

JISC Briefing Document

Background to the JISC Circular 04/06

The e-Framework and Service-Oriented Architecture

Abstract

This paper is a supporting briefing document for the JISC Circular 04/06.

The aim of this paper is to enable project bids to be aligned with the service-oriented approach formulated by the JISC/DEST e-Framework.

The paper background to the e-Framework from four perspectives:

- 1. Recent Background*
- 2. Technical Background*
- 3. Benefits*
- 4. Current Activity*
- 5. Implications for New Projects*

A set of open technical standards are emphasized in their importance to the interoperability and sustainability of new projects.

The JISC tool kit, demonstrator and reference model projects are emphasized in their role of providing a foundation for new projects.

The e-Framework Definitions and Terminology is emphasized as providing a theoretical basis for the integrity of the framework and the submission of new services.

1. e-Framework: Recent Background (1990-2006)

1.1 World Wide Web (1990s)

The success of the *World Wide Web* since the 1990s is partly attributed to widespread adoption of simple, open standards – particularly the HTML mark-up used for web pages and the HTTP protocol used between browser clients and servers.

The *human-to-machine* paradigm for the Internet is now pervasive with the use of browser software to access remote server based *Web sites*.

1.2 Service Oriented Approach (1990s)

In the 1990s, a service oriented approach (soa)¹ to integrating separate systems evolved along with specifications, such as DCOM, CORBA and Java RMI which enabled

¹ lower case 'soa', to distinguish it from Service Oriented Architecture, upper case 'SOA'

distributed systems to work together.

Advanced developments such as Jini in the Java world virtualised everything as a service, supporting dynamic registration and de-registration of services; dynamic discovery of services by clients; and dynamic downloading of adapters by clients.

However, as each technology had its particular draw-backs (platform specific and not object oriented; limited cross-vendor interoperability and too complex; Java specific) and all had difficulty working across organisational boundaries, none of them became dominant and hence had limited adoption.

1.3 Web Services (2000)

Lessons learned from the success of the Web, namely the adoption of open standards implemented on a ubiquitous technical infrastructure, are being applied to a machine-to-machine paradigm for the Internet, with the use of client programs to access remote server based Web services.

In 2000 the term Web Service was codified by the W3C consortium as a software system to support interoperable machine-to-machine interaction over a network.

By using the same underlying communication protocols as the Web, the expectation is that these machine-to-machine interactions are now placed to provide benefits to parallel the human-to-machine success of the World Wide Web.

1.4 Service-Oriented Architecture

Full scale Service Oriented Architectures arose as result of a thorough going application of the service oriented approach across the whole enterprise.

Wikipedia defines [Service-Oriented Architecture](#) as:

“In [computing](#), the term **Service-Oriented Architecture** (SOA) expresses a perspective of [software architecture](#) that defines the use of services to support the [requirements](#) of software users. In an SOA environment, nodes on a [network](#) make resources available to other participants in the network as independent services that the participants access in a standardized way. Most definitions of SOA identify the use of [Web services](#) (i.e., using [SOAP](#) or [REST](#)) in its implementation. However, one can implement SOA using any service-based technology.”

In the context of the e-Framework, the e-Framework supports a service-oriented approach and provides resources to help an institution build its own partial or complete Service Oriented Architecture, which is seen as the specific deployment and interrelationship of services which make up its IT infrastructure, together with its own governance, policies, prioritised implementation plans, procedures and guidelines.

1.5 Convergence of Web Services and Service-Oriented Architectures

As Web Services addressed the problems that were holding back the further development of the service oriented approach, and because those pursuing the development of Web services began to recognise the need for a wider architectural perspective, a broad consensus has emerged that Web Services provide a good platform for the development of SOAs.

However, although providing an agreed common platform for building a SOA, Web Services have yet to provide the same level facilities of earlier platforms that are needed to meet the technical requirements of various enterprise level applications. Much of the current development work on Web Service specifications seeks to address these shortcomings.

1.6 The e-Learning Framework (2003)

In 2003 the JISC funded Frameworks and Tools strand began funding projects to factor e-Learning functionality, such as assessment, into discrete Web services.

The service factoring, combined with a registry of services, example implementations, was called the JISC [*e-Learning Framework*](#).

Technical work within the e-Learning Framework is catalogued either as a *toolkit* or a *demonstrator*.

