eXtreme Architecture Framework Revisited

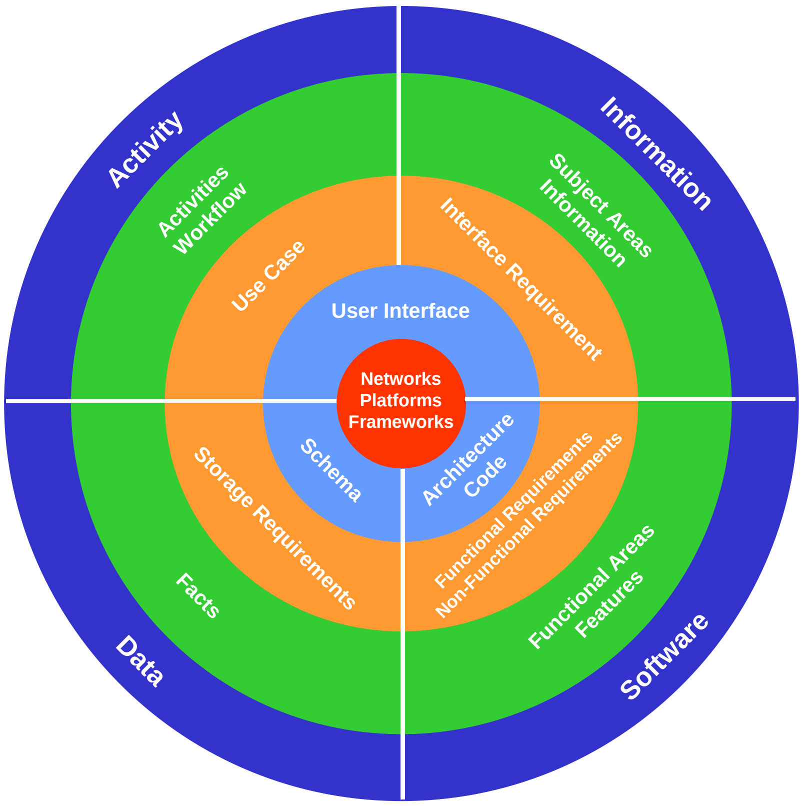
Phil Robinson
lonsdale@iinet.net.au

XAF and mandalas

During early iterations of the XAF we made extensive use of laminated reference cards that we would take along to meetings and use as a focal point for our discussions. We found that people responded well to the card and often pointed or gestured at particular areas of the framework to emphasise a point.

Each time we met to discuss the XAF we would have a copy of the reference card available that would quickly become the focal point of our discussion.

Appropriately, while trying to explain the benefits of the laminated reference card to an audience in Bangalore, the word "mandala" came into my head. Since our writing about the XAF already included a spiritual metaphor in the form of the Buddhist concept of following the "middle path", I was keen to explore this idea further.



The XAF presented as a circular "mandala"!

I was delighted when I looked up “mandala” in Wikipedia and found:

> *Mandala is a term used to refer to various objects. It is of Hindu origin, but is also used in other Dharmic religions, such as Buddhism…*
>
> *In practice, mandala has become a generic term for any plan, chart or geometric pattern that represents the cosmos metaphysically* ***or symbolically, a microcosm of the universe from the human perspective****…*
>
> ***In the various spiritual traditions, the mandala is frequently used as an object for focusing attention and as an aid to meditation.***

However, it seemed like most of the mandalas that I had seen were circular not rectangular, so I was inspired to create the circular version of the XAF which appears in the diagram above.

XAF and hardware

The participants for one of the courses I presented on the XAF included a number of hardware engineers. When I had finished describing the framework, one of them asked the inevitable question:

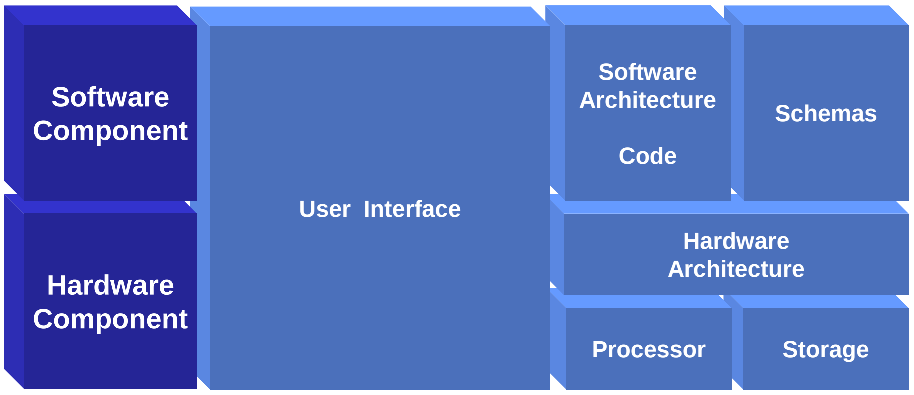
> *What about hardware components?*

My first reaction was to point to the *Platforms* and *Networks* elements in the *Technology* column and to try and convince them that hardware was covered by these two elements. I quickly realised that this was not going to work when another participant followed up with:

