Proceedings of the 3rd IEEE Conference on Engineering of Complex Computer Systems (ICECCS97), Villa Olmo, Como, Italy. September 8-12, 1997. pp. 200-205, IEEE Computer Society.

The Butterfly Methodology : A Gateway-free Approach for Migrating Legacy Information Systems

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

Abstract

The problems posed by mission-critical legacy systems - e.g., brittleness, inflexibility, isolation, non-extensibility, lack of openness - are well known, but practical solutions have been slow to emerge. Generally, organisations attempt to keep their legacy systems operational, while developing mechanisms which allow the legacy systems to interoperate with new, modern systems which provide additional functionality. The most mature approach employs gateways to provide this interoperability. However, gateways introduce considerable complexity in their attempt to maintain consistency between the legacy and target systems. This paper presents an innovative gateway-free approach to migrating legacy information systems in a mission-critical environment : the Butterfly Methodology. The fundamental premise of this methodology is to question the need for the parallel operation of the legacy and target systems during migration.

1. Introduction

Many organisations, although solvent and exhibiting future promise, must overcome serious challenges if they are to remain competitive in today's fast changing business and technological environments [4]. These organisations would have been among the first to adopt computer based information systems and leverage the related benefits to become industry leaders. These same information systems, referred to as legacy systems, are now a roadblock to progress. The problems these massively complex systems pose include :

- they cannot evolve to provide new functionality required for their host organisation to remain competitive;
- they run on obsolete hardware which is expensive to maintain and reduces productivity due to its low speed;
- maintenance is expensive, tracing failures is costly and time consuming due to a lack of documentation and a general lack of understanding of the internal workings of the systems;
- integration efforts are greatly hampered by the absence of clean interfaces.

Unfortunately, the replacement of legacy information systems is a far from straightforward process. Switching off a thirty year old system and plugging in a new feature rich replacement overnight is not an option. The effort involved in discovering exactly what a legacy system does, never mind how it does it, may take many man months. The development of the replacement system is perhaps the most straightforward task once the required functionality is clearly defined. However, the cut-over to the target system should cause as little disruption as possible to the current business environment. An extensive testing program must then be undertaken to ensure there are no inconsistencies between the output of the legacy system and its replacement.

The problems posed by legacy systems have to be solved. A partial solution, such as wrapping legacy applications and leveraging the legacy data onto the desktop through some form of federated database system or data warehouse, is not enough. Therefore research is currently being conducted into developing a safe and cost-efficient way to guide legacy system migration as a whole ([2], [5], [6], [8]).

In this paper a new approach to migrating legacy information systems is presented: the Butterfly Methodology. In the following section, current approaches to legacy system migration are briefly reviewed. The Butterfly Methodology is presented in Section 3. Section 4 briefly discusses the properties of the methodology. The concluding section presents a summary of findings and discusses a number of future directions.

2. Current research on legacy migration

Although legacy information system migration is a major research issue, there are only a limited number of general migration methods available. Tilley [8] discusses legacy system reengineering from several perspectives: engineering, system, software, management, evolution and maintenance. A framework for legacy system reengineering is proposed for each perspective. Using the system reengineering framework, the implication is that the legacy system will operate normally while the target system is developed independently. When the target system is complete, the legacy system will be shut down and the target system switched on. However, the proposed frameworks are presented at too high a level to be applied in practice and no consideration is given to the migration of legacy data.

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 is selected and the associated legacy systems are analysed. New applications are then developed to fit these processes. These guidelines recognise 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.

In their *Chicken Little* Methodology Brodie and Stonebraker ([1], [2]) propose an 11 step generic migration strategy employing complex gateways. In this method the legacy and target information systems are operated in parallel throughout the migration. Initially the target information system is very small, perhaps only one application with a very small database. However as the migration progresses the target system will grow in size until it performs all the functionality of the legacy system which can then be retired. During the migration, the legacy and target information systems interoperate to form the operational mission-critical information system. This interoperability is provided by a module known, in general, as a *gateway*, "a software module introduced between operation software components to mediate between them" [2].