Toolkits projects are implementations of service interface specifications in the form of service consumer ‘adapters’. These are written in specific programming languages and translate back and forth between the service interface and the programming language. They may also have service provider adapters and that can be used can be used by applications to provide a service interface. They may sometimes provide a service implementation. They also provide supporting documentation and other artefacts.

Toolkits protect applications from changes in the service protocol, simplify the task of implementing the protocol and, if they are configurable, also make it easier to upgrade systems e.g. with faster adapters, or when it is desirable to deploy new features such as reliable messaging and enhanced security mechanisms.

Demonstrator projects take the toolkit and trial it in a different institution to the one that created the toolkit.

Feedback and refactoring from demonstrator projects is fed back into the next iteration of toolkit development.

Toolkits and demonstrator projects collaborate to ensure the sustainability of the software using guidelines developed under an [*Open Source Maturity Model*](#) (OSMM).

Sustainability is further enhanced by the adoption of open technical standards by tool kit and demonstrator projects. Currently, the set of e-Learning standards and specifications

used by these projects is:

- . *IMS Learning Design*
- . *IMS Question and Test Interoperability*
- . *IMS Content Packaging*
- . *IMS Simple Sequencing*
- . *ADL SCORM 2004*
- . *IMS Reusable Definition of Competency*
- . *IMS Vocabulary Definition Exchange*

As work on toolkits and demonstrators progressed, it became apparent that supporting services were required and being developed outside the domain of e-Learning. Many of these supporting services were based around administration requirements and required the use of open standards and specifications that relate to learner information:

- . *IMS Enterprise*
- . *Exchanging Course Related Information (XCRI)*
- . *Open Source Portfolio Initiative*

There has also been uptake of schema within the domains of search, collaboration, repositories, feeds and social software:

- . *IMS Digital Repositories*
- . *Z39.50/SRW/SRU*
- . *Really Simple Syndication (RSS)*
- . *Friend of a Friend (FOAF)*
- . *IMS Resource List Interoperability*

Tool kits have also explored integration work with a number of open source learning environments. Notably:

- . *Moodle*
- . *Boddington*
- . *Learning Activity Management Systems (LAMS)*
- . *SAKAI*

The development of e-Learning Framework tool kits to include administrative, repository, search and other services indicated that enlargement to a more generalised approach was required: the *e-Framework*.

1.7 Service-Oriented Frameworks (2004)

In 2004 a paper by Rehak, Wilson and Blinco, [Service-Oriented Frameworks: Modelling the infrastructure for the next generation of e-Learning Systems](#), articulated how the benefits of Web services within a service-oriented architecture would be applied to the IT requirements of higher education.

This overall vision, which the authors called a *framework*, was to expose higher education IT resources as Web services. The services would then be consumed and combined in an

unlimited number of ways.

Guidance and exemplars for Web service consumption and combination that address common problems would be codified into *reference models*. The reference models would include generalised descriptions of *service designs* and work flows.

The vision of a framework of service designs, supported by reference models, found expression in a strand of the [JISC e-Learning Program](#) called the [Frameworks and Tools Strand](#).

JISC funding is enabling the benefits of the service oriented approach to be realised in the higher education sector through the supporting concept of a *service-oriented framework*.

1.8 the e-Framework (2005)

By 2005 it was clear that the initial conception of an e-Learning Framework managed by JISC was restricted in *geographical reach*, *organizational structure* and *domain scope*.

The eLearning Framework was enlarged to include international partners, with a governance body and lightweight operational structure tasked with creating and maintaining the e-Framework as an information or knowledge structure with a web site as its visible manifestation.

The scope was also expanded to include the domains of resource management, (where the JISC Information Environment had already defined a service oriented approach), research (where the Global Grid Forum and OGSA, the Open Grid Services Architecture, had adopted Web Services) and academic administration (e-Admin).

The e-Framework is now a joint activity between the UK JISC and the Australian Department for Education, Science and Training (DEST), which had been actively engaged with the e-Learning Framework on an informal basis. The new approach has been reflected in its re-branding as the [e-Framework for Education and Research](#), which, for brevity, is referred to as *the e-Framework*.

The aim the UK/Australian joint activity is to produce an evolving and sustainable, open standards based, service-oriented technical framework to support the education and research communities.

In 2005 work was commissioned to provide a theoretical basis for the e-Framework through the development of a *Definitions and Terminology* document. Initial [Factoring and Mapping the Research Domain](#) has begun.

