

Introducing Service Architecture Review Method: exploring the application of software architecture evaluation methods to public service reform

Simon Field *

Glamorgan Business School

University of Glamorgan

simon.field@lefert.com

Abstract

The drive to reform public services, often motivated by a desire to improve efficiency, quality, choice or citizen engagement, is frequently hampered by the difficulty of predicting the impact of new service designs. Changes in service design may affect multiple stakeholders, and the impact of change may vary across multiple service attributes. It is often therefore difficult to determine whether one service design is preferable to another. This paper presents a review method derived from those adopted by software architects to evaluate competing software architectures. It suggests that the domain of service design shares some significant characteristics with that of software solution architecture, and proposes the adaptation and application of evaluation and review methods that have proved successful in the software solution architecture domain.

* the author is a DBA student at Glamorgan Business School and Chief Technology Officer at the Office for National Statistics

For submission to the Public Policy and Administration / SAGE award

1 Introduction

This paper proposes the application of evaluation methods that have been developed for assessing computer software architectures to the domain of public service design. It takes as its starting point a definition of software architecture, and examines established methods and standards for describing software quality and evaluating software architectures.

Section 3 considers the domain of service evaluation, with a particular focus on public service design. It considers the complexity resulting from the multi-dimensional nature of impacts caused by changes to services, further compounded by the different perspectives of multiple stakeholders. It also examines the discipline of service design, highlighting the parallels between service design and software systems design that have been identified by numerous commentators.

Section 4 describes a proposed Service Architecture Review Method and accompanying Service Quality Attribute model, and the paper concludes with some suggestions regarding the future areas of research that may be opened up as a consequence of this work.

2 Software Architecture

The Software Engineering Institute at Carnegie Mellon University (SEI) has collected over sixty definitions of “Software Architecture”, publishing them under “Modern”, “Classic”, “Bibliographic” and “Community” categories on its web site. The majority of these definitions comes from the 1990's, the period when the concept of software architecture became widely discussed among the computer science community.

What is software architecture?

A seminal work from this period is *Software Architecture in Practice* by Len Bass, Paul Clements and Rick Kazman (Bass et al. 2003), the first edition of which was published in 1998. It defines Software Architecture as follows:

“The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationship among them.”

Whilst there is clearly not a single form of words that is universally accepted, this definition is widely quoted in associated literature, and lies towards the centre of the set of definitions collected by the SEI.

The authors highlight a number of implications of this definition. An architecture is an abstraction, defining a set of elements, but suppressing those details of the elements that do not relate to how they use, are used by, or interact with, other elements. The definition also implies that systems comprise more than one structure with critical relations between them. Following on from this is the implication that all systems have an architecture, and that the external behaviour of the elements that comprise a system are part of that system's architecture.

The growth in complexity of software systems over time has led to a recognition of the importance of abstraction as an aid to communication among the stakeholders of a system, especially during its specification and construction. A representation of the architecture becomes a *lingua franca* that all stakeholders can speak and understand, as well as serving as a technical “blueprint” for the system that is to be built, modified or analysed.

This growth has also led to increased interest in the relationship between the architecture of a system and the behaviour of that system. Since the architecture encapsulates the external behaviour of the components, it is a combination of those behaviours that will make up the external behaviour of the resulting system. It follows that different architectures can result in different behaviours that might be closer to, or further away from, the set of behaviours most desired by the stakeholders.

Software quality

The *IEEE Standard for a Software Quality Metrics Methodology* describes software quality as “the degree to which software possesses a desired combination of attributes (e.g. reliability, interoperability)” (Software Engineering Standards Committee of the IEEE Computer Society 1998). *ISO 9126* (International Organization for Standardization 1991) is an international standard for evaluating software quality. Part one of the standard (ISO 9126-1) sets out a quality model which classifies software quality in a structured set of characteristics and sub-characteristics (see Figure 1 below).

