Extreme Architecture: Architecture Components

By Phil Robinson and Floris Gout

IT groups have a pressing need to identify the things worthy of their attention.
In this and a second paper, we present an Information Technology (IT) Architecture framework that draws inspiration from legislation, enacted in 1996. The papers encourage a minimalist approach to IT Architecture by adopting a number of extreme points of view.

In this paper, we present a comparison between buildings, the traditional foundation of architecture, and the 'softer' systems associated with IT. The various building blocks of the framework are introduced.

In our second article, the full IT architecture framework is described using a single, uncluttered diagram. The framework is simple to describe and easy to recall. It organises 19 different elements that, taken together, define an IT architecture. This second article will be published in a future edition of Oracle Scene, and/or available at `http://www.ukoug.org/lib/`

The framework provides a point of reference for IT management, IT project managers, developers and operations staff who are frequently charged to do more with less in these lean times. These groups will find that the framework can offer them a much needed lifeline.

A couple of years ago, we both found ourselves working on strategic IT planning projects. Since both projects were similar, we began meeting on a regular basis to share ideas and plot strategies. We continued to meet long after the projects had ended, and began to realise that we were slowly defining an IT architecture framework that was worth sharing with a much wider audience.

Partly to emphasise that our framework is different to other frameworks (and partly to be unashamedly trendy) we coined the phrase 'extreme architecture'.

We use the word 'extreme' in two senses. Firstly we believe that, compared to other architecture frameworks, our framework is minimalist in the extreme. In contrast to other frameworks it is:

- Easy to describe
- Encourages an agile approach to architectural work products
- Unifies a number of disparate disciplines
- Provides a variety of stakeholders with a consistent view of an IT architecture.

The second sense in which we use extreme involves a bit of word play. We believe that our framework is in fact a very pragmatic approach that adopts the middle path thus avoiding the extremes of perfection and chaos!

Building

In his well known essay on software architecture, Frederick Brooks cites the building of the cathedral at Reims as a triumph of architectural vision[1]. According to Brooks, eight generations of builders sacrificed their own ideas in order to maintain the design integrity of the cathedral. In describing the cathedral, Brooks states, *'…the joy that stirs the beholder, comes as much from the integrity of the design as from any particular excellences.'*

Brooks offers the triumph of Reims as a sort of Holy Grail for software developers.

In stark contrast, the authors of the Big Ball of Mud Pattern[2], describe what they claim is the most frequently adopted software architecture. They use the metaphor of a shanty town to describe the architecture and point out that shanty towns are, *'…built from common, inexpensive materials and simple tools, using relatively unskilled labour.'*

The authors argue that there are a number of reasons why the big ball of mud is common:

- Pressure on project schedules and costs
- Inexperience, lack of skills, and high staff turnover
- The complexity and rate of change of the software requirements.

Confronted by the extremes of the Holy Grail and a big ball of mud, we find ourselves naturally drawn to the middle path. In doing so we believe we we are in good company. The Buddhist faith advises that, *'…avoiding the extremes, gives vision and knowledge and leads to calm, realisation, enlightenment…'*

Systems

Although quite useful at a superficial level, the building-software metaphor can only be taken so far. Buildings are static structures that interact minimally with their environment. In contrast, software systems are by very nature, highly dynamic and interactive.

The design integrity of a building is determined by factors such as geometric proportion, structural strength and the materials used for its construction.
In comparison, the design integrity of software depends on factors that are much more abstract and sometimes quite difficult to measure.

Another major difference is that software is always embedded in some other system. In some cases this is a hardware system such as a cellular phone or microwave oven. In other cases it is a (potentially much more complex) human activity system. In both cases, a detailed understanding of the system in which the software is embedded, is essential for a proper understanding of the software requirements. Our architecture framework focuses on software that is embedded in human activity systems.

Figure 1 shows how human activity systems and software systems are often decomposed into a hierarchy of systems. Industry sectors, consisting of a group of enterprises engaged in similar activities, are placed at the top of the hierarchy. Individual enterprises appear at the next level. These are underpinned by business processes, that are in turn supported by software applications. The software components used to construct applications appear at the bottom of the hierarchy.

