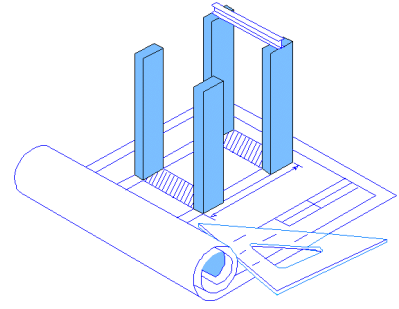


TASSC

technical paper



Software Architecture

Robert Armstrong, Gillian Adens

Tassc Limited
www.tassc-solutions.com

First Published: April 1999
Last Updated: January 2008

Copyright 1999-2010, Tassc Limited. All Rights Reserved.

This paper examines the purpose and intent of producing a software architecture, and reflects on its positive impact on software projects. We describe an approach for developing a software architecture that has been implemented and proved at a number of major organisations over a number of years.

What is Software Architecture?

*corporate-wide
infrastructure*

Software architecture provides a corporate-wide infrastructure for systems development. This gives a stable basis for long-term, medium to large-scale software development projects.

*links into
business change*

Software architecture has clear links into analysis of business processes and provides a mechanism for implementing the resulting business change initiatives. Being implementation focused, it also provides the rigour required for validating and fine-tuning the approach to business change.

*a framework for
projects*

Software architecture supplies a framework and reference point for individual development projects. It allows iterative development approaches, such as RUP (the Rational Unified Process), to contribute to strategic goals while maintaining focus on delivering business benefit in the short term. Commonality between projects is identified and can be exploited.

*provides
traceability from
business to
systems*

Software architecture shows how business-oriented models are implemented using a specific set of technologies. This provides traceability from requirements analysis through to detailed design and code, and allows technical issues to be addressed separately from business issues.

The Need for Software Architecture

strategic mobility

Strategic mobility is increasingly a demand of the environments in which many organisations operate. This demand is driven by:

- changes in the business practice of competitors
- the needs of third party sales channels or suppliers
- changes in customer behaviour
- mergers, acquisitions or take-overs
- changes to legislation or regulation
- increased competition
- a need to reduce time to market
- a need to exploit advances in technology
- the desire to limit the dependency on a single vendor or technology
- the introduction of new business practices

flexibility

Strategic mobility requires flexible, highly modular, component-based information systems that are developed cost effectively at minimum risk. Tassc consultants have been involved at all levels with many large organisations who are making a move from a more traditional mainframe technology base to a distributed client-server architecture.

legacy integration

Many of these organisations have been able to protect their existing investment in IS (Information Systems) by incorporating their legacy systems as an integral part of their server network. Rather than immediately replacing existing (and valuable) systems, they found that a slower, carefully planned migration is the most cost effective and low risk strategy.

Achieving Business Change

evolution rather than revolution

Tassc has considerable experience in helping companies plan and implement their transition to newer more flexible client-server technologies. We advocate evolution rather than revolution.

involve business analysts and end users

Our approach is to first understand the nature of the business and the role of IS, with a great deal of involvement from business analysts and end users. We can then begin to design and develop a corporate-wide infrastructure to support the needs of the whole business now and into the future.

integrate business and software architecture

In order to achieve business change, it necessary to model the business as it is now and how it will be in the future. These models and the techniques used to create them must be usable by the business for process and organisation redesign. They must also be usable by those who build information systems to support the business. That is, in order to achieve business change efficiently the software architecture and business architecture must be integrated.

use UML for business and software models

UML (the Unified Modeling Language) provides notations and diagrams that allow us to capture and analyse business process models, and subsequently document our decisions for the software architecture.

*follow best
practise*

Tassc promotes an approach to business modelling that is based on the latest thinking and extensive practice of implementing business change through information systems. The approach has the following features:

- easy to learn – the approach makes use of simple modelling techniques which can be taught very quickly
- easy to use – useful and simple models which allow in-depth analysis of the business can be developed and used very quickly
- robust – the models produced are based on the stable aspects of the business, and can be reused even after major business change
- practical – existing process, function and data models can be reused
- a set of integrated models with simple interrelationships is produced
- business and systems people speak the same language, the same models are used for business redesign and systems development
- duplication in business procedures is identified early
- provides a repository of business knowledge
- efficient – the business models feed seamlessly into solution development and their usefulness continues beyond the initial analysis and redesign
- manages complexity and scale
- implementation focused – the models are produced to a depth where the cost and viability of implementation can be judged

Whether building your own business model or validating an off-the-shelf generic business model, this approach to modelling the business provides understanding and enables change to be driven effectively by feeding seamlessly into software development.

Defining the Software Architecture

*highest level of
abstraction*

Software architecture defines concepts in the system at the highest level of abstraction. These abstractions form fundamental design components that encapsulate a clearly defined and stable set of responsibilities.

*identify key
business areas
as components*

The goal of the architecture is to identify subsystems and key business classes. This partitions the systems for the entire business into a number of separate components, each of which has responsibility for a major system feature, for example, Customer Service. For the architecture to fulfil its goals of driving both business change and system development, the components must be based on stable business areas and encapsulate core business concepts.