Characteristic	Sub-characteristic
Functionality	Suitability
	Accuracy
	Interoperability
	Compliance
	Security
Reliability	Maturity
	Recoverability
	Fault Tolerance
Usability	Learnability
	Understandability
	Operability
Efficiency	Time Behaviour
	Resource Behaviour
Maintainability	Stability
	Analyzability
	Changeability
	Testability
Portability	Installability
	Replaceability
	Adaptability
	Conformance

Figure 1: ISO 9126-1 Software Quality Model

The terms “attribute” and “characteristic” would appear to be interchangeable in this context, since it is clear that both the IEEE standard and the ISO standard are referring to the same concepts. Many more quality attributes have been proposed by others, with a catalogue of over sixty being listed in Wikipedia (wikipedia.org 2009)

In *Quality Attributes* (Barbacci et al. 1995) a technical report from the Software Engineering Institute at Carnegie Mellon University, the authors introduce a generic taxonomy of software quality attributes, and propose “an attribute-based methodology for evaluating software architectures” which involves analysing the trade-off between quality attributes offered by different possible software architectures.

Software evaluation

This proposal led to the development and further refinement of a number of software evaluation methods (Kazman et al. 1994),(Kazman et al. 2000),(Clements et al. 2001) founded on the hypothesis that the differences between alternative architectures for a system can be explored by relating them to the desired quality attributes of the system. The most widely used of these now is the *Architecture Trade-Off Analysis Method (ATAM)* (Kazman et al. 2000). The value of assessing different architectures lies in the high cost of changing a system once it has been developed. If a system fails to exhibit appropriate behaviour in a key quality attribute it is likely that an inappropriate software architecture has been selected. Correcting such a fault is often difficult, involving fundamental re-engineering of the solution.

Modern software development methods, such as *Rational Unified Process* (Jacobson et al. 1999) and *DSDM Atern* (DSDM Consortium 2008), place the development of the architecture of a system early in the life-cycle, prior to the more labour intensive activity of developing the software. By conducting an evaluation of possible architectures at this early stage, a project is able to ensure that the most suitable architecture is chosen before most of the resources are committed to the project, increasing the chances of a successful project, and maximising the chances that the system will satisfy the wishes of the system's stakeholders.

ATAM involves an intensive review by a team of stakeholders, which might include its developers, the system's owner, some users, and those who will have responsibility for running, operating and maintaining the system. The review team collectively develop a set of scenarios that are placed in the context of the key quality attributes by creating a utility tree (where each leaf node is a scenario, and the branch nodes are quality attributes). The tree is then analysed for each architectural approach that is under consideration, uncovering risks, sensitivity points and trade-off points in the tree and deriving a view of the overall utility of each approach. A major benefit of this evaluation method is that the assessment of different architectural approaches is directly related to those quality attributes that are most important in the eyes of the stakeholders for the system under consideration. It reflects the "multi-attribute" nature of a system's behaviour and exposes the inevitable trade-offs between the different attributes for any one solution.

The Office for National Statistics has developed a variation of this method to conduct its

own architecture reviews (Field 2009) (available at <http://www.unece.org/stats/documents/2009.05.msis.htm>) which evaluates the risk of failing to reach desired goals for quality attributes instead of attempting to measure degrees of utility. The effect is similar, in that it exposes trade-offs between attributes, but ones expressed in terms of risk rather than utility.

The software architecture evaluation methods described above can be characterised as “pre-implementation” evaluations. They evaluate a design before that design has been implemented, and are used to select the most suitable design with the aim of reducing the risk that the eventual implementation fails to satisfy the expectations of its stakeholders.

3 Public Services

Service evaluation

The complexity of evaluating public service reform has been the subject of much study, and it is beyond the scope of this paper to offer a comprehensive review of the relevant literature. However, it is interesting to note that a common theme that emerges from much of the literature is the complexity that arises from the multi-dimensional impact of making a change to a service.

In *Evaluating public management reforms* (Boyne et al. 2003, p.14), the authors highlight the difficulties of evaluating public services and the effects of change. These difficulties relate to the multi-dimensional nature of the criteria, and the authors point out that this is a problem shared with, and stemming from, the difficulty of evaluating organisational performance (Connolly et al. 1980). This is further complicated by the perspectives that different stakeholders will bring.

