Application of Case Based Reasoning to Legacy System Migration

D. O' Sullivan and R. Richardson[†]
Broadcom Éireann Research Ltd.
Kestrel House, Clanwilliam Place
Dublin 2
Ireland

J. Grimson, B. Wu, J. Bisbal, and D. Lawless
Dept. of Computer Science
Trinity College
Dublin 2
Ireland

1. Introduction

Systems migration is now a major issue in established IT departments, [Bene95, Wins94]. Typically the information systems of these organisations are large, old, mission critical, and have minimal, if any, documentation. These information systems define what we today call **legacy information systems**. The problems these systems pose to their host organisations are numerous and important :

- the systems cannot evolve to provide new functionalities required by the organisation
- they run on obsolete hardware which is expensive to maintain and reduces productivity due to the low speed of the old hardware
- maintenance is 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

There is thus an urgent need to provide tools, methodologies and techniques not only for accessing the data which is locked in these closed systems, but also for providing a strategy which will allow the migration of the systems to new platforms and architectures. The exigency of this requirement is all the more highlighted by the pending "Year 2000 problem" which will render almost all legacy systems unusable.

Following currently accepted methods for legacy system migration, few efforts have been successful, [Fing96]. Current methodologies, [Brod95, Rena96, Gant95, Till95], are generally characterised by a series of complicated steps. The complexity of individual steps derives from their generality as opposed to any specific tasks[1]. This leaves migration engineers constantly asking the question, "But how do I do that for my situation ?". There is no facility to learn from previously successful solutions.

In this paper we propose that case based reasoning, CBR, can play a very constructive role in legacy system migration. In the following section we briefly describe CBR and it's proposed role in legacy system migration. Any migration project involves both the migration of the legacy applications and the legacy data. In section 3 we present a more detailed discussion of the application of CBR in the migration of legacy applications. In section 4 we investigate the possible applications of CBR in the legacy data migration process. In the final section we present our conclusions and briefly describe our intentions for future work in our MILESTONE[2] project.

[†] To whom correspondence should be addressed

[1] One step from Brodies *Chicken Little* strategy is "Incrementally migrate Legacy Applications", [Brod95].

[2] MILESTONE is a collaborative project involving Trinity College Dublin, Broadcom Éireann Research, Telecom Éireann, and Ericssons which aims to provide a migration methodology supported by a generic tool-kit to aid migration engineers in the process of migrating legacy information systems to open (and/or distributed) systems.

2. Case Based Reasoning for Legacy System Migration

Case based reasoning, [Wats94, Kolo93, and Aamo96], has enjoyed widespread popularity as an alternative to expert systems for solution of problems of a certain type. CBR is useful in situations where a well understood model for the solution of a particular problem is unavailable. In these situations experts may be able to suggest solutions based on previous experience but would find it difficult to explain why such solutions are suitable or to formalise a useful set of rules for the solution of these problems in general. In these situations it is often more useful to compare a new problem to previously encountered problems, select a previously successful solution to a similar problem and to modify it to fit the new problem. This new solution can then be added to the set of previously successful solutions for future use.

We assert that the problem of legacy system migration is one which would benefit from the application of a CBR approach. In particular we are focussing on those issues that are central to all migration projects; understanding the source code of legacy applications and understanding the structure of legacy data. As can be appreciated it is very difficult to arrive at a simple generic methodology or set of rules that should be followed for all cases. In practice migration engineers learn over time to recognise patterns or features within legacy components. These features have in the past led them to adopt, (perhaps following trial and error), a particular course of action which results in a satisfactory solution. A CBR system could provide a framework to support this learning by experience process. In the following two sections we investigate in more detail the application of CBR to program understanding and data structure understanding.

3. CBR and Program Understanding

Program understanding is still a major stumbling block in the migration process. In our work we are investigating the potential of employing an Ontology in the creation of useful case representations of legacy code fragments. Such an Ontology would be used to model domain specific knowledge about standard programming routines. For example a bubble sort might be modelled within the Ontology as a relationship between the concept of pointers, the concept of array traversal and the concept of pointer comparison, a Quicksort might be modelled as a relationship between the concept of recursion, the concept of array partitioning and the concept of comparisons. A standard tax calculation could be modelled as a relationship between the concept of a selection (of both an allowance and a tax rate), the concept of a subtraction (of the selected allowance), and the concept of a multiplication (based upon a selected rate) and the concepts of allowances and tax rates. A simple ontology which could be used to model this calculation is shown in Figure 1. In this diagram concepts are represented by boxes and relationships such as *is selected by, is multiplied by, is a result of* etc. are represented as arrows. Each of these concepts would have links to triggers which would appear as various coding constructions in various programming languages, for example in C the concept of

