Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2004 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2004

Software Stability: Recovering General Patterns of Business

Aseem Daga
Brunel University

Sergio de Cesare
Brunel University

Mark Lycett
Brunel University

Chris Partridge
Brunel University

Software Stability: Recovering General Patterns of Business

Aseem Daga
Brunel University
Department of Information Systems and Computing
Uxbridge, Middlsex, UK
UB8 3PH
Aseem.Daga@brunel.ac.uk

Sergio de Cesare
Brunel University
Department of Information Systems and Computing
Uxbridge, Middlsex, UK
UB8 3PH
Sergio.deCesare @brunel.ac.uk

Mark Lycett
Brunel University
Department of Information Systems and Computing
Uxbridge, Middlsex, UK
UB8 3PH
Mark.Lycett @brunel.ac.uk

Chris Partridge
Brunel University
Department of Information Systems and Computing
Uxbridge, Middlsex, UK
UB8 3PH
Chris.Partridge @brunel.ac.uk

ABSTRACT

With re-engineering of software systems becoming quite pronounced amongst organisations, a software stability approach is required to balance the seemingly contradictory goals of stability over the software lifecycle with the need for adaptability, extensibility and interoperability. This paper addresses the issue of how software stability can be achieved over time by outlining an approach to evolving General Business Patterns (GBPs) from the empirical content contained within legacy systems. GBPs are patterns of business objects that are (directionally) stable across contexts of use. The approach is rooted in developing patterns by extracting the *business* knowledge embedded in existing software systems. The process of developing this business knowledge is done via the careful use of ontology, which provides a way to reap the benefits of clear semantic expression. A worked example is presented to show how stability is achieved via a process of 'interpretation' and 'sophistication'. The outcome of the process demonstrates how the balance that stability seeks can be achieved.

Keywords

Content Sophistication, Legacy Transformation, Content re-engineering and Ontology-Based Approach.

INTRODUCTION

Change in the business environment, often in connection with changes in technology approaches, causes entire software programs to be re-engineered – often at significant cost to the organisation (Fayad and Altman, 2001). Typically, re-engineering is oriented toward meeting emerging adaptability, extensibility and interoperability requirements. The cycle of change can often be destructive in nature, however, as existing systems become more complex and increasingly difficult to understand. With this background, the notion of software stability has emerged as a means of identifying the enduring aspects of a system while acknowledging the need for adaptability and extensibility (Fayad and Wu, 2002). Unsurprisingly, given the relative youth of the stability concept, many questions exist in relation to its purpose, role, achievement and the like.

This paper addresses the issue of how software stability can be achieved over time by outlining an approach to evolving General Business Patterns (GBPs) – patterns of business objects that are (directionally) stable across contexts of use. The approach called 'Content Sophistication' (CS), is rooted in developing patterns from the *business* knowledge that is embedded in existing software systems – and is based upon initial work described in Partridge (1996). CS is an ontology-based approach that focuses on the extraction of business content from existing system(s) and improving it along several dimensions.

The paper begins by noting the importance of the stability concept in the context of existing software systems. Section 2 outlines the motivation for the approach to stability and describes the approach to the evolution of GBPs, called sophistication. Section 3 describes the concepts of interpretation and sophistication. Section 4 provides an illustration of the sophistication process, via a condensed worked example, and describes how the process of evolution works. Section 5 notes

the theoretical and practical implications of the research. Lastly, conclusions and future direction of the research are presented.

STABILITY AND THE IMPORTANCE OF EXISTING SYSTEMS

It is stated that stability, currently expressed via business objects and enduring themes, has the potential to reduce or eliminate the re-engineering cycles commonplace in software engineering projects (Fayad and Wu, 2002). There is significant cost and effort associated with re-engineering, and much of the literature in the area focuses on the problems that the re-engineering effort aims to overcome. These problems include observations that (a) many applications are not easily integrated and were built during a prior era's technology, (b) plans and documentation are either poor or non-existent and (c) the architecture is insufficiently flexible to meet the challenges of anticipated future change (O'Callaghan, 1999).