> *But what about a keyboard or display?*

Thinking on my feet and moving across to the whiteboard, I started to improvise! It was pretty easy to see that a new hardware component row was required. Once I had added that to the framework, the new architecture elements seemed fairly obvious as well.

The result is the reworking of the XAF shown below.



When I stood back from the whiteboard and invited discussion, it became obvious that the *Software* column would need to be renamed as it now included the hardware *Processor* element as well as half of the *Hardware Architecture* element.

We debated whether the new name should be *Behaviour* or *Automation* but didn't reach a final conclusion.

We also realised that it was necessary to add the prefix "software" to the existing *Architecture* element in order to be consistent and distinguish it from *Hardware Architecture*.

A final insight resulting from the "experiment" of adding hardware components to the framework was the possibility that the *Software Architecture* element should in fact span both the Software and Data columns. It seemed obvious that a hardware architecture should include both processors and storage devices. However, the requirement for a software architecture to include both code and data schemas is not so obvious.

XAF and complexity

During one of the courses that I presented on the XAF, I was asked the question:

> *Why do you think that the Technology column is so much better managed than the other columns?*

Being somewhat stumped for an immediate answer, I tossed the question back to the participants and invited a discussion. During the discussion, the following points emerged:

- The other four columns are more "conceptual" or "abstract" in nature. Possibly becoming more so as we move leftwards from the *Data* column to the *Activity* column.
- It is much easier to manage an architecture consisting of concrete elements such as routers, modems and server racks. Managing an architecture consisting of more abstract elements such as *Activities, Workflows* or *Subject Areas* is far more difficult.
- The fact that managing abstract elements is difficult does not negate the fact that they should be managed. However, it does raise the question of whether we have created enterprises that are simply too complex for us to manage properly.

These ideas seem to be supported by the generally unsuccessful attempts by enterprises to manage the *Data* column. Data Administration was "the next big thing" in the 1990s. However, the reality has seldom lived up to the promise. Looking back, most practitioners would agree that there has been little improvement in the management of enterprise data since the 1990s.

As we move leftward across the XAF, success stories become even more elusive. Few enterprises can reach agreement on the detail of their business processes let alone establish proper governance of their business process architecture.

XAF and the Zachman framework

The most striking similarity between the XAF and the Zachman framework is the fact that both frameworks are presented as a matrix. This is not a coincidence; the developers of the XAF were heavily influenced by both the strengths and weaknesses of the Zachman Framework.

Rows - roles vs. systems

The first five rows of the Zachnman Framework represent roles (planner, owner, designer, builder and sub-contractor) with the final row representing the enterprise. In contrast the rows of the XAF represent systems (human activity, application and component).

We believe that there are a number of advantages to basing an architecture framework on systems rather than roles:

- Unlike roles, systems can be more precisely defined by a boundary.
- The allocation of responsibilities to individual systems and the interfaces between them encourages a consideration of interoperability.
- Systems encourage an objective systems engineering approach to architecture while roles are by nature more subjective.
- Systems can be decomposed into components that map better to an object-oriented, component-based, or service-oriented style of architecture.

Columns - interrogatives vs. views

The six columns of the Zachman Framework represent the English language single-word interrogatives (what, how, where, who, when, why). In contrast, the columns of the XAF represent architectural views (activity, information, software, data and technology).

We believe that there are a number of advantages to basing an architecture framework on views rather than interrogatives:

- The English language provides part of the rationale behind the Zachman Framework. The elegance of the framework for speakers of languages other than English may not seem quite so obvious. For example some languages, such as French have a seventh single-word interrogative (combien) meaning “how much”.
- Views are a widely used architectural mechanism. For example, the *IEEE 1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems* defines an architectural view as:

  > *a representation of a whole system from the perspective of a related set of concerns.*

- Many enterprise architecture frameworks (including XAF but not the Zachman Framework) have adopted the architectural views (business, information, application, data, and technical) originally described in the NIST report *Information Management Directions: The Integration Challenge*. This means that these views are widely accepted and understood.