a selection could be triggered by:	*if, case*
a comparison could be triggered by:	*=, <, >*
a subtraction could be triggered by:	-
a pointer comparison could be triggered by:	**p < *q* or *A[p] < A[q]*
an array partitioning could be triggered by:	*p+((q-p)/2)*

and the domain specific concept of

an allowance could be triggered by the literal *3660* or *7563*
a tax rate could be triggered by the literal *.281* or *.481*[3]

We propose to represent each case in the case base as a collection of features, each feature being a concept from the ontology. When classifying a segment of code, the code will be parsed and its constructions mapped to concepts within the ontology, this code segment can then be classified as being of the type of programming routine modelled in the ontology with which it shares most features. Depending on the situation a standard template for such a routine may be retrieved or the most similar previously classified code segment may be retrieved. As the set of features used to represent each case are concepts used to characterise each of the standard programming routines modelled in the ontology we predict that this representation will afford the classification algorithm a powerful discrimination ability, placing new cases strongly into one of the categories of the standard programming routines. The likely drawback to this approach is that cases which do not conform to the programming routines modelled in the ontology will be classified incorrectly or not at all. One possible solution would be to periodically revise the ontology to include models for new programming routines. Such a revision would be prompted by the appearance of poorly classified cases which clearly do not conform to any of the models in the ontology.

These factors are directly related to the completeness of the ontology and the level of detail modelled within it. In related work on the use of ontologies for concept based information retrieval it has been suggested that the trade-off between modelling effort and classification ability for ontologies should be decided relative to the particular problem at hand. A useful indicator of the correct granularity of an ontology may be an empirical measure of its ability to solve the task for which it was created.

> "Firstly, what level of granularity should an ontology be at, in other words how detailed should it be for a given task application? Clearly no ontology can make all possible distinctions between a pair of concepts. The answer lies in the precise characteristics of the language engineering task which is to be solved and in the relationship of the taxonomy to that task.....The effect of this on retrieval performance can be determined empirically although this does not indicate the correct level of granularity in an absolute fashion", [OSul95].

Just as no ontology can make all possible distinctions between a pair of concepts, neither can any ontology hope to capture the universe of all possible concepts. The completeness of an ontology should also be determined relative to its ability to perform the task for which it was created. The task for which our ontology of programming routines is intended may vary over time (as a result of new migration efforts on unforseen data), therefore the only feasible method for ensuring the completeness and correct granularity of this ontology is to revise it whenever the performance of the system falls below a certain threshold.

In effect, our approach attempts to embody some program understanding capabilities within a case representation. There have been a number of other approaches to program understanding such as constraint satisfaction [Wood96], however we are unaware of any attempts to combine Case Based Reasoning and program understanding in order to facilitate the sophisticated classification of program segments while building on previously classified examples.

[3] In Ireland 48.1% and 28.1% are standard tax rates and IRL3660 and IRL7563 are standard tax free allowances