Many existing systems, however, are now business critical – they deliver significant business value via the embodiment of substantial corporate knowledge in the form of business objects, processes, rules, events and the like. Moreover, the corporate knowledge that they embody is proven in the corporate working context on a daily basis. Given that many such systems have been in operation for years, we propose that the roots of stability lie in systems that exist and that re-engineering efforts can be seen as a proactive means for mining aspects of stability. The difficulty with mining stability from existing systems is that we have not had the conceptual tools to undertake such work – it is this area that our effort addresses.

THE CONCEPT OF SOPHISTICATION

The danger with current re-engineering projects lies in their inability to capture the business knowledge expressed through business content in a technology independent fashion. Existing approaches unlock a system view of the business from one technology set, make some changes to aspects of content and/or behaviour, and subsequently lock the revised view into another technology set (Bisbal, Grimson, Lawless, Wu, 1999; Brodie and Stonebraker, 1993; Ganti and Brayman 1995; Tilley and Smith, 1995). The effort does not generally consider what the system refers to from a business perspective. This consideration provides a key point of departure for understanding and generating business content that is stable. This inability to provide a stable business view has meant that there is a clear need for a different perspective, which is dealt here through the 'Content Sophistication' (CS) approach. The CS approach aims to extract business content (in the form of type and individual level content) from systems and improve this content along several sophistication dimensions. The CS approach operates at a computationally independent level, producing business models that are independent of any type of technology or platform.

The CS approach builds upon the work of the REV-ENG™ methodology (Partridge, 1996), which provides a re-engineering perspective on how to use business objects and General Business Patterns (GBP) to re-engineer existing systems into business models. The CS approach has redefined the REV-ENG™ process in a more systematic manner with well-defined workflows, activities, roles and responsibilities. Business objects refer to the things that exist in the real world and as a result have meaning and usage to the business (e.g., post code, location). GBPs are patterns that define a set of related business objects (e.g., the pattern Geo-Political Region (GPR) expresses a relation between location and postal code amongst others). Another important aspect is the ontological underpinning of the CS approach. CS is based on the philosophical notion of ontology (Honderich, 1995), which is regarded as a study of a set of things whose existence can be acknowledged by a particular theory or system of thought. This can be explained through a database example. The database can be viewed as the "theory" that recognizes the underlying set of things that form part of the database – its ontology. The primary aim is to seek truth and develop theories that provide a description of real-world entities and the relations that exist between these entities. The CS approach uses ontology as the basis for understanding, modelling and sophisticating content that corresponds to objects that exist in the real-world or business. The ontological aspects of the CS approach works at two levels. At an initial level ontology provides a process for analysing and clearly deciphering the real-world semantics (business knowledge) that is hidden in the content. A clear understanding of semantics of the business content clarifies the applications' ontological assumptions - the identification of the entities and relations that the content describes in the real world. At another level, to provide a model perspective, the relevant entities and relations of the ontology are modelled within a conceptual framework (ontological model), which maps the things in the ontology to their corresponding representation in the model. As a result the conceptual framework directly reflects the ontology. As such ontology provides a common conceptual framework across applications for analysing the objects their content describe.

Stability is achieved by applying the ontological process of interpretation and sophistication. In outline terms, interpretation is defined as identifying the business objects that the system commits to existing. The interpretation process works its way through the system identifying both the explicit and implicit business content. This process ensures completeness, i.e., all business content represented at a system level is captured in the final business model. For interpretation (and subsequent

sophistication) it is essential that the business content is analysed at two levels: (1) individual level content (such as the 'United Kingdom') and (2) type level content (such as 'Countries'). Interpretation offers the opportunity to understand the semantics of the content and highlights the system's view of the world (Application's Ontology). This provides the premise for the sophistication of the business content.