Cells - models vs. elements

One of the concepts underpinning the Zachman Framework is the periodic table used to classify chemical elements. The columns and first five rows of the framework provide a similar classification scheme for no less that thirty separate models of an enterprise. The sixth row of the framework classifies the actual implementation of the enterprise.

In contrast, the rows and columns of the XAF classify just nineteen “architectural elements”. These nineteen elements are intended to provide a minimalist answer to the questions:

> *Which elements of the enterprise do I need to be aware of and understand; and*
> *Which elements am I responsible for and need to manage?*

In other words, the XAF defines a minimalist enterprise architecture framework for governance of the enterprise.

We believe that there are a number of advantages to basing an architecture framework on a minimalist list of elements rather than models:

- Clearly, attempting to populate all of the Zachman Framework cells with the appropriate models is a gargantuan task for even a modest sized enterprise. The sheer scale of attempting this is sufficient to derail many enterprise architecture efforts.
- The XAF’s architecture elements offer more concrete guidance on precisely what is required in order to describe an enterprise. At the same time, it does not restrict how individual elements can be combined to create models of an enterprise as required.
- The Zachman Framework is intended to classify a number of independent models of an enterprise without defining the relationship between models in adjacent cells. In contrast, it is intended that the architecture elements of the XAF have some meaningful relationship with elements in adjacent cells.
- Each of the elements can be directly mapped to an element of the Unified Modelling Language (UML) standard thus providing the opportunity to standardise enterprise models and employ UML modelling tools.

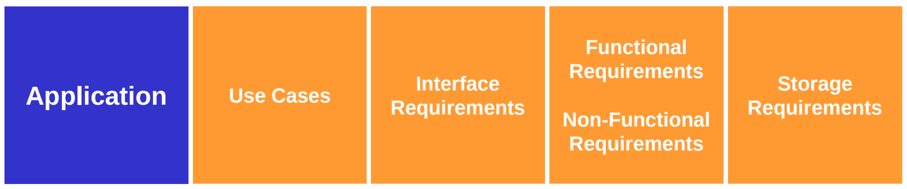
XAF and application architecture

An *Applications Architecture* is often a standard component of many enterprise architecture frameworks. For example, the TOGAF framework includes an *Applications Architecture*:

> *…this kind of architecture provides a blueprint for the individual application systems to be deployed, their interactions, and their relationships to the core business processes of the organization.*

However, it is not immediately obvious which of the XAF’s architectural elements should be used to describe an *Applications Architecture*.

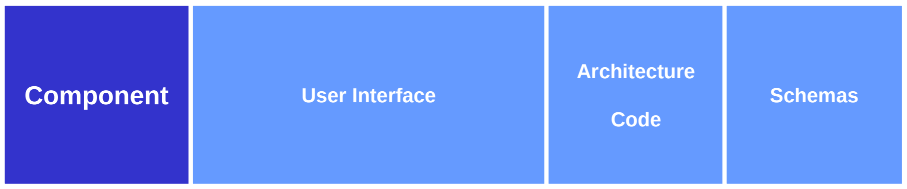
Recall that the XAF represents applications as systems having a clearly defined system boundary. The architecture elements that describe the application can be found inside the application's system boundary. These elements define the logical requirements for a single application.



The XAF architecture elements that describe an individual application

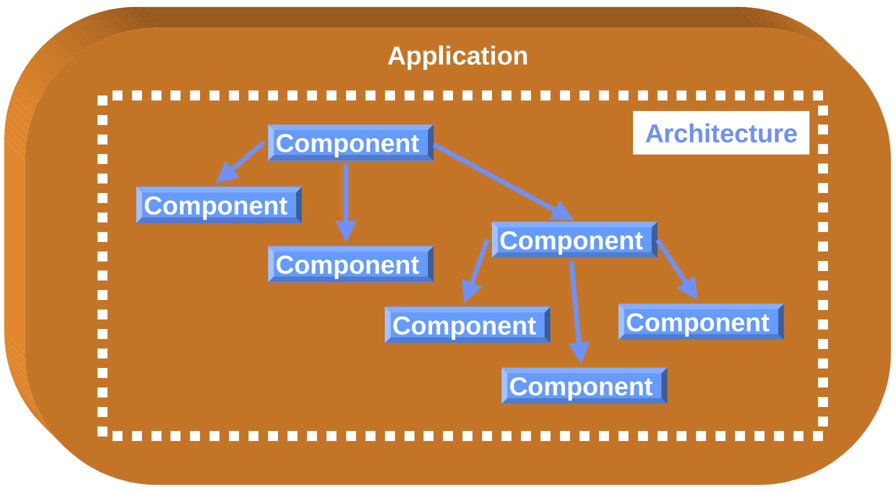
Applications can be decomposed into a number of individual components. In a similar manner to applications, components are represented in the XAF as systems having a clearly defined system boundary.

