

# Service Oriented Architectures in the Provision of Military 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 service interfaces at a high level of abstraction. Network Enabled Capability (NEC) is an initiative from the MoD to react to the rapidly changing conflict environment in which its forces must operate by using dynamic integration of assets to provide dependable military capabilities. The NECTISE (NEC Through Innovative Systems Engineering) project is responding to this need by investigating how loosely coupled services can be used to describe the functions and quality of service for heterogeneous assets and networks. SOA mechanisms for discovery, management and integration can be used along with rich service descriptions to support continuous delivery of capability.

## 1. Introduction

The Armed Forces need to be flexible, ready and rapidly deployable, with the application of controlled and precise force, to achieve realisable effects. To be successful in achieving this goal, NEC requires system integration of independent components that can evolve, operate in a dependable manner, managing system and component changes, cost effectively and connecting industrial, defence and pan-defence environments. NEC requires Network Enabling by connectivity, information sharing and networking people, assets, and procedures; and Capability requires identification of networks of people, assets and procedures to fulfil mission objectives.

In the future NEC battlefield, the architecture can provide the means to integrate systems of systems using service descriptions that include functional description and the Quality of Service (QoS) attributes, such as availability, accessibility, integrity, reliability, security, maintainability and resilience to name a few of the characteristics. 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]. However, in these

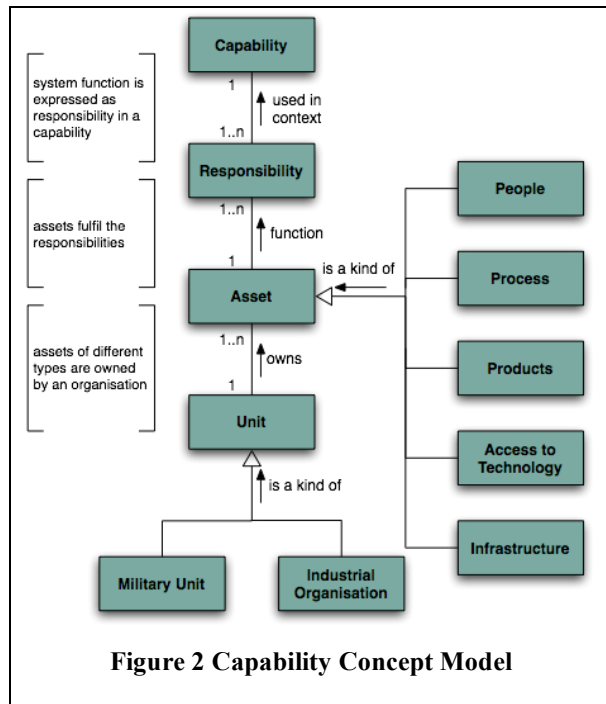
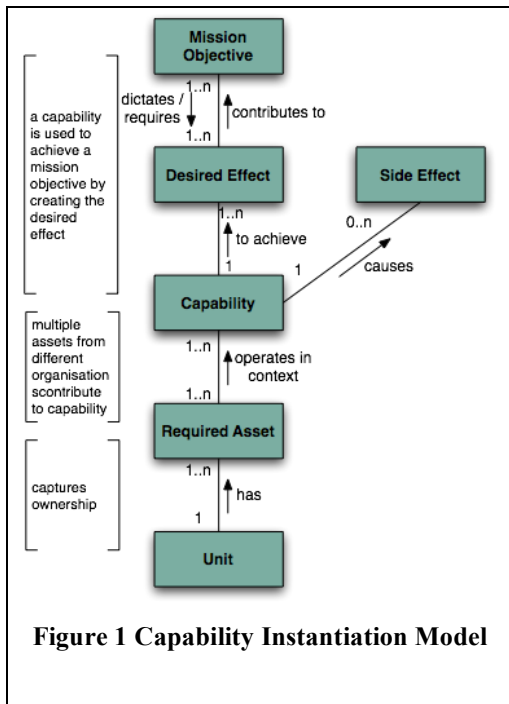
situations, using structured closed-world systems and making assumptions about total system behaviour can no longer meet requirements.

