XAF: A Minimalist EA Framework for an Agile Environment

by Floris Gout and Phil Robinson

As consultant-educators, we need to be able to quickly grasp an organization's architecture at different levels of abstraction. We have created a new framework, the Extreme Architecture Framework (XAF), in order to do just that. The XAF presents a unified view of human activity and software systems from the perspectives of business, information, and technology. The unified view strikes a balance between architectural perfection and the chaos that inevitably ensues when there is no architecture.

Of course, enterprise architecture (EA) frameworks already abound, and EA expert Jaap Shekkerman offers a useful inventory of the notable examples [13]. In presenting "yet another framework," it is not our intention to reinvent the wheel but rather to synthesize some of the best ideas in the field and inject a healthy dose of experience. The XAF has evolved over a period of time and has been heavily influenced by our consulting assignments. Early on, we sought to emulate the ideas advocated by Extreme Programming (XP) [1]. XP is a lightweight but highly rigorous approach to software development in which well-established best practices have been taken to the "extreme." For example, testing is good, so test first; code reviews are good, so pair program, and so on.

In similar fashion, the XAF is a lightweight, pragmatic approach to EA. It is lightweight because it removes the many "roles" (owner, planner, etc.) contained in some frameworks but keeps the architectural perspectives accepted by others. It is pragmatic because it ensures these perspectives are kept in all system views (the system views give architectural depth). The XAF shows all major work products needed by the architectural activities of governance, planning, and development.

The XAF answers the two questions our clients (IT managers and CIOs) ask most often:

1. "Which elements of the enterprise do I need to be aware of and understand?"
2. "Which elements am I responsible for, and which do I need to manage?"

ARCHITECTURE

A clear and simple definition of what constitutes architecture can be difficult to achieve. We like this definition of building architecture provided by retired architect Ean MacDonald:

> 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 the structure a work of art. [11]

In addition to noting that architecture is art, Roger Pressman describes architecture as representations of a software system that allows software engineers to:

> (1) Analyze the effectiveness of the design in meeting its stated requirements, (2) consider architectural alternatives at a stage when making design changes is still relatively easy, and (3) reduce risks associated with the construction of the software. [12]

Based on the opinions of these two architects, architecture provides both functional design and aesthetic appeal. It is useful to further explore metaphors that equate buildings with software systems and urban landscapes with enterprises.

BUILDINGS AND CITIES, SOFTWARE AND ENTERPRISES

Frederick Brooks cites the Gothic cathedral at Reims as a triumph of architectural vision [2]. Eight generations of highly skilled builders sacrificed their own ideas in order to maintain the design integrity. Brooks offers the triumph of Reims Cathedral as a sort of Holy Grail for software developers. In stark contrast, Brian Foote and Joseph Yoder, authors of the *Big Ball of Mud* pattern, use the metaphor of a shantytown as the most frequently adopted software architecture. In this view, buildings

are "built from common, inexpensive materials and simple tools … using relatively unskilled labor" [4].

We must take care, however, not to mix metaphors. Brooks's Reims Cathedral metaphor is based on the architecture of a single building, while Foote and Yoder's shantytown metaphor is simultaneously based on the architecture of buildings and a chronic lack of urban planning. Urban planning involves laying out the streets and providing services. Residents are free to build whatever style of house they desire so long as it conforms to the building standards and guidelines laid down for their area.

Canberra, Australia, which was designed by Walter Burley-Griffin, is a pinnacle of urban planning. In contrast to Canberra, a shantytown represents an extreme lack of urban design, where space is utilized with adverse effects on land and residents. Reims Cathedral and Canberra reflect the values of "architecture as art," glorification, and national pride. At the opposite extreme, the shack and the shantytown reflect architecture as non-art, squalor, and social disenfranchisement.

Confronted by these two extremes, we found ourselves drawn to what Buddhism calls the "middle path," which, "avoiding the extremes, gives vision and knowledge and leads to calm, realization, [and] enlightenment" [15]. Most architects and urban planners spend the majority of their time following the middle path, designing residential suburbs. The suburban house offers the perfect metaphor for this middle-path approach, situated as it is between the two extremes of perfection and chaos.

SYSTEMS AND INTEROPERABILITY

While useful, building and urban planning metaphors have significant limitations. Over time, buildings interact minimally with their environment. Services and access roads change infrequently.

In contrast to the static nature of buildings, human activity and software systems are highly dynamic and interactive. The design integrity is determined by factors that are far more abstract and often difficult to measure.