The architecture elements that describe the component can be found inside the application's system boundary. These elements define the physical construction of the component.



The XAF architecture elements that describe an individual component

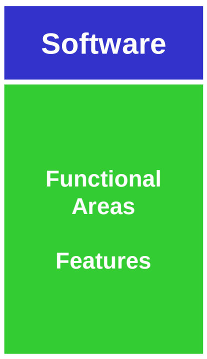
The *Architecture* element describes how individual components are composed into an *Application*.



Composition of individual *Components* into an *Application*.

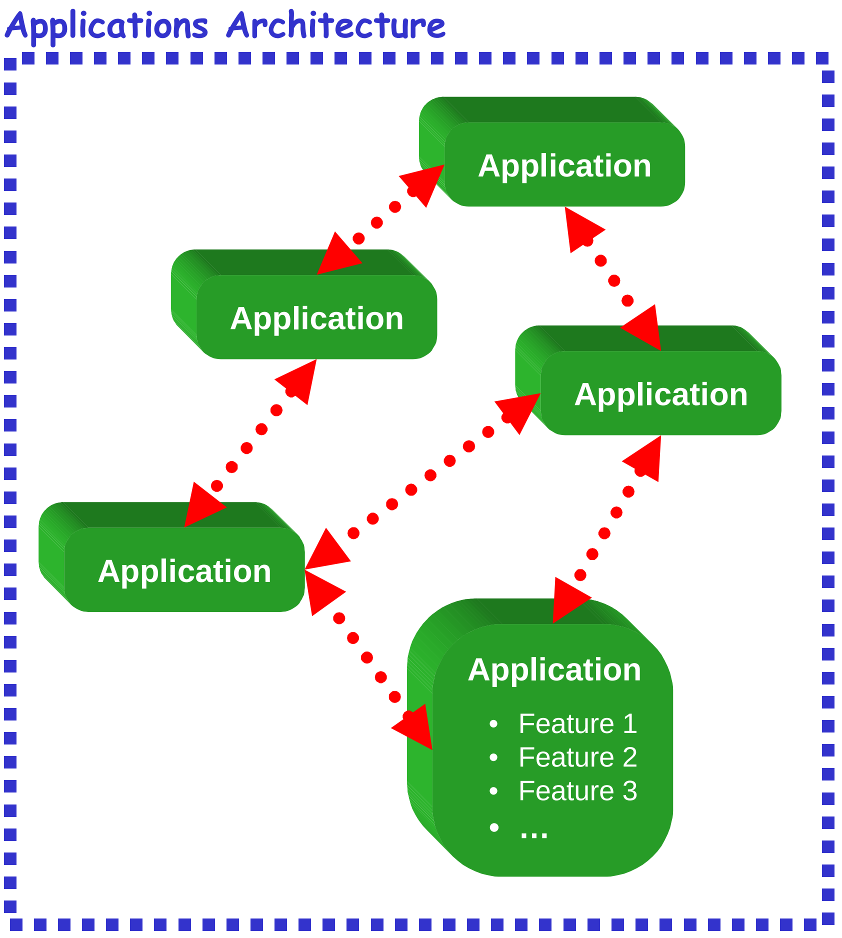
Since most enterprises have an extensive portfolio of applications, there is also a need to identify the individual applications and show how they are related to each other. Often this is required before the detailed requirements for each application have been defined.

For this reason the *Functional Area* element is used to define the *Applications Architecture*.



Functional Area architecture elements are groupings of *Features* that have a common purpose. An *Application* is a special case of a *Functional Area* that has a detailed *Architecture*, is implemented in program *Code* and is physically deployed.

The diagram below shows how an *Applications Architecture* can be described using *Functional Areas*. The *Feature* element can be used to provide a high-level overview of the application's requirements.