In 2006 the international evolution of the e-Framework continued with the addition of SURF, the JISC equivalent in the Netherlands, and the New Zealand Ministry of Research, Science and Technology as partners.

The JISC/DEST e-Framework now has the theoretical basis, geographic reach,

organizational structure and domain scope to provide strategic direction to technical effort to realise the benefits of service-orientation.

2. e-Framework: Technical Background

2.1 Web Services and Service Compositions

Early in the evolution of Web services, a compositional pattern emerged that was seen as the basic Web Service architecture. In essence, this proposed three systems: a *service provider*, a *service requester/consumer* and a *service registry*, which mediates the initial contact between the provider and consumer. Information about the service provider, such as its address and details of its interface first has to be recorded on the registry. A potential consumer uses the registry to discover, link to and communicate with the service provider (see more below).

As a pattern, it was derived from earlier service oriented approaches and adapted and applied to Web Services. As a Web service pattern, it can be implemented in many contexts. This pattern comes into its own when there are a large number of services, where new services are being added or replacing existing ones and where service consumers need to attach to several services of the same type. As with the Web, this 'loose coupling' of the client and the server enables a more dynamic, flexible and evolvable system.

To understand the recent development of SOA it is helpful to appreciate the recent evolution of Web services technology also.

Web services are commonly considered to have moved through *first generation* and *second generation* technologies.

Web services technology generations have seen a parallel maturing of SOA patterns: from *early* to *contemporary* SOA.

2.2 First Generation Web Services Technology

The first generation of Web services technology was focussed on the ability of a Web service interface to *publish* a description of itself. The technical specification for this feature is called *Web Service Description Language* (WSDL) and is the foundation to the two other important first generation features of Web services: the ability to be *found* by a client program and then be *bound* to a client.

Web services can be dynamically found because their WSDL descriptions allow them to be catalogued in Web services registries. The technical specification for this feature is the *Universal Description, Discovery, and Integration* (UDDI) specification. Although registries are an important feature for web services, limitations in the original UDDI specification have restricted its use in practice.

Web services can be dynamically bound to a client because their WSDL descriptions

enable the client to know the types of messages that can be sent and received from the Web service. The technical specification for this feature is the *Simple Object Access Protocol* (SOAP).

In an effort to promote best practice and high levels of interoperability between Web services, the [Web Services Interoperability Organization](#) (WS-I) has created a profile of the publish, find and bind specifications: WSDL 1.1, UDDI 1.0 and SOAP 1.1, along with the XML standards of XML 1.0 and XML Schema 1.0.

The WS-I provides the basis for the foundation of the *early WS design pattern*.

2.3 First Generation WS Design Pattern

Early implementations of the basic Web service architecture used the publish, find and bind features in the WS-I profile to define three roles for network endpoints in an early Web Service pattern: a service *provider*, a service *requestor* and a service *registry*.

Provider endpoints publish to a registry their web service description which includes their address and a description of the public interface to their underlying resource. It is helpful to think of the service provider as the “server-side” of a client-server relationship.

Requestor endpoints are responsible for binding to and invoking the remote service provider. It is helpful to think of the service requestor as the “client-side” of a client-server relationship and is sometimes called a service *consumer*.

Registry endpoints are responsible for advertising the published web service descriptions. It is helpful to think of the service registry as a kind of “matchmaker” between requestor and provider: once the registry makes the match, it can step out of the picture and let the rest of the interaction happen between the provider and its requestor consumer.

2.4 Second Generation Web Service Technology

After early implementation effort with the first generation of Web service technologies and the early Web Service pattern, it became apparent that further work was required before Web services were ready for enterprise computing.

The early pattern was found to be too limiting for complex business processes, and the first generation technology support for mission critical features, such as security, was insufficient.

The recognition of the limitations of first generation technology led to a slew of specification drafts, collectively known as the WS-* specifications.

The WS-* standards are being developed slower than most would like, but this is an artefact of adopting open industry standards rather than developing proprietary technologies.

Some of these WS-* specifications are now implemented in web service platforms. The issue of limiting support for complex business processes is addressed by the Business Process Execution Language (WS-BPEL), and security features (WS-Security) are being added to tools and runtimes.

