Service-Oriented Architecture

{Randall Perrey | Mark Lycett}@brunel.ac.uk
Department of Information Systems and Computing, Brunel University, Uxbridge UB8 3PH

Abstract

This paper attempts a definition of service derived by unifying its usage in practice. In doing so it identifies participant's perspective as separate from actual different usages. Using this as an aid, the paper analyses the criteria applied in the use of service and considers how these can be aligned to derive commonality underlying the usage. Also included is a brief outline of the way in which a service-oriented architecture following such a model would affect the practice and process of developing large-scale, distributed software systems.

Defining Service for Architecture's Sake

Finding a unified, applied convention for 'service' with which to develop ideas about a service-oriented architecture is proving difficult. A popular approach has been to define service in the abstract and by doing so attempt to drive the usage [1, 3]. This is not an unproven approach in general but here the rush into the field has produced little workable collateral for the practitioner attempting to implement a service-oriented architecture. The approach taken by this paper is to start with the use of the concept 'service' as used in industry and derive the bounds of what service could become in the more abstract framework of architecture.

'Service-Oriented Architecture' is not used here in a purely structural metaphor. Architecture can mean both a high level pattern for the arrangement and dependencies of technical modules as in [2] and, as used here, the overarching approach to the development and instantiation of the functional units that make up a system e.g. [4]. Much work has been done in this area on web services so understandably there is a substantial bias in the literature towards the particulars of web architecture. Here service and service architecture are not specifically web oriented but more about the abstractions used in large, usually distributed, software system developments. The aim of this paper is to understand and align the usage of service and present a high level overview of a possible service-oriented architectural approach that arises from such an alignment.

Scoping Service

New concepts, in this case abstractions, are usually likened to established (or at least popular) concepts for the purposes of understanding, but then suffer the problem of slowly being corrupted by them. To avoid remaking past mistakes it is important to be clear about what services are not, in addition to what they are. By carefully defining the bounds of the concept of service in this way, we can avoid scope creep and corruption.

One damaging assumption for service is that it is another term for component. Many of the good practices that were expounded in component-based development and integration methodologies are relevant but services are freed from the ties to platform or language specific technical frameworks [5]. Some definitions of service state that they can be implemented by components and thus tacitly suggest they are just another term for interfaces. A service can include an interface but it makes no assumptions that the consumer can accommodate the implementation mechanisms in the way that interfaces traditionally have done. A service operates under a contract / agreement which will set expectations, and a particular ontological standpoint that influences its semantics. Such complex, human comprehensible facets cannot be implemented by a collection of function signatures such as is the situation for interfaces in current commercial component frameworks. Another association is with any internet enabled application. This seems to be more of a marketing ploy and so is something simply to be rejected. A particularly vexing similarity is with a business process. This paper proposes a usage that exploits the similarities and differences with business processes but the issue remains ever open.

In order to clarify and then unify what a service is going to be for the purposes of a service-oriented architecture, the first step is to address the current, different usages of service in the field.

Services as Perspectives

The concept of perspective is the key to reconciling the different understandings of service. Business participants view a service (read business service) as a unit of transaction, described in a contract, and fulfilled by the

business infrastructure. The presumptions and semantics of service are connoted from business experience and this shapes the perspective. Technical participants view a service as a unit of functionality with the semantics of service described as a form of interface. These different views are not necessarily contradictory provided there is commonality underlying the perception (See fig 1); otherwise their usage will probably be conflicting. The utopia of service-oriented architecture is to structure the technical infrastructure in such a way that it explicitly provides business services and as such can be restructured to provide for new or upgraded business services on demand.

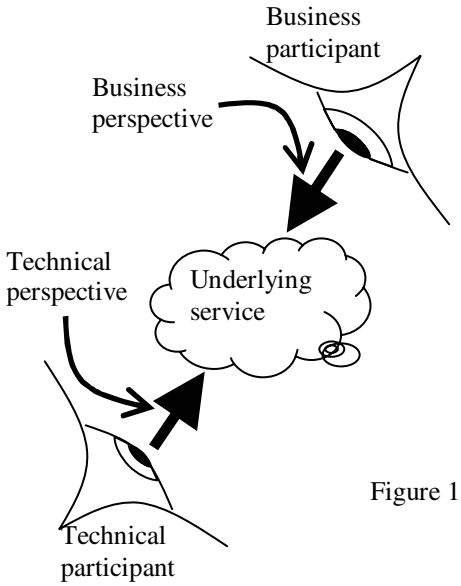


Figure 1

Current Practice

Currently the situation is that while understanding of the term service is not unified in practice, there is a tendency to presume that it is for the purposes of business / IT crossover conversations about architecture. Leaving this presumption unquestioned, however, admits confusion and inconsistency into any shared usage. We believe that there are two intertwined dimensions of difference in the disparate understandings of service. There are different perspectives of service (see above). There are differences in the criteria that each uses in their identification of service. Problems arise when the different and conflicting criteria are cloaked by the different perspectives of the participants.