Fundamental to the XAF is the idea that an enterprise is a collection of human activity and software systems:

- An **industry sector** is a system of interacting enterprises that are all engaged in a similar type of activity.

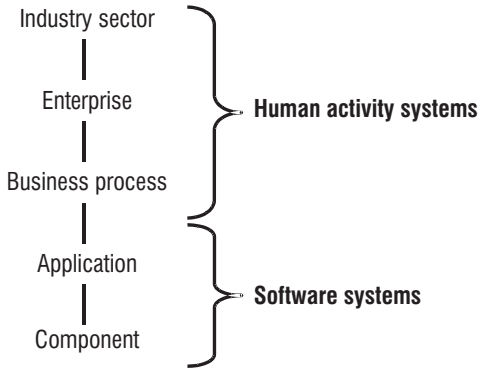

Figure 1 — Enterprise systems.

- An **enterprise** is a collection of individuals performing systematic and purposeful activities in support of a well-defined mission.
- A **business process** is "a sequence of activities that produce outputs of value to the customers of the process" [6].
- A **software application** is a mechanism for packaging and physically deploying a collection of software functions.
- A **software component** is a piece of software that can be easily replaced by another piece of software and can potentially be reused in a number of different applications.

The hierarchy of system types, shown in Figure 1, is often mistaken for a hierarchy of systems (i.e., industry sectors are composed of enterprises, enterprises are composed of business processes, and software applications are composed of software components). While this perspective was convenient in the past, the contemporary business environment now invalidates it:

- Business processes are commonly outsourced and thus are not nested inside a single containing enterprise.
- People outside an enterprise, such as customers, agents, and suppliers, now interact directly with the enterprise's software applications. Software applications are not nested inside business processes.
- Modern software development techniques emphasize reusable software components. A reused software component is not nested inside a single software application. The trend toward Web services means that an entire industry sector can use the services provided by a single software component, thus invalidating the entire hierarchy of systems.

We believe that it is more realistic to view the human activity and software systems of an enterprise as a number of independent and overlapping systems (see Figure 2). Independent, overlapping systems are able to interact with each other in an unrestricted manner, leading to the overriding requirement for interoperability. Interoperability, in turn, requires a high level of structural integrity. Needless to say, high levels of interoperability are unlikely to be found in systems that resemble a shantytown.

ARCHITECTURAL VIEWS

Describing a large building, town, or city is a complex task. Architects and urban planners present their plans using a number of complementary views. One view could show the proposed subdivision of land. Additional views might include transportation, public amenities, or environmental concerns.

For similar reasons, enterprise architectures are also typically presented as a number of complementary views. Perhaps the best-known set of architectural views are those described in the Zachman Framework [17]. The US National Institute for Standards and Technology (NIST) architecture framework describes a different set of architectural views [3].

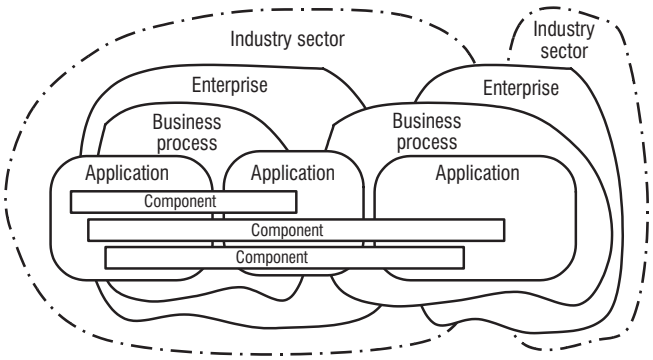

Figure 2 — The nature of human activity and software systems.

The OMB Reference Architecture

Following the enactment of the US Clinger-Cohen Act, the NIST architectural views were endorsed by the Office of Management and Budget (OMB). The OMB reference architecture has three major components, as shown in Figure 3.

The enterprise architecture consists of five architectural views that together describe the enterprise:

- Activity architecture[1] (the high-level business activities and workflows)
- Information architecture[2] (supports the business activities)
- Software architecture[3] (supports the activity and information architectures)
- Data architecture[4] (describes the logical and physical structure of the software-maintained data stores)
- Technology architecture[5] (describes the technical environment in which software executes)

The technical reference model is a comprehensive list of the generic services provided by the enterprise's technology infrastructure and includes items such as data interchange services and data management services, among others. The Open Group Architectural Framework (TOGAF) includes a detailed example of a technical reference model [14].