An example of *Chicken Little's* general migration architecture is shown in Figure 1 (modified from [2]). Data is stored in both the migrating legacy and the growing target systems. A *forward gateway* is employed to enable the legacy applications access the database environment in the target side of the migration process and a *reverse gateway* is employed to enable target applications to access the legacy data management environment.

In most cases, gateway *co-ordinators* have to be introduced to maintain data consistency. However, as Brodie and Stonebraker themselves recognise, maintaining update consistency across heterogeneous information systems represents a complex technical problem which has no general solution and is an open research challenge [2]. Thus it seems that to apply the *Chicken Little* approach would represent a major challenge to any migration engineer.

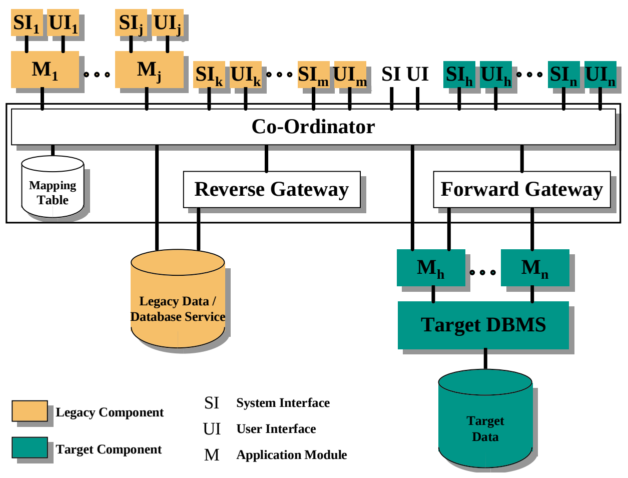

Figure 1 An Example of Chicken Little's General Migration Architecture

In summary, the few complete migration methodologies available are either so general that they omit many of the specifics or are too complex to be applied in practice. Little focus is given to legacy data migration in most methodologies. The *Chicken Little* Methodology offers the most mature approach. However, the need for the legacy and target systems to interoperate during the migration process via the gateways proposed adds greatly to the complexity of an already complex process and is also a considerable technical challenge in itself. This paper will explore a gateway-free approach to legacy system migration.

3. The Butterfly Methodology

The objective of the Butterfly Methodology is to guide the *migration* of a mission-critical legacy system to a target system. The methodology eliminates, *during the migration*, the need for system users to simultaneously access both the legacy and target systems, and therefore, to keep consistency between these two (heterogeneous) information systems.

3.1 Overview

The Butterfly Methodology is being developed as part of the MILESTONE project, an ongoing collaborative project involving Trinity College Dublin, Broadcom Éireann Research, Telecom Éireann, and Ericsson which started in July 1996. The Butterfly Methodology is based on the assumption that the data of a legacy system is logically the most important part of the system and 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(s). Thus, the Butterfly Methodology separates the target system development and data migration phases, thereby eliminating the need for gateways. To this end, several new concepts are introduced : *Legacy SampleData, Target SampleData* and *Sample DataStore; TempStore; Data-Access-Allocator; Data-Transformer; Termination-Condition* and *Threshold Value.*

Legacy SampleData is a representative subset of the data in the legacy data store. *Target SampleData* is transformed from the *Legacy SampleData*. A *Sample DataStore* stores the *Target SampleData* based upon the target system data model. The *Sample DataStore* is employed to support the initial development and testing of all target system components (except for data). Figure 2 illustrates the initial stage of migration using the Butterfly migration. Figure 3 shows a scenario during the legacy data migration.

