

Service Oriented Architectures in the Delivery of Capability

Duncan Russell, Jie Xu

{duncanr | jxu}@comp.leeds.ac.uk

School of Computing, University of Leeds, UK

Abstract

Service oriented architecture (SOA) is becoming established in computing as a means to integrate processing and access data across organisations. In general architecture terms, services are used to loosely couple assets in systems by describing interfaces at a high level of abstraction. The abstract service descriptions are independent of the different service implementations, which are instantiated by the assets. Dynamic integration of assets can be captured using the service interface descriptions, which promotes loose coupling by only 'binding' to service implementations when needed. For Network Enabled Capability (NEC), services can be integrated to provide executable capability. Service descriptions need to be rich enough to enable mission planning and evaluation of architecture. The descriptions also need to be high-level enough to give the means to evaluate and improve the dependability of supplied services by making assets interchangeable and evolvable to fulfill continuous delivery of capability. This paper proposes that extending service descriptions in SOA in terms of functional and non-functional attributes of assets can improve dependability in achieving delivery of capability.

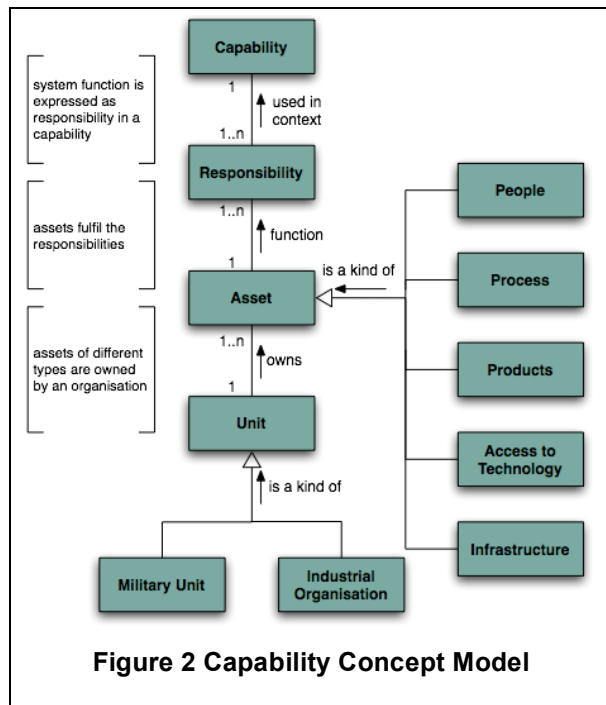
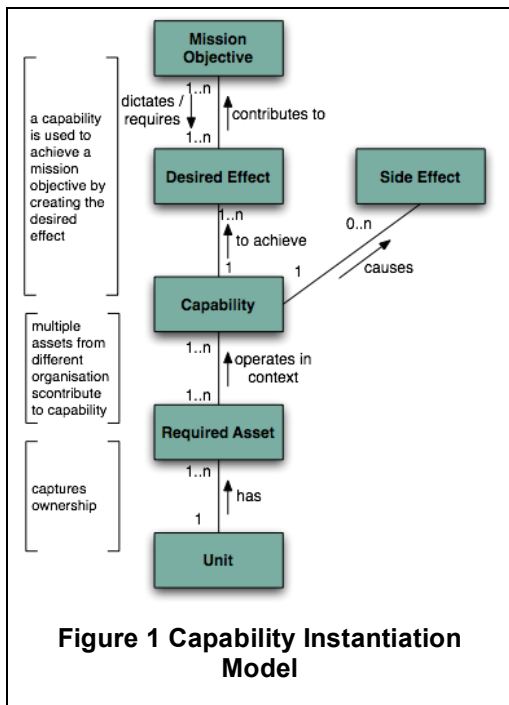
Introduction

Network Enabled Capability is the integration of assets to fulfil a mission objective. Network enabling assets with common communications is only part of the problem. Assets need to be integrated in context, to assist in human activity and provide dependable inter-operation. The network enabled architecture will need to integrate systems of systems in a flexible manner, identifying the assets that provide the functionality and characteristics of the task. For NEC, large scale system and integrated of systems of systems need to cope with fast paced changes and operate in unknown and dynamic environments. However, in these situations, using structured closed-world systems and making assumptions about total system behaviour can no longer meet requirements.