The standards profile is a collection of standards that fully specify the generic services identified in the technical reference model. Standards are fundamental to interoperability between systems.

[1]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.

[2]It is common practice either to 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 information architecture that are not accommodated well in the data architecture. Nonelectronic records are often a component of the information requirements for a business activity. However, the value of including nonelectronic records in a data architecture is questionable because they are not manipulated and stored using information technology.

[3]In the OMB 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.

[4]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.

[5]The technology architecture includes components such as hardware platforms, operating systems, database management systems, networking software, and the communications infrastructure.

THE EXTREME ARCHITECTURE FRAMEWORK

The XAF is a matrix of independent, overlapping systems and OMB architectural views. Each cell in the matrix contains a number of architectural elements to organize architectural content.

We chose the architectural views from the OMB enterprise architecture for the XAF because they are easy to understand. The OMB reference architecture does not exhaustively define an enterprise. A full understanding of an enterprise must include elements such as business strategy, culture, and values. These facets will influence the architecture in much the same way that financial, political, and social factors influence urban planning. However, we are seeking a minimalist framework.

Figure 4 shows the complete XAF with the 18 architectural elements added to the framework. Some of the cells have been grouped together when they share similar content across a number of rows of columns. The most obvious examples are the grouping of Sector, Enterprise, and Process into a single row representing human activity systems and the grouping of the entire Technology column into a single cell.

A necessarily brief description of the 18 architectural elements is given below. The high-level descriptions facilitate the mapping of the XAF to definition standards such as the Unified Modeling Language or the Integrated Definition Methods series.

- **Activities:** describe the business activities performed within a sector, enterprise, or business process

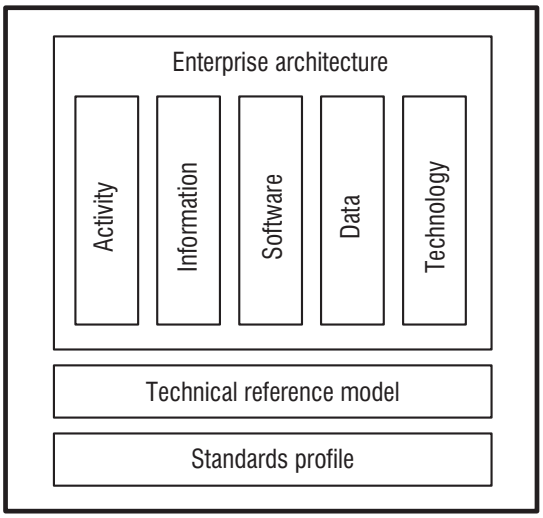

Figure 3 — The OMB Architecture Framework.

- **Workflows:** describe the flow of physical objects and information between business activities
- **Subject areas:** classify and group information requirements, business objects, and storage requirements that have a common theme
- **Information requirements:** describe the information required or produced by an activity
- **Functional areas:** classify and group functional requirements, nonfunctional requirements, interface requirements, and use cases that have a common purpose

	Activity	Information	Software	Data	Technology
Sector	Activities Workflow	Subject areas Information requirements	Functional areas	Business objects	Networks Platforms Frameworks
Enterprise					
Process					
Application	Use cases	Interface requirements	Functional requirements Nonfunctional requirements	Storage requirements	
Component	User interface		Architecture Code	Schemas	

Figure 4 — Extreme Architecture Framework.

- **Business objects:**[6] represent the concepts of interest within the Sector, Enterprise, or Process
- **Use cases:** describe the interaction of a software application between an actor and the software [8]
- **Interface requirements:** describe the application's interface with a user or another application
- **Functional requirements:** describe the mandatory capabilities, actions, and behavior of a proposed software application
- **Nonfunctional requirements:** describe the requirements of a proposed software application that are not related to its capabilities, actions, or behavior; e.g., quality attributes (performance, usability, security, etc.) and application constraints (software platform, external environment, etc.)

Partitioning the framework into rows and columns provides valuable insight into the governance and development tasks of EA.