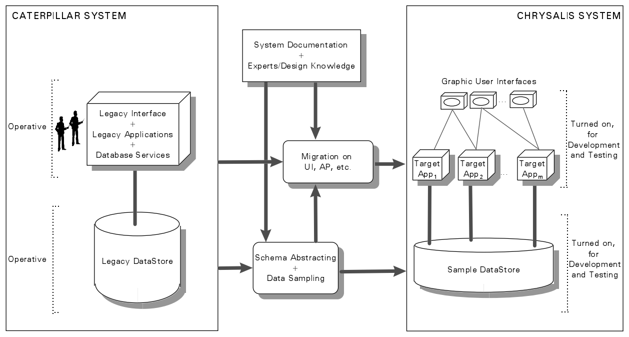
CATERPILLAR SYSTEM
CHRYSALIS SYSTEM
System Documentation + Experts/Design Knowledge
Operative
Legacy Interface + Legacy Applications + Database Services
Migration on UI, AP, etc.
Graphic User Interfaces
Target App1
Target App2
Target Appm
Turned on, for Development and Testing
Operative
Legacy DataStore
Schema Abstracting + Data Sampling
Sample DataStore
Turned on, for Development and Testing

Figure 2 Building Up the Target System

Using the Butterfly Methodology, when the legacy data migration begins, the legacy datastore is frozen to become a read-only store. All manipulations on the legacy data are redirected by the *Data-Access-Allocator (DAA).* The results of these manipulations are stored in a series of auxiliary datastores: *TempStores (TS).* 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). (See Figure 3.)

A *Data-Transformer*, named *Chrysaliser*, is employed to migrate the legacy data to the target system. *Chrysaliser* is responsible for transforming the data from its legacy format to the data model format of the target system. This transformation will depend on the target and legacy schemas.

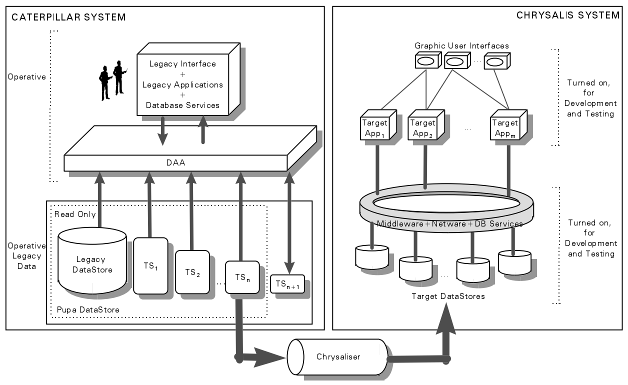

Figure 3 Migrating Data in TempStore TS_n

Chrysaliser first transforms all data in the frozen legacy data store (TS_0) to the target system. While this data is being migrated, the *DAA* will store results of manipulations on the legacy data to the first TempStore TS_1. When all data in the legacy data store has been migrated, TS_1 is frozen to be read-only. *Chrysaliser* will then begin to transform TS_1 to the target system. The *DAA* will now store results of manipulations on legacy data in a new TempStore, TS_2. When TS_1 has been successfully migrated, TS_2 will be frozen and migrated and so on.

When the size of the current TempStore is less than or equal to the *Threshold Value* (represented by ε), the amount of time needed to migrate the data in this TempStore is sufficiently small to allow the legacy system to be brought down without causing any serious inconvenience to the core business. The *Threshold Value* will probably be determined by the administrator of the legacy system. The *Termination-Condition* of the Butterfly Methodology is met when TempStore TS_n has been fully transformed and, at the same time, there exists a TempStore TS_{n+1} such that size(TS_{n+1}) $\leq \varepsilon$ ($n \geq 0$). Thus using the Butterfly Methodology, at no time during the migration process will the legacy system by inaccessible for a significant amount of time .

The DAA and Chrysaliser are essential elements of the Butterfly Methodology. The construction of DAA and Chrysaliser will differ depending upon the type of the legacy system migrating (e.g., file system, network database, or relational database). The placement of DAA and Chrysaliser is also a critical factor and affects the degree of complexity and the length of the whole migration process. Possible placement options include, for example, embedding them into a file system, implementing as an extended part of a transaction processing system within a DBMS, or above the file or DBMS systems as filters. The authors realise that this is a critical issue and research results from the MILESTONE project will be available in the future.