The authors go on to consider two criteria that have been proposed by proponents of public choice reform: “efficiency” and “responsiveness”. These are each shown to be multi-dimensional with, for example, cost, quantity and quality all being potential dimensions of “efficiency”, and “responsiveness” being measurable from a variety of stakeholder perspectives (Boyne et al. 2003, pp.15-23). They identify a third multi-dimensional criterion for evaluating public service reforms, “equity”.

In *Excellence and Fairness* (Cabinet Office Strategy Unit 2008, p.12), a paper that sets out the UK Government's approach to improving public services, four criteria are proposed for

evaluating the extent to which public services can be considered “world class”: “delivering excellent outcomes”, “offering personalised approaches”, “being fair and equitable” and “offering good value for money”. No formal definition is offered, and whilst these criteria are expressed differently from the three mentioned in the previous paragraph, the examples given in the paper show that they are similar multi-dimensional criteria, and that they share many of the same constituent dimensions.

Evert Vedung (Vedung 1999, p.36) proposes a taxonomy of evaluation models that clearly illustrates the multi-dimensional nature of the problem facing public service evaluators. Among its eleven evaluation models are models for goals, side-effects, stakeholder concerns, client concerns, productivity and efficiency measures.

In addition to the multi-dimensional criteria that might be used to judge a service reform, the service will have multiple stakeholders, each of whom may have a distinct perspective and different objectives for that service.

An example public service stakeholder model can be found in *Beyond Boundaries: Citizen-Centred Local Services for Wales* (Welsh Assembly Government 2006, p.4) (commonly known as “the Beecham review”), which examines the Citizen Model advocated in the Welsh Assembly Government's vision for public services, *Making the Connections* (Welsh Assembly Government 2004). It includes the diagram given in Figure 2 below, representing the relationship of services with the public.



Figure 2: The relationship of services with the public

Taken together with those involved in commissioning, designing and delivering services (only some of which are reflected in the above diagram), this is a starting point for creating a more complete and commonly accepted view of public service stakeholders. It is essentially the same set considered in *Evaluating public management reforms* (Boyne et al. 2003, pp.21-23).

The evaluation of a service reform must therefore explore multiple dimensional impacts from the perspectives of multiple stakeholders.

In the context of services, the term “evaluation” is controversial when applied to an examination of a service *before* its implementation. In *Public Policy and Program Evaluation*, the author insists that “evaluation is retrospective”, and that “prospective appraisals (i.e., scrutinies of courses of action considered but not yet adopted even as prototypes), are not included in my definition”(Vedung 1999, p.7). He argues that to include prospective (*ex-ante*) assessments in evaluation would be to allow the concept of evaluation to “become too diluted”.

However, he does acknowledge that some leading theorists argue that prospective assessment (*ex ante* assessment) does belong to evaluation (Vedung 1999, p.7). Among them, Rossi, Lipsey and Freeman adopt a broader definition of “program evaluation”. They include the design of the programme among the five domains that may be assessed as part of an evaluation (Rossi et al. 2003, p.29).

Service design

Among the earliest publications to recognise service design as a distinct activity was *How to Design a Service* by G. Lynn Shostack (Shostack 1982). In it, the author characterises services as “processes” and distinguishes services from products as existing only in time (whereas products exist in both time and space), and as being unable to be possessed. He goes on to apply the analogy of molecular modelling to describing complex entities that may be made up of multiple services and products. The concept of a *Service Blueprint* is introduced, essentially a process model representing the service to be delivered. The author highlights the relationship between services and computer software systems:

“Since a service is basically a process, service blueprinting rests, as it must, on systems that have been developed to deal with processes, acts and flows.

Three systems are relevant: time/motion or methods engineering; PERT project programming; and computer systems and software design.” (Shostack 1982, p.56)

and later:

“what happens in a computer is often analogous to what must happen in order for a service to be successfully rendered.” (Shostack 1982, p.57)

More recent development of tools and methods for service design has seen the application of use case modelling, first developed to support systems design in 1986 by Ivar Jacobson (Morelli 2002).

The link between software systems and services goes beyond the fact that services often depend upon systems for their successful delivery. Both services and systems are composed of components that relate to each other, and are performed in time. As Shostack identified, delivery of services and execution of software systems both involve performing processes.