The use of flexible architectural descriptions for NEC enables the functional requirements to be identified and the non-functional characteristics can be used in the selection of assets to deliver the required capability. Flexible architectures provide an abstracted view of the function that is removed from the concrete instances of assets. An abstract view of the task can be used for planning and handling unanticipated system changes in the delivery of capability by offering different solutions to achieve the same or similar mission objective. Defining capability in terms of function aids mission planning, coping with a changing environment to maintain capability, understanding the impact of new requirements on existing assets, how best to use them and what options may be available.

Capability can be broken down into functional elements that can be fulfilled by one or more networked assets. By describing the functions in terms of services, SOA can be used to define capability. Quality of Service (QoS) captures the attributes and constraints of the service operation, such as availability, accessibility, integrity, reliability, security, maintainability and resilience to name a few. These can be applied to descriptions of services, so that different assets can fulfil the same service function, but provide a different QoS. A simple example is a parcel courier, where delivery is offered within different durations. Underlying the delivery service are many different assets, people and procedures that may be interchangeable to suite the distance, size of parcel and time for delivery.

In the NEC battlefield, the architecture will need to integrate systems of systems using service descriptions that include functional description and the QoS attributes. These QoS attributes are important measures that need to be monitored in use, but also need to be known for mission planning and acquisition. The challenge for architectures in NEC is to express known



characteristics alongside unknown or variable attributes, using monitoring to evaluate an architecture through its lifetime in unknown and variable situations [1].

Capability

Capability can be modelled as the combination of the assets required to achieve a mission objective, as shown in Figure 1. This is the implementation model that represents a combination of concrete asset classes that achieve the desired effect contributing to a mission objective and may also cause some side effects. The required assets are each owned by a unit, which is defined in Figure 2 as either a military or industrial organisation.

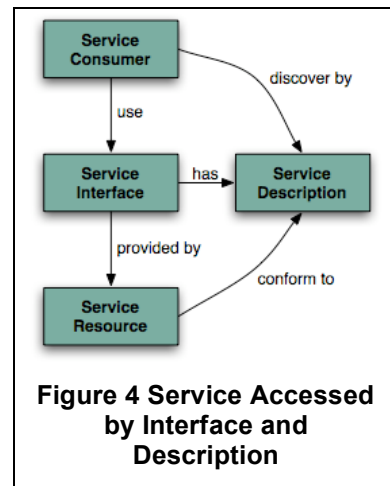
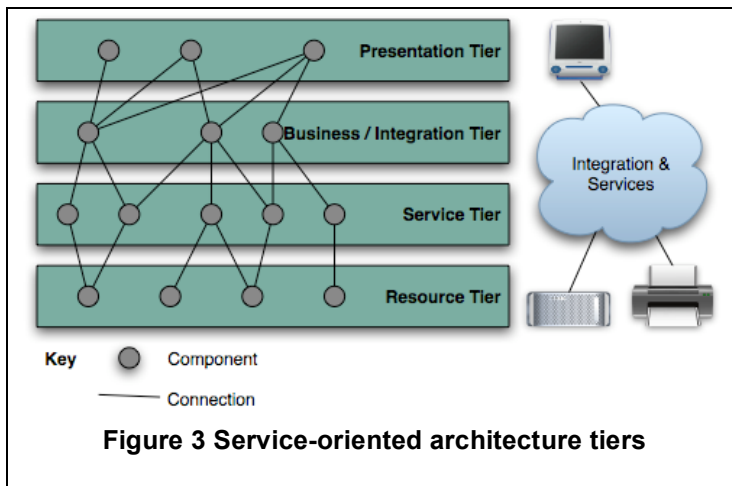
The capability concept model in Figure 2 shows how assets can be combined at a conceptual level to fulfil capability. This diagram shows an additional level of indirection between the assets and capability. By extracting the responsibilities that assets have to achieve a capability, the indirection illustrates that different assets can be exchanged to provide similar responsibilities. The assets are systems and components that may be humans, hardware, software, communication equipment, as well as other resources, including those that describe the Defence Lines of Development (DLod) [2].

