**Global Co-Operation in the New Millennium**
The 9th European Conference on Information Systems
Bled, Slovenia, June 27-29, 2001

# USING PROBLEM-DOMAIN AND ARTEFACT-DOMAIN ARCHITECTURAL MODELLING TO UNDERSTAND SYSTEM EVOLUTION [RESEARCH IN PROGRESS]

**T. R. Addis**

University of Portsmouth,School of Computer Science & Mathematics,
Mercantile House, Hampshire Terrace,PORTSMOUTH, PO1 2RG
Tel.: +44 (0) 23 92 845141, Fax:+44 (0) 23 92 843030
Tom.Addis@port.ac.uk

**G. H. Galal**

University of North London, School of Informatics and Multimedia Technology,
166-220 Holloway Road, LONDON, N7 8DB
Tel.: +44 (0) 20-7753-3229, Fax: +44 (0) 20-7753-7009
Galal@acm.org

## ABSTRACT

*The authors describe on going research to uncover the architectonic nature of artefacts and see how these may be related to high-level, but also grounded, model of the original problem domain. We thus propose both a General Systems Architectural Model (GSAM) that reflects all the patterns of connectivity that may be found within any program and a Grounded Systems Engineering Method (GSEM) that captures the architecture of a problem domain. We hypothesis that there will be an isomorphism between the architecture of an artefact and the problem domain. We also hypothesis that for an artefact to be continually modified it has to conform to certain architectural features that can be abstracted from the domain. The purpose of the study is to examine a wide range of real programs (those designed for a job) and the problem domain that they address and from this examination determine the validity of the model and hypotheses by uncovering features and measures that define the architecture of the system. It is proposed that these features and measures could be used to make predictions about the flexibility, robustness and reliability of software. Such predictions would be used to formulate and close the design control loop. We hope that this work will lead to a universal constructional theory for systems.*

## 1. INTRODUCTION

Complex systems are perceived to evolve in often unpredictable, and hence unmanageable ways. This perception is caused by the fact that the requirements for such systems are rarely stable, which means that systems need to evolve continuously. A factor that exacerbates the problem is that most current software-engineering methods assume a final and complete specification; changes in requirements are regarded as evil

that should be resisted or better, eliminated. The result is an inevitable dramatic increase in maintenance cost and, more seriously, an inability to modify systems quickly enough to respond to changing needs.

## 2. POSSIBLE CAUSES OF THESE PROBLEMS

Conventional systems design is driven by immediate required functionality (e.g. Yourdon 1989). The result is that the structure of the designed system predominantly satisfies the functional needs that were specified at the outset of the design exercise. The longer-term and non-functional objectives, such as maintainability, have been traditionally addressed by design heuristics such as low coupling and high cohesion (Myers, 1975; Yourdon and Constantine, 1979). This usually leads to adaptability properties that are local to the level at which they were applied: the elementary structural components. Changes that are triggered at the user and requirements levels are very difficult to accommodate and often require far reaching changes.

Another source of the methodological difficulty is that the current approaches to software design assume that good global structures will emerge from local combination rules. The assumption that an adaptable architecture will emerge by following local structuring rules has not been proven. The current approaches fail on at least four counts:

1. innovative design does not follow the strict top down decomposition heuristic,
2. global coherence cannot be achieved through considering local measures,
3. there is no measure of global coherence or system integration on which to base design decisions,
4. there is no support of modelling the domain in a way that informs the process of architecting the system to produce emergent, long-term, adaptable behaviour.

The problem seems to be that the design of complex systems is mainly too narrowly focused on technical performance. Further, the activities of analysis and design typically only skim the surface of the problem when searching for required functions and performance characteristics. Normally, little effort is expended in trying to develop a deeper understanding of the nature of the evolution of the problem domain, and the types of functionality changes that it typically generates. Experience suggest that some aspects of the problem domain change more frequently and in a qualitatively different way from others. There are no methodological tools that explore the types of dynamics most problem domain exhibits. There are no published studies of how changes in the problem domain may be related changes in the system before they are needed; we need to make predictions of change. In particular, the predictions are needed in ways that will ultimately guide designers in the design of new systems.

## 3. IDEAS TOWARDS A SOLUTION

Our particular angle on this problem is that the enormous complexity of systems requires us to focus on how a problem domain can be modelled. This should be done in a way that sheds light on any isomorphism that may exist between the domain dynamics and system evolution. The investigation needs to be done at a level of abstraction that is higher than the programming language constructs. We have observed that architectural thinking about buildings is concerned with the overall configuration, relating it very clearly to the context, and its own emergent properties (see Stevens et al, 1998). We used the term "architectonic" to describe levels in a system that relate to their susceptibility to change (see Frampton, 1995).

Our ideas have been stimulated by two particular analogies. The first is that of ‘learning’ buildings. Brand (1994) argues that buildings that are relatively more adaptable exhibit a number of constructional layers that change at different speeds and that are loosely coupled to each other. The second analogy is that of urban layouts. Hillier (1996) shows how the spatial configuration of a city correlates quite strongly with patterns of space use. The most integrated streets tend to attract the largest volume of pedestrian movement, which in turn strongly influences the land use patterns, such as retail, housing and entertainment.