Sophistication can be defined as the process of improving a business model - by removing the gap that exists between the system's view of the real world and the things as they exist in the real (business) world (Migrated Business Ontology). This gap is called the sophistication gap and by removing this gap, sophistication aims to provide better theories and a more precise representation of the real world. Sophistication thus provides the underpinnings for stability and improvements lie along the following dimensions:

- Explanatory power. The ability of the improved model can give increased meaning to the things and the relationships expressed.
- Fruitfulness. The degree to which the improved model can meet currently unspecified requirements or is easily extendable to do so.
- Generality. The degree by which the scope of the types in the improved model can be increased without the loss of information.
- Objectivity. The ability of the model to provide a more objective (shared) understanding of the world.
- Precision. The ability of the improved model to give a more precise picture of the business object: in particular, to index a thing to its mode of existence as opposed to its mode of representation and/or application.
- Simplicity. The degree by which the model can be made less complex.

Analysis of existing systems leads to the observation of sophistication gaps, categorised as follows. For each category, techniques and heuristics have been developed to liberate a model along the dimensions above.

- Generalisation. Redundancy analysis of things to produce a smaller collection that is more general but which typically contains more inherent information.
- Interpretation. Index-based analysis where information is treated as content rather than as a representation in order to tie that content back to what is represented (understanding the mode of existence).
- Destratification. Relationship-based analysis of things to better express the natural overlaps between them, removing the constraints of artificial hierarchies.
- Temporalisation. Spatio-temporal analysis of things to express natural temporal stages (relationships between things in time) in a systematic fashion.

THE PROCESS AND OUTCOMES OF SOPHISTICATION

The identification of sophistication gaps and action taken in respect of them leads to the development of business objects and patterns of business objects, called General Business Patterns (GBPs). Sophistication gaps and actions are best understood in the context of a small example. To highlight its practical significance, CS is explained through an example of Sophistication Instance (SI). SI is a CS artifact whose purpose is to enable the benefits of content sophistication to be easily presented to and understood by a business audience. The SI presented in this section is part of the re-engineering work undertaken to renovate a large financial system, henceforth referred to as 'Isure' and the company as 'Megabank'. Megabank is a large information technology solution and services company with interest in a number of vertical markets. Isure is a large financial system developed, maintained and managed by Megabank. The system originally written in state of the art technology around 20 years back is however rendered monolithic, based on obsolete technology and difficult to manage today. Client feedback, market directions and the long-term business objectives of the company have resulted in a business goal to migrate the system towards modern technical platforms. This has presented a situation to model the systems in a technology agnostic fashion, thus insulating the risk of being dependent on any specific technology. The SI described here concentrates on GPR-TME GBP, a combination of the GPR and Time GBP, as it is regarded as simple enough to be easily understandable, while also being rich enough to demonstrate the feasibility of the approach. The GPR-TME GBP has emerged from the process of applying CS to the existing system.

Once the decision is made to renovate the system, it is important to lay down the groundwork for CS. The target system is divided into manageable work units called segments. Figure 1 shows a typical segmentation of the Isure system. Isure can be divided into a number of segments or work units, deemed necessary when analysing the system. Each segment is further

composed of one or more fragments that display similar characteristics corresponding to the segment, e.g., the GPR segment consists of fragments CNTRY (Country Detail) and CNTGM (Country Group).