These two figures illustrate part of the conceptual model of capability. To use this practically, knowledge of the assets and procedures required to achieve objectives needs to be captured. The high level model can be used to link wide ranging activities in the identification and use of capability. For long-term strategy, it can be used to identify valuable assets in acquisition planning. At the operational level, different combinations of assets can be selected to achieve a mission objective, while at the tactical level, the model supports the selection of assets to deal with changes during operation.

Continuous delivery of defence capability requires management of precision, speed, agility, deployability and sustainability [3]. The capability model attempts to illustrate that capability can be defined at an abstract level, but to achieve NEC, networking is required to combine assets in a flexible and controlled manner. The next section presents Service Oriented Architecture as one method of flexibly structuring networked assets to deliver higher functional capability.

Service Oriented Architecture

Service Oriented Architecture (SOA) provides one means to facilitate the matching of assets to capabilities. In SOA, assets are described by the service they provide. Services are



combined to provide further functions that can be offered as services. Further integration of services provides functional capabilities. This integration of services can be seen to be similar to the conceptual model of capability, where a capability is formed by the integration of responsibilities. The description of responsibility is similar to a service described by its function and quality of service attributes.

The capability model uses the description of the responsibility of an asset as a level of abstraction to decouple the concrete assets from the general description of capability. This generalisation allows different types of assets to contribute similar responsibilities. In SOA, an asset described as a service (by its function) can be generalised such that different types of asset may also provide a similar service. To represent this we need to understand what a service is and how services are combined in a service oriented architecture.

The OASIS group defining the SOA standard uses the term capability in the definition of a service [4]:

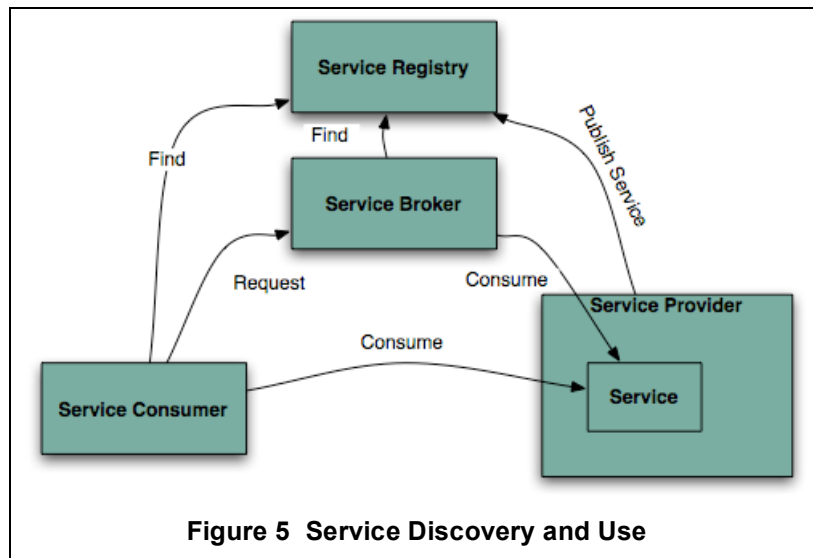
The noun “service” is defined in dictionaries as “The performance of work (a function) by one for another”. However, service, as the term is generally understood, also combines the following related ideas:

- *The capability to perform work for another*
- *The specification of the work offered for another*
- *The offer to perform work for another*

These concepts emphasize a distinction between a capability and the ability to bring that capability to bear. While both needs and capabilities exist independently of SOA, in SOA, services are the mechanism by which needs and capabilities are brought together.

