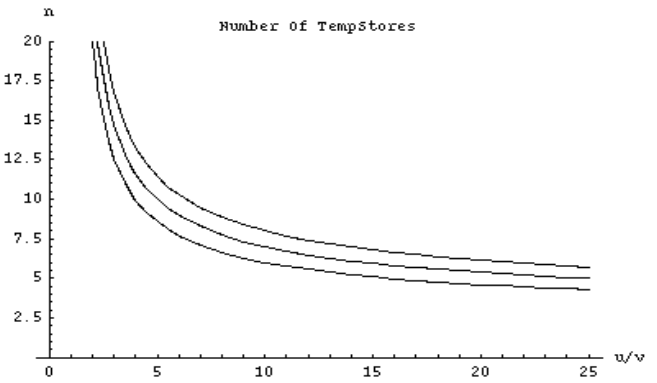

Figure 8 The Relationship Between *n* and $\frac{u}{v}$

It can be seen that when X_0 = 1 billion bytes and **u** is 5 times **v** it will take 9 TempStores for the data migration; when **u** is 10 times **v**, it will only take 6 TempStores, etc..

Following on from this we see that it is possible to predict how long it will take to perform a given legacy data migration. Once again from equation (2), we can see that the time needed to complete the migration of TempStore TS_i can be represented by:

$$t_i = \frac{X_0 \left(\frac{v}{u}\right)^i}{u} \qquad (6)$$

Therefore, the total time of the whole data migration can be estimated based on the speed of u and v before the migration begin.

5.2 The migrated data set

A final observation is with regard to the migrated data set of the Butterfly methodology. Any data migration methodology would have to ensure that following the migration the target data is equivalent to the initial legacy data[5].

By the time the Butterfly methodology process terminates, the following equation holds:

$$\text{Target Data} = \text{Legacy Data}^{6} \qquad (7)$$

It can be seen from the methodology in section 4.2 that at any moment n (≥ 1) within step 4; the following equations hold:

$$\text{Target Data} = \sum_{i=0}^{n-1} TS_i \qquad (8)$$

$$\text{Legacy Data} = \sum_{i=0}^{n} TS_i \qquad (9)$$

Therefore,

$$\text{Legacy Data} = \text{Target Data} + TS_n \qquad (10)$$

From step 4.6 and other phases of Butterfly methodology (Section 3), equation (10) implies equation (7).

6. Conclusions and Future Directions

In this paper a new approach to the problem of legacy system migration has been presented : the Butterfly methodology. The migration process as a whole is a very complex procedure encompassing many different fields of research. The focus of discussion of this paper was thus necessarily limited. The proposed Butterfly methodology applies to the whole process of legacy system migration with the main focus specifically on the migration of legacy data in a mission-critical environment. The Butterfly methodology is a simple, safe, and open approach to this problem. It represents a departure from current thinking on how legacy systems as a whole can be migrated to new architectures.

---

[5] MILESTONE is aware that 'dirty-data-cleaning' may be needed here, but it is another research issue and beyond the scope of this paper.

[6] MILESTONE is aware that the quality of the legacy data will have an impact on its migration. However, this is an issue for which no clear solution exists and should be addressed elsewhere. To the authors' knowledge, no existing migration methodology has addressed this issue. The completeness of Butterfly is illustrated by the fact that if all legacy data is required in the target, then it can be done.

The main difference between Butterfly and other proposed migration methodologies, is that Butterfly methodology is a gateway-free approach. It eliminates, during migration, the need for system users to simultaneously access both the legacy and target systems. There is therefore, no need to keep consistency between these two (heterogeneous) information systems as Butterfly methodology always stores live data at legacy system side! In practice, using gateway-based approaches, gateway co-ordinators [3] have to be introduced to maintain data consistency. However, as Brodie and Stonebraker point out "Update consistency across heterogeneous information systems is a much more complex technical problem with no general solution yet advised, and it is still an open research challenge" [3]. Thus to design and implement a gateway co-ordinator is a task without general methods or even guidelines. In contrast, generic mechanisms of Butterfly's data transforming engine : DAA and Chrysaliser for legacy (flat file, hierarchical) systems have been developed [8].

MILESTONE is an ongoing project, working with real life legacy systems in Telecom Éireann, the Irish national telecommunications service provider. Immediate future work for MILESTONE includes further investigating the framework tool-kit which supports Butterfly methodology. Future work also includes an effort to implement the *Chrysaliser* Data Transformer and the *DAA* Data-Access-Allocator subsystems for a migration process, and criteria and techniques to develop a Sample Datastore. MILESTONE is aware that many factors such as the structure of the TempStores and the placement of the Chrysaliser and DAA will affect the migration process and is investigating these issues. A number of subsequent practical experiments will provide results to illustrate the relationships between the u, v, X_0, and ε system variables. A trial migration applying Butterfly methodology to a legacy system is being planned, and results of it will be available in the future.

References

[1] Bateman A. and Murphy J. "Migration of legacy systems", School of Computer Applications, Dublin City University, working paper CA - 2894.

[2] Brodie M. and Stonebraker M., "DARWIN: On the Incremental Migration of Legacy Information Systems", Distributed Object Computing Group, Technical Report TR-0222-10-92-165, GTE Labs Inc., March 1993, http://info.gte.com/ftp/doc/tech-reports/tech-reports.html.

[3] Brodie M. and Stonebraker M., "Migrating Legacy Systems Gateways, Interfaces and the Incremental Approach", Morgan Kaufmann, 1995.

[4] Fingar P., and Stikleather J., "Distributed Objects For Business", SunWorld Online, April 1996.

[5] Ganti N., Brayman W., "Transition of Legacy Systems to a Distributed Architecture", John Wiley & Sons Inc. 1995.

[6] I. Sommerville, "Software Engineering Environments", 5th Ed., Addison-Wesley 1995.

[7] B. Wu, D. Lawless, J. Bisbal, J. Grimson, Vincent Wade, R. Richardson and D. O'Sullivan , 'The Butterfly Methodology: A Gateway-free Approach for Migrating Legacy Information Systems", in Proceedings of the 3rd IEEE Conference on Engineering of Complex Computer Systems (ICECCS '97), Villa Olmo, Como, Italy.

[8] B. Wu, D. Lawless, J. Bisbal, J. Grimson, Vincent Wade, R. Richardson and D. O'Sullivan, 'The mechanisms of DAA and Chrysaliser for Butterfly methodology', Technical report, Department of Computer Science, Trinity College, Dublin.