As the complexity of web services specifications has increased, a school of thought has developed suggesting the overhead of the WS-* and SOAP specifications is too great for some simple network operations. This community advocate a lighter weight approach to web services for cases where a network call to simply *retrieve, delete, create or modify* a resource is required. This lightweight approach to web services, known as Representational State Transfer (*REST*), is used to describe any simple web-based interface that uses HTTP and XML without employing the overhead of the first or second generation specifications – WSDL, SOAP, UDDI and WS-*. This *RESTful* approach is typically employed when a requestor client uses the HTTP operations of *GET, DELETE, PUT* or *POST* to work with a resource such as an XML document. Implementation experience with first generation web services, the RESTful approach and the WS-* specifications has led to a *contemporary SOA* identified as a pattern from building on top of this web services infrastructure.

2.5 Second Generation Contemporary SOA

With implementation experience, patterns of repeating approaches to technical challenges become apparent.

A major area of further development has been the focus on business processes as the determinant of the services needed and their coordination. By decoupling the coordination from the underlying services, it makes it quicker and easier both to define processes and to modify them when needed. This results in SOAs in principle being able to provide more flexible infrastructures that enable organisations to adapt more readily to changes in their environment or in their strategic direction.

A number of recent developments in Web service specifications are aimed at supporting business processes through the coordination of multiple services, further strengthening the case for Web services as the preferred platform for SOAs.

At the lower level of designing individual services, implementation experience with early SOA, has resulted in a set of eight principles emerging that facilitate building contemporary SOAs.

The eight *principles* recommend that Web services should be designed to be:

1. *loosely coupled*
2. *stateless*
3. *autonomous*
4. *discoverable*
5. *abstract*
6. *described by a formal contract*
7. *reusable*

8. *composable*

In addition to these principles, hard won examples of *good practice* have emerged from implementation experience which suggest that services should be designed to be:

- *coarse grained*
- *document style*
- *aligned with business processes*

Services that are designed to be *loosely coupled* and *stateless* will not have dependencies on other services.

Services that are designed to be *autonomous* have governance over an underlying resource and maintain an explicit control boundary over that resource.

Services that are discoverable and have an abstract description can then use the loose coupling, statelessness and autonomy principles to promote the reuse of the service in infinite number of service workflows or service compositions.

Services that are *coarse grained* are good candidates for *scaling* as they enforce the transmission of a small number of large messages, rather than the heavy network traffic of many small messages.

Document style services use XML documents as input and output messages. *Remote Procedure style (RPC)* services use small, fine-grained units of information, such as strings and integers, as input and output messages.

Services designed in the *document style* promote more *robust* and *flexible* frameworks. Services that are designed in the *document style* also promote scaling. A document style service will exchange and process a few, information-heavy XML documents, rather than making many simple calls to a *remote procedure (RPC) style* service.

If there is a change in requirements, document style services can remain unaffected as often only the document schema needs to be amended. In the case of an equivalent RPC style, the serviced calls themselves have to be changed, requiring a new service interface to be published and re-implemented by both the service provider and consumer.

With the contemporary service-oriented architecture, the IT community has arrived at a set of principles and practices for building on open standards and an already deployed server infrastructure available on the Internet.

The stage is now set to realise the potential of machine-to-machine interaction to parallel the success of human-to-machine interaction of the World Wide Web.

3. e-Framework: Benefits

3.1 Specific and General Benefits of a Service Oriented Approach

The 2004 paper by Rehak, Wilson and Blinco (see above) articulated the *specific* benefits of a service-oriented approach as *value propositions* to four stakeholder groups: *policy makers, institutional IT departments, academic communities* and *suppliers/vendors*.

In addition to these specific benefits, the service-orientated approach is considered to be a general candidate solution to the following scenarios:

- *Where a problem domain has a distributed nature. Resources exist at various network endpoints or are maintained by different institutions.*
- *Where a problem domain requires integration of resources on heterogeneous platforms or in legacy systems.*
- *Where data needs to be made widely available beyond the core application that maintains and generates the data.*
- *Where a problem domain requires a solution that includes automation.*
- *Where a problem domain requires a solution to be agile and respond to policy changes.*
- *Where a business/institutional process is not core and has potential for outsourcing.*
- *Where an institution wishes to avoid being “locked in” to a dependency on a particular vendors product.*

3.2 Unexpected Benefits of Service Oriented Approach

In addition to these general benefits and the specific value propositions outlined in the 2004 paper, there is now recognition that when services are made available using the principles and practices of service orientation, services can be reused, combined and consumed in unexpected ways.