3.2 The Butterfly Methodology phases

In terms of the Butterfly Methodology, a legacy system migration can be divided into six major phases. Each phase consists of a number of individual, normally independent, migration activities. Within each phase, some activities are more crucial to the success of migration than others. This paper concentrates on data migration because this is the particular focus of the Butterfly Methodology.

- **Phase 0:** Prepare for migration.

Once the decision to migrate a legacy system has been made, the next stage is to prepare everything for the migration. Although many issues essential to a migration project have to be clarified at this stage, the Butterfly Methodology considers the user requirements for migration and target system determination to be most important. The main activities within this phase are listed in Figure 4.

Phase 0:
0.1 Get the migration preliminary requirements;
0.1.1 Determine user requirements;
0.1.2 Determine benchmarks for measurement of migration success;
0.2 Determine the target architecture;
0.3 Prepare the target hardware system;

Figure 4 Migration Activities in Phase 0

- **Phase 1**: Understand the semantics of the legacy system and develop the target data schema(s).

The activities identified within this phase are listed in Figure 5. Activity 1.5 to finalise the migration requirements is needed as it may not be possible to identify all the requirements until the legacy system has been understood.

Phase 1:
1.1 Understand the legacy interfaces, identify redundancies and determine the function of the target interfaces;
1.2 Understand the legacy applications, identify redundancies and determine the function of the target applications;
1.3 Understand the legacy data; identify redundancies and determine to-be-migrated data;
(optional) 1.4 Identify and understand interactions with other systems;
1.5 Finalise the migration requirements;
1.6 Develop the Data-Access-Allocator (DAA);
1.7 Develop the target data schemas and determine the mapping rules.

Figure 5 Migration Activities in Phase 1

A wide range of tools have been developed to assist in this reverse engineering area and it is likely that more will be developed in the future [7]. The Butterfly Methodology will take advantage of these tools and develop new tools only if it becomes absolutely necessary. One such tool is Data-Access-Allocator (DAA) developed by activity 1.6 which will be used to redirect all manipulations of legacy.

- **Phase 2**: Build up a Sample Datastore, based upon the Target SampleData, in the target system.

The main activities of this phase are to determine the legacy SampleData and to develop the Chrysaliser. Initially, the legacy SampleData will be transformed by Chrysaliser to form the target Sample DataStore. This will be used to develop and test the target system. Figure 6 lists the activities involved in this phase.

Phase 2:
2.1 Determine the Legacy SampleData;
2.2 Develop Chrysaliser;
2.3 Transform the Legacy SampleData into the Target SampleData and building the Sample DataStore;

Figure 6 Migration Activities in Phase 2

- **Phase 3**: Incrementally migrate all the components (except for data) of the legacy system to the target architecture.

"Forward" system engineering principles and methods will be one of the guidelines for migration in this phase. The Sample DataStore, built up in Phase 2, will be used to support the cycle of the 'design-develop-test' for newly developed target components. Figure 7 lists activities in this phase .

Phase 3:
3.1 Migrate legacy interfaces;
3.1.1 Migrate/develop a fragment of target interface;
3.1.2 Test against Sample DataStore for Correctness;
(optional) 3.1.3 Validate against User's requirements;
3.2 Migrate legacy applications;
3.2.1 Migrate/develop a target application;
3.2.2 Test against Sample DataStore for Correctness;
(optional) 3.2.3 Validate it against User's requirements;
3.3 Migrate reusable legacy components;
3.4 Integrate Target components/system;
3.5 Test Target components/system for Correctness;
3.6 Validate Target components/system against User's requirements;
(optional) 3.7 Train users on target components/system;

Figure 7 Migration Activities in Phase 3

- **Phase 4**: Gradually migrate the legacy data into the target system and train users in target system.

This phase is mainly devoted to legacy data migration and is the core part of the Butterfly Methodology. The legacy data will be gradually migrated into the target system by introducing a series of TempStores, the Data-Access-Allocator (DAA) and the data-transformer (Chrysaliser). The activities of this phase are listed in Figure 8.