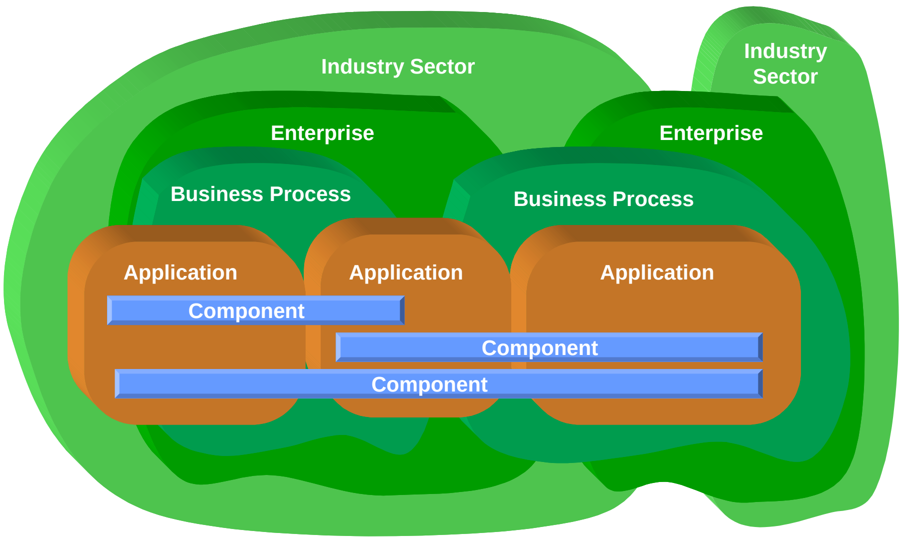
Using XAF elements to describe *Applications Architecture*

The XAF and service-oriented architecture (SOA)

SOAs are intended to permit the construction of ad-hoc applications by "orchestrating" (linking and sequencing services) a number of software services. The orchestration of services is guided by the details of the business processes that the application will support.

The hope for SAO is that the cost of orchestrating applications will become progressively cheaper as more applications are created. In theory, at some point all the software required to satisfy the requirements of a new application will already exist and so the cost of creating the application will actually be close to zero.

XAF is actually based on the concepts underpinning SOA as the human activity and software systems of an enterprise are viewed as a number of "independent" and "overlapping" systems as shown the diagram below.



XAF's view of systems as "independent" and "overlapping"

The benefit of adopting this point of view is that independent, overlapping systems are able to interact with each other in an unrestricted manner. Specifically, a software component can be designed to provide a service that can be used by one or more applications.

In the XAF services are described by the architectural elements of the components row and application "front-ends" by architectural elements of the application row.

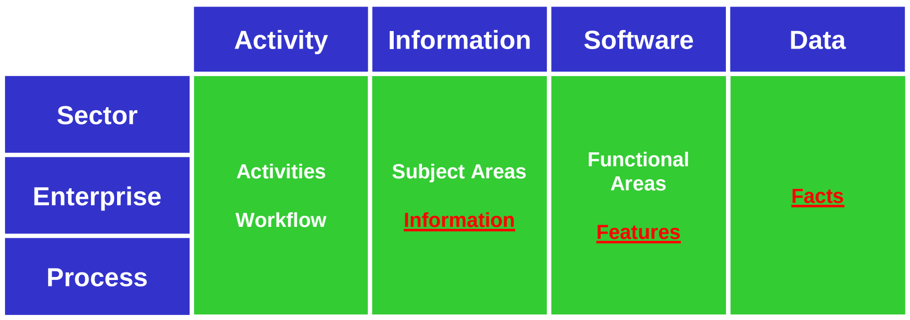
Components representing services have some specific characteristics:

- they are isolated, non-hierchical units of functionality;
- there are no "calls" or dependencies existing between them;
- they implement human recognisable functions such as making a booking or placing an order;

- their granularity is much larger in scope than traditional components or software classes (but, if the granularity is too large, it may limit opportunities for reuse); and
- they are described by metadata, such as WSDL, and invoked by protocols, such as SOAP.

Recent changes to XAF

There have been a number of recent changes to the XAF. All of these changes have been confined to the human activity systems row.



Changes to XAF

- The *Information Requirements* architecture element has been renamed to *Information*. It retains the same definition as the old element.
- The *Business Objects* architecture element has been replaced by a new element *Facts*. The definition of the *Facts* architecture element is:

  Definitions of concepts and the facts that describe and relate concepts. Types of concept include:
  - *events, transactions, time periods, agreements and contracts;*
  - *the roles of people, organisations, places and things;*
  - *the actual people, organisations, places and things; and*
  - *ways of classifying all of the above.*

- A new architectural element *Features* has been added. The definition of the *Features* architectural element is:

  An informal description of a Use Case; or software requirement (interface, functional, non-functional or storage requirement).