- **Storage requirements:** describe data that will be permanently stored (persistent data) and may include detailed data element definitions.
- **User interfaces:** the screens, reports, and Web pages that a user interacts with
- **Architecture:** various high-level views of a software application[7]
- **Code:** the human-readable source code that defines the software and binary code that is executed
- **Schemas:** the electronic data store in terms of the records (or tables) and relationships between the records
- **Networks:** the mechanisms used to connect platforms, which permit the transfer of data and invocation of remote services
- **Platforms:** the hardware, firmware, system software, and middleware required to deploy and execute a software application
- **Frameworks:** standard component models and/or reference software architectures such as J2EE or .Net

ARCHITECTURE CONTENT

The body of the matrix can contain different types of content. The content for an architectural element might consist of (but is not restricted to):

- A model, list, or definition of any of the actual architectural elements (e.g., a list of the core business processes, a list of business objects relevant to a sector, or a data model for an application)
- An assessment or SWOT [Strengths – Weakness – Opportunities – Threats] analysis of the current state of an element (e.g., an assessment of data quality associated with a database). Assessments might also refer to:
  - — A potential risk (e.g., low customer satisfaction associated with a complex business process)
  - — A potential reward (e.g., a reduction in procurement costs associated with effective data interchange with suppliers)
- A vision of some future state of an element and how it will contribute to business strategies and goals (e.g., data will be seamlessly transferred between different business processes)
- A strategy or course of action to achieve the vision (e.g., using a "hub and spoke" architecture strategy to integrate software applications and facilitate data transfer)
- An underlying principle associated with an element (e.g., the principles of minimizing data redundancy and duplication)

The framework as described here contains architectural representations and planning information. We are currently experimenting with synthesizing balanced scorecard techniques [9, 10] and planning models described by the Business Rules Group [5]. Readers familiar with

[6]Strictly speaking, the "objects" of object orientation have relevance in three places in the framework. The persistent business objects described here belong in the Data column; business objects that have behavior (as well as state) belong in the Activity column (we actually don't recommend that activities be modeled in this way, but some users of the framework may prefer this approach); and software classes and components belong in the Component row.

[7]Christine Hofmeister and her coauthors identify views for the underlying conceptual organization, individual modules, organization of the source code, and runtime deployment of software [7].

Radical Project Management [16] will recognize items common to both the architectural framework and a business case. The business case contains content for architectural scope in its project objectives and outcomes. For example, a business case for an application would contain objectives for implementing user interactions, interfaces to other applications, functional requirements, data storage requirements, and technology considerations. The application row in the XAF contains these elements. Any framework must provide the architectural content items to support the work products for governance, planning, project management, and development.

GROUPING FRAMEWORK CELLS

The framework cells are grouped into the architectural perspectives and system types. In addition to this "standard" grouping of cells, the matrix cells can be grouped in a number of other ways. As we will see, partitioning the framework into rows and columns provides valuable insight into the governance and development tasks of EA.

Rows: The Systems Development Lifecycle

Grouping the cells into rows reflects the major disciplines associated with the systems development lifecycle (see Figure 5).

The business modeling row describes what the enterprise does and how its activities are supported by its software systems. This is the context for the individual software applications. The requirements definition row defines the requirements for a single software application. The content of this row also reflects typical architectural content of a requirements definition document. The software construction row describes the physical artifacts that together implement a single software application. Technology is considered at all three levels.

Columns: Enterprise Architecture Governance

The logical choice of names for the columns tends to reflect the management disciplines frequently tasked with EA governance (see Figure 6).

The process improvement column includes the elements that focus on business process reengineering projects or continuous improvement initiatives. The information management column includes the elements needed to properly manage effective use of information. The software portfolio management column includes the elements that define an organization's software portfolio. A major concern of those who focus on this column is integration of disparate custom-developed software and packages.

The data administration column includes the elements that define an organization's electronic databases. This discipline drives data quality improvement and data integration initiatives. The infrastructure management column manages hardware and software platforms, networks, and the technical frameworks. While this discipline is concerned with guaranteeing the smooth running