The unintended and novel synergies of services are informally referred to as service *mash-ups*. Service *mash-ups* are part of a wave of innovation on the Internet, informally referred to as *Web 2.0*, that enables *social and collaborative software* to be built.

It is anticipated that this type of software innovation will provide key benefits in the academic community where *social constructivist* learning styles are emphasized or where *remote collaborative working* is required.

With the benefits of service orientation identified, as well as emerging valuable unintended synergies, the JISC is supporting the development of the e-Framework on a broad front.

4. e-Framework: Current Activity

4.1 JISC e-Learning Programme

In addition to the Toolkits and Demonstrator projects which are already associated with the e-Framework, another group of more recent JISC projects are being aligned with the e-Framework. The JISC *Reference Model projects*, started in April 2005, are being submitted to the e-Framework to provide direction for future work.

4.2 JISC Reference Model Projects

In September 2006 the first round of [JISC Reference Model Projects](#) will conclude. The projects will present an approach to articulating a domain area using a service-oriented perspective.

Each reference model is a service-oriented description of a collection of use cases, workflows, service descriptions, toolkits and other artefacts specific to the issues of a particular problem domain.

The intention is that each reference model will enable new work within the field to build on a service-oriented view of the domain and ensure that new projects are aligned with the strategic direction of the e-Framework.

There are six reference model projects, addressing the domains of:

1. *Assessment*
2. *Learning Activity Management*
3. *Course Information Exchange*
4. *Course Validation*
5. *e-Portfolios*
6. *Personal Learning Environments*

A service-oriented view of the assessment domain is explored in the [e-Learning Framework Reference Model for Assessment](#) (FREMA) and similar perspective of the domain of learning design and learning activity management is articulated in the [Learning Activity Reference Model](#) (LADIE).

The [Exchanging Course Related Information](#) (XCRI) reference model has pioneered an XML schema for representing academic courses. The XCRI work has strong associations with reference models that take a service-oriented view of the course validation and student progression: the [Course Validation Reference Model](#) (COVARM) and the [ePortfolio Reference Model](#).

The reference model for [Personal Learning Environment](#) (PLE), like Web 2.0 approaches and service mash-ups, explores the possibilities of a persistent environment for life-long learners that can interact with institutional systems and services. It also looks for the novel service synergies and combinations that arise when academic resources are factored as discrete services.

There is a strong expectation that the outputs of each of the reference models will inform the direction of new projects to ensure alignment with the principles and practice of service-orientation within the e-Framework.

5. e-Framework: Implications

5.1 JISC e-Learning Programme and SOA

The JISC e-Learning programme, following government initiatives including the DfES e-

strategy and the HEFCE e-Learning strategy, articulates a number of priorities. These general strategic priorities include:

- *Support for lifelong learning*
- *Cross-institutional learning*
- *Workplace learning*
- *Personalisation of learning*
- *Part-time and distance learning*
- *Removal of administrative barriers to progression*
- *Re-use and sharing of learning resources*

The service oriented approach of the e-Framework is technically aligned with these general strategic emphases. When institutional resources are factored as autonomous services, *cross-institutional* and *workplace* collaboration is technically enabled, which in turn supports *progression* and *lifelong learning*. When web services are built using the schemas of open technical standards, technical interoperability is realised, which in turn promotes the *re-use and sharing of learning resources*.

In addition to these general strategic priorities, specific aims for new work include:

- *e-Portfolio*
- *e-Assessment*
- *Personalised learning*
- *e-Admin for learning and teaching*

The work of the e-Learning Framework tool kits and demonstrator projects, and the e-Framework reference model projects, is technically aligned with these specific strategic emphases. There are several tool kits and a specific reference model for the *e-Portfolio* domain, the *e-Assessment* domain and *Personalized Learning*. *e-Admin* has been addressed in a number of tool kits and the areas of course information description, validation and exchange have reference models.

The close mapping of the benefits of the e-Framework's service oriented approach to the general strategic aims of the JISC e-Learning programme, along with a body of existing work to support specific priorities, has strong implications of the new project work.

5.2 Alignment of New Project Work

A body of existing work that supports the specific strategic priorities is already aligned with the e-Framework, and new projects will want to build on this foundation.