The NECTISE project is jointly funded by the EPSRC and BAE Systems. It involves ten universities and is addressing the question of how BAE Systems delivers NEC to the UK MoD, taking account of the aims summarised in the 2005 Defence Industrial Strategy [2]. To tackle this NECTISE contains four topic groups:

- Through-Life Systems Management
- Systems Architecture
- Decision Support
- Control and Monitoring

Within the Systems Architecture topic group there is an investigation into using SOA to support the dynamic military environment and delivery Network Enabled Capability.

Practical evidence shows that system architecture is the most important factor that affects both a system's functionality as well as its non-functional aspects, such as scalability, flexibility and dependability. Architectural models should therefore be the starting point for any development of NEC systems. Numerous studies have demonstrated that system architectures are effective in assisting the understanding of broader system concerns by abstracting away from the details of a system [3]. This is achieved by employing architectural styles appropriate for describing systems in terms of components, the interactions between components and the properties that regulate the



composition of components. Systems and components can be human beings, hardware, software, communication equipment as well as other resources.

In particular, architectures for NEC systems must consider the viewpoints of all capability stakeholders; connectors between independent components and/or systems must describe their interoperability; and configurations describe the evolutionary acquisition of capability through successive system development cycles.

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 integration of systems of systems need to cope with fast paced changes and operate in unknown and dynamic environments.

## 2. Capability

Military capability is the ability to achieve a specified 'wartime' objective and includes four major components: force structure, modernization, readiness and sustainability [4]. Examples of different levels of capability are: defend UK, capture and defend hilltop, and

deliver and administer medical treatment. In this architectural view (Figure 1), capability can be modelled as the combination of the assets required to achieve a mission objective. This is the instantiation 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 uses responsibility as an additional level of indirection between the assets and capability. Capabilities can be broken down by functional elements and the functions are assigned as responsibilities that assets provide in order to achieve a capability. The indirection between assets and capability illustrates that different assets can be exchanged to provide similar responsibilities, albeit with different qualities of capability delivery. 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) [5]. The DLoD are crosscutting concerns that cut across different systems, namely Training, Equipment, Personnel, Information, Concepts & Doctrine, Organisation, Infrastructure and Logistics.

Figures 1 and 2 illustrate part of the conceptual model of capability. To use this practically, knowledge of the assets required to achieve military objectives need to be captured. This high level model of capability can be used at the strategic level [5] to identify valuable assets in acquisition planning. At the operational level [5], different combinations of assets can be selected to achieve a mission objective, while at the tactical level [5], 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 [6]. 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.

### 3. 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 and 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

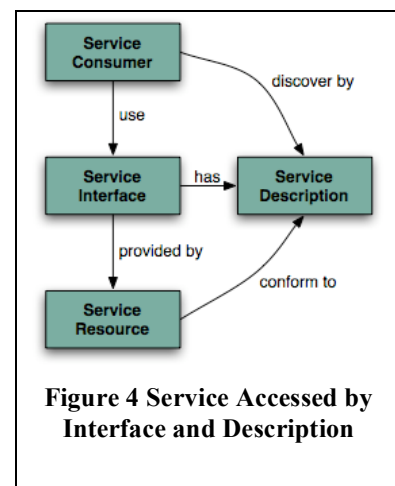
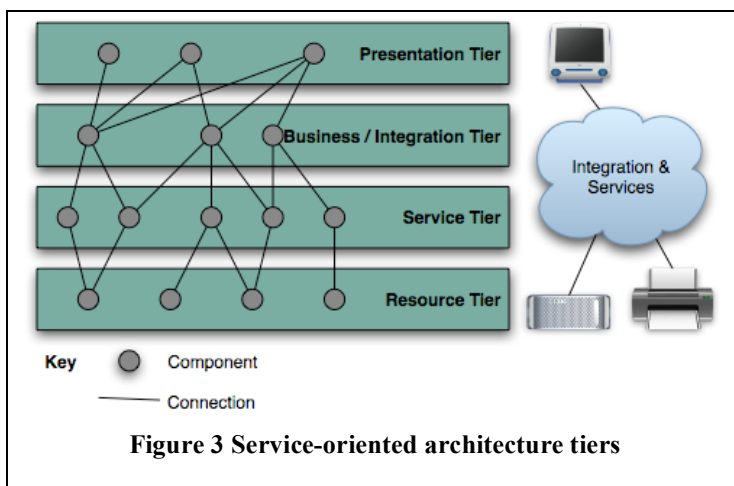
the general description of capability. This generalisation allows different types of assets to contribute similar responsibilities. In SOA, an asset described as a service (by its function) can be generalised such that different types of asset may also provide a similar service. To represent this we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