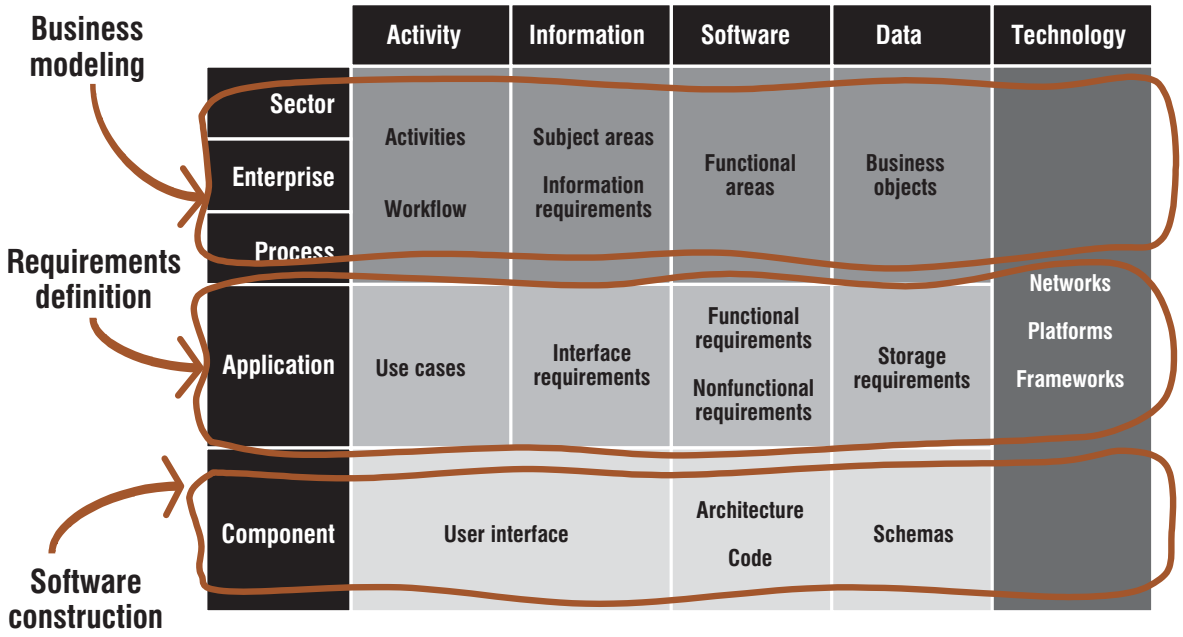

Figure 5 — Areas of architecture content.

of technology infrastructure, it is also responsible for technology conversion and rationalization projects.

Mapping Business and IT Responsibilities

These disciplines are distributed across the organization, and tension between business and IT groups is commonplace. The reason for the tension becomes clearer when the responsibility for different columns is highlighted on the XAF; business groups are usually responsible for the Activity and Information columns, while IT groups are usually responsible for the Data and Technology columns (see Figure 7).

So what of the Software column? It is true that IT actually constructs the software portfolio. However, one could argue that business groups must be actively involved in management of the software portfolio if it is to meet their needs. We conclude that both groups must share responsibility for the software portfolio. The Software column represents the boundary between the two groups where most benefit will be gained from collaboration and joint responsibility; thus it has the potential to be an "axis of joy." But in enterprises that cannot resolve the tension between their business and IT groups, this column represents an "axis of sorrow."

Managing Conflict Where Rows Meet Columns

The individual cells of the XAF are where the lifecycle and management disciplines intersect (see Figure 8). They also highlight areas where cooperation is required and conflict is common. For example, the intersection between the Component row and the Data column involves interaction between software developers and data administrators. Many readers will be familiar with situations in which a software developer designs a data structure that does not conform to enterprise standards. Often the developer has compelling reasons to do this, but it will inevitably bring him or her into conflict with the data administrator.

The XAF offers a way out of this dilemma by providing a concrete artifact through which both parties can present their arguments while at the same time acknowledging the opposing point of view. For instance, the data administrator might show how the developer's schema needs to be altered to meet an enterprise business object model. On the other hand, the developer could show the data administrator how the schema would assist in meeting an interface requirement.