Identifying Business Services

- Business visible – The thing your customers pay for
- Technology agnostic – WHAT is to be implemented, but not HOW
- Defined by context NOT size – Any example cannot be assessed outside of its context as per the above points

As an example of this last point:- when providing the service of creating a new insurance contract a likely step will be to first give a quote in which case 'quote' is not considered a separate service; whereas if a quote is of value to the customer, e.g. for comparison purposes, then it can be regarded as a service in its own right.

Identifying Technical Services

- Technology solutions already exist – Agnosticism is an illusion
- Service is functionality encapsulated and abstracted from context

Examples such as 'print service' illustrate these points in so much as there was no concept of such a service before there were encapsulated printer drivers and there is only a print service because what is to be printed and how (i.e. type and location of printer) are safely abstracted away as variables of the service.

Whilst there are differences between business and technical services in the value proposition and in the visibility, the most notable difference is the complete reversal of the significance of context to the definition. Resolving the first two differences could lie in something as straightforward as being explicit about the expected granularity, reuse and optimisation capabilities of service. The latter is not so straightforward. The business perspective is working in fluid contexts, looking to exploit market opportunities and expecting business services to come into existence or be reconfigured as necessary. Meanwhile the technical perspective is looking for efficient, reusable, long-lived services and technical functionality is best optimised when the context is static. Context can never be irrelevant to service but limiting its pivotal role is essential for a coherent picture of service.

The previous example of a potential business service of 'quote for insurance' is very unlikely to be implemented in more than one technical way because it would be considered poor optimisation. Technical stakeholders might ask "What is the optimised mechanism for quoting?" and while business stakeholders can answer for today's business model they will not want to be held to this if and when the model changes, scooping 'quote' into or out of the role of a business service.

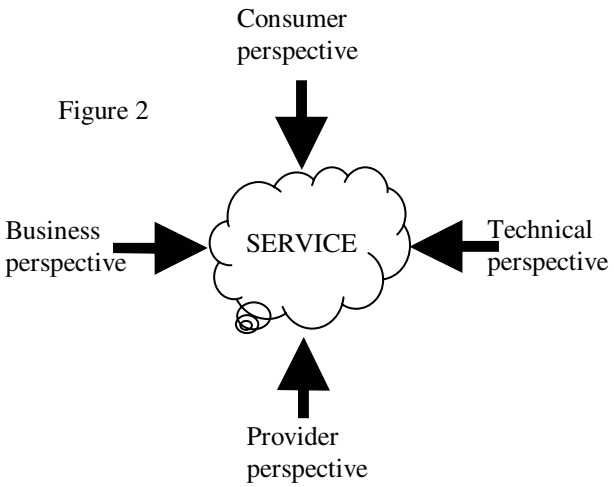
More Perspectives on Service

Service is a useful concept if it describes the boundary between a consumer understandable value proposition and a technical implementation. The value proposition will be a matter of perspective for the consumer of the service and the technical implementation will be the responsibility of the provider for which they too will have a perspective. Consumers may wish to know some detail about the

performance of the service to satisfy themselves as interested parties in a transaction but this does not affect the principle here. The use of the term technical is used in a loose sense and should not be taken as defining the level at which a service operates.

Putting the more established consumer and provider perspectives orthogonal to the business and technical perspectives on service (See fig 2) gives a complex picture which must be resolved with an underlying shared understanding if useful work is to be done by a service-oriented architecture.

Figure 2

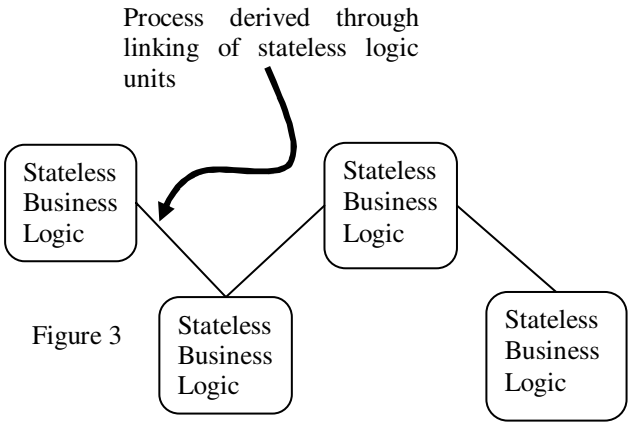


Provision of Service

The technical provision of services is not a uniform activity but is skewed by prevalent technology solutions. User interface Services are commonly provided via well defined frameworks such as Microsoft Visual Basic controls and data management by relational DBMS. Even when these actual technologies are not used the services tend to be described through reference (albeit indirectly) to them. Business process, business logic and business rules are not served well by existing technology solutions and no leading ideas command the field. To complicate the issue the user interface and database technologies have tended to leak-over in piecemeal fashion offering part solutions to logic and process and, in doing so, hindering the development of a core approach.