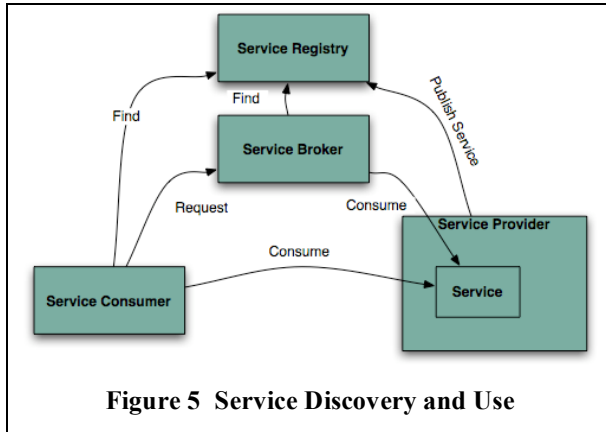
The OASIS group defines the SOA standard using the term capability in the definition of a service [7]:

*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 executes 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 service interface, there is a description of the service. This describes the interface and qualities of the operation and management of





the service. The qualities, which may be specific to the service resource, 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. SOA uses a mechanism by which service providers and service consumers can be matched to each other. 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 system design cycle [8], i.e. deciding the implementation solution to meet functional requirements. However, to cope with changing demands caused by new requirements and changing environments, binding later allows for a more flexible design and system implementation. Ultra-late binding, where a service is discovered and instantiated at the time of execution, allows for a system to respond to changing requirements in real-time.

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.

One military example of the different stages of dynamic binding is the use of weather sensors on a naval ship. Weather information is obtained from on-board sensors and combined with external information such as Met Office [9] forecasting data. This is used for operational planning and in the control of weapons. Currently, applications are designed around to specific sensor types, delivering data for temperature, barometric pressure, wind speed, etc. Using service descriptions of sensors in a SOA means applications can use dynamic discovery and flexible integration. Changes to the system components can be tolerated and therefore managed. Thus, sensor failure or destruction, together with design upgrades, can be incorporated and the application can even be extended to use sensors across different platforms (i.e. aircraft, ground-based stations and other ships).

The dynamic binding of assets to service delivery in SOA is achieved by abstracting service function from the 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 [10], 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.

### 3.1 Relationship with Service-oriented Computing

Service-oriented computing is based upon middleware infrastructure that simplifies connections between heterogeneous systems. The middleware handles common methods of data description and open standards for data interchange means different platforms can be connected without creating bespoke interface handling. This reduces a potential problem of  $O(N*N)$  custom integrations to a  $O(N)$  problem [11].

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). Comprehensive service descriptions are also necessary for service management, enabling service negotiation, composition, and substitution [12]. Current SOA technologies, such as web services [13], 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) [14].

The semantic web community [15-17] is working towards enhanced descriptions of services that support the description of a service function and QoS attributes. Projects such as IBHIS [18] 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 [19]. 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 [20].

Research into Grid Computing [21] has demonstrated dynamic resource discovery and use in SOA. In addition, Grid Computing activities have investigated and developed in many areas that are necessary to produce NEC systems of systems, namely:

- Security [22, 23]
- Virtual Organisations [24-26]
- Fault tolerance [22, 27]
- Service Composition [26, 28]

These research areas provide results useful for building dynamic systems, however, more work is required to develop architectural concepts to deliver continuous military NEC.

## 4. 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 upon 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; consumers can use ultra-late binding to select services, as demand requires. To improve service discovery and management of through-life service, delivery services can be described using formal

semantics that allow similar services to be compared and selected in terms of functionality and quality of service.