In Figure 3, the integration of services produces useful applications presented to the users’ needs. On the lowest tier the resource execute the functions that are presented as services for integration. The resources are executed via a service interface, shown in Figure 4. The interface provides a consistent view for using different resources. In addition to the interface, there is a description of the service. This describes the service interface and qualities of the operation and management of the resource. These are the QoS attributes that can be used to provide assurances of the service behaviour and at the integration level these can be combined to provide assurances about the delivery of capability.

In delivering capability, the SOA concept is applied to facilitate the matching of capabilities and needs. Figure 5 shows how a service provider (the owner of the resource) can publish the availability of a service type in a registry. The service registry holds the service description, including a definition of the interface. The service consumer searches the registry for appropriate services to fulfil the requirements. Once found, the service consumer will *bind* to the service before use, which relates to the selection of assets in the previously presented capability model.



One advantage of SOA is the support for the different temporal options for binding service implementations. Early binding of assets to functionality is traditional in the design cycle. However, to cope with changing demands caused by new requirements and changing environments, binding later allows for a more flexible design. Ultra-late binding, where a service is discovered and instantiated at the time of execution, allows for a system to respond to changing requirements.

Within a military context, this approach provides major advantages for integration both for planning and operation. Provision of service definitions enables selection of equipment to be deployed on the battlefield and ultra-late binding enables dynamic reconfiguration and upgrades.

The dynamic binding of assets to service delivery in SOA is achieved by abstracting service function from providing assets, by defining common service elements that are provided by different assets and asset types. The generalisation of service definitions creates a loosely coupled system, where the architecture supports changing assets. One advantage of loose coupling has already been discussed in dynamic and late binding of assets to service types. As shown in Figure 5, the binding requires the service consumer to find services that fulfil a requirement. The consumer would typically combine different service types to fulfil an overall process, to form part of a capability.

Loose coupling and dynamic binding means that SOA has the following characteristics:

- Service integration. Services are defined as composable functions, similar to component architecture [5], and can be combined to form higher levels of functionality and deliver capability.
- Service discovery. Service providers offer services in a loosely coupled architecture to consumers for dynamic composition. The consumer requires discovery mechanisms to locate and bind before utilising services.
- Service reconfiguration. Services can be adapted to meet consumer requirements at bind time. During service discovery, the consumer and provider may negotiate terms of service delivery involving quality of service parameters. For instance, reliability may be increased using redundancy, or response time improved using demand-led dynamic allocation of resources.
- Service evolution. By abstracting the interface from the service implementation, a service can adapt to changes in its environment and the demands of the service consumer. Selecting appropriate resources at the time of service execution allows the resources to be updated and adapted without interrupting service availability. This supports continuous service delivery and therefore, continuous delivery of capability.

To enable loose coupling, services need to be described so that service consumers can discover applicable service functionality and relevant QoS attributes (such as accuracy, confidentiality, timeliness, availability, cost and legal factors). Current SOA technologies, such as web services [6], have limited scope for service description. Functional behaviour and input-output formats can only be recognised by using a pre-defined naming convention. The

restrictive naming convention limits service usage to pre-arranged consumer-producer relationships and defines the service only in terms of its acceptable data types, methods (functions), message formats, transport protocol and end-point uniform resource identifier (URI) [7].

The semantic web community [8-10] is working towards enhanced descriptions of services that support the description of a service function and QoS attributes. Projects such as IBHIS [11] have investigated enhancing service description to dynamically discover new services that provide data for personal health records. An information broker can build increasingly detailed records of people as more services become available. Data types are described semantically to allow uniform comparison and presentation methods.