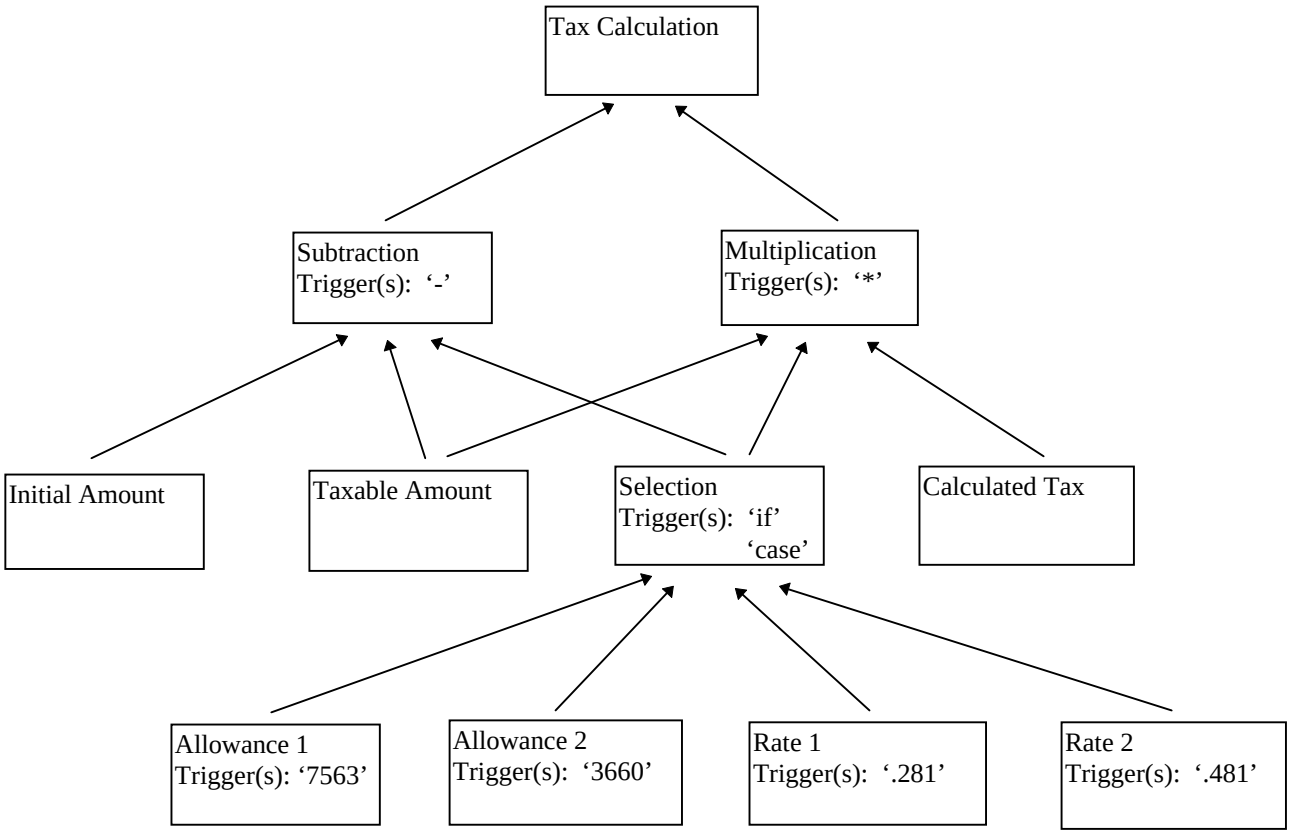

Fig 1 : A simple Ontology representing a Tax Calculation Routine

4. CBR in Database Reverse Engineering

The process of reverse engineering legacy databases involves designing a target schema which is conceptually equivalent to the legacy schema. This has proven to be a very difficult process to model. For the purposes of this discussion we will divide the process into the following components :

- Extracting the data structure from the legacy system
- Conceptualising this physical/logical schema to an equivalent conceptual schema
- Forward engineering from the conceptual schema to a new physical schema

Each of these steps can benefit from the use of case based reasoning.

4.1 Data Structure Extraction

The degree of difficulty involved in the data structure extraction process depends greatly on the data model of the legacy system. If a database management system is used, be it relational, network, or hierarchical, the process can be as simple as analysing the Data Definition Language, DDL, and examining the data dictionary. Unfortunately the situation is much more complex for the more prevalent standard file format, for which no computerised descriptions of structure exists. Individual source programs must be examined in order to detect partial structures of the files. Very often data structures are hidden (refer to Fig 2), optimisation constructs are introduced (e.g. padding for address alignment or record splitting when the size is greater than the page size), and specifications are left to be procedurally encoded.

Initial Record Structure :

```
01 Employee
    02 Key          pic X(19)
    02 Name         pic X(25)
    02 Age          pic 9(10)
    02 Sex          pic X
    02 Department   pic X(15)
```

Coded Record Structure :

```
01 Employee
    02 EmpKey       pic X(19)
    02 Filler       pic X(51)
```

Fig 2 : Example of Structure Hiding

It can, as a result, be very difficult to recover the original data structure. However, by examining source code statements, (e.g. the variables used to fill the "filler" variable in Fig 2), the structure can be inferred. Any padding can be discovered by knowing how large a page is and examing what is placed in the "filler" variables.