In the light of this conclusion, it is worth re-examining the definition of software architecture to see whether it might be amended to apply equally to services:

“The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationship among them.”

Substituting the words *service* or *process* for the words *software*, *program* and *computing system* results in a definition that is worthy of closer examination:

“The architecture of a service is the structure or structures of the service, which comprise process components, the externally visible properties of those components, and the relationship among them.”

The implications highlighted by the Software Engineering Institute in the context of software architecture would appear to be equally valid in the context of Service Architecture: the architecture is an abstraction defining a set of elements; services comprise more than one element with critical relations between them. These are

conclusions already reached by Shostack, Morelli and other contributors to the field of Service Design (Tassi 2009) (School of Architecture and Design at Aalborg University 2009). We may therefore reasonably reach the same conclusion for Services that Bass et al. reached for software systems: *that all services have an architecture*.

In the same way that well articulated systems architectures perform a valuable role in the development and maintenance life-cycles of software systems, so Service Architectures can fulfil the same role for services.

4 Service Architecture Reviews

It is beyond the scope of this paper to explore all the potential benefits of considering an architectural perspective when designing services, but one of those worthy of more detailed consideration is the possibility of applying the architecture review methods developed in the context of software systems, as outlined in Section 2 above, to Service Architectures.

Service quality attributes

Given the similarities between services and software systems and the methods applicable to their development, it is perhaps not surprising to see strong similarities between the quality attributes typically considered in software evaluation (described in Section 2) and the dimensions, perspectives and attributes that are typically considered in the field of public service evaluation (described in Section 3). A re-reading of the ISO standard for software quality (International Organization for Standardization 1991), the model of which appears in Figure 1 above, with the context of services instead of software systems in mind, supports this. Of the dimensions or attributes discussed by the various service evaluation authorities cited in Section 3 above, it is perhaps only *equity* (Boyne et al. 2003) that cannot easily be classified within the ISO 9126-1 model. The authors go on to conclude that equity is itself a composite attribute, and cite the conceptual framework of Le Grand (Le Grand 1982) as a means of breaking it down to its constituent attributes. If any of these are desired outcomes of a new service, then it is suggested that these could be classified within the Suitability sub-characteristic of the ISO 9126-1 model.

In the same way that a system can be developed to a number of different architectural approaches, each of which will offer a different set of trade-offs among the quality

attributes, so, it is suggested, can a service be delivered according to a number of different service architectures, similarly offering different trade-offs among the service quality attributes. A benefit of adopting an analysis and evaluation of architectural approaches to delivering a new service prior to its implementation is that stakeholders can assess alternative service solutions in terms of the quality attributes and outcomes they most highly value. The use of such an evaluation approach *ex-ante* will create a benchmark of expectations that can form the basis for any *post implementation* evaluation, while also improving the alignment of the service design to the intended programme outcomes (hopefully increasing the chances of a successful programme).

Service Architecture Review Method

It is proposed that a Service Architecture Review Method can be adopted that applies either of the methods outlined by SEI (*ATAM*) (Kazman et al. 2000) and ONS (Field 2009), using ISO 9126-1 as the basis for an accompanying service quality attribute model. The language of ISO 9126-1 is perhaps a little systems-oriented, but it should be noted that both review methods cited above involve mapping the generic quality attribute model to project-specific scenarios (in the case of *ATAM*) or requirements (in the ONS case). In each case, these will be expressed in the business language with which project stakeholders should be fully conversant.

One further distinction between the two methods is the use of utility in the case of *ATAM* and risk in the ONS case. It is suggested that service designers, especially those engaged in the design of public services, might be more conversant with the language of risk, and that a solution that draws attention to the risk of failure to achieve desired outcomes may be more appropriate in the case of public services, many of which involve delivery of critical, even life-saving, services. The proposal is therefore that the method adopted by ONS for systems architecture reviews can form a Service Architecture Review Method in conjunction with ISO 9126-1 as a basis for associated Service Quality Attributes.