Alignment of projects to the e-Framework will be realised when new work adheres to two principles: *adopting a service oriented approach* and the *use of open technical standards*.

Another important factor, when designing applications, is to adopt a mindset that, rather than thinking in terms of a monolithic application, thinks in terms of splitting out functionality into services under the guidance of the principles for service design outlined above. A key factor is whether the functionality is likely be of value to other applications, or whether it is unique to this specific application. If the latter, there may be no point in

separating out as a service, but if the former, it becomes a candidate for a reusable service.

As it develops, the e-Framework will increasingly provide support and guidance for this and will itself be extended by projects *submitting relevant outputs and outcomes to the e-Framework*. To that end, all projects are asked to contribute, as appropriate, business cases, use cases and good practice guidelines, as well as information about the services definitions they have used or developed, to the knowledge base which the e-Framework is developing to support institutions seeking to adopt a service oriented approach.

Open technical standards are widely seen as having played a crucial role in the success of the World Wide Web, and will play a similar role in the success of Web services in general and of the e-Framework. New project work should, by default, make use of both the technical standards for web services (see sections 2.2 and 2.4), and the domain specific standards for e-Learning, e-Admin and e-Research (see section 1.4).

However other technical approaches are permissible, under appropriate circumstances, e.g. where existing standards are already in use (such as Z39.50), or where Web services (SOAP or REST) do not yet meet performance or functional requirements such as for streaming large data sets or for secure transactions, although these issues are currently being worked on.

Submission to the e-Framework will be realised when projects factor functionality into services, build services using the design principles and practices of SOA (see section 2.5) and describe services using e-Framework Definitions and Terminology (see section 1.5). The governance structures of the e-Framework will facilitate and manage submission of services to framework. The submission process aims to ensure the *integrity* of the e-Framework, but is neutral to the technology choices used by concrete service implementations. Guidance should be sought from Programme Managers and Programme Support Staff

5.3 Technology Choices for New Project Work

New projects that align themselves with e-Framework will face decisions on implementation technology and design about which the framework is silent. As well as choosing an implementation platform (typically *Java* or *Microsoft .NET*), the question of *web service style* will need to be addressed. Broadly speaking there are four implementation styles:

- *SOAP remote procedure call style*
- *Other approaches*
- *HTTP REST style*
- *SOAP document style*

Many advocates of SOA discourage the use of remote procedure style. There are also other approaches, such as the use of Z39.50 in library systems. However, most projects will face a choice between an HTTP RESTful style and a SOAP document style.

Service designs based around a requirement to consume or modify an *information*

resource will typically use a HTTP RESTful style. The resource is typically an XML document identified with a *uniform resource identifier (URI)*.

Service designs based around a requirement to invoke an operation will typically use a SOAP style. The operation is identified in the WSDL description of the service.

In a student record system, for example, a *resource based service* to retrieve and modify the XML for a student's record might be designed in the RESTful style.

In a timetable system, an *operation based service* to schedule students to classes might be designed in the SOAP style. While student records (the resources) are an important aspect of the operation, they are not the central focus of the service.

The RESTful service design will typically have a focus on *nouns* (in the above example: *a student*), the SOAP style will typically have a focus on *verbs* (in the above example: *to schedule*).

Technical overhead and complexity plays a part in design decisions also. The online bookseller Amazon offers its APIs in both RESTful and SOAP style, with 85% of the usage taking the RESTful approach. There is growing evidence that the RESTful approach is gaining ground with technical teams, but more sophisticated applications are likely to need the additional facilities that can be supported by SOAP.

6 Summary

The principles and practices for using Web services within contemporary service-oriented architectures have been identified.

General, specific and unexpected benefits that result from the use of service-oriented architectures have also been identified.

The UK JISC and the Australian DEST are working to realise the benefits of service-orientation within academic IT in support of education and research.

The JISC/DEST e-Framework has the theoretical basis, geographic reach, organisational structure and domain scope to provide strategic direction to realise the benefits of service-orientation.

The use of open technical standards is important in ensuring sustainability and interoperability of new project work.

Recent JISC work on tool kits, demonstrators and reference models is important in guiding new project work and promoting alignment with the e-Framework.

The e-Framework Definitions and Terminology provides a theoretical basis for the integrity of the framework. The e-Framework Web site will facilitate submission new service definitions to the framework.