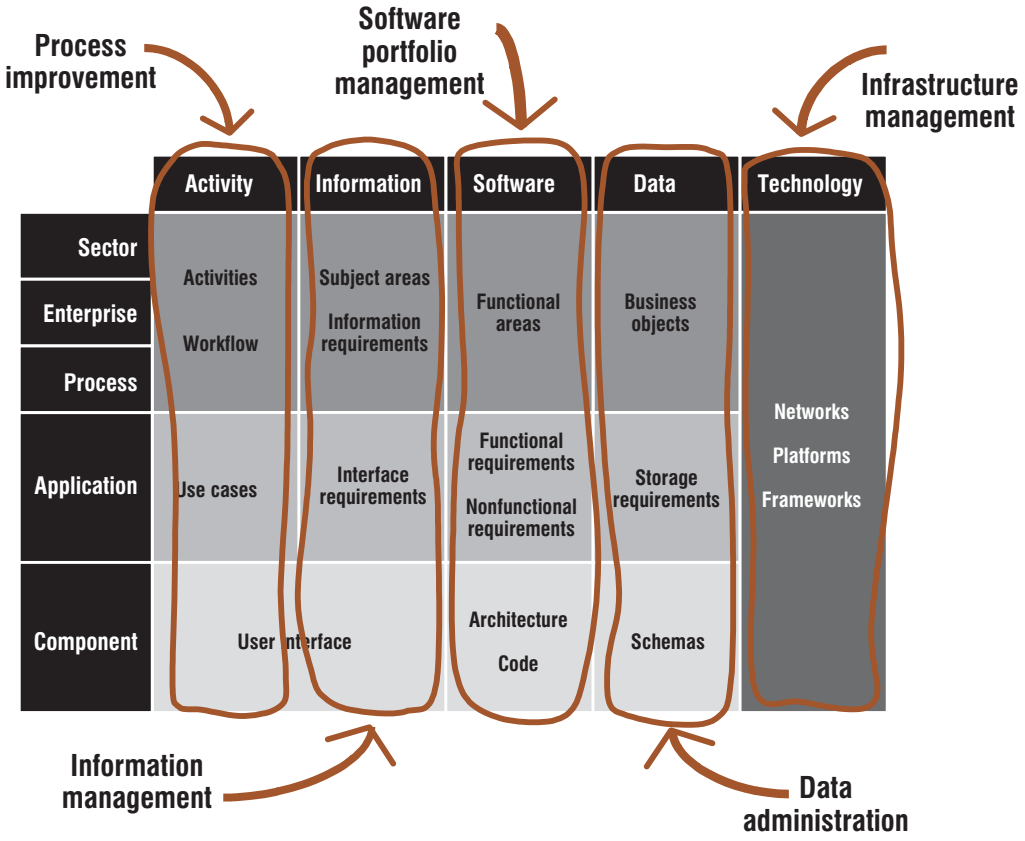

Figure 6 — Management discipline areas of architecture content.

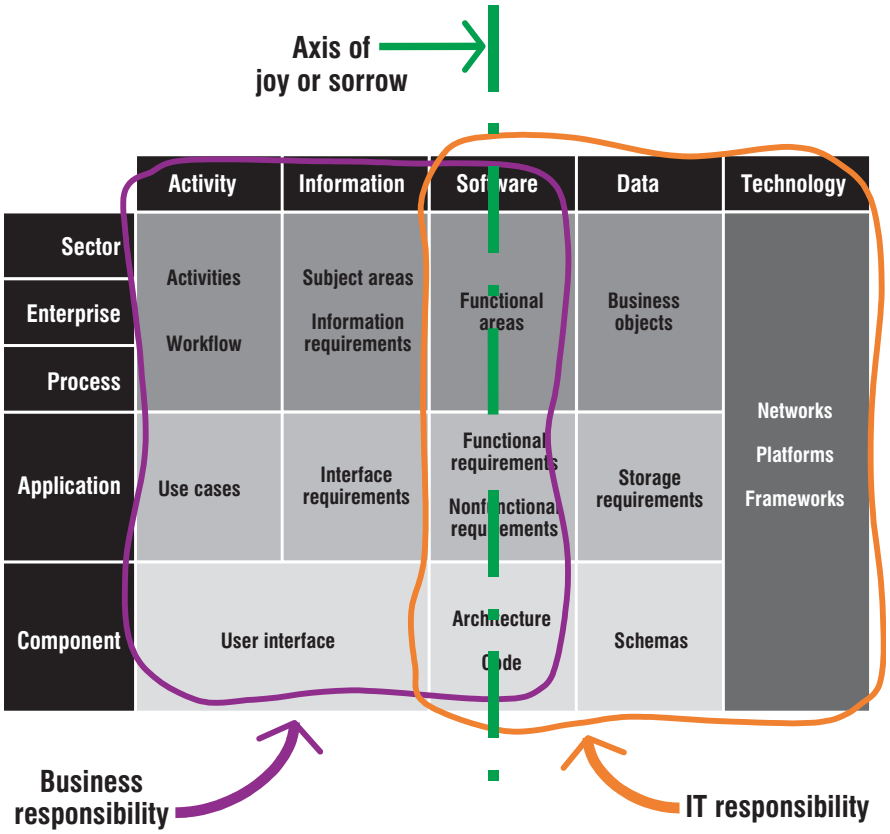

Figure 7 — Business and IT ownership.

