



The e-Framework Technical Approach — a Rationale

Early work of the e-Framework Partnership has addressed multiple challenges posed by service-oriented approaches to ICT systems analysis and design. Such approaches offer great benefits in terms of flexibility, suitability for purpose, and potential cost savings. They are, however, complex, and require considerable expertise to implement.

The e-Framework Partnership has developed a common approach to the description of service-oriented design and analysis. The core objective of this work is to facilitate sharing experience and expertise more effectively across national boundaries, in addition to sharing across domains of learning and teaching, research, and administration. The early results of this work are stored in an open international service knowledgebase that is built around community contributions. The e-Framework maintains the content to assist international education and research agencies and communities in planning, prioritising and implementing IT infrastructure more effectively.

Consider an illustrative example from a common desktop software application:

A word processor application offers a 'spell checker' function. This function only exists within the word processor; other applications do not come with a spell checker and thus do not provide this function. In order for other applications to make use of the spell checker, a number of options exist. First, a human could copy the information to be spell checked from one application to the word processor, verify the spelling, and copy any alterations back to the first application. Secondly, the software company providing the other application could be approached to add a spell check function. If that company also implemented the word processor, then it should not be difficult to re-use the spell checker library in the other application. However, if the word processor and the other application are owned by different companies, or implemented in different programming languages, then the task of adding the spell checker becomes much more complicated and time consuming.

In the service-oriented view of software design, only one spell check service is needed. This service exists somewhere on the internet, and provides a standard interface by which other applications can make use of it. Thus any company could potentially use the spell check service; and the means of the standard interface should eliminate any problems caused by implementing in different languages. The spell check service will not have any knowledge itself of why it is being called and would be ignorant of the applications that are calling it.

The e-Framework, or something like it, is very important if a service-oriented approach is to be taken to a *strategic* level. In a world that is based around many discrete, inter-

operable services, a common syntactic and semantic understanding of those services and the data that they exchange is an absolute requirement. In an increasingly internationalised education and research landscape, this understanding needs to transcend national boundaries. It also needs to cross domain or work area boundaries, which have a tendency to create “silos”, limiting the effective interoperability of ICT systems. Such silos act to duplicate code and cost, rather than facilitating the elimination of duplicated code, thereby lowering cost and increasing the flexibility of ICT systems over time. If the promise of service orientation is to be fulfilled, a way to enable common understanding amongst practitioners and developers from a variety of different communities is essential. They need to know what is being shared, both from a point of view of ‘production’ (“Here’s my service; you can use it...”) and from the perspective of ‘consumption’ (“I want to use your service ...”).

Interoperability — Problems and Solutions

Consider first a world where ‘just’ the data is available for others to use. This is a world that is not service oriented, but where, nevertheless, a few services do exist. These services are purely “accessing” services – services that allow access to some data that they ‘manage’. Any consumer – that is, any program going to one of those services and asking for data – needs to understand three things:

- How to communicate with the service (FTP? CORBA? A proprietary TCP interface?)
- What the format of the data is (XML? Attribute/value pairs in a text file?)
- What the data means (Student record? Test result? Raw research data?)

This was much the state of the world in the late 1990s. Communication was described in RFCs and developers would use a process of trial and error to obtain data from services and manipulate it. The adoption of common specifications and standards allowed members of a community to agree on semantics. It should be borne in mind, however, that such standards could be interpreted in different ways by different parties, leading to further barriers to interoperability. Often those different interpretations would be informally shared between some members of a community, but little would be formally recorded. Overcoming such barriers was frequently costly.

Whilst much is offered by service-oriented systems in terms of flexibility, in a service-oriented world, these issues, and the overhead they carry, are amplified:

- Is there a shared understanding of communication and semantics?
- Is there a complete lack of understanding of communication?
- Is there understanding of communication but lack of understanding of semantics?

Where a shared understanding of communication and semantics exists, little or no difficulty presents itself. The e-Framework, through Service Expressions and Expression-based SUMs, provides constructs for achieving and building on this.

In the second case, the parties cannot communicate at all – and thus are at least aware that a problem exists – which will lead to a dialogue.

The third case is particularly problematic. The parties can communicate, but do not necessarily have shared understanding of the meaning of the conversation they are engaged in. A trivial example would be sending data to a service that claimed to multiply numbers together, and found instead that it had added them. Semantic misunderstandings are frequently, of course, far more subtle and less easy to recognise than this. If knowledge of what services do is implicit, or open to interpretation, then problems can – and will – occur. The e-Framework provides constructs that facilitate the reduction or elimination of this type of problem.

The e-Framework provides a series of mechanisms by which services may be modelled in an implementation- or domain-neutral manner, and provides a consistent vocabulary by which services may be described. The individual components of the e-Framework describe the behaviours of the different services, as well as showing the precise interfaces to them. The e-Framework also demonstrates how different services may be combined to support a business process within education and research.

This helps to create the conditions where a service-oriented approach can be deployed on a strategic scale. The syntax and semantics of services, and their data, is explained explicitly. Nothing is left to the trial and error approach of the past, because the services themselves are properly documented.

The e-Framework service-oriented approach has been developed by the Partnership over the last three years. It embodies expertise and experience from multiple communities, but remains a living, developing approach. Community feedback is solicited around this rationale.

This work is licensed under the Creative Commons Attribution-Share Alike 2.5 Australia license. To view a copy, visit <http://creativecommons.org/licenses/by-sa/2.5/au/> or send a letter to 171 2nd Street, Suite 300, San Francisco, California, 94105, USA