Future work will investigate how SOA mechanisms defined from service-oriented computing can be used to capture the service functionality and QoS attributes of military assets. Then, using methods of dynamic integration, compose network-enabled assets into capability. Management of service delivery using middleware should use QoS attributes to provide assurances about the delivery of a service and therefore assurances about the delivery of capability. Part of the future work will investigate using QoS parameters to evaluate architectures before and during operation. The overall objective is to support dependable, continuous delivery of capability using SOA to combine function and QoS of interchangeable and evolvable 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, *Defence Industrial Strategy: Defence White Paper (CM6697)*, UK Ministry of Defence, 2005. <http://www.mod.uk/DefenceInternet/AboutDefence/CorporatePublications/PolicyStrategy/DefenceIndustrialStrategyDefenceWhitePapercm6697.htm>.
- [3] Shaw, M. and D. Garlan, *Software architecture : perspectives on an emerging discipline*. Upper Saddle River, N.J.: Prentice Hall. 1996, xxi, 242 p.
- [4] *Military Capability Definition (from DoD)*, 2007. <http://usmilitary.about.com/od/glossary/terms/m3958.htm>.
- [5] UK Ministry of Defence, *The Acquisition Handbook, Edition 6 - October 2005*, UK MoD, 2005. <http://www.ams.mod.uk/ams/content/handbook/maintext.pdf>.
- [6] 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>.
- [7] 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>.
- [8] Mealy, G.H., *The system design cycle*, in *Proceedings of the second symposium on Operating systems principles*. 1969, ACM Press: Princeton, New Jersey.
- [9] *Met Office: Weather and climate change*, 2007. <http://www.metoffice.gov.uk/>.
- [10] 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.
- [11] Joshi, R., Closer to the Edge. *IET Electronics Systems and Software*, 2007(April/May 2007): p. 14-18.
- [12] O'Sullivan, J., D. Edmond, and A. Ter Hofstede, What's in a Service? Towards Accurate Description of Non-Functional Service Properties. *Distributed and Parallel Databases*, 2002. **12**(2-3): p. 117-133.
- [13] 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.
- [14] Turner, M., D. Budgen, and P. Brereton, Turning software into a service. *Computer*, 2003. **36**(10): p. 38-44.
- [15] *Semantic Web enabled Web Services (SWWS)*, SWWS, 2003. <http://swws.semanticweb.org/>.

- [16] Berners-Lee, T., J. Hendler, and O. Lassila, The Semantic Web. *Scientific American*, 2001(May 2001). *DAME*, in *IEEE International Conference on Services Computing, 2005. (SCC 2005)*. 2005: Orlando, Florida. p. 139-146.
- [17] Decker, S., *SemanticWeb.org*, SemanticWeb.org, 2003. <http://www.semanticweb.org/>.
- [18] 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.
- [19] Hill, M., *Service Taxonomy and Service Ontologies Deliver Success to Enterprise SOA*, SYS-CON Media, 2006. <http://webservices.sys-con.com/read/175385.htm>.
- [20] Fensel, D., Ontology-based knowledge management. *IEEE Computer*, 2002. **35**(11): p. 56-59.
- [21] Foster, I. and C. Kesselman, *The grid 2 : blueprint for a new computing infrastructure*. 2nd ed. San Francisco, Calif.: Morgan Kaufmann. 2004.
- [22] Yang, E. and J. Xu. Integrating an Attack Tolerant Information Service with Taverna. in *Proceedings of 4th U.K. e-Science All-Hands Meeting*. S.J. Cox, (ed) 2005. Nottingham, UK.
- [23] Dada, J.O. and A. McNab. ShibVomGSite: A Framework for Providing Username and Password Support to GridSite with Attribute based Authorization using Shibboleth and VOMS. in *Proceedings of 5th U.K. e-Science All-Hands Meeting*. S.J. Cox, (ed) 2006. Nottingham, UK.
- [24] Hiden, H., et al. The GOLD Project: Architecture, Development and Deployment. in *Proceedings of 5th U.K. e-Science All-Hands Meeting*. S.J. Cox, (ed) 2006. Nottingham, UK.
- [25] Periorellis, P., et al. GOLD Infrastructure for Virtual Organisations. in *UK e-Science All Hands Meeting 2006*. S.J. Cox, (ed) 2006. Nottingham, UK: National e-Science Centre.
- [26] Russell, D., P.M. Dew, and K. Djemame, *Service-Based Collaborative Workflow for*