A more detailed discussion and analysis of the data migration approach can be found in [9].

Phase 4:

4.1 Incorporate DAA into legacy system;
4.2 Create TempStore TS_1; set TS_0 to read-only;
4.3 Migrate all the data in TS_0 into the target datastores through the *Chrysaliser.* While this is taking place, all access to the legacy data store is redirected by the *DAA* and results of manipulations are stored in the new TempStore, TS_1. Continue until TS_0 has been fully migrated;
4.4 Create TempStore TS_2; then set TS_1 to read-only;
4.5 Migrate all the data in TS_1 into the target datastore(s) through the *Chrysaliser.* All access to the legacy data is redirected by the *DAA* and manipulation results are stored in the new TempStore TS_2. Continue until TS_1 has been fully migrated;
4.6 Repeat step 4.4 and 4.5 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}) $\le \varepsilon$;
4.7 Freeze the entire legacy system and migrate all the data in TS_{n+1} into target datastore(s) through the *Chrysaliser;*
4.8 Train users for the target system;

Figure 8 Migration Activities in Phase 4

- **Phase 5**: Cut-over to the completed target system.

The last phase of the Butterfly Methodology is the cut-over phase. Once the target system has been built up and all the legacy data have been migrated, the new system is then ready to run.

Phase 5:

5.1 Cut over to the completed target system.

Figure 9 Migration Activities in Phase 5

3.3 The workflow of the Butterfly Methodology

The Butterfly Methodology divides legacy system migration into six main phases. Each phase consists of a number of individual, normally independent, migration activities. Within each phase some activities can proceed in parallel with others. However, some activities will be inter-dependent (i.e. activities 2.1 and 2.3; 4.3, 4.4, 4.5 and 4.6). In general these six phases should be conducted in the sequence given which is based on the natural principles of system development. Normally the migration requirements for each legacy system will be unique to the system and therefore the migration workflow will also often be unique to the system. Hence, the Butterfly Methodology provides a flexibility for migration engineers. When desirable, activities can be conducted independently in an order different to the sequence given. Figure 10 shows a possible organisation of the Butterfly Methodology workflow.

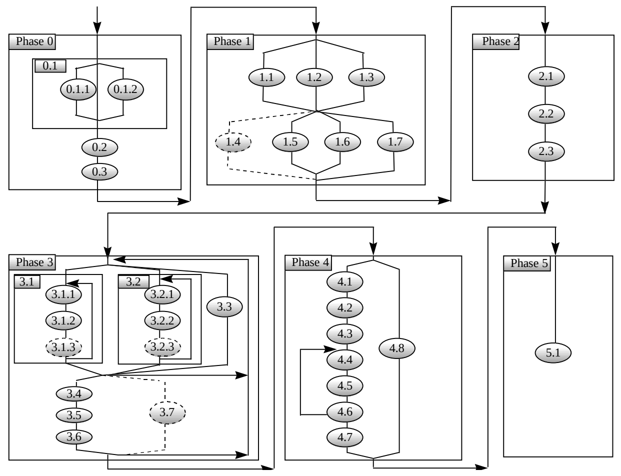

Figure 10 A Workflow of Butterfly Methodology

4. Discussion

The salient properties of the methodology can be summarised as follows:

- **clearly defined and strong support given to testing:** It has been documented in many case studies that up to 80% of time spent on a reengineering project can be made up of testing [3]. Each step of the Butterfly Methodology can be completely and successfully tested in practise. Target applications can be exhaustively tested against actual data held in the Sample Datastore.
- **flexible:** The methodology does not refer to any particular migration tools (except for DAA and Chrysaliser). Therefore, a choice of the most applicable tools can be made from the wide range currently available and these tools may be reused in different migration projects.
- **the total duration of the data migration can be estimated:** This is mainly due to the fact that the migration time of legacy data can be very clearly determined from the volume of the initial legacy data (X_0) in the legacy system together with the speeds of DAA and Chrysaliser. In the planning stages of any migration project this type of information is invaluable.
- **minimum intervention between the legacy system and the target system:** Only a single-way data transformation is needed for on-line operation, the rest can be done off-line.
- **minimum interruption to legacy system:** The legacy system will continue to operate as normal throughout the migration until the last TempStore has reached the pre-determined threshold value ε.