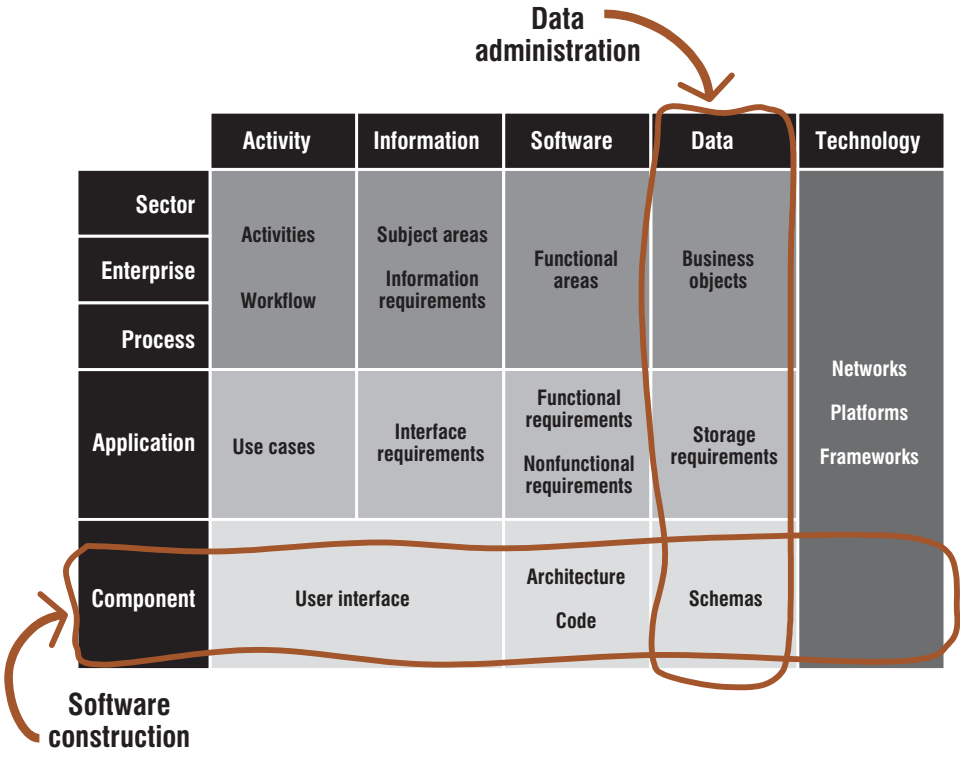

Figure 8 — The intersection of disciplines.

The XAF gives all parties a consistent framework for identifying and visually mapping out areas of responsibility and shows how one person's area of responsibility relates to another. In cases where responsibility is not clear, it provides a focal point for negotiation. The XAF is a common place to return to in the case of recurring disputes.

We know of architecture groups that work in splendid isolation. Their elaborate models never make one iota of difference to the business managers, business analysts, software developers, or IT infrastructure groups.

WHY EXTREME?

How can a framework that evolved out of a search for the middle path be called "extreme"? The XAF is extreme because it exaggerates the best aspects of other architecture frameworks:

- **The XAF is easy to describe.** The framework is based on a simple five-by-five matrix. The body of the matrix is populated with just 18 architectural elements. The framework can be presented on a single reference card. In contrast, many architecture frameworks are complex and difficult to describe. The documentation for TOGAF Version 8 runs to 313 pages — try getting people's attention with that!
- **The XAF encourages an agile approach to architectural work products.** Each of the architectural elements can be described using anything from a simple bullet-point list to a detailed UML model. In contrast, many architecture frameworks advocate the creation of a large number of elaborate and detailed models. The Zachman Framework identifies no less than 36 "primitive" models.
- **The XAF unifies a number of disparate project disciplines.** Each area is focused on a particular discipline but retains the context of its relationship to the other elements. In contrast, we know of architecture groups that work in splendid isolation. Their elaborate models never make one iota of difference to the business managers, business analysts, software developers, or IT infrastructure groups.
- **The XAF offers a simple, consistent view to the various parties involved in the management of enterprise resources.** This encourages shared understanding between disparate groups by presenting an area of "common ground" that everyone can understand. In contrast, some frameworks organize models according to various roles. This tends to encourage redundant descriptions of elements at different levels of detail. For example, the Zachman Framework answers the questions what, why, when, how, where, and who from the perspective of five different roles.

Enterprise architecture is beset by extremes. At one end of the spectrum, the would-be builders of corporate cathedrals will settle for nothing less than perfection. At the other, the builders of chaotic shantytowns grab the first solution that comes to hand. The suburban house, with its connotations of self-reliance, affordability, and pragmatism, is much more relevant to the modern enterprise than the abundance, privilege, and order of Reims Cathedral or the disenfranchisement, poverty, and chaos found in a shantytown. In dealing with both software architecture and enterprise architecture, the Extreme Architecture Framework offers a middle path for those businesses and IT groups that cannot afford the comprehensive nature of some published frameworks and wish to avoid the chaos of not having one at all.