In addition to the hierarchical decomposition of systems described above, systems are often presented as being nested one inside the other. The nested view implies that a system can only interact with its container system or the other systems sharing the same container. This implies, for example, that a business process interacts in a limited way with the business process of a different enterprise, or that a software application supports a single business process.

While restrictions such as these may have been workable in the past, they are unacceptable constraints in a modern business environment. For example:

- It is common for business processes to be outsourced. When this happens it means that a number of enterprises interact with a single business process.
- It is becoming common for people outside an enterprise, such as customers, agents and suppliers, to interact directly with the enterprise's software applications, thus bypassing the internal business processes.
- Modern software development techniques emphasise the creation of reusable components. The goal is to reuse a component many times across a variety of software applications.

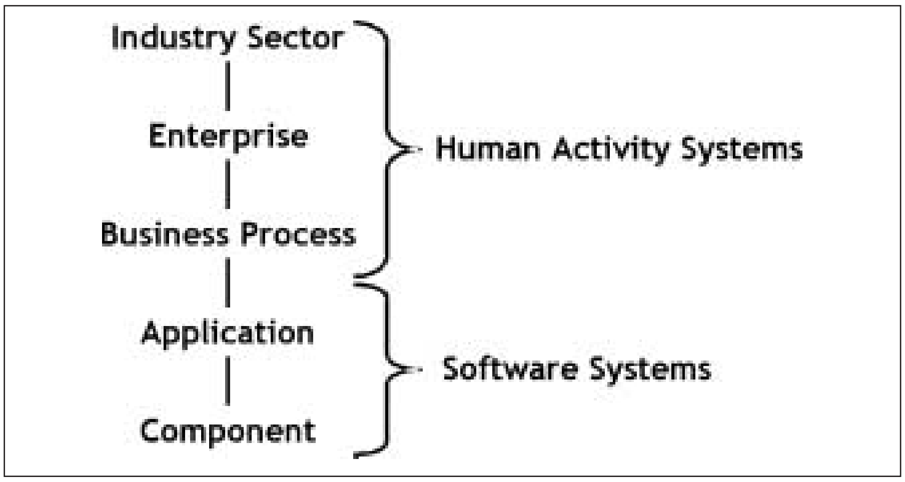

Fig. 1: Hierarchy of Subsystems

For these reasons, we believe that it is more realistic to view human activity and software systems in terms of a number of independent and overlapping systems. Our view as illustrated in Figure 2 below.

Independent, overlapping systems are able to interact with each other at all levels of the hierarchy. For example:

- A software application might support a number of different business processes.
- Some of the business processes may span several different enterprises.
- A software application may correlate information flowing from a number of different enterprises.
- A software component may be designed to provide a suite of generic services that are used by one or more industry sectors.

This view of human activity and software systems leads to an overriding requirement for interoperability between the individual systems. Interoperability can be defined as; *'…the ability of a system to successfully interact with other, specified systems.'*

Architecture

High levels of interoperability are unlikely to be found in software systems that resemble a shanty town or big ball of mud. As the builders of Reims knew, to achieve a high level of design integrity, it is necessary to develop a detailed architectural plan and stick to it.

A clear and simple definition of what constitutes architecture can be difficult to achieve. We like this definition of building architecture that was provided by

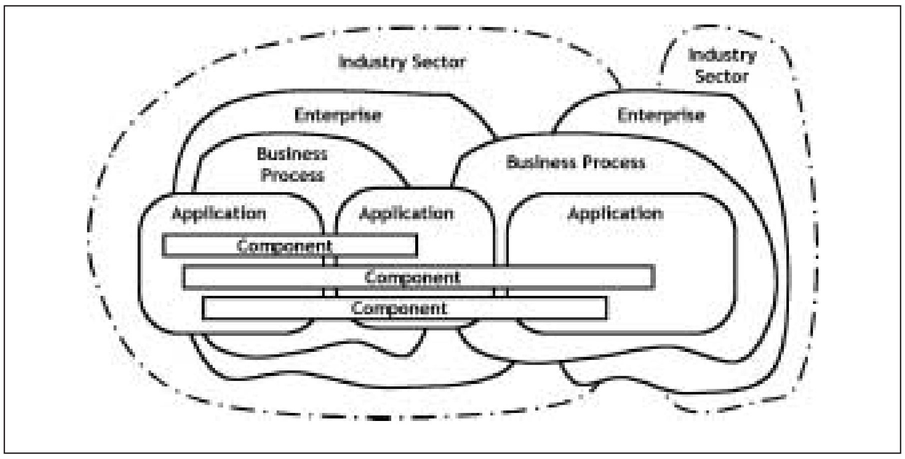

Fig. 2: The Independent and Overlapping Nature of Human Activity and Software Systems