The method involves a collaborative effort from a team of stakeholders and knowledgeable peers. These should be drawn from the stakeholder roles shown in Figure 2 above, together with those responsible for commissioning, developing, owning and running the service, plus some peer expertise if it can be identified. There are four phases to the

process:

1. Completion of Matrix 1
2. Workshop Preparation
3. The Review Workshop
4. Completion of the Review Report

Relating requirements to quality attributes

Matrix 1 is a cross-reference between the Service Quality Attributes and the service business requirements. An early phase of the service design method will involve the documentation and prioritisation of business requirements, and as soon as these are available in draft form they should be classified according to the relevant Service Quality Attributes, selected from the ISO 9126-1 model.

The matrix is simple, with the cells being marked to indicate a connection or a strong connection between individual requirements and quality attributes. The purpose of this exercise is to ensure that all the relevant quality attributes are represented by at least one requirement. The completion of the matrix need not involve all stakeholders, but would typically be accomplished in a meeting including the service owner, programme manager and programme business analyst or service designer. One reason for completing this matrix in a separate, early, exercise, is that it can lead to the identification of gaps in the set of business requirements that will need further analysis work to fill.

The review workshop

The dominant activity in a review is the review workshop. This should have an appointed chair or leader, for whom facilitation skills are more important than domain knowledge. Since the workshop is likely to involve between twelve and eighteen stakeholders for a whole day, preparation for the workshop is an important activity to ensure that meeting logistics and relevant papers or presentation material are prepared in advance.

Part one of the workshop involves introductions, a description of the business problem or goals of the new service and a presentation of the competing solution approaches. It is assumed that a number of solution approaches, based on different service architectures,

have been developed to a sufficient level to allow them to be described, and their strengths and weaknesses to be assessed.

Part two of the workshop involves the collaboration of all stakeholders in conducting a risk trade-off analysis of the competing solution approaches. This is accomplished by collective completion of a further matrix, this time with the business requirements from Matrix 1 as the rows, and the competing solution approaches as columns. The cells in the matrix are used to record an agreed numeric representation of risk.

Each requirement is discussed, considering the context of each solution approach in turn. The aim of the discussion is to arrive at an agreed representation of the risk that the requirement under consideration will not be satisfied by that particular solution approach. The ONS risk assessment method involves breaking risk down into “likelihood” and “impact”, and treating the product of these two values as the overall risk. This is just one approach; what is important is the arrival at an agreed quantifiable measure of estimated risk.

By the end of this analysis, the review team will have completed all of the cells in the matrix. Examination of the matrix will reveal the trade-off between requirements that each solution option represents. Colour-coding different levels of risk in the matrix can help highlight these trade-offs.

Experience of applying this method in the context of software systems has shown that the most valuable outcome from the process of completing the matrix is often the list of issues, actions and mitigations that are identified and noted during the lengthy discussions rather than the content of the matrix itself. Another key benefit of the process is the generation of consensus among the team concerning where the greatest risk lie, and which requirements are the most important. It is not uncommon that the architecture review workshop is the first time that this broad group of stakeholders has met together on the same day in one location.

The ONS method also includes completion of a third matrix to assess architectural alignment to corporate architectural principles. The use of architectural principles for enterprise architecture is becoming more common, and where a set of common principles have been adopted, this assessment can give a further perspective on the relative merits of the competing solution approaches.

The architecture review report

Following the workshop, the chair of the workshop prepares the architecture review report documenting recommendations, issues and the outcome of the matrix analysis for each solution. This report would typically be presented to the programme board to guide the decision on which solution approach will be adopted for the new service.

5 Future Work

This paper has set out the case for applying software architecture review methods to service designs, based on an analysis of the similarities between software systems and services. The proposed Service Architecture Review Method has been described, and adoption of an accompanying service quality attribute model, which was originally developed for software quality, has been recommended. Whilst the ONS review method has been applied to evaluations of systems for several years, and ATAM, upon which the ONS method is based, has been in widespread use for over ten years, neither has been seriously applied to the domain of service design (as far as the author is aware).

The next stage of this research project is the application of the Service Architecture Review Method to a number of public service design projects, and the refinement of both the method and the service quality attribute model in the light of that experience.

As indicated in Section 4 above, there is also scope to explore the impact of conducting rigorous service architecture reviews prior to the implementation of a new or changed service on the ability of a review team to evaluate the resulting service post-implementation.