REFERENCES

1. Beck, Kent. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 2000.

2. Brooks, Frederick P., Jr. *The Mythical Man-Month: Essays on Software Engineering*. 20th anniversary ed. Addison-Wesley Professional, 1995.

3. Fong, Elizabeth N., and Alan H. Goldfine. *Information Management Directions: The Integration Challenge*. National Institute for Standards and Technology, 1989.

4. Foote, Brian, and Joseph Yoder. "Big Ball of Mud." In *Pattern Languages of Program Design 4*, edited by Neil Harrison, Brian Foote, and Hans Rohnert. Addison-Wesley, 1999.

5. Hall, John, Keri Anderson Healy, and Ronald G. Ross (eds.). *The Business Motivation Model — Business Governance in a Volatile World*. Business Rules Group, September 2005 (www.businessrulesgroup.org/second_paper/BRG-BMM.pdf).

6. Hammer, M., and J. Champy. *Reengineering the Corporation: A Manifesto for Business Change*. HarperCollins, 2003.

7. Hofmeister, Christine, Robert Nord, and Dilip Soni. *Applied Software Architecture*. Addison-Wesley Professional, 1999.

8. Jacobson, Ivar. *Object-Oriented Software Engineering: A Use Case-Driven Approach*. Addison-Wesley Professional, 1992.

9. Kaplan, Robert S., and David P. Norton. *The Balanced Scorecard: Translating Strategy into Action*. Harvard Business School Press, 1996.

10. Kaplan, Robert S., and David P. Norton. *Strategy Maps: Converting Intangible Assets into Tangible Outcomes*. Harvard Business School Press, 2004.

11. MacDonald, Ean. E-mail communication, 5 October 2003.

12. Pressman, Roger. *Software Engineering: A Practitioner's Approach*. 6th ed. McGraw-Hill, 2005.

13. Schekkerman, Jaap. *How to Survive in the Jungle of Enterprise Architecture Frameworks*. Trafford Publishing, 2004.

14. The Open Group Architecture Framework (TOGAF) "Enterprise Edition" Version 8.1 (www.opengroup.org/architecture/togaf8-doc/arch/).

15. Thera, Piyadassi. *The Buddha: His Life and Teaching*. Buddhist Publication Society, 1982 (www.buddhanet.net/pdf_file/lifebuddha.pdf).

16. Thomsett, Rob. *Radical Project Management*. Prentice Hall PTR, 2002.

17. Zachman, John A. "A Framework for Information Systems Architecture." *IBM Systems Journal*, Vol. 26, No. 3, 1987.

Floris Gout was a late starter in information technology. After eight years as a professional dancer with the West Australian Ballet Company, he worked in engineering on large iron ore, petroleum, and gas projects in Western Australia and Malaysia until 1987.

Following those two exciting careers, Mr. Gout earned a bachelor's degree in information science and has since worked across industry sectors in a range of IT roles. His major interests lie in project management and modeling, and he has modeled systems at the enterprise and software application levels. Mr. Gout has developed a skill of reverse-engineering a database and assessing its alignment to business objectives and IT governance. His project management interest focuses on risk and rewards and demonstrating the alignment of an architectural model to the business case.

Mr. Gout can be contacted at floris@floris.com.au.

Phil Robinson is Lonsdale Systems' principal trainer and consultant. Mr. Robinson has worked in IT in a variety of roles since 1975. He has been involved in the planning, analysis, and implementation of a diverse range of business, scientific, and technical information systems. Mr. Robinson is an experienced workshop facilitator and trainer and has earned a reputation as a lucid and knowledgeable presenter. He has presented training courses for clients in Australia, Thailand, Hong Kong, Singapore, and Indonesia.

Mr. Robinson has authored numerous courses for industry and three university units. He has also published two books on programming Apple computers, which were subsequently translated into German and French. More recently, Mr. Robinson coauthored two award-winning articles describing an original organizational theory and was appointed to the editorial board of Work Study*, the UK journal in which they were published.*

Mr. Robinson has a degree in electrical engineering and has trained as a group work leader. He has lived and worked in Thailand and speaks a little ("nid noi") of the Thai language.

Mr. Robinson can be contacted at E-mail: lonsdale@ii.net; Web site: www.lonsdalesystems.com.