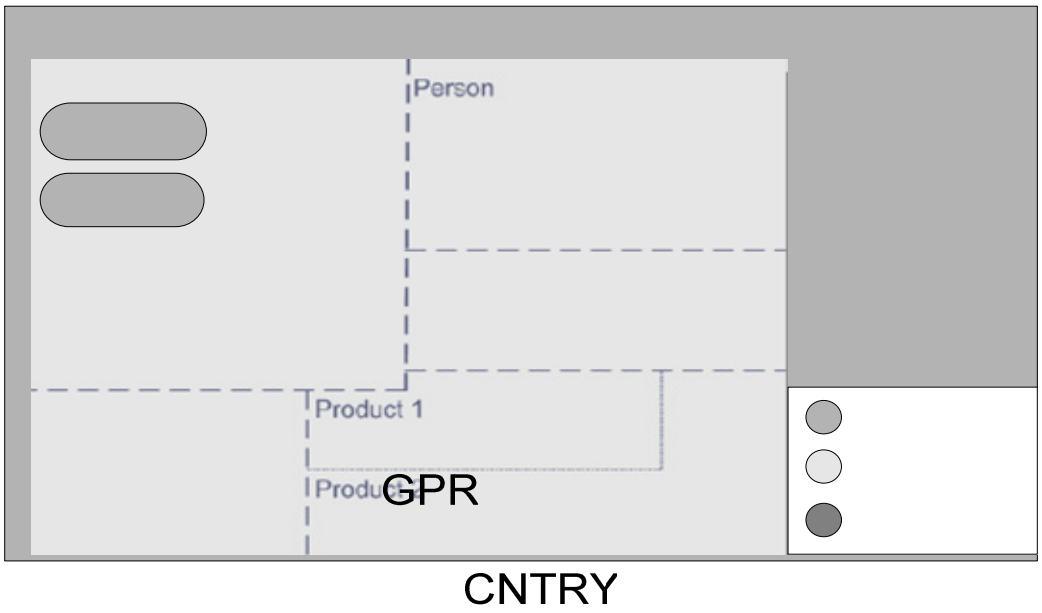

Figure 1. Typical System Segmentation

The part that this SI is focusing on is a lack of generalisation (one of the categories of lack of sophistication) in the fragment CYDRY - Country Standard Holiday. It also illustrates how the process can involve a number of types of sophistication. In this case, the central sophistication type is generalisation, but destratification is also involved. The SI part contains the fragment CYDRY, the fragment CNTRY - Country Table upon which CYDRY is dependent and CYDRY's dataitems Day-Month-s (Holiday Date).