Consequently, the legacy system will never be inaccessible for a significant amount of time.

- **promotes parallel activities:** It can be seen that Phase 4 does not have to wait until Phase 3 has completely finished. The employment of a Sample DataStore allows planners to establish two distinct co-operative project teams, one to work on the target system development and the other to manage the migration of legacy data. The Sample DataStore also provides an excellent platform for training users on the target system before the migration is complete. This could greatly alleviate the difficulties typically associated with the cut-over stage.

From a pragmatic point of view, the main factor which will determine whether or not this methodology is usable, is the value of $\frac{v}{u}$ (here **u** is the speed of Chrysaliser transforming the data, and **v** is the speed of the DAA building up new TempStores). If **v** = 0, the methodology reverts to a *Cold Turkey* [2] migration where all the data is migrated in one massive operation. If **v** > **u**, then the migration process will never finish. Other factors relevant to the success of the methodology include:

- a thorough understanding of the legacy and target systems;
- an accurate and concise sample datastore;
- a fast chrysaliser;
- an efficient Data-Access-Allocator.

5. Conclusions and future directions

In this paper a new approach to the problem of legacy system migration has been presented. The migration process as a whole is a very complex procedure encompassing many different fields of research. The focus of discussion 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 offers a new, gateway-free approach to this problem. It represents a departure from current thinking on how legacy systems as a whole can be migrated to new architectures.

Immediate future work includes further investigating the framework tool-kit which supports the Butterfly Methodology to identify the necessary tools for each step of the method. In addition, the *Chrysaliser* Data Transformer and the *DAA* Data-Access-Allocator will be implemented, and criteria and techniques to produce a Sample Datastore will be developed. Many factors such as the structure of the TempStores and the placement of the Chrysaliser and DAA will affect the migration process and research investigating these issues is ongoing. A number of subsequent practical experiments will provide results to illustrate the relationships between the u, v, X_0, and ε system variables.

6. References

[1] M. Brodie and M. Stonebraker, 'DARWIN: On the Incremental Migration of Legacy Information Systems', Technical Report TR-022-10-92-165 GTE Labs Inc., http://info.gte.com/ftp/doc/tech-reports/tech-reports.html, March 1993.

[2] M. Brodie and M. Stonebraker, 'Migrating Legacy Systems: Gateways, Interfaces and the Incremental Approach', Morgan Kaufmann Publishers Inc. 1995.

[3] G. Dedene and J. De Vreese, 'Realities of Off-Shore Reengineering', IEEE Software, pp. 35-45, January 1995.

[4] P. Fingar and J. Stikeleather, 'Distributed Objects for Business: Getting started with next generation of computing', SunWorld Online, Vol. 10 No. 4, http://www.sun.com/sunworldonline/swol-04-1996/swol-04-oobook.html, April 1996.

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

[6] ESPRIT Project - Lancaster University, 'RENAISSANCE Project - Methods & Tools for the evolution and reengineering of legacy systems', http://www.comp.lancs.ac.uk/computing/research/cseg/ projects/renaissance, November 1996.

[7] J. Schmidt, ' A Practical Implementation with Migration Tools', Chapter 10 of 'Migrating Legacy Systems: Gateways, Interfaces and the Incremental Approach', By M. Brodie and M. Stonebraker, Morgan Kaufmann Publishers Inc. 1995.

[8] S. R. Tilley and D. B. Smith, 'Perspectives on Legacy System Reengineering', http://www.sei.cmu.edu/ ~reengineering/lsyree, November 1996.

[9] B. Wu, D. Lawless, J. Bisbal, J. Grimson, V. Wade, R. Richardson and D. O' Sullivan, 'Migrating Legacy Systems : From a Caterpillar to a Butterfly', Trinity College Technical Report, January 1997.