Semantic descriptions play an important part in quality of service attributes. As services are composed to form higher level services and ultimately create capabilities, appropriate levels of service description are required [12]. For example, tracking a visual target may require a camera with a certain resolution. The data returned from the service should be described to have a degree of accuracy (using the resolution metric). The 'return picture' service from the camera can be dynamically integrated in a targeting system that controls the firing of a weapon. By aggregating the measures and metrics of QoS attributes from the integrated services, the overall capability will provide a degree of lethality. Other overall QoS measures, such as dependability, can also be produced from the aggregate metrics. Semantic technology using ontologies allows descriptive terms to be related and understood by computers [13].

Conclusion

In this paper, a conceptual model for capability has been related to service oriented architecture. Service functions can be integrated to form higher-level functionality and provide capability. The architectural style of SOA provides mechanisms to improve system dependability by supporting reconfiguration and evolution through loose coupling. Loose coupling relies on service consumers discovering appropriate services to integrate and meet capability requirements. This means that service providers must produce services that meet consumer demands and that services are described in terms that can be understood by consumers. To meet demands services can be adapted and updated. And consumers can use ultra-late binding to select services, as demand requires. Services can be described using formal semantics that allows like service to be compared for functionality and selected in terms of quality of service.

Future work will investigate enhancing methods to describe service functionality and QoS attributes to allow military assets to be composed into capability, concentrating on assets that are networked. QoS attributes should provide assurances about the delivery of a service; part of the future work will investigate methods of evaluating architectures before and during operation. The overall objective is to support dependable, continuous delivery of capability by understanding the contribution in function and QoS of different assets.

Acknowledgements

The work reported in this paper has been supported by the NECTISE project jointly funded by BAE Systems and the UK Engineering and Physical Sciences Research Council Grant EP/D505461/1.

References

- [1] Russell, D., N. Looker, and J. Xu. SOA, Dependability, and Measures and Metrics for Network Enabled Capability. in *IET Forum on Capability Engineering: At Home and Abroad* 2006. London, UK: IET.
- [2] UK Ministry of Defence, *The Acquisition Handbook, Edition 6 - October 2005*, UK MoD, 2005. <http://www.ams.mod.uk/ams/content/handbook/maintext.pdf>.
- [3] UK Ministry of Defence, *Network Enabled Capability JSP777 Edition 1*, UK Ministry of Defence, 2005.

<http://www.mod.uk/DefenceInternet/AboutDefence/CorporatePublications/Reports/OtherPublications/NEC/Jsp777NetworkEnabledCapability.htm>.

- [4] OASIS, *OASIS Reference Model for Service Oriented Architecture V 1.0*, OASIS, 2006. <http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf>.
- [5] Szyperski, C., D. Gruntz, and S. Murer, *Component software : beyond object-oriented programming*. Component software series. New York, NY ; London: ACM Press : Addison-Wesley. 2002.
- [6] Alonso, G., et al., *Web Services - Concepts, Architecture and Applications*. Data-Centric Systems and Applications, ed. M.J. Carey and S. Ceri. London: Springer. 2004.
- [7] Turner, M., D. Budgen, and P. Brereton, Turning software into a service. *Computer*, 2003. **36**(10): p. 38-44.
- [8] *Semantic Web enabled Web Services (SWWS)*, SWWS, 2003. <http://swws.semanticweb.org/>.
- [9] Berners-Lee, T., J. Hendler, and O. Lassila, The Semantic Web. *Scientific American*, 2001(May 2001).
- [10] Decker, S., *SemanticWeb.org*, SemanticWeb.org, 2003. <http://www.semanticweb.org/>.
- [11] Budgen, D., P. Brereton, and M. Turner. Codifying a Service Architectural Style in *28th International Computer Software and Applications Conference (COMPSAC 2004)* 2004. Hong Kong, CHINA: IEEE Computer.
- [12] Hill, M., *Service Taxonomy and Service Ontologies Deliver Success to Enterprise SOA*, SYS-CON Media, 2006. <http://webservices.sys-con.com/read/175385.htm>.
- [13] Fensel, D., Ontology-based knowledge management. *Computer*, 2002. **35**(11): p. 56-59