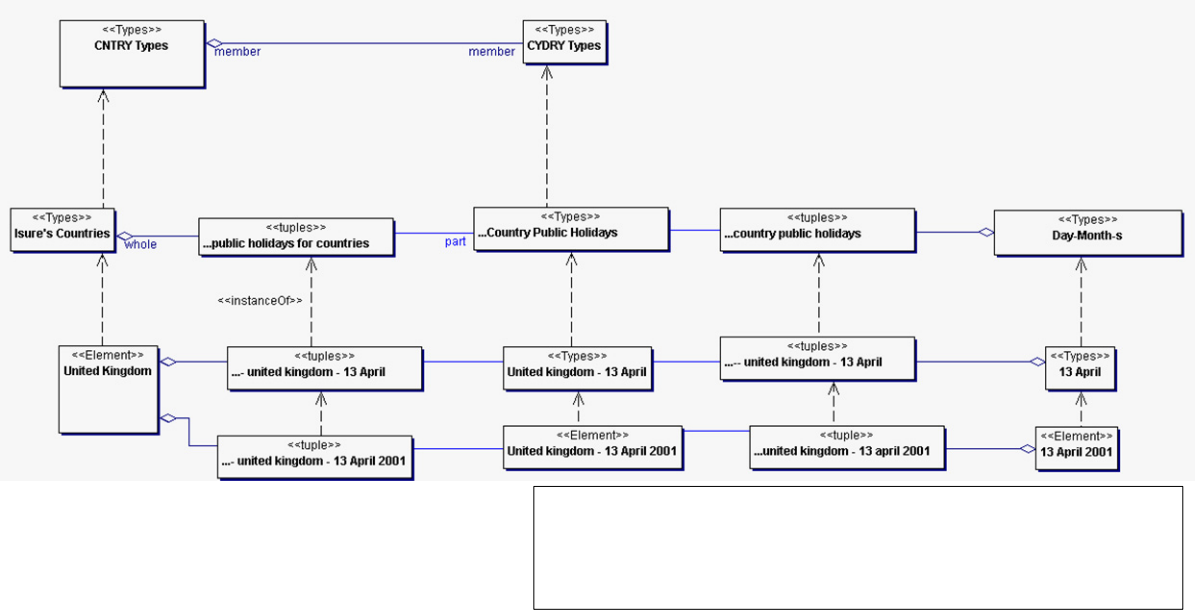

Figure 2. Isure System Fragment

Figure 2 provides an example interpretation of a fragment related to public holidays in the existing system. The CYDRY fragment records the days and months of all the public holidays for a particular country. Note that the types CNTRY (Country Detail) and CYDRY (Country Holiday) refer to interface specifications. The interpretation process has revealed a generalisation gap between the Isure Application Ontology and the Migrated Business Ontology. The interface CNTRY is implementation indexed, which means an implicit assumption is made while designing the system that the countries stored in the CNTRY Fragment are disjoint. In reality however, there are countries that are not disjoint, the United Arab Emirates (UAE) containing Abu Dhabi is an example or the United Kingdom (UK) containing Northern Ireland. This type of disjointness constraint is common in computer systems and is known as stratification. The disjointness constraint does not specify which countries are allowed to be set up in CNTRY, merely that whatever countries are set up must be disjoint. Where two countries overlap, the disjointness constraint allows either of the two to be set up, but not both. This can lead to different implementations having different incompatible CNTRY tables, even though the application level CNTRY description is identical. This situation is called implementation indexing – where the table is indexed to the implementation and even though the application level description is identical, different implementations will have different tables (that are likely incompatible). The interface CYDRY is dependent on CNTRY and thus inherits the index on implementation and the common business pattern (a kind of reflected stratification). As a result different implementations will have different country public holiday calendars, which cannot be combined. Another gap that has emerged as a result of the interpretation process is that the CYDRY's Day-Month-s (Holiday Date) attribute is constrained to representing only country holidays and also only days, which restricts the kind of Public Holiday that CYDRY can represent.

Types:
Tuples:
Elemer
Tuple: I

Given an understanding of the operational requirements of Isure, a number of competency questions (CQ) were asked of the system that it could not accommodate:

- Can public holidays for both nesting (*e.g. United Kingdom – 13 April 2001*) and nested countries (*e.g. Northern Ireland – 19 March 2001*) be represented?
- Can non-country public holidays (e.g. *Valencia - 22 January 2001*) be represented?
- Can non-day public holidays (e.g. *Turkey - 10 February 2001 half-day*) be represented?
- Is it possible to recognise when a public holiday in a whole country (e.g. *United Kingdom – 13 April 2001*) impacts upon residents of a part country (e.g. *Northern Ireland*)?

The process of sophistication was then undertaken in the following manner, dealing with one aspect at a time. Firstly, country was destratified so that the countries and their public holidays from the different implementations could be combined. Secondly, countries were generalised so that non-country public holidays could be included. Lastly, days were generalised to periods so that non-day holidays could be included. The result is shown in Figure 3, which is a significant improvement in that the model is:

- More universal in that it is not implementation indexed as the original Isure system was – it will now work over multiple implementations.
- More general as it works at the level of Geo-Political Regions rather than stratified country and public holiday patterns.
- Simpler in that it no longer needs to deal with the stratified country and public holiday patterns.
- More precise in that it is (a) a more faithful representation of the common-sense notion of a public holiday and (b) does not include the spurious Isure public holidays stratification.