Ean MacDonald[3], a retired architect said, *'Architectural design is the simultaneous resolution and solution of the various architectural problems; that can include location, aspect, and prospect, sun, wind, and weather, materials and method, finance, function, and form, to which may be added a dash of flair that can make a structure work of art.'*

An equally appealing definition for IT architecture was offered by Jim Sinur of the Gartner Group at the 1996 ZIFA Annual Forum,
'If you can implement a drawing in more than one way, you have architecture.
If you can implement a drawing in only one way, then the drawing contains exact specifications for instantiation, and you have a design.'

Surprisingly, a formal definition of IT architecture can be found in legislation. In February 1996, the US Congress passed the Information Technology Management Reform Act of 1996, which is widely known as the Clinger-Cohen Act.
It included a formal definition of IT Architecture that has now passed into law, *'An integrated framework for evolving or maintaining existing information technology and acquiring new information technology to achieve the agency's strategic goals and information resource management goals.'*

A key aspect of the Clinger-Cohen Act's definition of IT Architecture is its reference to an integrated framework. In fact, the word framework is frequently used in relation to IT architectures. A framework describes an underlying structural arrangement that can be used as the starting point for developing an IT Architecture.

Perhaps the best-known architecture framework is the Zachman Framework for Enterprise Architecture, proposed by John Zachman in 1987[4]. The framework is presented as a matrix of six rows and six columns. Each row of the matrix describes a perspective of the architecture as seen by various enterprise roles (planner, owner, designer, builder, out-of-scope, and the functioning enterprise). The column headings consist of what Rudyard Kipling called the 'six honest serving men' (what, why, when, how, where, and who).
Each cell of the matrix contains a primitive model that describes a single aspect of the architecture. Thus a fully populated Zachman Framework consists of thirty-six independent views of an enterprise.

The Zachman Framework is a comprehensive, rich and intellectually appealing approach to classifying architectural models, but it is not always easy to understand and apply.

Another noteworthy architecture framework is The Open Group's Architectural Framework (TOGAF). As well as describing the framework, TOGAF also includes a detailed methodology that can be applied to the development of an IT Architecture. (TOGAF was in fact based on yet another framework, the now defunct Technical Architecture Framework for Information Management (TAFIM) developed by the US Department of Defense.)

Following the enactment of the Clinger-Cohen Act, the US Office of Management and Budget (OMB) issued a directive that also included a brief description of an architecture framework. We chose this framework as the starting point for our framework because we appreciate its simplicity in comparison to the Zachman Framework or TOGAF.

As the diagram above illustrates, the OMB architecture framework has three major components:

- An Enterprise Architecture, which defines the relationships between an enterprise's business activities, information systems, and its information technology infrastructure.
- A Technical Reference Model, which provides a generic description of the services provided by an enterprise's information technology infrastructure.
- A Standards Profile, which is a collection of information technology standards that precisely define the services identified in the Technical Reference Model.

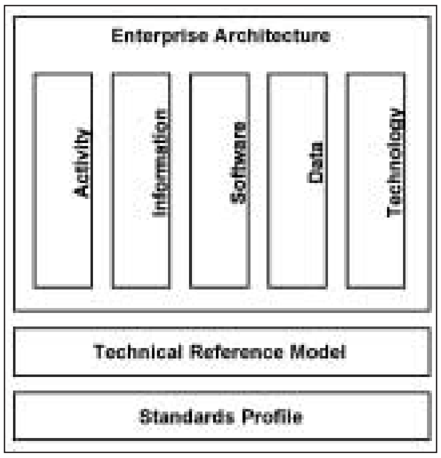

Fig. 3: The OBM Architecture Framework

The Enterprise Architecture is further decomposed into five sub-architectures:

- An *Activity Architecture*[a], which describes an enterprise's high-level business activities and workflows.
- An *Information Architecture*[b], which describes the information required to support the business activities described in the *Activity Architecture*[c].
- A *Software Architecture*, which describes the software that is required to support the Activity and Information Architectures.
- A *Data Architecture*[d], which describes the logical and physical structure of the enterprise's software-maintained data stores.
- A *Technology Architecture*[e], which describes the technical environment in which software executes.

a *In the OMB memorandum, this architecture is called the Business Process Architecture. We have changed the name because we have already used the phrase business process to describe a type of system.*

b *It is common practice to either combine the Information Architecture with the Activity Architecture and call it a Business*

Architecture or to include information requirements in the Data Architecture. We do not like the first approach, as the phrase Business Architecture sounds rather vague and nebulous. As far as the second approach is concerned, we feel it does not highlight the fact that information is related to business activity, while data is more closely related to information technology. In addition, there are issues associated with an Information Architecture, such as the management of non-electronic records that are not accommodated well in the Data Architecture.

c *In the OBM memorandum, this architecture is called an Applications Architecture. We have changed the name because we have already used the word application to describe a type of system.*

d *The Software Architecture and Data Architecture together could be viewed as the definition of an Information Systems Architecture. In fact, this composite architecture would appear to be a much better candidate to be named Applications Architecture.*

e *The Technology Architecture includes components such as hardware platforms, operating systems, database management systems, networking software, and the communications infrastructure.*

The Technical Reference Model is a comprehensive list of all the generic IT services that are available to an enterprise. The list includes items such as:

- Data interchange services
- Data management services
- Graphics services
- Directory management services;
- Network services
- Operating system services.

The Technical Reference Model groups the services into logical classifications, rather than identifying specific products or solutions (see TOGAF [v] for an example of a technical reference model).

The Standards Profile is a collection of standards that fully define the services identified in the Technical Reference Model. Standards are fundamental to the achievement of interoperability between systems. The standards are often classified in the same way as the services identified in the Technical Reference Model. Internally developed guidelines, de facto standards and formal international standards can all be included in the Standards Profile.

In this article we have introduced the basic building blocks of our extreme architecture framework.

References

1 Brooks Jnr, Frederick P., The Mythical Man-Month, 1995, Addison-Wesley.

2 Foote, B. and Yoder, J., 'Big Ball of Mud', Pattern Languages of Program Design 4, 1999, Addison-Wesley.

3 MacDonald, Ean, Notes on a conversation about building architecture. Perth, Western Australia. October 2003.

4 Zachman, John A. 'A Framework for Information Systems Architecture.', IBM Systems Journal, vol. 26, no. 3, 1987, IBM. Available `http://www.zifa.com/` [2002, 1 June].

5 The Open Group Architecture Framework (TOGAF), Version 8, 'Enterprise Edition', 2002, The Open Group. Available at `http://www.opengroup.org/products/publications/catalog/i912.htm`

About the Authors

Floris Gout gained his Bachelor of Applied Science (Information Science) at Edith Cowan University in Perth, Australia. Whilst studying at ECU he worked at the University of Western Australia, building research databases for epidemiological studies. Floris was then employed at the Department of Justice from 1991 till 1999. He became Data Administrator and was Project Manager for its first data warehouse. Floris has been an independent contractor since 1999 and he is still enjoying new and creative challenges. He can be contacted at floris@floris.com.au

Phil Robinson has been involved in the planning, analysis and implementation of a diverse range of business, scientific and technical information systems. Phil is an experienced workshop facilitator and has led numerous workshops in the course of his consulting assignments. Phil has presented training courses for organisations in Australia, Thailand, Hong Kong, Singapore and Indonesia. As well as presenting courses, Phil has authored numerous courses for industry and three University units. He has also had two books published on programming Apple computers. The books were published in a number of countries including the USA, UK and as translations in Germany and France. More recently, he co-authored two award-winning articles describing an original organisational theory. He can be contact at Lonsdale@iinet.net.au

This paper represents a major collaborative effort that organises not just the ideas of the two authors but also the many inspiring people they have worked with.

This article was originally printed in the AUSOUG's Foresight magazine.