*start with high
level functional
decomposition*

A traditional functional decomposition is a useful start point for identifying the business areas that partition the model. This is generally a precursor to an object-oriented design in which the components logically partition data as well as functionality.

*move to an
object-oriented
model of
components and
services*

Each component comprises a set of classes whose instances co-operate to support the services offered by the component. Portions of the entire system (objects residing in several components) are orchestrated to perform the use cases supported by the system. This is particularly true of cross-functional processes, which utilise and combine the services provided by a number of components.

Developing the Software Architecture

requires significant design effort

For strategic projects, developing the software architecture is of fundamental importance and therefore requires significant design effort. The architectural framework ought to be general purpose and resilient to all but major changes to requirements.

partition component responsibilities

Each application will typically require the services of a number of components. Therefore, it is preferable to build an overall analysis model, which partitions responsibilities to appropriate components, and to design each application as a client of these components. This approach necessitates software architecture at this stage to avoid future rework as the system evolves. We advise that organisations start to produce the overall software architecture in parallel with the business object modelling required by specific projects, since it is not normally practical to suspend projects while this process takes place.

identify opportunities for re-use

Major applications typically focus on a small number of components. This provides an opportunity to build large elements of reusable business-oriented components as part of the normal project development process. This obviously requires planning, an understanding of the overall framework for software development and a corporate commitment to build for reuse. However, by using currently scheduled projects, the benefits of reuse emerge earlier and any overhead of building for reuse is minimised.

integrate legacy components

In many large organisations, a key requirement is to gain full value from existing systems regardless of their implementation technology. A component strategy, based on developing components incrementally through projects, must take into account how existing systems are used and provide a method for gaining maximum value from them. Existing systems and legacy technologies can then become a valuable part of the component strategy.

component interfaces should be natural, logical and cohesive

A primary goal of a business component is to provide a single logical set of related business classes. In this way, a coherent structure of inter-related function and data is developed which can support multiple applications in a consistent and robust manner. In our experience, if the components are designed in isolation, a biased design is produced which cannot accommodate new business requirements. Functionality is incorporated into one component that would more naturally be the responsibility of other components. Component interfaces should be natural, logical and cohesive.

Using Generic Software Architectures

a shortcut

A number of 'off the shelf' software architectures which represent the activities of particular types of business area are available. For example, it is possible to buy models of insurance companies, utilities and supermarkets. These models are sold as being a shortcut to providing full understanding of the business and to speed up the process of business transformation.

criteria for success

The advantage of buying a model rather than developing one from scratch is undeniable provided that a number of criteria are met:

- there must be a good closeness of fit between the generic model and the organisation
- the model should be up-to-date
- the supplier should have broad and deep experience of the business area which is captured in the model
- the models should capture best industry practice
- the users should be comfortable with the model supplier learning from the business and consequently enhancing the generic model

- the models should be technology independent
- the models should not be biased towards the selling of software packages
- it should be easy to build solutions independently of the model supplier
- it should allow business-oriented components to be identified and their inter-relations and dependencies understood
- it should be easy to adapt the model to the special circumstances of the business
- the model should be easily extendible to provide differentiation and innovation

The Approach

the team

We would propose to establish and work with a core architecture team comprising a minimum of a business architect and a senior business analyst. The size of the team depends on how quickly results are required, the level of skill that the organisation can provide and how much work from previous analysis exercises can be reused.

In most circumstances we would recommend starting with a small team and scaling up later if necessary. We would also advise that a review team be established (with representation from the various business departments) and be available to the architecture team to provide an independent assessment.

development method

A combination of three approaches is used:

- top-down decomposition
- bottom-up composition
- technical architecture mapping

top-down

Top-down decomposition partitions the business into the major functional areas with the dependencies between each area modelled. Much of this activity is likely to have been addressed already in previous business modelling exercises.

bottom-up

Bottom-up composition aims to identify all candidate business classes and aggregate them into natural clusters according to their roles and responsibilities. The top-down view is reconciled with the bottom-up view.

map to technical architecture

Client-server architectures are modelled by allocating different types of business and technical services to different parts of the physical infrastructure and defining the methods of communication among these elements. The classes identified in the models are characterised into the different types of business services and mapped into the technical architecture.

class diagrams provide a static view of the system

The software architecture defines the component partitioning, and states responsibilities and relationships. Initially, high-level class diagrams, which identify key business classes in each component, will be developed. This provides an overall static view of the system.

activity diagrams and use case models describe business requirements

In addition to the class models, the key business processes for the core business of the organisation will be identified and modelled using activity diagrams.

High-level use case models, which represent the key steps in these processes, will be developed.

These processes and use cases will, in general, cross component boundaries.

interaction diagrams provide a dynamic view of the system

The activities and use cases discovered in the business process modelling exercise provide a dynamic perspective on the system. Interactions between key business classes within each component are driven by use cases. The static and dynamic models are inter-dependent and are best developed in parallel.

a framework from static, dynamic, physical and organisational perspectives

The software architecture provides a framework that describes the business from four perspectives:

- static
- dynamic
- physical (client-server model)
- organisational (business process view)

These models allow software development to be co-ordinated with business improvement.