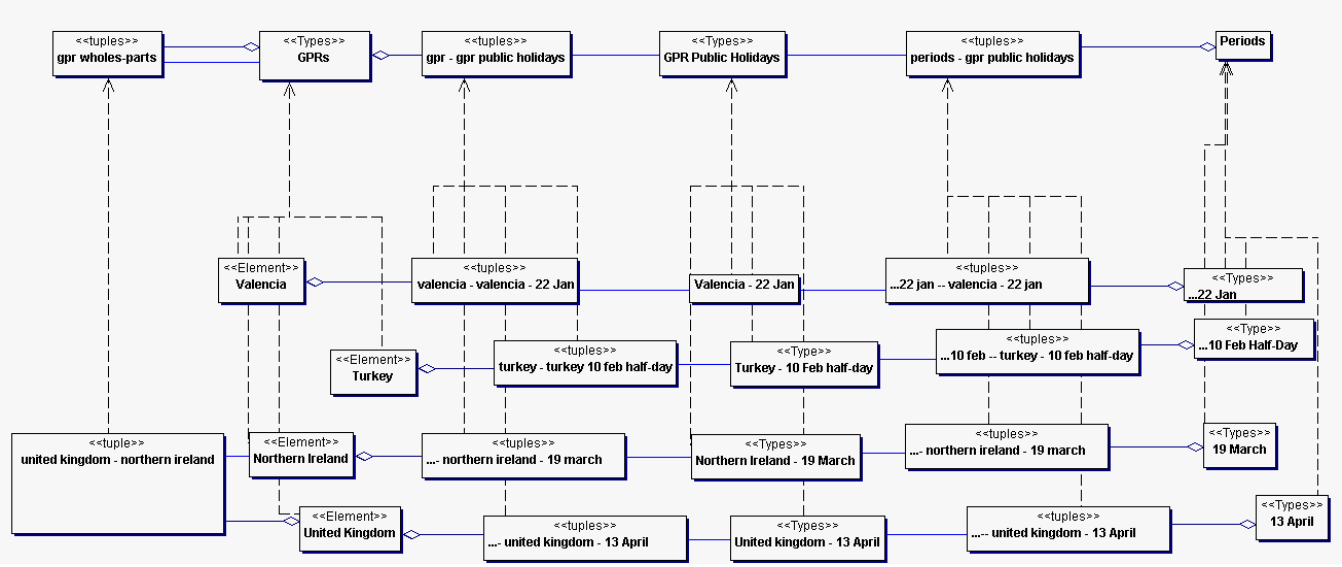

Figure 3. Generalized Country and Public Holidays

GBPs emerge from a process of harmonising the many fragments of the system selected for analysis and both business objects and GBPs can be envisaged as (directionally) stable. In this particular case, Isure was examined against an existing GBP called GPR. From a GBP perspective, stability is achieved via a process of constant comparison. This approach has strong roots in the grounded theory approach (Glaser et al. 1967), where theory emerges from analysis of empirical content and is refined in the context of additional empirical content until a saturation point is reached (i.e., new content does not impact the model). To provide a basis for comparison, the ontology is represented within its conceptual framework. While defining the conceptual framework is outside the scope of this paper, representing the ontology within its conceptual model is beneficial for the following purposes. While on one hand, this allows a clear articulation of the entities and relations among those entities, on the other hand, the framework provides the starting point for analysing the additional empirical content and what entities their content describes.

In this case, (a) theory can be considered as a model that is conceived of and taken to be true (Bhaskar, 1978) and (b) empirical content is drawn from existing financial service systems. True equates to more sophisticated in this case and the notion of saturation needs to be considered from a pragmatic perspective. Consequently, we conceive of sophistication in terms of maturity levels – where a single GBP might be sophisticated enough to provide interoperability across several

systems within an organisation, several systems across organisations and/or several systems across several domains for example.

THEORETICAL AND PRACTICAL IMPLICATIONS

The stability aspect within current re-engineering work remains quite low for a variety of reasons. The technology focus of the current approaches coupled with negligence towards modelling aspects of business content has meant that systems remain stable for a very short period of time. The CS approach provides a stable alternative to the current re-engineering approaches and in light of this argument, the theoretical and practical implications of the CS approach are as follows:

- **Stable General Business Patterns and Business Objects** - The CS approach through its sophistication process aims to provide a stable set of General Business Patterns and Business Objects that can be similarly recognised across various systems and also different organisations. For example, the destratification of Country has resulted in a more universal and practical view of Country and as a result is more stable that others.
- **CIM focus –** The CS approach works at the level of Computationally Independent Models (CIM), which highlight the need for well developed business models to clearly portray the needs and requirements of the business and aims to separate the business concerns from the application concerns.
- **Semantic richness of business content –** Applying the CS approach results in business content which is semantically richer and more sophisticated and as a result can cater for future requirements that might emerge later on. Generalising the 'Countries' business object to GPR increases the semantic richness of the content and also means that there are fewer objects to manage.
- **Flexible and effective re-engineering -** The CS approach is completely flexible, as it is not tied to any particular migration or design tool. Considering the model-based approach of CS, any design tool, depending on the needs and familiarity can be used. Moreover, given that the CS approach does not hinder on the general working of the system, the system can remain operational while business models are developed using CS. The effectiveness of the CS approach is proportional to the scope and the amount of time spent in analysing the existing system. Increases in scope can lead to better opportunities for sophistication of the business content, which over time provide a more significant payback as economies of scope and scale are increasingly achieved.