Figure 3 describes an architecture for technical services providing business logic and process and assumes these are isolated from the well-provisioned areas of user interface and data management. Such an assumption raises issues such as data management of transient data (state) required for execution of business process. Here we argue that data persistence should be orthogonal (effectively out of this picture because it would be whenever and whatever is required) and that data management technologies should recede from leaking into process and instead concentrate on providing joined up management of disparate persistence requirements.

Figure 3



Services and Business Processes

The business and technical perspectives can be aligned without resorting to the kind of recursive decomposition that pushes the definition of service into the academic abstract. The technical services are constructed into stateless business logic units which are comprehensible but have no (or limited) business process connotations. These are composed into more external-customer-focussed business services by orchestrating them into business processes (see fig 3). The orchestration into process is currently hidden in this picture within the simple lines that link the boxes. These lines represent significant architectural policy information that, we suggest, will require intelligent connectors encapsulating active functionality to implement. The advantage of these stateless logic services is that they can be made to exploit the context independent expectations of technical services but can be composed relatively quickly, possibly dynamically, into business processes provisioning business services that can opportunistically capitalise on a particular business context.

Using this model, the required business services can be identified by business participants in the appropriate timescale for the market. The business processes required to provision those services can be defined taking into account any implication of having different flavours of the service; e.g. gold, silver, bronze. The technology supporting facets of the process can be decomposed into business logic services which can be refined in collaboration with technical participants. This can be done independently of the business service definition exercise since the context is not important to these technical services. The business process connectors will be generic mechanisms that require policy and context information (that will need to be defined) to connect the logic units

together, using the appropriate protocol, in a semantically consistent way.

Building the business logic services and composing them back up into processes that will support the business services requires a particular approach to software development. The approach mandated by component-based development is a suitable starting point but as mentioned earlier the two approaches are not the same. Components rely on a technical framework within which to operate. Component in the same framework can work well but there is not one standard for frameworks and all those available suffer from one or another limitation. The lack of successful generic mechanisms for interoperation between components in different frameworks illustrates the problems of optimising for performance without regard to the original philosophy that led the move from monoliths in the first place. Service orientation does not refer to a technology or expect an implementation mechanism. Mining services from legacy applications does not require their division into smaller units but rather the isolation of key logical function points and their extensive description. Substantial up-front resources will be required to create the intelligent connectors and their environment. The selection of these will depend on the legacy infrastructure but it is envisioned that they will always drive towards greater decentralisation rather than draw all resulting solutions towards their implementation. This demands an approach to architecture that realises and values the significance of external, configurable context metadata. This metadata is essential to ensure semantic consistency of the information items and the correct business process protocol and provides the commonality underlying the service architecture.

The Advantages

Such an approach would pay substantial dividends for an organisation when the market or business context changes in such a way that demands infrastructure changes. Now redefining business services and processes is a case of defining the context metadata, agreeing policy decisions and identifying what architectural mechanisms to utilise. The supporting infrastructure of each service can be assessed individually and therefore in parallel for any upgrades needed. Once this is done the business process can be realised by reconnecting the appropriate stateless logic services to suit using the generic connectors which have been reconfigured from the context metadata. New business processes can be defined in a similar way, reusing the stateless logic services and augmenting the context metadata sufficiently to enable (re)use of generic connectors. New stateless logic service can be introduced as and when needed in a demand driven environment. This is another contrast to the component reality in which new functionality for a given framework is delivered when suppliers are prepared to release.

There is nothing new in each point of this architecture but the collection demands a significant shift in thinking from current approaches. If we are finally to realise the promise of agile, relevant, efficient information systems through technology then we should expect that we cannot compromise the paradigm shift and opt instead for a renaming and reshuffling: that has not worked to date.

References

[1] P. Allen, *Realising e-Business with Components*. London: Addison Wesley, 2001.

[2] J. Bosch, *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach*. Harlow: Pearson Education Ltd, 2000.

[3] C. Lovestock, S. Vandermerwe, and B. Lewis, *Services Marketing*: Prentice Hall, Europe, 1996.

[4] M. Shaw and D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*. Upper Sddle River, NJ: Pearson Education, 1996.

[5] D. Sprott, "Service Oriented Process Matters," in *CBDi Forum Newsletter*, 2002.