Architecture reviews are just one of the tools and methods that have been evolved over the past ten years following the development of software architecture as a distinct discipline. IT architecture has already given rise to the discipline of Enterprise Architecture, whose reach includes service and business function architecture. However, there may be other “crossover” solutions that have been developed for software architecture but whose applicability to services and service architecture has yet to be fully understood. It is beyond the scope of the current project to explore these in any detail, but it is suggested that this may prove to be a fruitful avenue of exploration for future research projects.

References

- Barbacci, M. et al., 1995. *Quality Attributes*, Software Engineering Institute, Carnegie Mellon University.
- Bass, L., Clements, P. & Kazman, R., 2003. *Software Architecture in Practice (2nd Edition)* 2nd ed., Addison-Wesley Professional.
- Boyne, G. et al., 2003. *Evaluating Public Management Reforms: Principles and Practice*, Open University Press.
- Cabinet Office Strategy Unit, 2008. Excellence and Fairness: Achieving world class public services. Available at: http://www.cabinetoffice.gov.uk/strategy/publications/excellence_and_fairness/report/html.aspx [Accessed January 12, 2009].
- Clements, P., Kazman, R. & Klein, M., 2001. *Evaluating Software Architectures: Methods and Case Studies*, Addison Wesley.
- Connolly, T., Conlon, E.J. & Deutsch, S.J., 1980. Organizational Effectiveness: A Multiple-Constituency Approach. *Academy of Management Review*, 5(2), 211-218.
- DSDM Consortium, 2008. *DSDM Atern The Handbook* 1st ed., DSDM Consortium.
- Field, S., 2009. Evaluating architectural options: ONS experience of conducting architecture reviews. In *MSIS 2009*. Oslo: United Nations Economic Commission for Europe.
- International Organization for Standardization, 1991. *ISO/IEC 9126 - Software engineering product quality*, International Organization for Standardization.
- Jacobson, I., Booch, G. & Rumbaugh, J., 1999. *The Unified Software Development Process*, Addison Wesley.
- Kazman, R. et al., 1994. SAAM: A Method for Analyzing the Properties of Software Architectures.
- Kazman, R., Klein, M. & Clements, P., 2000. *ATAM: Method for Architecture Evaluation*, Software Engineering Institute, Carnegie Mellon University.
- Le Grand, J., 1982. *The Strategy of Equality: Redistribution and the Social Services*, Routledge.
- Morelli, N., 2002. Designing Product/Service Systems: A Methodological Exploration. *Design Issues*, 18(3), 3-17.
- Rossi, D.P.H., Lipsey, M.W. & Freeman, D.H.E., 2003. *Evaluation: A Systematic Approach* Seventh Edition., Sage Publications, Inc.
- School of Architecture and Design at Aalborg University, 2009. Designing Product Service

System - home. *Service Design Wiki*. Available at:
<http://servicedesign.wikispaces.com/> [Accessed July 27, 2009].

Shostack, G.L., 1982. How to Design a Service. *European Journal of Marketing*, 16(1), 49.

Software Engineering Standards Committee of the IEEE Computer Society, 1998. *IEEE Standard for a Software Quality Metrics Methodology*, IEEE.

Tassi, R., 2009. Service Design Tools | Communication methods supporting design processes. Available at: <http://www.servicedesigntools.org/> [Accessed July 27, 2009].

Vedung, E., 1999. *Public Policy and Program Evaluation*, Transaction Publishers.

Welsh Assembly Government, 2004. *Making the Connections: Delivering Better Services for Wales*,

Welsh Assembly Government, D.F.P.S.A.P., 2006. Beyond Boundaries: Citizen-Centred Local Services for Wales. Available at:
<http://new.wales.gov.uk/topics/improving/services/strategy/beyondboundaries/?lang=en> [Accessed December 17, 2008].

wikipedia.org, 2009. List of system quality attributes - Wikipedia, the free encyclopedia. *Wikipedia list of system quality attributes*. Available at:
http://en.wikipedia.org/wiki/List_of_System_Quality_Attributes [Accessed July 25, 2009].