Although, the CS approach provides a number of benefits over the existing re-engineering approach, the success of the CS approach is affected by several factors including the availability of both type and individual level content, business and technical expertise of the existing system, and business modelling knowledge. The adoption of the CS approach is coupled with a necessary learning curve. This learning curve is however necessary as the approach requires a new way of thinking about systems development and so new development rules apply and new competences are required. Moreover, the level of sophistication and its associated benefits are directly proportional to the application of the CS approach within real systems. Since the approach is firmly grounded in content, the benefits are greater when the models are generalised out of several systems.

SUMMARY AND CONCLUSION

Software stability concentrates on balancing the seemingly contradictory goals of stability over the software lifecycle with the need for adaptability and extensibility (Fayad and Wu, 2002). This paper has outlined an approach for evolving stable business objects and General Business Patterns from empirical reference content drawn from existing software systems. Using a small example of a sophistication instance we have part-illustrated the process of sophistication and described how the outcomes produce business objects and patterns that are (directionally) stable; and more explanatory, fruitful, general, objective, precise and simple. The example has sought to illustrate fruitfulness and objectivity in particular.

The interested reader will have noted two things however: (1) we are operating at a computationally independent level and (2) business content provides the focus of the work. The purpose of working at a computationally independent level is one of ensuring that the outcomes can be tailored to any particular technology-set without being wed to it – that 'wedding' remains a challenge for both industry and academia. While the Model Driven Architecture (MDA) underpins our thoughts, the whole purpose of our approach is to liberate meaning in the business sense in order to ensure that systems represent the business in the most precise and universal manner possible. In that light, we have purposely sought to separate what there is (business content) from what the application does with it (application behaviour) as a positive step in achieving adaptability and extensibility.

Our work related to application behaviour is much less mature at this point in time, but is inherent not least in the fruitfulness of content – it is apparent that the model in Figure 2 works across multiple implementations and is rich enough semantically

to both handle the existing system functionality and, at least, the functionality implicit in the competency questions above. Future research will be aimed at identifying further general business patterns (e.g., a Product GBP) and consolidating the ones already identified. While on one hand, this will allow the CS process to mature, on the other hand, at the same time it will test the resilience of patterns already identified. Further work will also focus on the development of a software tool to automate CS activities as well as combining CS with application-level development.

REFERENCES

1. Bhaskar, R. (1978) A Realist Theory of Science, The Harvester Press, Sussex.
2. Bisbal, J., Grimson, J., Lawless D. and Wu B. (1999) Legacy Information Systems: Issues and Directions, *IEEE Software*, 16, 5, 103-111.
3. Brodie, M. and Stonebraker M. (1993) DARWIN: On the Incremental Migration of Legacy Information Systems, GTE Labs Inc, March, TR-022-10-92-165.
4. Fayad, M. and Altman, A. (2001) An Introduction to Software Stability, *Communications of the ACM*, 44, 9, 95-98.
5. Fayad, M. and Wu, S. (2002) Merging Multiple Conventional Models in One Stable Model, *Communications of the ACM*, 45, 9, 102-106.
6. Ganti, N. and Brayman W. (1995) Transition of Legacy Systems to a Distributed Architecture, John Wiley and Sons, New York.
7. Glaser, B. G. and Strauss, A. L. (1967) The Discovery of Grounded Theory: Strategies for Qualitative Research, Weidenfeld and Nicholson, London.
8. Honderich T. (ed.) (1995) Oxford Companion to Philosophy, Oxford University Press, Oxford.
9. O'Callaghan, A. J. (1999) Migrating Large-Scale Legacy Systems to Component-Based and Object Technology: The Evolution of a Pattern Language, *Communications of the Association of Information Systems,* 2, 3.
10. Partridge, C. (1996) Business Objects: Re-Engineering for Reuse, Heinemann, Oxford.
11. Tilley, S. R. and Smith D. B. (1995) Perspectives on Legacy Systems Re-engineering, Software Engineering Institute, Carnegie Mellon University: 15213-3890, 1-133.