Besides *structure hiding* the code can also give clues to constraints in the legacy data structure which should be extracted and enforced in the target schema. Referential integrity, cardinality of relationships, uniqueness constraints and secondary identifiers are just some such constraints that are left to be enforced in a procedural manner. The structure of these procedures are generally both simple and standard, and can thus be easily recognised, refer to fig 3.

Over time patterns of legacy code and system specific variables can be identified as relating to the underlying data structure. By recording these features, possibly in an Ontology as described in section 3, or simply as they are, individual cases can be built up over time. These cases can then be used by the migration engineer in the future to alleviate the complexity of the extraction process.

```
01 Supplier
    02 Name             pic X(20)
    02 Address      pic X(25)
```

```
01 Part
        02 Part_No      pic X(20)
        02 Color                pic X(5)
        02 Sup          pic X(20)


Procedure Division              (pseudo)

        read Supplier(Name = X) into Supplier
        if found(S) then
                Part.Part_No := ...
                Part.Color := ...
                Part.Sup := Supplier.Name
        endif
```

Fig 3 : Example of Referential Integrity in Source Code

4.2 Conceptualisation of the extracted data structure

The extracted data structure is typically still in a form that is a poor semantic reflection of its original conceptual schema. In this phase a number of transformations are performed to rediscover the full semantics of the legacy database. The resulting conceptual schema is typically expressed in an Entity Relationship model. A transformation is grossly speaking the process of replacing a data structure with another one which is semantically equivalent to the former. Examples of transformations include the replacement of one-to-many relationship types by reference attributes, the introduction/replacement of multivalue or compound attributes, the transformation of entities into attributes, etc.. Transformations can be carried out for one of two reasons. Firstly to remove constructs that were designed for optimisation purposes, e.g. merge/split records, denormalisation, derived variables, etc.., refer to fig 4, and secondly, to arrive at the most expressive, simple, and readable conceptual schema, refer to fig 5.

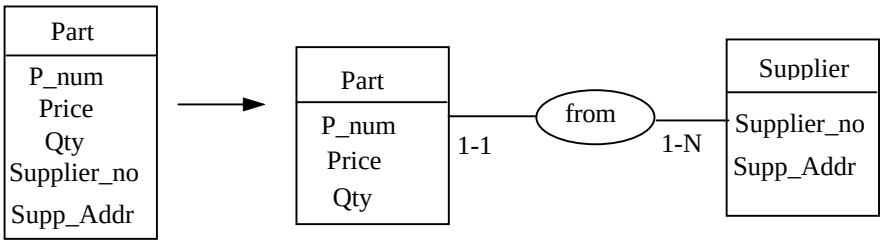

Fig 4 : Denormalisation

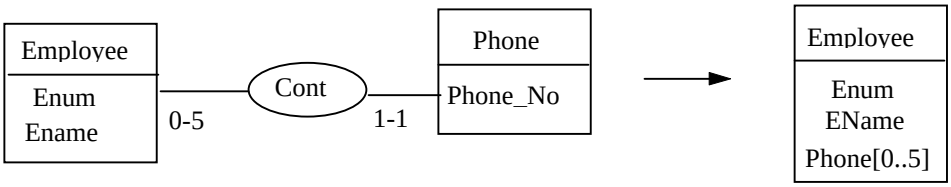

Fig 5 - Simplify by introducing of multivalued attributes

The biggest problem in the conceptualisation process is the fact there is no 1-1 mapping between conceptual structures and their physical/logical compliant translation. A conceptual construct can be translated into several physical/logical structures, while several different conceptual constructs can be translated into the same physical/logical structure. We believe a CBR system could be employed to decide which transformations to apply and in what order to apply them. Features of a schema are simple to illicit (eg a transitive functional dependency), and can thus be readily compared to pre-recorded features of previous cases. The entire process can be made considerably easier by CASE tools such as DB-MAIN, [Hain96], which graphically illustrate schemas and automatically apply any number of transformations to a schema.

4.3 Forward engineering from the conceptual schema to a new physical schema

This step is the reverse of the conceptualisation process, i.e. employing reversible transformations to derive a physical schema, (generally in the Object Oriented data model but possibly the Relational model), which is semantically equivalent to the conceptual schema. There is once again an excellent opportunity to use a CBR system to aid the migration engineer in choosing which transformations to apply and in what order to apply them.