Putting these two analogies together, we reach the position that it would be desirable to model a problem domain in a way that tells us something about the relative stability of categories of its elements. This is akin to establishing the normative case about how a building is used. We would also like to model the domain to find out how particular modelling perspectives help us in pinpointing the layers that give rise to desired emergent behaviours, such as adaptability. This is akin to establishing what is the equivalent of the 'urban grid' layer in a technical system, that gives rise to other potentialities, and what are the best primitives to model that layer.

There are thus two things to model: the problem domain and the deep architecture of the system. The system can be modelled using 'constructive' methods that focus on the qualitative data that arise naturally within most problem domains, say by interviewing stakeholders. The hidden architecture of the artefact can be retrieved through a standardised representation of the concepts embedded therein.

### 3.1 The Hypothesis

Our hypothesis is that there is an 'architectonic' relationship between the domain model as it is embedded within the artefact and the domain model as elicited from the stakeholders. First we need to model the original problem domain at a relatively high level of abstraction, and with the necessary rigour. If we then model the technical artefact (the system) in a way that shows how it reflects the problem domain we can expect certain isomorphisms to be found. These isomorphisms suggest to us the way in which the original problem domain might govern the evolution of the technical artefact. Further, if a designer had a way of measuring how his/ her design coheres with some model of the problem domain then it's more likely that systems designed with this knowledge will ultimately exhibit a higher degree of adaptability.

### 3.2 The Approach

Our line of attack towards testing this hypothesis we will integrate elements of functional modelling of the systems with the modelling of the original problem domain using qualitative, constructive research methods, and ideas from urban and architectural morphology. The qualitative, constructive research method we chose is the Grounded Theory method from the social sciences (Strauss and Corbin 1998) has been evolved into the Grounded Systems Engineering Method or GSEM (Galal and McDonnell 1997; Galal and Paul, 1999).

Reasoning about deeper structures embedded in software requires a uniform means of describing them to facilitate comparison between software artefacts, and against domain models elicited from stakeholders. A functional approach (using functional representations) serves us with such uniform means for extracting and describing the domain models embedded in systems.

Addis and Addis (2001) have effectively used functional techniques and representations to extract and model software structures. This has been generalised into a General Systems Architecture Model, or GSAM, which proposes the three layers of 'Basic Domain Functions'; 'Intermediate Functions' relating to the more changeable requirements; and 'Interface Functions' as the least stable layer. Figure 1 illustrates a general architectural model using entity relationship diagram symbols (Page-Jones, 1988). In this case, the entities are the function sets classified into their architectural roles in the system. These are:

- the basic domain functions (fundamental elements of a generalised domain, usually directly mapped onto a programming language) that are fixed,
- user intermediate functions that are created by the user to bridge the gap between the generalised domain and the more specific, transient functionality of the problem domain,
- the user interface functions that integrate the system with the problem domain,
- the side effects triggered by the use of the functions.

The cardinality relationships describe a recursive structure of parent and children functions. These structures influence different kinds of flexibility as shown by the braces in the diagram. For example, the more inter-

mediate functions there are will suggest the need for a greater extension of the domain functions. Such a need implies that the domain functions are not adequate to bridge the gap between the domain and the computer. The number and complexity of these intermediate functions are a measure of the tension that exists between the problem domain and the computer system (say). The combination of function decomposition and expected number of children per function measure the complexity of the functions at all levels. The more the decomposition (up to a point) of the functions the easier it is to interpret the program/system.

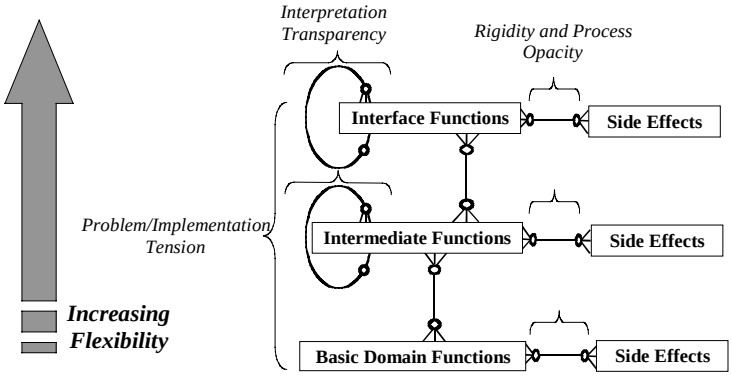


***Figure 1:*** *A General Systems Architectural Model (GSAM)*

This model acts as synthesising framework towards the extraction of embedded architecture. Note that it relates to the layered model of buildings presented above. We intend to examine this relationship closely. Our aim is to compare the domain models, obtained in two different directions: from the original domain, and from the artefact studies, to detect the architectonic shifts and aspects of stability. The comparison thus helps us to detect domain-artefact isomporphisms in a way that guides the design of new systems as well as the differential design, deployment and management of extant ones. The elements of our approach are:

1. Modelling the original problem domains addressed by the programmed artefacts supplied by our industrial collaborators. For this we use GSEM that has the benefit of producing constructs expressed in very simple diagrams and natural language.
2. Modelling the domains implicit in the systems using the GSAM model based on work on functional modelling by one of the principal investigators (Addis & Addis, 1998, 2001).
3. Developing a scheme for measuring the level of integration (or relatedness) in both models developed in 1 and 2 above. Perhaps using the spatial analysis techniques developed in the work of Hillier (1996), and in the general direction suggested in the work of Hillier & Galal (1999).
4. Using fuzzy logic, Baysian analysis and any other appropriate statistical techniques, we will develop measures of isomorphism between the original domain model (1) and the domain model implied by the system (2), after it has evolved over time. A similar idea has been successfully extended and combined with some advanced intelligent approaches, such as fuzzy logic, to take into account the inherent uncertainty in complex systems (Gegov, 1996, Jamshidi, 1997).
5. By experimenting with varying levels of abstraction and representational primitives, we expect to be able to pinpoint an appropriate level that best relates the architectonic nature of the system to the problem domain. This guides the designer to the optimal way of investigating and modelling the domain to extract features, or invariances, that should be clearly and directly mapped onto the architecture of the artefact.

If our hypothesis is proven correct, it will form the foundation for further testing on a wider variety of cases. This will ultimately lead to the development of a control scheme with the software designer as a controller and the software as the plant under control. This scheme will allow the designer to regularly compare the architectural features of the domain against the desired architectural features of the system. The designer can then apply corrective actions where necessary. The aim is to ensure that decisions taken by the designer are in line with the corporate (global) and long-term goals of the problem domain.

## 4. CONCLUSIONS

This paper, which describes work currently underway, has presented our views on how coarse-grain characteristics of the domain (domain architecture) may be correlated to the way in which a software artefact has evolved to serve it. The core hypothesis being that there is a certain degree of isomorphism between the architecture of the domain, and that of the artefact. Identifying such isomorphism and the optimal way to model is the subject of the forthcoming phase of this research. The way our research differs from the software engineering mainstream view is in the high level at which we consider software architecture, and the position that the architecture of evolved software artefacts can be clearly correlated to some model of the architecture of the domain it serves. To test our hypothesis, we use theories, concepts and techniques from functional programming, the social sciences and urban and architectural morphology. We hope to report of further progress in future writings in this area.

## REFERENCES

Addis T. R. and Townsend Addis J. J. (1998) '*A Functional Schematic Interpreter: an environment for Model Design*'. International Journal of Systems Research and Information Science, Vol. 7, Pub Gordon Breach Science, ISSN 0882-3014, Pages 263-299.

Addis T. R. and Townsend Addis J. J. (2000) *'Avoiding Knotty Structures in Design: Schematic Functional Programming.'*, International Journal of Visual Languages & Computers, Vol. 12, Pages 22.

Brand, S. (1994). *How buildings learn: What happens after they're built.* London, Phoenix Illustrated

Frampton K. (1995) '*Studies in Tectonic Culture – The Poetics of Construction in Nineteenth and Twentieth Century Architecture.'* Cambridge, Mass: MIT Press.

Galal, G. H. and J. T. Mcdonnell (1997). "Knowledge-Based Systems in Context: A Methodological Approach to the Qualitative Issues." AI and Society 11(1-2): 104-121

Galal, G. H. & Paul, R. J. (1999) *'A Qualitative Scenario Approach to Managing Evolving Requirements'* Requirements Engineering Journal 4 pp92-102.

Gegov, A. (1996) '*Distributed Fuzzy Control of Multivariable Systems'* , Kluwer Academic Publishers: Dordrecht, Netherlands.

Hillier, B. (1996). Space is the machine - A configurational theory of architecture. Cambridge University Press: Cambridge.

Hillier, B. and G. H. Galal (1999). Capturing Emergence: surface patterns and deep structures. Paper on invited keynote speech 4th IEEE Symposium on Requirements Engineering, Limmerick, Ireland, IEEE Computer Society.

Jamshidi, M., (1997) *'Large Scale Systems: Modelling, Control and Fuzzy Logic'*, Prentice Hall: Englewood Cliffs.

Myers, G. J. (1975). *Reliable Software Through Composite Design.* Van Nostrand Reinhold Company: New York.

Page-Jones, M. (1988) 'The Practical Guide to Structured Design' 2nd edn. Prentice Hall.

Stevens, R., P. Brook, K. Jackson & S. Arnold, (1998). Systems Engineering - Coping with Complexity. Prentice Hall: London.

Strauss, A. and J. Corbin (1998). Basics of Qualitative Research, Techniques and Procedures for developing Grounded Theory, Sage: California.

Yourdon, E. (1989). *Modern Structured Analysis.* Englewood Cliffs, N. J., Yourdon Press/Prentice-Hall: Englewood Cliffs, N.J.

Yourdon, E. and L. C. Constantine (1979). Structured Design, fundamentals of a discipline of computer program and system design. Prentice-Hall: Englewood Cliffs, N.J.