5. Conclusions

In this paper we have highlighted the crucial role CBR can play in the process of legacy system migration. The nature of the problem, i.e. the absence of a usable model, makes it particularly amenable to the solution by experience approach proposed by CBR. We have concentrated on the particularly difficult issues of legacy application understanding and the legacy data understanding. Other approaches to this problem to date have been so complex as to preclude their application in a practical generic solution.

MILESTONE is an ongoing project and the proposals introduced in this paper are the current focus of attention. The project is working with real life legacy systems in Telecom Éireann, the Irish national telecommunications service provider. Immediate future work includes the construction of domain dependent Ontologies and further investigations into possible case representations for legacy data structures. There are a number of ontologies available at present which may serve as starting points in the construction of domain specific ontologies. The Princeton Wordnet, [Mill90a, Mill90b], is an excellent general purpose concept ontology which may be used to relate programming specific concepts to real-world concepts at a high level. The Enterprise Ontology, [Usch95], is a collection of terms and definitions relevant to businesses which may provide a useful high-level framework for the organisation of programming routines specific to certain business domains. The Knowledge Systems Laboratory, [KSL94], within the Department of Computer Science at Stanford University provide a library of shareable ontologies available over the internet. Some of these ontologies such as ‘Abstract-Algebra’, ‘Basic-Matrix-Algebra’ and ‘Component-Assemblies’ could be exploited within a domain specific ontology of programming routines.

References

[Aamo96] : Aamodt A. and Plaza E., “Case-Based Reasoning: foundational Issues, Methodological Variations, and System Approaches”, Artifical Intelligence Communications, Vol. 7, No.1, 1996.

[Bene95] : Bennett K., “Legacy Systems: Coping with Success”, IEEE Software, January 1995, pp. 19 - 22.

[Brod95] : Brodie M. and Stonebraker M., “Migrating Legacy Systems Gateways, Interfaces and the Incremental Approach”, Morgan Kaufmann, 1995.

[Fing96] : Fingar P., and Stikleather J., “Distributed Objects For Business”, SunWorld Online, April 1996.

[Gant95] : Ganti N., Brayman W., “Transition of Legacy Systems to a Distributed Architecture”, John Wiley & Sons Inc. 1995.

[Hain96] : Hainaut J-L., Englebert V., Henrad J., Hick J-M., and Roland D., “Database Design Recovery”, Proc. of the 8th Conference on Advance Information Systems Engineering, CAiSE’96, Springer-Verlag, 463-480.

[Kolo93] : Kolodner J. L., “Case Based Reasoning”, Morgan Kaufmann.

[KSL94] : “Shareable Ontologies Library” at http://www-ksl.stanford.edu/knowledge-sharing/ontologies/README.html

[Mill90a] : George A. Miller, Richard Beckwith, Christiane Felbaum, Derek Gross, and Katherine Miller, "Introduction to WordNet : An On-line Lexical Database", International Journal of Lexicography, Vol. 3, No. 4, 1990, 235 - 244.

[Mill90b] : George A. Miller, "Nouns in WordNet : A Lexical Inheritance System", International Journal of Lexicography, Vol. 3, No. 4, 1990, 245 - 264.

[Osul95] : O’Sullivan, D., McElligott, A., Sutcliffe, R.F.E, ‘Augmenting the Princeton WordNet with a Domain Specific Ontology’ in Proceedings of the IJCAI ‘95 Workshop on Basic Ontological Issues in Knowledge Sharing 19-21 August, 1995, Montreal, Canada.

[Rena96] : “The RENAISSANCE Project”, http://www.comp.lancs.ac.uk/computing/research

[Till95] : Tilley S., “Perspectives on Legacy System Reengineering”, http://www.sei.cmu.edu/~reengineering/lsyree

[Usch95] : M. Uschold, M. King, S. Moralee and Y. Zorgios, “The Enterprise Ontology”, at http://www.aiai.ed.ac.uk/~entprise/enterprise/ontology.html, 1995.

[Wats94] : Watson I. and Marir, F., “Case-Based Reasoning: A Review”, The Knowledge Engineering Review, Vol. 9, No. 4, 1994.

[Wins94] : Winsberg P., “What about Legacy Systems”, Database Programming and Design, Vol. 7, No. 3, 1994.

[Wood96] : Wood S. and Yang Q., “The Program Understanding Problem: Analysis and a Heuristic Approach”, Second Working Conference on Software Engineering, March, 1996, 78-84.